Collaborative Competition for Crowdsourcing Spaceflight Software and STEM Education using SPHERES Zero Robotics

by

Sreeja Nag

B.S. Exploration Geophysics, Indian Institute of Technology, Kharagpur, 2009
M.S. Exploration Geophysics, Indian Institute of Technology, Kharagpur, 2009

Submitted to the Department of Aeronautics and Astronautics and the Engineering Systems Division
in Partial Fulfillment of the Requirements for the Degrees of

Master of Science in Aeronautics and Astronautics
and
Master of Science in Technology and Policy

at the
Massachusetts Institute of Technology
June 2012

Signature of Author _____
Department of Aeronautics and Astronautics and Engineering Systems Division
May 11, 2012

Certified by _____
Jeffrey A. Hoffman
Professor of Practice in Aeronautics and Astronautics
Thesis Supervisor

Certified by _____
Olivier L. de Weck
Associate Professor of Aeronautics and Astronautics and Engineering Systems
Thesis Supervisor

Accepted by _____
Eytan H. Modiano
Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Accepted by _____
Joel P. Clark
Professor of Materials Systems and Engineering Systems
Acting Director, Technology and Policy Program

Collaborative Competition for Crowdsourcing Spaceflight Software and STEM Education using SPHERES Zero Robotics

by

Sreeja Nag

Submitted to the Department of Aeronautics and Astronautics and the Engineering Systems Division
On May 11, 2012 in Partial Fulfillment of the Requirements for the Degrees of

Master of Science in Aeronautics and Astronautics
and
Master of Science in Technology and Policy

ABSTRACT

Crowdsourcing is being researched as a technique to develop small-scale spaceflight software by issuing open calls for solutions to large crowds of people with the incentive of prizes. There is widespread investment of resources in the fields of Science, Technology, Engineering, Mathematics (STEM) education to improve STEM interests and skills. This thesis tackles the dual objectives of building crowdsourcing cluster flight software and educating students using collaborative gaming and competition, both in virtual simulation environments and on real hardware in space. The concept is demonstrated using the SPHERES Zero Robotics Program which is a robotics programming competition. The robots are nanosatellites called SPHERES – an experimental testbed to test navigation, formation flight and control algorithms - onboard the International Space Station (ISS). Zero Robotics allows students to access SPHERES through a web-based interface and the robust programs run on the hardware in microgravity, supervised by astronauts. The apparatus to investigate the influence of collaboration was developed by (1) building new web infrastructure and an Integrated Development Environment where intensive inter-participant collaboration is possible, (2) designing and programming a game to solve a relevant formation flight problem, collaborative in nature - and (3) structuring a tournament such that inter-team collaboration is mandated. The web infrastructure was built using crowdsourcing competitions too, to demonstrate feasibility of building software end-to-end through crowdsourcing. The multi-objective design of experiments had three types of collaborations as variables – within matches (to achieve game objectives), inter-team alliances and unstructured communication on online forums. The data used to evaluate objective achievement were simulation competition scores, website usage statistics, post-competition surveys and satellite telemetry from ISS hardware demonstrations. All types of collaboration showed positive influence on the quality of solutions achieved. Educationally, they showed mixed results and lessons on improving their process of implementation for more impact have been documented. Overall, this thesis ratifies the applicability of the developed framework for crowdsourcing spaceflight software and educating students and maps the utility of collaboration in this framework. A systems dynamics model for generalizing the framework into other programs for simultaneous crowdsourcing and education outreach has been proposed and management policy concerns highlighted.

Thesis Supervisor: Jeffrey A. Hoffman
Title: Professor of Practice of Aeronautics and Astronautics
Thesis Supervisor: Olivier L. de Weck
Title: Associate Professor of Aeronautics and Astronautics and Engineering Systems

Everything I am and ever hope to be, I owe to my mother.

Acknowledgements

First and foremost, I want to sincerely thank my MIT advisors, Prof. Olivier de Weck and Prof. Jeffrey Hoffman for consistent belief in my capabilities and for their guidance. I am particularly grateful to them for going out of their way on many occasions to provide unwavering support of my pursuit of unexpected opportunities and ever-evolving interests. Their dedication to their students is admirable and very deeply appreciated.

I particularly acknowledge Jacob Katz and Alvar Saenz-Otero from the Zero Robotics (ZR) team, without whose leadership and creative, hard work, this program would have reached even close to the heights it is at today. Alvar and Prof. Dave Miller inducted me into the SSL and to the SPHERES ZR team and gave me an opportunity in the awesomeness called "space" - an experience I can never be grateful enough for. In the last two years, I have learned more from Jake on a day-to-day basis than I have from any other person at MIT, be it on technology, programming or the American professional culture. ZR would be incomplete without the consistent support of our collaborators. As student lead of the program for over a year, I identify with the program and my cohort has grown to identify the program with me. I would like to acknowledge Ira Heffan and Jason Crusan for helping with the program and indirectly my thesis. Last but not least, the data and conclusions mentioned here would be incomplete without the hundreds of students and mentors who have participated in, provided feedback and helped improve ZR.

Very deep thanks to Alan Natapoff for helping with my statistical analysis. Social science research was a new field and books could teach me only so much in the short period, his input was timely and very helpful. A part of Chapter 3 in this thesis has been read through and improved by Dave Scott from NASA Marshall Space Flight Center as part of a review board for my IEEE Aerospace 2012 paper, thank you.

A special thanks to the Technology and Policy Program (TPP) family. It is one of the most international programs at MIT and an incredible opportunity to meet and absorb cultures from all over the world. TPP is not just about the classroom; but so much more through culture nights, iAmbassador lunches, leadership retreats, personal guidance from Ed, Sydney and Krista and the always-active OS mailing lists! When I first saw the program's flyer the summer of 2007 and googled

it, I felt an instant connection. And the program has been everything I had imagined and more. My summer internship of 2010 was an especially rewarding experience, thanks to Leopold Summerer for supervising my first work experience in Europe and allowing a ton of travel along with.

Generous funding for my work and study at MIT has come from the MITEI-Total Energy Fellowship, NASA Headquarters, DARPA InSPIRE and the MIT Gordon Leadership Program. Work can never be satisfying through content alone. Colleagues have played an integral part of my educational experience. Thank you, Daniel for support whenever I called for it, Zoe and Danielle who have helped me navigate the internship arena outside MIT very successfully, Sydney for being the last man standing with me when the sun sets on the SSL cluster, Farah for spearheading the Quals movement and introducing me to Aero/Astro-related social events, Alessandra for being my advice-offering sounding board, Gireeja for helping me formulate my ZR crowdsourcing ideas in the larger context of the field and many other SSLers and SERGers for making my association with the two groups a very memorable one.

Finally, and most importantly, I would like to thank my family and friends for being my rock through the turbulent sea of excitement and the directionless free-fall called Grad School. A special mention for Diviya for being my Cambridge family, for Nihit, Harshad, Siddharth, Shamel, Aditya, Arun, William, DC, Srinath, Abhijit, Deepak, Amrit, Sohail, Rachel, Grace, Sheekha, Sharmistha, Farrah and Sayamindu for being fun company as well as friends in need. My world would be incomplete without my parents, grandparents and Arish, I owe the contented solace behind my energy and passion to them.

Table of Contents

List of Figures

List of Tables

List of Acronyms

2D	Two dimensional. Refers to games where only 3 DOF movement of SPHERES is possible or competitions where 2D games are played
3D	Three dimensional. Refers to games where 6 DOF movement of SPHERES is possible or competitions where 3D games are played
API	Application programming interface
ARG	Alternate Reality Games
CS-STEM	Computer Science, Science, Technology, Engineering, Mathematics. Often referred to as only STEM in the thesis.
DARPA	Defense Advanced Research Projects Agency
DOF	Degrees of Freedom
ESA	European Space Agency
EU	European Union
FF	Formation flight
GUI	Graphical User Interface
HS	High School
IDE	Integrated Development Envrionment
ISS	International Space Station
JEM	Japanese Experiment Module in the ISS
MS	Middle School
NASA	The National Aeronautics and Space Administration
r	Pearson's Correlation Coefficient
RR	Round Robin
SoI	NASA Summer of Innovation
SPHERES	Synchronized Position Hold Engage Reorient Experimental Satellites. Singular usage refers to a single satellite
SPHERES Challenge	The annual ZR tournament for HS students
TC	TopCoder Inc.
UI	User Interface

US	United States of America
ZR	Zero Robotics

Chapter 1 –
Introduction

"Adults worry a lot these days. They worry especially about how to make other people learn more about computers. They want to make us all 'computer-literate'. 'L i t e r a c y' means both reading and writing, but most books and courses about computers only tell you about writing programs. Worse, they only tell about commands and instructions and programming-language grammar rules. They hardly ever give examples. But real languages are more than words and grammar rules. There's also literature—what people use the language for. No one ever learns a language from being told its grammar rules. We always start with stories about things that interest us." – Marvin Minsky [1]

What if students were invited on board to solve real-world problems that space scientists and engineers are scratching their heads over? Would they learn the required grammar on their own and justify their 'education'? Could they produce something useful for the scientists and engineers?

With satellite constellations augmenting traditional monolithic satellites for an increasing number of missions, annual satellite launches at hundreds a year and with over 300,000 pieces of space debris larger than 1cm in size in near-Earth orbit, fuel-efficient and robust satellite cluster flight algorithms are required more now than ever before. Scientists and engineers in the distributed satellite systems domain are working to come up with better algorithmic solutions and they could use more help. Crowdsourcing is an emerging methodology by which problems are opened up to crowds of people through an open call to solve these problems with the incentive of prizes for the best solutions [2]. To teach students to interact with and contribute to a technological world, computing, science, technology, engineering and math (CS-STEM) education is becoming very important and nations cannot afford to lag in this area [3]. Furthermore, since humanity is using technology to become more globally social and in turn contributing to more social technology, peer motivation and collaboration is increasingly being used to accelerate problem solving [4] and learning [5].

This thesis addresses the question of whether the two distinct problems of developing cluster flight software using crowdsourcing and improving STEM education can be solved using a single combined program. The thesis examines in depth the role of collaborative competition in the design of such a program. The specific research questions addressed by this thesis are:

1. To what extent is it possible to combine the development and implementation of high performance satellite cluster flight algorithms using crowdsourcing with enhanced CS-STEM education?
2. What is the potential impact of various collaborative competition mechanisms in team-based competitions to achieve *both* better technical results and improved learning outcomes?

The questions will be addressed by a proof of concept followed by statistical analysis of data obtained from two years of competitions. The proof of concept is given by an end-to-end development and demonstration of the SPHERES Zero Robotics Program [6]. The proposed framework will additionally be explained through a systems dynamics model. Recommendations are made based on lessons learned from the development, operations and analysis of the program and the developed model, backed by similar studies in the relevant literature.

1.1. Research Framework

The research framework developed for this thesis has been derived from the TPP Thesis Manual [7] and studies on the design of social experiments [8][9].

1.1.1. Research Questions

The research questions for this thesis are:

1. To what extent is it possible to combine the development and implementation of high performance satellite cluster flight algorithms using crowdsourcing with enhanced CS-STEM education?
2. What is the potential impact of various collaborative competition mechanisms in team-based competitions to achieve *both* better technical results and improved learning outcomes?

The stakeholders in this research effort are the cluster flight scientific community, who want to find better algorithms and develop better spaceflight systems, and students and educators who would benefit from broader outreach and better quality of STEM education. Measuring the value delivered

to both of these stakeholders by the framework of simultaneous crowdsourcing and educational programs is the prime task associated with answering the above question.

1.1.2. Proof of Concept

To prove that (1) cluster flight algorithms and software can indeed be crowdsourced and (2) participants can be educated using the same program, the SPHERES Zero Robotics Program (ZR) was used. ZR is an international robotics programming competition where the robots are SPHERES satellites in the International Space Station. Students and amateur enthusiasts play the challenging games first on a high fidelity simulation and then on real SPHERES hardware in microgravity, and therefore demonstrate flight-capable programs. ZR tournaments consist of student teams playing games in a competitive format, through elimination rounds in simulation or ground hardware and in a final round on the ISS to determine the tournament champion. In order to program the SPHERES, manage their teams and participate in competitions, the students have access to an elaborate website, integrated development environment (IDE) and an online simulator. The ZR program was modified in 2011 to solicit complex trajectory tracking algorithms, and several methods of collaboration were introduced in the previously competitive-only tournament structure. The games were designed such that by writing programs and by implementing them during a tournament-style competition, the participants contribute to developing high performance cluster flight software. Furthermore, the entire web interface for ZR was developed using crowdsourcing contests in collaboration with a commercial company called TopCoder Inc., based on a prototype developed at MIT. The intent was to prove that end-to-end crowdsourcing of spaceflight software, i.e. developing the web interface *by* crowdsourcing and then using it *for* crowdsourcing is possible and beneficial.

1.1.3. Research Methods

The research methods used in this thesis are:

1. Case study of the web interface development using TopCoder crowdsourcing contests
2. Design of social experiments [8][9] using the ZR Tournaments
3. Statistical analysis of the educational value of the ZR Tournaments and the effect of collaborative competition on crowdsourcing and education

4. Data analysis of satellite telemetry returned after hardware operations of SPHERES on the ISS based on well-established methods and standards
5. Systems Dynamics modeling to explain the causal effects of the overall framework of crowdsourcing and education

As mentioned in the previous section, the web interface for ZR was developed through collaboration with TopCoder. The case study highlights the methodology used to run the contests, analysis of the data from the contests in terms of participation, attainment of prizes and quality of the product and the lessons learned through the process.

The design of social experiments draws from the design of experiments (DOE) framework [10] applied to non-random, human participants. It was framed along the lines of an observational, quasi-experimental study and not as an active experiment on human subjects with distinct control and test groups. The objective functions were the value delivered to the cluster flight software development community (through crowdsourcing) and the value delivered to STEM Education. Three types of collaboration mechanisms were considered, which in DOE language translates to three variables or factors with two levels each. Metrics were defined to measure the objectives and the variables. The value of these metrics helped assess whether the simultaneous crowdsourcing and education framework has the potential to satisfy all its stakeholders. Additionally, it helped assess the effect of collaboration on this framework and teach lessons on how the program can be improved in future years. The 'value' of the above metrics was assessed using statistical analysis of data gathered during the ZR Tournaments – through simulation statistics on the IDE, website usage statistics, the satellite telemetry obtained during operations in the ISS and surveys taken after the tournament.

Finally, the systems dynamics model was developed to pull together the lessons learned through the process of designing, developing, operating and analyzing the ZR program, highlight the advantages of the simultaneous crowdsourcing-education model and at the same time pointing out management policy precautions regarding its application.

1.1.4. Logic

The Wheel of Science approach to research [11], introduced in 1971, categorizes research into two distinct subsets of reasoning and drawing conclusions: Deduction and Induction. Deduction is the logical model [8] in which specific expectations, or hypotheses, are first developed on the basis of underlying principles, experiments are then conducted to accept or reject these hypothesis and the observations give what is known as an ideographic conclusion (based on ideals) – as shown in Figure 1 on the arc to the right. It is also called the Cartesian model after Rene Descartes and is the preferred model of analysis in this thesis. Induction is the logical model in which general principles or theories are developed and inferred from specific observations obtained from experiments and the conclusion drawn is known as a nomothetic one – as shown in Figure 1 on the arc to the left. Induction is also called the Baconian model after Sir. Francis Bacon and is used very sparingly in this thesis and only when an established framework of methods already exists to prevent directionless collection of data.

The research in this thesis is to explore and prove viability of the concept of crowdsourcing spaceflight software development and at the same time engaging and educating school students. A parallel intent is to evaluate the effect of collaborative competition mechanisms in the proof of concept studies. Therefore, proof of concept and collecting the ZR Tournaments data can also be seen as a critical experiment [7]. The Wheel of Science approach applied to the contributions of this thesis can be categorized as:

1. Deductive/Ideographic conclusions on:
    a. The influence of collaborative competition based on theories of collaboration effects on gaming, crowdsourcing and STEM interest within the competitive tournaments
    b. Benefits of the ZR Program due to combined crowdsourcing and STEM Education based on the individual theories of both objectives
2. Inductive/Nomothetic conclusions on:
    a. The case study of developing the ZR web interface using TopCoder-directed crowdsourcing contests based on an earlier prototype developed at MIT
    b. Tournament demographics and student and mentor satisfaction using essay-type feedback received after each ZR tournament season.

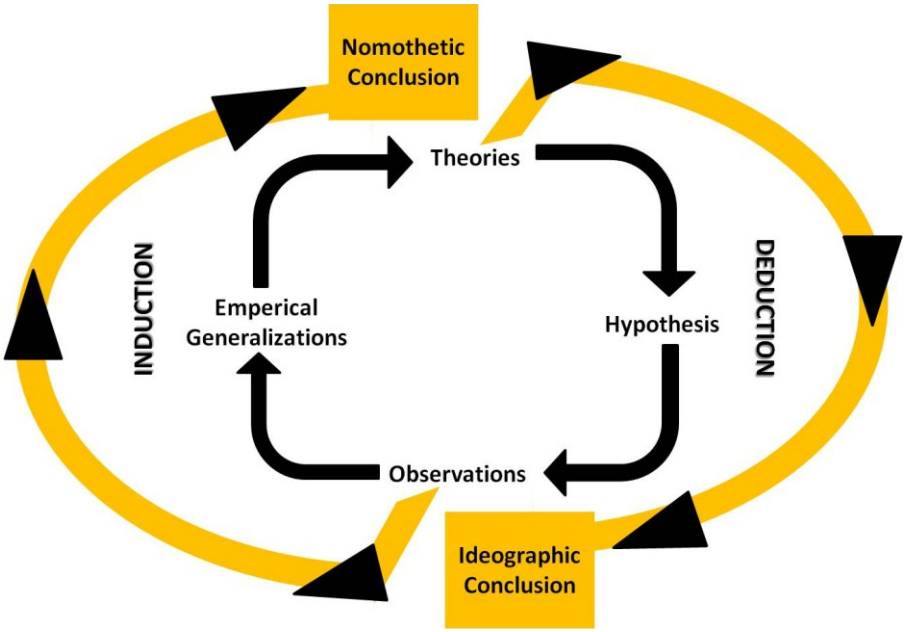


Figure 1: The Wheel of Science to describe the Social Research Processes [11]

The research in this thesis is primarily ***exploratory*** [8] since it attempts to answer the 'what' questions (Is this possible? What could make it better? What are the effects of various factors on it?). In trying to answer the 'what' questions, it also tries to answer the 'why' questions for the observations made so that the lessons can help design the framework better and refine the methods of analysis for the future of not only the ZR program but similar efforts. Furthermore, the development of the web infrastructure for the ZR tournaments, presented as a case study [12], was done using existing TopCoder crowdsourcing methodology, which has been described and explained. In this capacity, the research is also ***explanatory*** and ***descriptive***. Descriptive hues can also be seen in some chapters which define the overall ZR program and its operations, but this has been curtailed significantly or included in the appendix, so that the reader is not distracted from the central thesis. Technical details on the ZR program [6] and specifications and capabilities of the SPHERES satellites [13] have been described in Chapters 3, 4 and 5.

1.2. Document Overview

This section provides a brief overview of each chapter of the thesis. Chapter 2 introduces the main areas of interest of this research effort, provides a detailed literature review of each area, followed by the analysis of gaps in literature and how the thesis fits into it. Chapter 3 describes the web infrastructure for the Zero Robotics program and its development using TopCoder crowdsourcing contests, based on an MIT prototype. The chapter has been presented as a case study of the methodology followed, analysis of the data from the contests (obtained by querying the TopCoder SQL database with permission) and the lessons learned from it. Chapter 4 introduces the concept of the ZR Tournaments, their components, how they leverage the developed web infrastructure and how they were used to answer the research questions. It also develops the metrics for evaluating the thesis objectives. Chapter 5 is the main results chapter, which discusses the analysis of data generated by the ZR Tournaments – simulation statistics on the IDE, website usage statistics (both obtained by querying the ZR SQL database), satellite telemetry obtained during operations in the ISS (obtained with permission from NASA Marshall Spaceflight Center) and surveys (obtained through SurveyMonkey databases) – to draw research conclusions at the intersection of crowdsourcing, education and collaboration. The chapter wraps up with documentation of lessons learned for the future. Chapter 6 introduces a systems dynamics model for the simultaneous crowdsourcing and education framework with collaboration, makes general recommendations on how maximum benefits can be reaped and finally, based on literature review and lessons learned, discusses management policy implications.

Chapter 2 –
Background and Motivation

This chapter presents the background literature on the three areas of interest in this thesis. This review led to the identification of a gap in research at the intersection of these areas. A research objective was framed to fill up the gap. The motivation behind framing the research hypothesis and the effort invested in analyzing it and proving it has been described here.

The three main areas of interest in this thesis are:

1. Crowdsourcing to develop innovative solutions to cluster flight problems for satellites
2. CS-STEM Education of the next generation workforce
3. Collaborative Mechanisms within Gaming and Competition

Crowdsourcing is a blanket term used for many open source development efforts on open innovation platforms in the recent past both in research and industry[1]. Some famous examples of such efforts have been Wikipedia, Linux, Fold It and companies such as Innocentive, Threadless and TopCoder[1]. In the context of this research effort, crowdsourcing is defined as the methodology by which a well-defined problem is attempted to be solved by announcing it as an open call for solutions to crowds of people with the incentive that the best solutions will be awarded prizes. There is no restriction on the methods that the crowds can use to solve the problem, but there may be a time limit given to come up with a solution and constraints on the ways in which the proposed solutions are submitted.

Satellite formation flight is the concept that multiple satellites (e.g. satellite constellations) can work together in a group to accomplish the objective of one larger, monolithic satellite [14]. A satellite constellation is a group of artificial satellites - a set of physically independent, "free-flying" modules that each collaborate on-orbit to collectively achieve a certain level of system-wide

[1] A comprehensive list of open innovation platforms for R&D, marketing, design and ideation, collective intelligence and trend prediction, human resources and freelancers, open innovation software, intermediary services, creative co-creation, corporate initiatives such as product ideas, branding and design, peer production and public crowdsourcing is available at: http://www.openinnovators.net/list-open-innovation-crowdsourcing-examples/. Last accessed on April 24, 2012.

functionality. One of the key requirements of a satellite cluster[2] with multiple physical entities, a type of constellation, is the need for all the modules to fly within a specific range of each other (communication range, sensing range, data transfer range, etc.) in orbit in order to be functional. This requires solutions to multi-body problems in Earth orbit, precise determination of position and time, and sometimes relative and absolute attitude and orientation, advanced control algorithms, trajectory planning and many other issues.

CS-STEM is an acronym for Computer Science (CS), Science, Technology, Engineering and Mathematics. CS-STEM Education refers to efforts invested in bringing students and young professionals, the next generation workforce, up to speed in the fields of CS-STEM and therefore be prepared to address the grand challenges of the 21st century. A recent editorial in the Science Magazine [15] defined STEM Education as, "*For most, it means only science and mathematics, even though the products of technology and engineering have so greatly influenced everyday life. A true STEM education should increase students' understanding of how things work and improve their use of technologies. STEM education should also introduce more engineering during precollege education. Engineering is directly involved in problem solving and innovation, two themes with high priorities on every nation's agenda. Given its economic importance to society, students should learn about engineering and develop some of the skills and abilities associated with the design process.*" Given the current generation's dependence on digital and media technologies, a nation's economy depends upon its people's ability to contribute computationally to its challenges. Computer science has moved up the ranks rapidly and found its spot as an important part of STEM education.

Gaming, in the obvious sense, is the act of playing a game. In the context of this research effort, gaming is defined as the act of playing a game using an online interface or inside a virtual world. ***Collaborative gaming*** and associated competition refers to the recent gaming phenomenon called 'massively multiplayer online role-playing games' (MMORPGs). MMORPG is a genre of role-playing video games in which a very large number of players interact with one another within a virtual game world. Revenue for the gaming industry is generated largely through subscriptions and sometimes through advertising. In 2008, the consumer subscription spending on subscription MMORGGs in North America and Europe was over $1.4 billion [16].

[2] The words 'constellation' and 'cluster' as well as 'formation flight' and 'cluster flight' may be used interchangeably in the thesis and in literature. The intended definition of cluster for this thesis is the one mentioned here and cluster flight indicates formation flight for satellite clusters.

The ***motivation*** for the research objective presented in this thesis was developed as introduced in Figure 2 and the research objective itself will be focused on in Section 2.5.

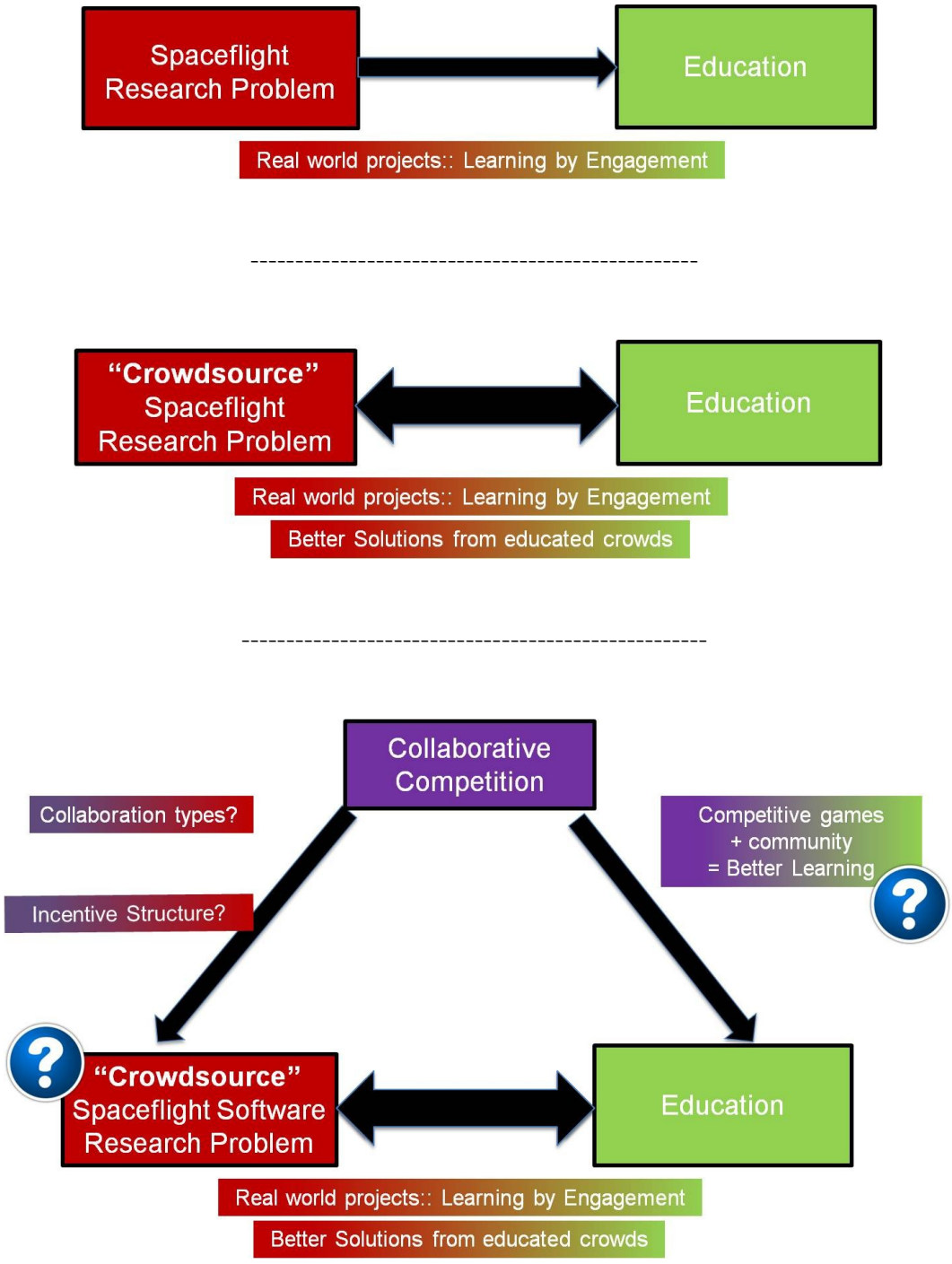


Figure 2: Conceptualization of Thesis Motivation

The ***first block*** in the Figure 2 shows STEM education achieved through engagement of students using real-world projects such that they can learn by doing. In the context of satellite constellation and cluster flight fields, this implies allowing student access to relevant spaceflight research problems such that they can learn about CS-STEM fields by solving real-world problems with known solutions. The ***second block*** introduces the concept that, since students are allowed access to real-world problems and learn by solving them, it would be a win-win for both students and the research community if the research problem provided to them was something yet to be solved or improved within the research community (see bidirectional arrow). Given the rising number of missions involving small satellite constellations, the demand for robust guidance, navigation and control (GNC) algorithms for flying these constellations through an increasingly crowded space environment is increasing. Therefore, if difficult, unsolved GNC and path-planning problems for small satellites were crowdsourced i.e. opened up to crowds of students to solve with the incentive of prizes, both stakeholders would benefit from the most successful solutions. Thus, what is of interest in crowdsourcing is not the average quality of submitted solutions but the best most capable and potentially most innovative solutions. In terms of CS-STEM outcomes on the other hand we are interested in raising the average level of knowledge and learning of all participants. The students learn by engagement with a difficult real-world problem and the crowdsourcers, who put the problem out there to be solved, get many potential solutions to it. The ***third and last block*** of the figure explores the effect that collaborative competition has on both the above objectives: STEM education, where the stakeholders are students, mentors and educators, as well as the quality of crowdsourced solutions, where the stakeholders are the scientific community that is looking for the solution to the spaceflight problem being crowdsourced. It questions the appropriate collaboration mechanisms between the crowds of participants i.e. students and appropriate incentive structures that would produce a positive effect on both objectives: innovation and learning. While the thesis motivation has been generically described for 'spaceflight problems', the problems targeted in the thesis are, specifically, formation flight problems for satellite clusters.

The ***research motivation*** of this thesis can therefore be summarized as exploring the *proof of concept* that it is possible to crowdsource a cluster flight algorithm or software used for small satellites *while at the same time* educating students in CS-STEM simultaneously. Further, the thesis also analyses the impact of such an initiative on crowdsourcing and educational objectives and measures the effect that various types of *collaboration* among the participants have on these objectives.

2.1. Crowdsourcing

The term 'crowdsourcing' was introduced by Jeff Howe in 2006 in a Wired magazine article. He later went on to define the term as: '*Simply defined, crowdsourcing represents the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call*' [2]. Most generally, a person or organization with a problem to be solved invites a crowd to come up with solutions and offers incentives for contribution. Crowdsourcing has been classified into various typologies based on the aims of the practice [17]: Problem solving (crowd wisdom), creative input (crowd creation), opinion polling (crowd voting), outsourcing tasks (crowd production) and raising money (crowd funding).

2.1.1. Historical Applications

Challenging crowds to compete to achieve a difficult goal by providing the incentives of prizes has a long history and has led to many successful competition solutions (hence, the terms 'challenges' and 'competitions' will often be used interchangeably). In 1714, the English parliament, seeking to solve the difficult problem of accurately determining ships' longitude at sea, created a Board of Longitude to oversee the offer of a prize of 20,000 pounds to anyone who could solve the problem. While the Parliament could have directly funded astronomical research efforts, they instead chose to offer a prize to anyone who could solve the problem. John Harrison, a self-taught clock maker developed an improved clock design that would be accurate at sea and eventually won the prize [18]. In 1775, a prize of 100,000 francs was offered by the French Academy of Sciences for the production of alkali soda ash (sodium carbonate) from salt (sodium chloride) [19] to soften water for washing purposes. A surgeon, Nicholas Leblanc, developed a process that some have since characterized as the beginnings of the modern chemical industry[3]. In 1919, a $25,000 prize was offered by hotel magnate Raymond Orteig to the first person to fly non-stop between New York and Paris. In 1927, Charles Lindbergh won that prize, landing 2½ hours ahead of schedule [20].

[3] It is interesting to note, however, that both Harrison and Leblanc had trouble collecting on their prizes, Harrison due to the resistance of the astronomical establishment that was holding out for an astronomical solution and Leblanc due to the French Revolution.

2.1.2. Recent Applications

Crowdsourcing applications in the non-aerospace industry are on the order of hundreds in the recent past. The MIT Center for Collective Intelligence has several projects going where large-scale socio-technical problems are solved by opening them up to external networks and relationships. Among their most influential projects are: Climate CoLab [21], which attempts to harness the collective intelligence of large numbers of people to address the problem of global climate change, and Deliberatorium [22], which explores the integration of ideas from argumentation theory and social computing to help large numbers of people enumerate the issues, ideas, and tradeoffs for complex problems with much greater signal-to-noise and much more systematic organization than existing (e.g. forum, wiki, or idea-sharing) technologies. CoLab allows people to register on their website as guests and create climate change related debates, argue on existing debates or rate the existing arguments. The intellectual input from people then feeds into a model that generates charts for the time projection of temperature, sea level, mitigation or damage cost as a percent of GDP and physical impacts on water, agriculture, health, etc as a function of temperature rise. Crowdsourced solutions are also invited to turn knobs for action which can be factored into the mitigation aspects of the model. The Deliberatorium aims to create an argument map with moderators for questions/problems, potential solutions, pros and cons, etc. in several layers.

In the fields of biology, crowdsourcing is currently being used to solve difficult protein folding, synthetic biology, neurobiology problems that cannot be optimized globally by any algorithms, by bringing crowds of humans in the loop. A new startup, called Eyewire, has opened up the problem to crowds using their website - http://eyewire.org/. The challenge is to map the neural connections of the retina by analyzing images that were acquired using serial electron microscopy at the Max Planck Institute for Medical Research in Heidelberg, Germany. Dozens of players log in everyday to play the game and there is a live leaderboard that dynamically reports the scores. The human eye is one of the prime examples of irreducible complexity, a point of debate in evolution theory, and, more relevant to this thesis, has neural connections that have baffled neurobiologists for years. The most successful biological crowdsourcing breakthrough in the recent past has been achieved using Foldit. Foldit is an online puzzle video game based on protein folding where the game objective is to fold the structure of selected proteins to the best of the player's ability, using various tools provided within the game. The score is a metric of the structure's stability and the highest scoring solutions

are analysed by researchers, who determine whether or not there is a native structural configuration (or native state) that can be applied to the relevant proteins, in the "real world". Using Foldit, gamers have deciphered the molecular structure of a key protein that retroviruses like HIV need to multiply [23] - an achievement that scientists believe will aid in the development of new AIDS drugs. The largely non-scientist gamers came up with an accurate model of the so-called protease molecule in three weeks while biochemists had been trying to create such a model for more than a decade.

A recent example of the use of large-scale innovation tournaments in the aerospace industry is the X-Prize competition. On October 4, 2004, the X PRIZE Foundation awarded a $10 million prize to Scaled Composites for their craft SpaceShipOne [24]. Aerospace designer Burt Rutan and financier Paul Allen led the first private team to build and launch a spacecraft capable of carrying three people to 100 kilometers above the earth's surface, twice within two weeks, the first humans to achieve this feat. SpaceShipOne exceed the altitude of 100 kilometers but did not achieve orbital velocity.

U.S. Government agencies can now use competitions to reach out to thousands of citizens, which is why the White House has been encouraging agencies to consider the use of challenges as a policy tool. At the outset of his Administration, President Barack Obama signed the Memorandum on Transparency and Open Government, committing the Administration to creating a more transparent, participatory, and collaborative government. In Sept. 2009, the President released his "Strategy for American Innovation" calling for agencies to increase their ability to promote and harness innovation by using policy tools such as prizes and competitions [25]. On Dec. 8, 2009, the Director of the Office of Management and Budget (OMB) issued the Open Government Directive, which required executive departments and agencies to take specific actions to further the principles established by the President's memorandum, including to develop an Open Government Plan that should "include innovative methods, such as prizes and competitions, to obtain ideas from and to increase collaboration with those in the private sector, non-profit, and academic communities [26]. In January 2011, the America COMPETES Act [27] was reenacted, which authorized all government agencies to conduct challenges and competitions.

Competitions must be designed to meet their intended goals. There is no single type of challenge that can fulfill all needs. A program that is solely intended to educate the public about a topic will be designed differently than a challenge that is created to obtain an innovative solution. To explore

these differences, NASA created the NASA Tournament Lab (NTL) in collaboration with Harvard Business School and TopCoder to use open innovation Competitions to solve problems within the NASA scientific and research community, and to reach beyond the walls of the research centers and engage the world to help solve its challenging and complex problems [28]. Some examples of successfully crowdsourced (crowd wisdom) NTL problems are:

- NASA required the development of a robust software algorithm that would efficiently recognize vehicles in aerial images [29]. A set of 1000 images containing vehicles and 3000 images containing only background were provided as test cases. The algorithm submissions were tested against a larger set of data. After the problem had been selected and framed, a three-week competition was held on the TopCoder platform. During the competition, 139 programmers from around the world participated by submitting 549 total submissions. The preliminary data analysis by the NASA team showed that the top five solutions were a significant improvement over their current algorithms, employing "state of the art computer vision methods." NASA is currently working on integrating the winning submissions into their own solution.

- NASA's Space Life Sciences Directorate required the development of a software algorithm that would solve a "backpack problem," which consited of recommending the ideal components of the space medical kit included in each manned space mission [30]. As mass and volume are restricted in space flight, the medical kit has to be designed in a way such that both expected and unexpected medical contingencies can be met through the resources in the kit as well as be attuned to the characteristics of the space flight and crew. The challenge was to develop a software algorithm that, based on mission characteristics, would minimize mass and volume and provide the resources necessary to minimize poor health outcomes or the necessity of premature mission aborts. After the problem had been selected and framed, a 10-day competition was held on the TopCoder platform. During those 10 days, 439 programmers from around the world participated by submitting 5994 program submissions. The preliminary data analysis by the NASA team showed that the solutions developed by the leading entries far surpass the current state of the art internal to NASA in terms of computation time (30 seconds as compared to 3 hours), diversity of technical approaches and robustness. After the competition ended, NASA researchers reviewed the top 5 highest scoring code submissions by

looking at the actual code and documentation stating that *"The amount of useful code developed in such a short amount of time really made us reconsider some of the ways that we write software*" [30]. The NASA team was not able to directly import the code into their software because their model was created with the SAS software analytics package, but they converted elements from the winning submissions to develop a new algorithm to design the medical kids used in space missions. Thus, crowdsourced solutions may require additional work for adaptation or integration into a larger host system.

- NASA wanted to generate ideas for new applications to allow exploration and analysis of the NASA Planetary Data System (PDS) databases - http://pds.nasa.gov/. While rich in depth and breadth of data, the PDS databases have been developed in a disparate fashion over the years with different architectures and formats; thereby making the integrated use of the data sets difficult. Consequently, a challenge faced by NASA and the research community is to maximize the usefulness of the enormous amounts of PDS data and identify ways to combine the data that is available to generate interesting applications (e.g., visualizations, analysis tools, educational applications, mash-ups). The goal of this challenge was to generate ideas for these applications. Submissions included a description of the overall idea, a description of the target audience, the benefits of the application for the target audience and the nature of the application (how should the application be implemented?). Overall, submissions were expected to be around 2-3 pages of text including figures and tables and images. No code or software was necessary. Prizes included a $1,000 grand prize and three $500 runners-up prizes. A $750 "community choice award" selected by the community also was awarded. There were over 40 submissions received, with the winner proposing an application concept that was focused on a PDS documents parser, processor and validation tool that could be used to identify what areas, parameters, and objects of the planetary systems are well researched and what objects are "white spots," meaning that the data is sparse and more research is needed [31]. Future competitions will include implementing the winning idea.

More recently, NASA launched an international competition to develop space software applications using http://spaceappschallenge.org on April 21-22, 2012 with events across seven continents (Antarctica included) and in space. The apps competition will bring people together to exploit openly available data collected by space agencies around the world to create innovative solutions to longstanding global challenges. Open data includes statistics, facts and other information that is

freely available to the public. Teams will compete with others around the world to use open data to design innovative solutions to a predetermined series of global challenges. Participants will be free to develop mobile apps, software and hardware, data visualization, and platform solutions that could contribute to space exploration missions and help improve life on earth. In October 2010, the European Space Agency unveiled the Space Game online (http://sophia.estec.esa.int/thespacegame/) for the World Space Week aimed to improve interplanetary trajectories. The intent was to 'watch' humans design complex interplanetary trajectories (e.g. using creative flybys, gravity assists) while optimizing time of flight and delta-V and improving the intelligence of computer algorithms using the insights gained from all the submissions. The website received more than 3,000 hits in the span of the Space Week and received 592 solutions. Since 2010, there have been 2 missions in the Space Game where 6,254 users have submitted 2,454 solutions to ESA.

To summarize, competitions and more recently, crowdsourcing competitions, have had a successful history in spurring innovation and solving problems creatively (crowd wisdom) and in large numbers (crowd production). The government and NASA have only recently tapped into the power of competitions to organize their enormous amounts of available information, identify and solve complex problems and to democratize the innovation process. The role of the internet in enabling crowdsourcing and competitions has been paramount.

2.2. Formation Flight for Satellite Clusters

A satellite constellation is a group of artificial satellites - a set of physically independent, "free-flying" modules or entities that each collaborate on-orbit to collectively achieve a certain level of system-wide functionality. They may communicate with each other, are aware of at least a subset of each others' states, operate with shared control and complement each other in terms of overall functionality. A satellite cluster is a constellation that needs to maintain a certain amount of proximity with each other and must fly in formation accordingly. While each satellite in a constellation has traditionally been considered a self-sustaining entity in terms of the entire spacecraft bus (everything minus the payload), a new paradigm design in constellations called fractionated spacecraft allows almost all subsystems of a satellite to be distributed among different physical elements. Each module in a fractionated spacecraft is composed of various subsystems, and

thus a fractionated spacecraft might consist of separate modules responsible for power generation & storage, communications, payload, and so on. In 2008, DARPA began a program called the System F6 Phase 1 (for Future, Fast, Flexible, Fractionated, Free-Flying Spacecraft) aiming to generate a new paradigm for space systems, especially in the responsive space sector [32], [33]. The large, monolithic spacecraft of today are not typically designed for responsiveness and have other drawbacks (e.g. delay cascading in manufacturing), which a fractionated spacecraft approach could potentially eliminate or reduce. This approach allows for a disruptive change in how satellites are built and how they will be used, since the establishing of a space infrastructure lowers the entry barrier for satellite building and allows for resource sharing. The main idea is to further modularize satellites up to the point where the monolithic spacecraft can be decomposed into a network of wirelessly linked modules, all separate smaller spacecraft, flown in a cluster while providing the same or more capabilities than a single spacecraft. The concept is assessed in [34] mainly regarding its influences on the aerospace sector, resulting from standardization and mass production.

One of the key requirements of a satellite constellation or fractionated spacecraft is the need for all the modules to fly within a tight ellipsoid in orbit in order to be functional. This requires solutions to multi-body problems in Earth orbit, precise determination of position and time, advanced control algorithms, trajectory planning and a host of other issues. There have been many instances of successful constellation formation flight. In the late 1990s, the US Air Force began the conceptual design of TechSAT-21, which was to demonstrate the ability of several satellites to replace a large monolith in an interferometric radar mission. Although the program was cancelled later, it provided a rich resource of literature on formation flight technology. NASA demonstrated Enhanced Formation Flying (EFF) via their first formation flight mission in 2000 called the Earth Observation 1 (EO-1), which flew in formation with LandSat-7, an Earth environment satellite launched in 1999. NASA's New Millennium Program, of which EO-1 was a part, was thus a great success and it paved the way to many technological FF milestones, which has now led to the plan of the Terrestrial Pathfinder Mission (TPF). In TPF, a virtual space interferometer system with a 1km baseline will be implemented to detect and analyze the light from distant stars. NASA has also planned MMS (Magnetospheric Multi-Space) and SIRA (Solar Imaging Radio Array) as future formation flying missions. In 2002, the Gravity Recovery and Climate Experiment (GRACE) supported by NASA and DLR demonstrated formation flight by a pair of satellites to measure the Earth's gravity field and its temporal variations. The European Space Agency (ESA) has proposed the following

formation flying projects: PROBA-3 with 3-axis stabilized pair of satellites [35], DARWIN to study the origins of life with one leader and 4 follower satellites and SWARM to study the Earth's magnetic field with 3 satellites.

A representative example of required features in formation flight is that of collision and kinetic threat avoidance. In a space environment that is getting increasingly crowded, another key requirement is collision avoidance from other satellites, orbital debris or even anti-satellite missiles. The US Space Surveillance Network is tracking more than 19,000 Earth-orbiting man-made objects more than 10 cm in diameter, of which roughly 95% are considered debris [36]. There are also an estimated 300,000 additional man-made objects in Earth orbit measuring 1-10 cm and more than a million smaller than 1 cm. In February 2009, a defunct Russian Cosmos satellite collided in space with a commercial Iridium satellite, not only causing destruction but also adding to the debris already present in space. In June 2007, NASA reported manoeuvring its $1.3 billion Terra satellite to avoid a piece of Fengyun-1C debris. Antisatellite (ASAT) missiles have been technologically demonstrated since 1960, when a US U-2 spy plane was destroyed by a USSR ASAT [37]. The US tested its Air-launched miniature vehicle (ALMV) in the early 1980s to demonstrate 'kinetic kill'. Thereafter, in spite of oscillating treaties such as the Outer Space Treaty (1967), Anti-Ballistic Missile Treaty (1972) and the ban on ASAT testing (1986), the third generation ASAT systems were developed. Most recently, in 2007, China tested the kinetic kill technology of its ASAT system by shooting down one of its own satellites. Collision avoidance has been dealt with in past literature, although very rarely for completely distributed systems. The historic approach has been linear programming where the obstacles and required formation is treated as a constraint and modelled as a MILP [38]. NASA's Jet Propulsion Laboratory have a collision avoidance approach based on the Bouncing Ball algorithm (BB) and Stalemate which adopts a heuristic approach to multiple satellite reconfigurations [39]. ESA's PRISMA satellites, under the contract of the Swedish Space Corporation have robust collision avoidance algorithms for autonomous formations where separation and nominal guidance are solved for analytically, since the satellites have near-circular orbits [40], [41]. Another approach has been to propagate uncertainty covariances to calculate the probability that the relative displacement between two objects is less than a 'collision metric' [42]. Princeton Satellite Systems has come up with distributed guidance laws for low, medium and high autonomy of constellations, solved for using linear programming [43].

However, most of the approaches outlined above are either centralized algorithms or assume the presence of a 'captain' or 'master' to assess the distributed inputs from the less intelligent agents. Unique collision avoidance manoeuvres using fully distributed satellite systems have been implemented at MIT in the Space Systems Laboratory. One algorithm works by predicting the closest point of approach and then overriding regular satellite controls to move the satellites in a direction perpendicular to the collision to avoid it [44]. The theoretical concepts of the algorithm has been tested on a nano-satellite testbed called the SPHERES facility on board the International Space Station [45] developed by MIT SSL. More recently, distributed collision avoidance and threat avoidance has been distributed in theory using different types of artificial potential functions (APF) whose parameters were determined by a method called "equilibrium shaping" [46]. The equilibrium shaping method as well as the application of neural networks to formation flight will soon be tested on the SPHERES testbed on the ISS.

The previous two paragraphs are intended to serve as an example of a single formation flight feature and the depth of literature and algorithms that are being developed to address it. There are many such features that need the attention of the scientific community, as problems to be solved or bettered. Moreover, since the paradigm of space agencies is moving from large, monolithic spacecraft to constellations and clusters of smaller satellites or even fractionation, the need for better formation flight guidance, navigation and control algorithms is more pronounced than ever before.

2.3. STEM Education

CS-STEM Education, as defined earlier, is education in the fields of computer science, science, technology, engineering and mathematics. The terms STEM and CS-STEM are used interchangeably from Chapter 3 onward in this thesis. The intent is always to mean CS-STEM since computing is now considered an indispensible 21st century skills (will be explained in this section). The idea that a revolutionary, hands-on method of education is required to create and maintain students' interest in STEM has been floating about since decades. Children learn by doing and thinking about what they do, an idea supported by Marvin Minsky in his book 'Introduction to Logo Works'. Minsky believes good STEM education is not only that which teaches students to use and learn about new technology, but also gives them the tools to modify technology to suit their own needs.

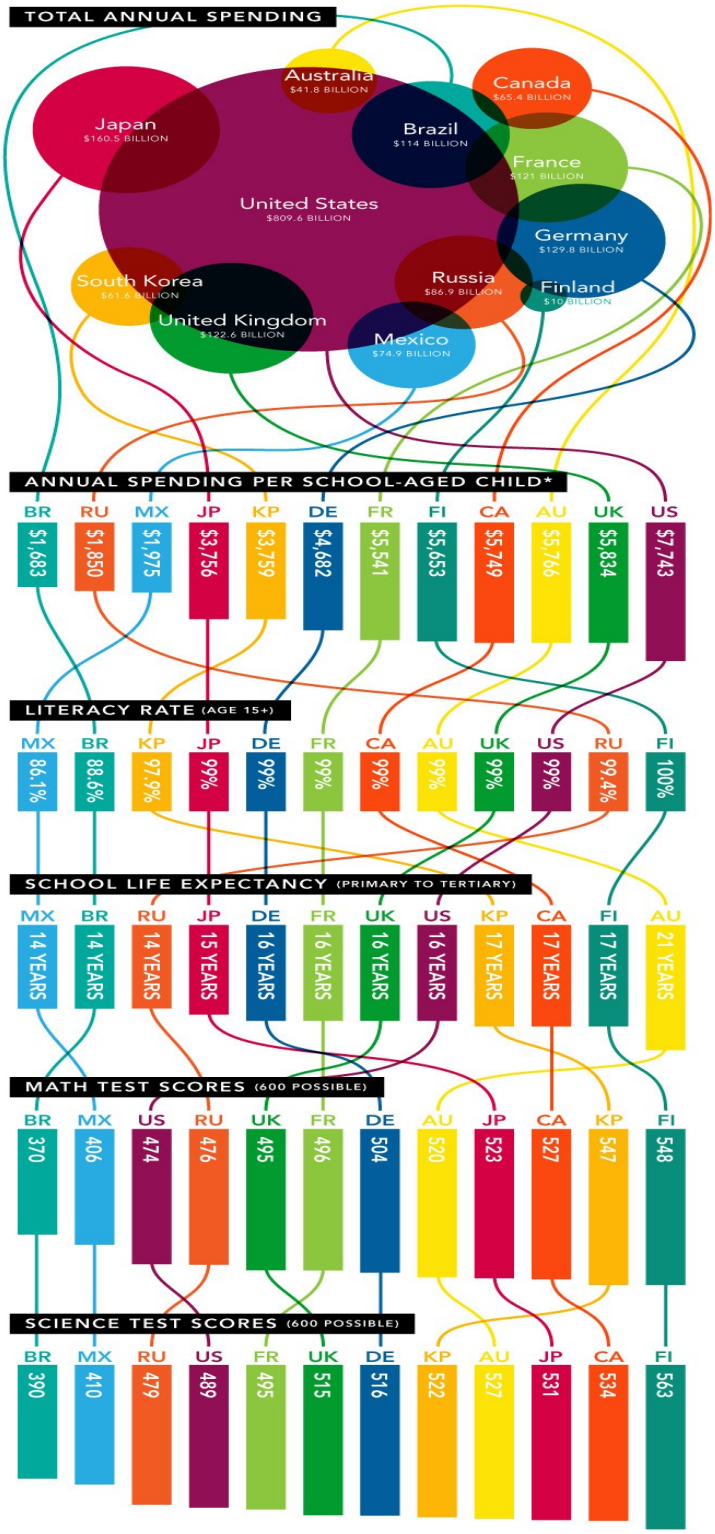


Figure 3: Infographic prepared by the Infographic by Master of Arts in Teaching, USC. References cited in the text.

The United States, in spite of being the largest spender on education in the world has among the lowest Science and Math test scores in the developed world. The 2007 TIMSS[4] scores showed that 15 percent of the US. fourth-graders and only 10 percent of U.S. eighth-graders scored at or above the advanced international benchmark in science [47]. Figure 3 shows an infographic prepared by the University of Southern California using data published by the OECD [48], CIA [49] and UN from 2003 through 2009. It compares the education spending in 12 countries with their corresponding literacy rate, school life expectancy and math and science test scores (standardized). The students surveyed for the figure for test scores were 15 years old. Each country has been plotted in a single color such that it can be traced down. It can be seen that although the USA is the highest spender i.e. circle with the largest area at the top and also has the highest per school child expenditures of $7,743, it progressively shifts to the left going downward i.e. it gets outperformed.

While hiking up U.S. performance in math and science is important, computer science skills also require a special mention here. "21st Century skills", the new buzzword in education, refers to a growing global movement to redefine the goals of education, to transform how learning is practiced each day, and to expand the range of measures in student achievement, all in order to meet the new demands of the 21st Century [50]. Literature shows that incorporating critical thinking, problem solving and communication into the teaching of core academic subjects is indispensible to 21st century learning. Moreover, the three core skills [51] required are:

1. Life and career skills (flexibility, adaptability, initiative, self direction, communication, social and cross-cultural interaction, productivity and accountability, and leadership and responsibility).
2. Learning and innovation skills (critical thinking and problem solving, communication and collaboration, and creativity and innovation applied to imagination and invention).
3. Information media, digital media and technology skills.

[4] The Trends in international Mathematics and science study (TiMss) is an international assessment and research project designed to measure trends in mathematics and science achievement at the fourth- and eighth-grade levels as well as school and teacher practices related to instruction. Since 1995, TIMSS has been administered every 4 years. TIMSS 2011, the fifth study in the series, will involve students from more than 60 countries, including the United States. TIMSS is sponsored by the International Association for the Evaluation of Educational Achievement (IEA) and managed in the United States by the National Center for Education Statistics (NCES), part of the U.S. Department of Education.

Note that 21st century skills call upon not only pure math and science but the ability to use and manipulate computer technology. To enforce the point further, while the 3 R's of "reading, 'riting, and 'rithmetic" were deemed essentials of mandatory public schooling in the 19th century, 21st century literacy is *defined* by 4 R's [52]: Reading, 'riting, 'rithmetic and 'rithms, the fourth R being algorithms or basic computational skills. Therefore, any sort of STEM Education effort should certainly involve computer science and media interaction (i.e. CS-STEM) as well as promote communication, leadership, critical thinking, imaginative problem solving as soft skills.

Space exploration and technology has always been a region of fantasy to everyone, especially children, and space-related activities are therefore excellent motivators to learning and fostering interest in STEM. Not surprisingly, education of the next generation workforce has always been one of NASA's mission goals. Two of six goals released as part of NASA's 2011 Strategic Plan have direct relevance to STEM and education [53]. For instance, Goal 6 states: "*Share NASA with the public, educators, and students to provide opportunities to participate in our mission, foster innovation and contribute to a strong National economy.*" It directly calls upon the agency to create opportunities for broad outreach and student involvement in projects. Goal 3: "*Advance aeronautics research for societal benefit*" indirectly refers to educational advancement too, since a society's future depends on the education of its citizens and their ability to use their education to contribute to the economy.

Since the 1980s, NASA has played a very beneficial role in directed space education outreach in the United States, inspiring students and teachers across the nation. Two of NASA's largest educational programs: the NASA Explorer Schools (NES) and NASA Spaceward Bound programs are examples of outreach (in the past and currently ongoing) to promote student interest in science, technology, engineering, math, and geography (STEM-G) careers [54]. The ISS since its starting stages has been extensively used to conduct research by universities, and extra effort is being invested in getting students involved with the onboard activities. The NASA "International Space Station Education Concept Development Report [55] states: *"Utilizing the International Space Station National Laboratory for education is an effort initiated in response to the 2005 NASA Authorization Act, which designated the U.S. segment of the ISS as a national laboratory".* The report reveals a framework where goals are laid out in a pyramid structure: inspire a large number of students, engage a set of them, and educate a sub-set of these.

However, a truly revolutionary education program should and would inspire a large number of students, allowing many to learn by directly *engaging* them. Since it began operations, the ISS has accommodated a number of education experiments. *Engagement* is a critical first step for education and while multiple programs have reached a substantial number of students via demonstrations and videoconferences with astronauts, these have traditionally not allowed students to become *engaged* in actual research activities, but represented more or less a one-way flow of information.

Bob Rogers, founder and Chairman of BRC Imagination Arts and winner of the NASA Public Service Medal, when developing NASA's master plan for the exploration of Mars as part of the Mars Exploration Program Analysis Group, presented five strategies for public and student engagement [56]. The presentation, summarized by Mark Craig, makes three important points for effective *engagement*:

1. Effective and massive public engagement has important benefits beyond increased support. It enhances work force retention, morale and recruiting because "It's nice to be a part of something famous". It enhances "spin control" of unplanned events because it establishes a compelling context. The most profound benefit is that it builds a "psychological highway to space". If done well, public engagement builds the exploration and opening of the space frontier into the Nation's DNA.
2. Engagement is best achieved to the broadest audience through the use of a 'story'. As people are engaged by a story, goals in the story need only be important to the protagonists (us). Said in reverse, if people are not engaged by a story, explaining why our goals should be important to them will never be enough.
3. "Story" is an effective mechanism for dealing with potential showstoppers such as loss of interest after major accomplishments (Apollo 12 syndrome). It is also key in sharing the experience of space exploration because it takes people with us emotionally, beyond just visual and tactile experiences.

Important components in making great education possible are international collaboration in development of interest in space, and providing easily accessible information and development of programs that will motivate the next generation workforce. Space Exploration educators across the globe are confronting challenges and embracing opportunities to educate and prepare students for an increasingly interconnected world. Collaboration is in the interest of the US as well. A recent

National Research Council (NRC) Space Studies Board report [57] acknowledges that *"US problems requiring best efforts to understand and resolve are global in nature and must be addressed through mutual worldwide action"*. The report notes that educating *"a capable workforce for the 21st century is a key strategic objective for the US space program"*. It further recommends that the International Space Station (ISS) be utilized fully for education and research, echoing a similar educational recommendation in the Augustine Commission Report [58].

2.4. Collaborative Gaming and Competition

Games have been around as long as human history has been documented. They allow us to build worlds that specifically tap into our evolutionary senses. Stuart Brown [59] observed animal play in the wild, where he first conceived of *play as an evolved behavior* important for the well-being and survival of animals, especially those of higher intelligence. Play, he concluded, has been known to pique human curiosity (exploration play), cause community collaboration (social filling play), charge better performances (adrenaline pumping play) and bring out the creative best in people (imaginative play). Jane McGonigal from 42 Entertainment that produced the record-breaking *'I love Bees'* has researched the reasons for games bringing out the best in people [60]. The positive outcomes of games, she suggests, are blissful productivity, urgent optimism, working in a collaborative environment and toward something agreed upon as an 'epic win'. Furthermore, the common theme among all the gaming blockbusters of today is the fact that they all break into reality [61]: FarmVille lets Facebook users play with their real friends, Guitar Hero lets music lovers play the game while playing music real-time on a real instrument, Nintendo Wii or the Microsoft Kinect use a real console to translate real actions into a video game. The internet, being the best platform for broadcast as well as conversation, has been the critical facilitator of games entering real lives of communities of people worldwide. The introduction of reality in games – picked up and virally spread by alternate reality games - has made the reasons to play them stronger and shown strong correlation between behavior in games to rational, economic behavior in real life [62]. Games are great tools to pique human productivity and reward the brain [63] because they provide easy-to-monitor bars of progress (e.g. An evolving Avatar), multiple short and long term aims, an easy link of consequences to actions, elements of uncertainty to keep the user's interest, windows of enhanced attention as users race for a predefined goal and a crowd of players to play with or against.

Competitions based on the concept of games can organize individuals to work toward a common objective with the incentive of a monetary or non-monetary reward. Individuals with a diversity of skills can participate in the task, with participants picking up and contributing in tasks they are best at. Collaboration allows individuals to work together to achieve larger goals. However, meaningful development through competitions requires a careful balance of competition and collaboration to achieve its goals. One of the important tenets of this thesis is that competition and collaboration atre not mutually exclusive. While big competitions 'challenge' the public with a difficult objective, a series of smaller challenges can be used to engage multiple participants if the challenge structure includes collaboration. Collaboration among the participants allows for the accomplishment of larger tasks by multiple people, and for the performance of each participant to be improved by learning from others. There are a number of ways to bring collaboration into a competitive model, while retaining the benefits of competition. MMORPGs, or Massively multiplayer online role-playing games as introduced before, always have the common feature of social interaction. The games are designed such that some degree of team work is required in order to achieve game objectives. Strategies are decided upon by communication via typed conversation and due to the large online forum available, players often find like-minded players to collaborate with. While some individuals may be outcasts in the real world, they can become whomever they want in these virtual worlds, and can find other players with similar interests and personalities. In one survey, 39.4% of males and 53.3% of females felt that their MMORPG companions were comparable to or even better than their real world friends [64].

Breaking all these virtual, collaborative games into reality, while keeping the excitement and story mentioned above, is the concept of Alternate Reality Games (ARGs) [65] e.g. *I love Bees* from 2004 which had over 600,000 players. ARGSs have an "*interactive narrative that uses the real world as a platform and uses transmedia to deliver a story*" that may be altered by participants' ideas or actions in the virtual or real world. Players interact directly with characters in the game, solve plot-based challenges and puzzles, and collaborate as a community to analyze the story and coordinate real-life and online activities. ARGs generally use multimedia, such as telephones, email and mail but rely on the Internet as the central binding medium. The stereotype of a gamer as a lone and asocial individual has been disproven [66]. On personality tests, gamers have proven to be more extroverted, open, and conscientious than non–game players [67]. Moreover gamers prefer to play with people they already know turning the game into a social experience and may even make, confirm and maintain

friendships and relationships through gaming [68]. In summary, gaming has become a collaborative phenomenon to achieve the required game objectives and is far more than adversarial competition. Such games provide tremendous potential to tap into the several million strong gaming community worldwide to help solve puzzles when judiciously articulated in the language of the game (objectives, incentives, rules etc.).

2.5. Gap Analysis and Research Motivation

This chapter began with a brief introduction to three areas of interest for this thesis and a potential research problem that connects them. Sections 2.1 through 2.4 discussed the available literature that documents the progress made in the individual areas of interest. While crowdsourcing has been discussed as a standalone topic in Section 2.1, crowdsourcing conducted in this thesis is specifically leveraged to solve cluster flight problems. Advances in formation flight for satellite clusters have been discussed in Section 2.2. This section discusses the research that has been conducted in the overlapping areas between the three areas of interest and how the thesis research fits into this context. Figure 4 shows a snapshot of important pieces of literature under each of the areas of interest and their overlaps. Note that the structure of the figure is kept the same as Figure 2 so that the thesis motivation can be tied closely to the gaps in literature.

The ***red ellipse*** in Figure 4 (lower left) refers to some representative crowdsourcing literature. For example, the Mars Crowdsourcing Experiment [69] refers to a game concerning the annotation of semantically rich features of Mars using photographic data of the Martian surface transmitted from the Mars Reconnaissance Orbiter (MRO). The players were scored on the basis of precision and the experiment cleverly utilized an area where human perception exceeds the capabilities of computers. With the goal of achieving a similar objective, Clickworkers' Interactive [70] presented a web-based platform for collecting massive amounts of data from a volunteer workforce tasked with analyzing data captured by the High Resolution Imaging Science Experiment (HiRISE) instrument on the MRO. As described in detail in Section 2.1, NASA runs a large scale Tournament Lab [71] in collaboration with Harvard Business School and TopCoder Inc. to help solve problems using crowdsourcing. Crowdsourcing is therefore meant to be introduced as a methodology by which complicated problems can be solved by crowds, interacting with each other (or not) in whatever way they choose [72]. Prizes are allocated to the best solutions. As described in Section 2.2., with the

growing number of distributed space systems projects and missions, precise, robust and efficient formation flight (FF) algorithms are becoming harder and more necessary to solve. The MIT Space Systems Laboratory operates a microgravity testbed - where FF algorithms can be tested on autonomous nanosatellites aboard the ISS with the help of astronauts – called the SPHERES. SPHERES is therefore a unique opportunity to test developed algorithms much as they would behave in actual outdoor spaceflight.

Potential Problem Statement #1:

To solve cluster flight problems using the crowdsourcing methodology with the advantage of testing the best developed algorithms on the SPHERES testbed

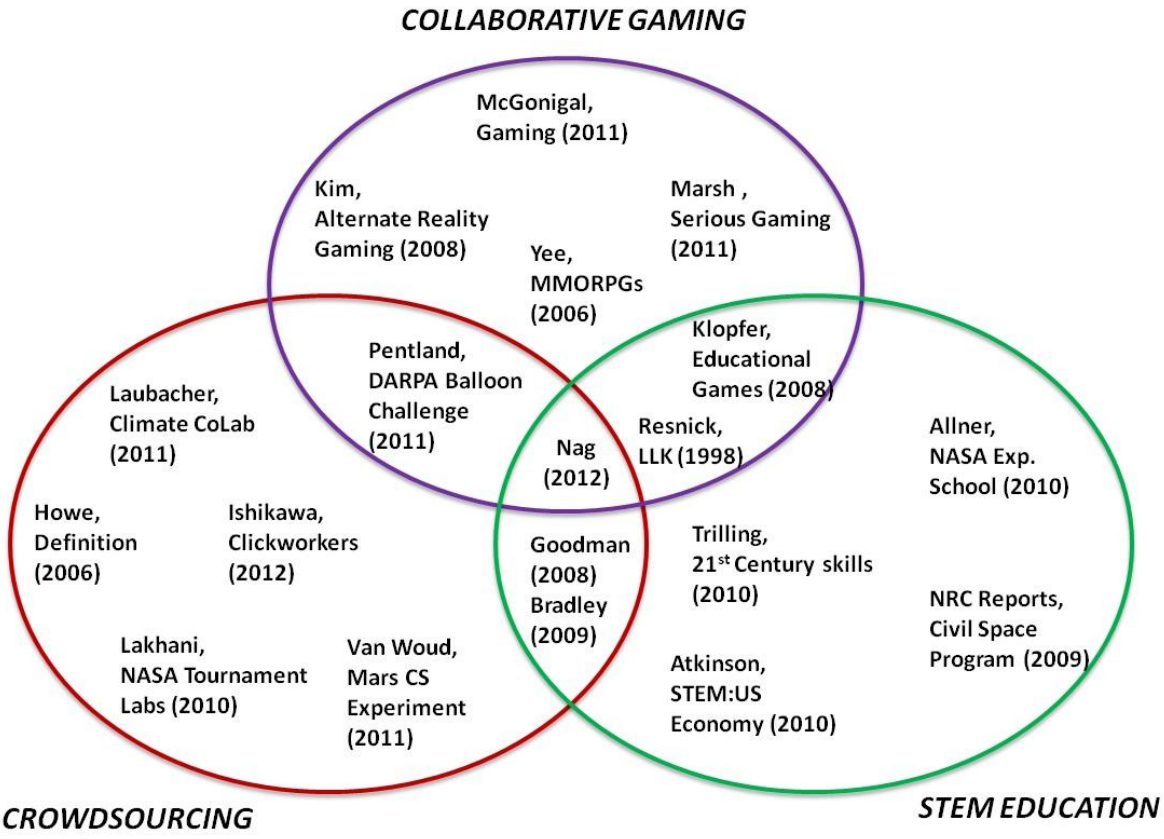


Figure 4: Research Venn Diagram for 'Filling the Gap'. Colors correspond to those in Figure 2.

The ***green ellipse*** in Figure 4 (lower right) refers to some representative STEM Education literature. After-school educational initiatives such as FIRST Robotics [73] and NASA Explorer

Schools [54] are aimed toward increasing student interest in STEM fields, and raising the probability that they will pursue the same in college and graduate school to contribute to these fields. Reports from the National Research Council [57] and the Information Technology and Innovation Foundation [74] strongly encourage the use of government dollars to teach STEM skills, not STEM facts, and create interdisciplinary and entrepreneurial STEM students. "*In the digital world, many of the distinctions between designers and users are becoming blurred. We are all, to some extent, designers now*" [75]. The digital world can therefore be the white canvas for students to learn from and create within, therefore learn by engagement. Furthermore, reports have shown that engagement is motivated by experiences that extend friendships and interests, by self-directed and peer-based learning and learning through "*hanging out, messing around and geeking out*" [76].

Potential Problem Statement #2:

To educate the next-generation workforce in 21st century skills especially in STEM and computer science through hands-on engagement and real-time problem solving with peers

To tap into the engagement aspect of education *and* solve problems at the same time, i.e. the overlap of the ***red and green ellipse*** in Figure 4, there have been initiatives to open up unsolved problems to crowds of students. The iGEM competition [77] challenges participating students to specify, design, build, and test simple biological systems made from standard, interchangeable biological parts for important advances in medicine, energy, and the environment. The competition began within MIT in 2003, went international in 2005 and in 2010 had 130 participating teams from all over the world. The intent is to not only to develop stable and operating biological systems but also to help construct and educate a society that can productively apply biological technology. Similarly, the Spectral Game [78] is a web-based individual player game where players match chemical molecules to interactive spectra such as NMR and mass spectrometry. The game continues until the player gets a spectrum validation wrong; players may report issues with the spectra and/or be allocated a score and shown a leaderboard. Crowdsourced curation efforts have resulted in the deletion or re-association of certain spectra from the database and have allowed the curators to re-reference the spectra. Educationally, the game has been used to teach NMR Spectroscopy to an undergraduate organic chemistry class at Drexel University. Since literature has shown that crowdsourcing and education are possible using the same combined program, the idea of combining crowdsourcing for spaceflight software development and STEM education was explored.

Potential Problem Statement #3:

To solve cluster flight problems using the crowdsourcing methodology AND educate the next-generation workforce in CS- STEM and computer science through the same program

The ***purple ellipse*** in Figure 4 (top) refers to some representative collaborative gaming literature, as discussed in Section 2.4. Phenomena such as globalization, increasing trends to outsource high-level cognitive tasks, and the need to participate effectively in addressing complex world problems are changing how we think, learn, work, and collaborate. A few paradigm shifts from industry age-working to knowledge-age working have been in the nature of the problems (systemic problems framed and solved by transdisciplinary collaboration instead of problems solved by one discipline) and interaction /collaboration (shared professional interests instead of physical proximity) [79]. With millions of gamers collaboratively solving games every day [65], [68], it is hard to overestimate the opportunity of creative labor that the gaming user base has to offer. Collaborative gaming overlaps with *both* crowdsourcing efforts as well as STEM Education efforts.

Crowdsourcers have tapped into the power of collaboration to solve massive and complex problems that could not have been solved by a single person i.e. the overlap of the ***red and purple ellipse*** in Figure 4. For example, the DARPA Red Balloon Challenge [4] was to find and submit to DARPA the coordinates of 10 moored, 8-foot, red weather balloons at 10 previously undisclosed fixed locations in the continental United States for a prize of USD $40,000. A team led by MIT solved this problem within 9 hours by crowdsourcing it to an expanding network of unknown people, who progressively joined the team by being invited via the internet by someone on the team. MIT declared that they would give $2,000 to the first person to send them the coordinates, $1,000 to the person who invited them, $500 to the person who invited them and so on, in reduced geometric progression. Another example is being explored in the world of collaborative gaming - 'The World Without Oil' is an ARG (Alternate Reality Game) tackling a real-life issue of a global oil shock, backed with precise environmental technology information and econometrics [80]. Therefore, revisiting the original question of crowdsourcing spaceflight software on the SPHERES platform, it is thus worthwhile to explore the effects of collaboration on the quality of crowdsourcing.

Potential Problem Statement #4:

To solve cluster flight problems using the crowdsourcing methodology with collaboration among the competitors

STEM Educators have tapped into the power of collaborative learning, which has proven to be far more than the sum of its parts [79] i.e. the overlap of the ***purple and green ellipse*** in Figure 4. Eric Klofper's lab [81] at MIT is focused on making online games and simulations for students to learn through digital media. The Lifelong Kindergarten group [82] at MIT has built revolutionary, collaborative educational tools such as the programmable Lego block, the founding stone to MindStorms, and Scratch, a programming language that makes it easy for anyone to create interactive stories, animations, games, music, and art and share their creations on the web. Scratch has over a million registered users, with the peak age group being 13-17 years, and nearly 2.5 million projects uploaded since January 2007. Data analysis from Scratch has shown that design-based activities, such as creating interactive stories and games, cultivates computational thinking [83]. It promotes a *learning* spiral such that learners imagine what they want to do, create a project based on their ideas, experiment with their creations, share their ideas and creations with others, and reflect on their experiences – all of which leads them to imagine new ideas and new projects [84].

Potential Problem Statement #5:

To educate the next-generation workforce in 21st century skills especially in STEM and computer science through collaboration among the students

As explained in Sections 2.1 through 2.4 and discussions above, most of the individual areas and their overlaps have been covered through previous academic literature or real-world demonstrations. Problem Statement #2 and #5 have been addressed to significant depths since they pertain to CS-STEM and 21st century education alone. Problem Statement #1 and #4 have been addressed for various problems such as synthetic biology, planetary feature detection, chemistry, etc. However, the usage of crowdsourcing, *collaborative or otherwise*, to seek cluster flight algorithms and then test them on a nano-satellite testbed in space is a very new approach, the technicalities of which are being explored in detail as part of a separate research effort beyond the scope of this thesis. The focus of this thesis is on quantifying the combined impact on the quality of formation flight solutions and learning outcomes produced by this approach. Since education is a prime area of interest for this

research, we focus our attention toward Problem Statement #3 and the overlap areas of Problem Statement #4 and #5. Having now provided the motivation, the research objective of this thesis can be summarized below:

Thesis Research Statements

---

1. **Provide a proof of concept that crowdsourcing of cluster flight problems as well as CS-STEM Education is possible and beneficial using the same program**
2. **Analyze the effects of participant collaboration through different mechanisms on both crowdsourcing and CS-STEM Education**
3. **Recommend management policies for Spaceflight Software Development efforts combined with Education efforts**

The proof of concept in Research Statement #1 includes development of a web infrastructure for the program such that widespread participation is possible (Chapter 3), program operation where problems are crowdsourced and students are educated (Chapter 4) and analysis of results of the program operations in terms of performance in simulation, performance on the SPHERES nano-satellite testbed in space, participation satisfaction and other educational metrics (Chapter 5). The effects of collaboration in Research Statement #2 are analyzed by developing metrics for the objectives of crowdsourcing and education and the collaboration variables (Chapter 4) and the analysis of program results in terms of those metrics (Chapter 5). Lessons learned from the crowdsourcing effort, both in developing the program, running and analyzing it, and educational initiatives have influenced the policy recommendations in Research Statement #3 (Chapter 6).

Chapter 3 –
Apparatus Development: SPHERES Zero Robotics Web Infrastructure

Expanding on Research Statement #1, this chapter describes the web infrastructure required for the program, enabling large crowds of people/students to participate in crowdsourcing efforts as well as educational ventures, and the development of this infrastructure. The ZR infrastructure development itself is also a demonstration of crowdsourcing in itself, since it was done through contests in collaboration with a commercial crowdsourcing company called TopCoder Inc. It has therefore been presented as a descriptive and exploratory case study by itself [12] in Section 3.5 onwards. While the mentioned contests to develop ZR's web infrastructure were conducted using TopCoder's methodology and infrastructure, the author played an important role in framing the requirements, communicating with participants, evaluating the results and integrating the final product of each contest. Hence, this chapter is not only a description of apparatus development of the program tools required to achieve the thesis objectives but also an active case study.

SPHERES Zero Robotics is a DARPA-initiated endeavor under the umbrella program called InSPIRE[5] to develop spaceflight software by crowdsourcing. It is a robotics programming competition where students learn to write programs that control a satellite in space using a web browser. The robots are miniature satellites called SPHERES (Synchronized Position Hold Engage Reorient Experimental Satellites) – an experimental testbed developed by the MIT Space Systems Laboratory (SSL) operating on the International Space Station (ISS) to test control and navigation algorithms in microgravity. The Zero Robotics participants compete to win a technically challenging game by programming their strategies into the SPHERES satellites. The game includes command

[5] DARPA eyes crowdsourcing to develop new ideas aiming at the democratization of innovation. The Red Balloon Challenge [4] was just the beginning. Crowdsourcing cluster flight software to expand the capability of microsatellites – SPHERES - operated onboard the ISS is the next step. The Inspire program has four elements: electromagnetic formation flying; vision-based relative navigation; a design study for "Exo-SPHERES" microsatellites that could fly outside the ISS; and a design challenge to involve high school students in the development of algorithms for the SPHERES spacecraft. The ultimate objective is to ensure that a substantial portion of the populace has the requisite information, motivation, and opportunity to participate in the development of spacecraft cluster control algorithms for real on-orbit hardware operating in a zero-gravity environment [85].

and control problems of interest to MIT, DARPA and NASA. Students use either a graphical editor or a C editor to write code, and then simulate their program and see the results in a flash animation. The simulation uses a high-fidelity 3D model of the SPHERES satellites and allows a triage of the most promosing algorithms or approaches to be demonstrated on orbit. Since astronaut time, battery power and CO_2 tank capacity is limited on orbit, only the most promising solutions are tested in microgravity. Astronauts assist in running the final competition on the ISS (where in the SPHERES satellites perform maneuvers as programmed by the students) and interact with the students via a live video broadcast.

3.1. SPHERES

The SPHERES program began in 1999 as part of an MIT Aero/Astro undergraduate class. Prototypes were built by the student class in 2000, flight satellites were delivered in 2003, and launched to the ISS occurred in 2006 [13]. SPHERES became one of the first educational programs that launched student-designed hardware to the ISS. SPHERES consists of a set of tools and hardware developed for use aboard the ISS and in ground-based tests: three nanosatellites, a custom metrology system (based on infrared and ultrasound time-of-flight measurements), communications hardware, consumables (tanks and batteries), and an astronaut interface. They operate aboard the ISS under the supervision of a crew member (Figure 5).

The ground-based setup consists of a set of hardware analogous to what is in the Station: three nanosatellites, a metrology system with the same geometry as that on the ISS, a research oriented GUI, and replenishable consumables. Due to gravity the ground-based testbed is implementated on a flat floor, allowing to exercise three out of six degrees of freedom. The SPHERES satellites implement all the features of a standard thruster-based satellite bus. The satellites have fully functional propulsion, guidance, communications, and power sub-systems. These enable the satellites to maneuver in six degrees of freedom (6-DOF), communicate with each other and with the laptop control station, and identify their position with respect to each other and to the reference frame. The laptop control station (an ISS supplied standard laptop) is used to collect and store data and to upload new algorithms. SPHERES uploads new algorithms (ahead of time) and downloads data (after the session) using the ISS communications system. Figure 6 shows a picture of a SPHERES satellite and identifies its main components. Physical properties of the satellites are listed

in Table 1. There are two communication channels for data transmission: the SPHERES-to-Laptop (STL) channel to transmit data and telemetry to the laptop station and the SPHERES-to-SPHERES (STS) channel used for inter-satellite communication, enabling cooperative and coordinated maneuvering between satellites during tests. Each channel is on an independent radio frequency, either 868 MHz or 916 MHz. The communication bandwidth is limited to a total of 70 packets per second, where each packet is 32 bytes. This must be shared among all of the satellites in operation. The communication delay between sending and receiving is usually on the order of a few milliseconds, but can be up to 200 ms at worst case due to the Time Division Multiple Access (TDMA) protocol. The maximum frequency of data transmission within the SPHERES communication system is 5 Hz. The amount and frequency of data transmission possible with the SPHERES hardware was a limiting constraint in the development of algorithms for the SPHERES and for their usage as robots in a game.

Figure 5: Astronaut and MIT alum Gregory Chamitoff operates 3 SPHERES aboard the ISS

Diameter	0.22 m
Mass (w/tank & batteries)	4.3 kg
Max linear acceleration	0.17 m/s^2
Max angular acceleration	3.5 rad/s^2
Power consumption	13 W
Battery lifetime (replaceable)	2 hours

Table 1: SPHERES Physical Properties

The SPHERES Position and Attitude Determination System (PADS) consists of inertial sensors and ultrasound beacons and receivers. Inertial sensors include three single-axis gyroscopes and three single-axis accelerometers, providing three-axis inertial measurements. The ultrasound system consists of 24 ultrasound receivers and one beacon on each satellite. There are five external wall-mountable beacons. Estimation is based on sequenced time-of-flight measurements from the beacons to the receivers to determine range. A state estimator is then used to provide real-time position, velocity, attitude, and angular rate information for each SPHERES satellite at a rate of up to 5 Hz.

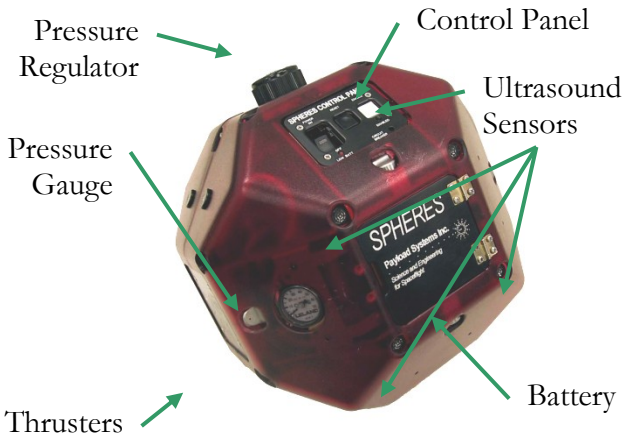


Figure 6: A SPHERES Satellite

SPHERES was designed to be a permanent *facility* aboard the ISS, not just a single experiment, by following a set of design principles learned from previous MIT SSL experience [13]. To provide the ability to involve multiple scientists in a simple manner, a SPHERES Guest Scientist Program was created [86]. This program consists of a test development framework, a robust and flexible interface to the SPHERES flight software, a portable high-fidelity simulation, two laboratory test beds and data analysis utilities, and supports the efforts of geographically distributed researchers in the development of algorithms. SPHERES software consists of an embedded system (SPHERESCore) and additional user-selectable library function. SPHERESCore is responsible for handling interrupts and interfacing with the hardware [87]. The library functions such as math utilities, etc., provide guest scientists with the ability to use pre-defined utility functions to expedite programming and testing. The coding language used on the hardware is C, while code for the simulation is in

MATLAB. The Zero-Robotics program expands the Guest Scientist Program with a simplified interface and a high-fidelity back-end online simulation so that students at many different grade and skill levels can program the satellites.

3.2. History of Zero Robotics and Modification of the Program

The Zero Robotics (ZR) competitions draw significant inspiration from FIRST Robotics [73] and share common goals, including building lifelong skills and interest in science, technology, engineering, and math through project-based learning. FIRST Robotics concentrates heavily on the development of hardware, has a registration fee and does not have any space-related components. Since SPHERES concentrates on the development of software, Zero-Robotics complements FIRST Robotics by providing students an avenue to further develop their software skills, with the incentive that the software they develop will be tested by robots and astronauts in space at no cost to participants.

In fall 2009, the SSL conducted a pilot program of the Zero Robotics competition with two schools/10 students from northern Idaho [88]. In 2010, Zero Robotics was a component of NASA's Summer of Innovation, a nationwide program targeted at encouraging STEM education for middle school students. During this competition, 10 teams and over 150 students from schools in the Boston area worked for five weeks to program the SPHERES to compete in an obstacle course race. In the fall of 2010, Zero Robotics conducted a nationwide pilot tournament for high school students named the Zero Robotics SPHERES Challenge 2010. Over 200 students from 19 US states participated as part of 24 teams. The objective of the game was to complete the assembly of a fictitous solar power station by maneuvering a satellite to dock with a floating solar panel and then bring it back to the station to finish the mission before the opponent does.

The 2010 tournament was designed purely for STEM education and outreach. The two SPHERES satellites in each match, controlled by opposing participants, engaged in direct head-to-head competition. Participating teams in 2010 competed as individual teams throughout the entire tournament, and there were no extensive community forums where they could exchange knowledge or converse with each other. Thus, the 2010 tournament emphasized “pure” competition. An

external forum plug-in was provided on the website, but due to the inherent competitive nature of the game, it was not very widely used. To address the thesis research question, the 2011 tournament was designed to achieve both crowdsourcing objectives i.e. solve a hard cluster flight problem, as well as STEM objectives, i.e. educate students and outreach. In 2011, three different types of collaboration mechanisms were introduced. First, the game was designed such that teams that programmed their SPHERES would be encouraged to collaborate during the match to achieve game objectives (i.e. crowdsourcer objectives) and would gain more points than those that did not. Since collaboration was meant to be rewarded more than winning, the competition structure was that of a round robin where the team with the maximum cumulative points won the competition, not the one with the maximum number of wins. Second, halfway through the tournament, there was a mandatory requirement that selected teams had to form alliances of 3 teams each and submit integrated projects per alliance for all competitions after that. Third, the 2011 tournament had extensive community forums where teams could exchange ideas, educate each other, challenge each other to informal games and share projects to work on collaboratively. The ZR program was therefore modified in 2011 to be steered in a direction that would help evaluate its impact on both crowdsourcing and CS-STEM Education and furthermore, assess the impact of collaborative competition on both these objectives. One of the key sources of insight in this thesis is the side-by-side comparison of the 2010 and 2011 tournaments.

2010	2011
Purely for STEM Education	For Crowdsourcing and STEM
In Game Competition => Elimination	In Game Collaboration => Round Robin
Individual Teams	Individual Teams + Alliances
No Community Forum	Extensive Community Forums

Table 2: Comparison of Zero Robotics competitions in 2010 and 2011 to highlight the introduction of collaborative competition mechanisms

3.3. A System Representation of Zero Robotics

To allow crowds of students to use the SPHERES high-fidelity simulator, write spaceflight-capable programs and interact/collaborate with each other, an online environment was required. Spaceflight software development through Zero Robotics, therefore, occurs for existing spaceflight hardware in

two stages, as shown in Figure 7: (1) Building the web infrastructure for the programming competitions – circled in red - by leveraging a crowd of thousands of software developers, and (2) the programming competitions themselves – within the blue box - when thousands of students contribute to writing SPHERES software which is subsequently tested. Both stages are demonstrations of crowdsourcing using different classes of participants and with different objectives.

The goal of the Zero Robotics tournaments is to develop cluster flight algorithms (specifically for SPHERES but that can be generalized to small satellites) at the same time as promoting STEM Education. As depicted in Figure 7, the students who participate in the tournaments are the input into the Zero Robotics 'system' and the output are the mentioned research objectives, STEM education and satellite software or algorithms. The 'blue box' of crowdsourcing therefore has a dual impact of algorithm development *and* education. The 'system' includes a game which is available through the ZR Web Infrastructure, which in turn is comprised of a website, tutorials, online community forums, team management tools, tournament management and participation tools and programming environment where students can create, save, edit, share, simulate and practice as well as submit computer code for competitions. Thousands of developers competed in TopCoder crowdsourcing contests to creatively design the web infrastructure for these students (crowd creation) and assemble the software components, developed sequentially and in parallel, to build a robust web framework to allow for writing and testing satellite control programs online (crowd production). The feedback of the students, as they participate in the tournaments, serves to improve the web infrastructure - 'red circle' in Figure 7. This chapter discusses the *process of developing the ZR Web Platform* with the intent of making MIT's SPHERES simulator accessible and providing a persistent community platform for crowds to interact and write spaceflight-capable software for SPHERES. The TopCoder methodology has been discussed in detail and data from the crowdsourcing contests has been analyzed, to highlight the role of competitions in enabling space research amateurs, from non-technical personnel to software developers, to create space mission software of value to the space community. It serves as a case study for commercial crowdsourcing operations.

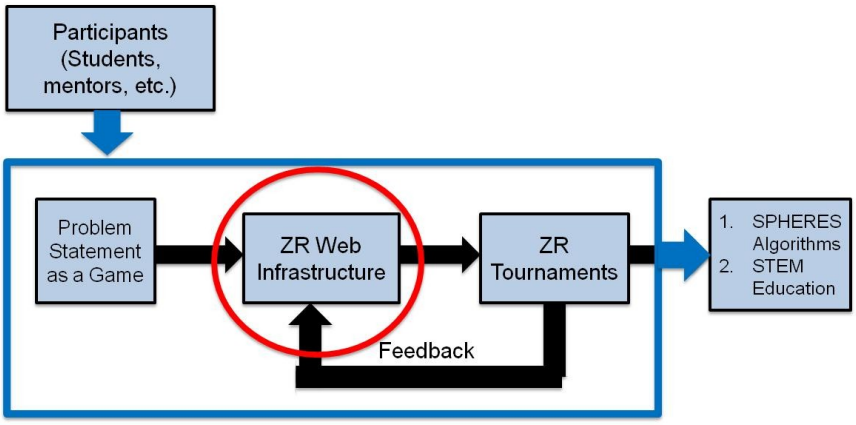


Figure 7: Zero Robotics System Diagram

TopCoder is a commercial company that uses a mix of competition and collaboration within their online community of over 300,000 developers, who voluntarily register on TopCoder's website, to make scalable, cloud-based software systems. Thousands of developers competed in TopCoder contests for prize money. Section 3.4 briefly describes the components of the developed infrastructure and the following section 3.5 discusses the process of designing and running the TopCoder crowdsourcing contests. Finally, Section 3.6 presents the analysis of the results of the contests and lists the lessons learned through the development effort. The ZR Web Infrastructure may be considered a capital non-recurring investment. Once it is ready, the recurring efforts or investments comprise only of designing and uploading specific game software (such that the games can be played using the existing infrastructure) and rules supporting the Zero Robotics tournaments.

3.4. Zero Robotics Web Infrastructure

The following section will briefly list the components of the ZR Web Interface – the red circled section of Figure 7. The web interface[6] comprised of a programming interface and several other tools such that participants of the ZR Tournaments could program the SPHERES satellites, submit their programs for competitions and for teams to interact with each other (to achieve the collaboration and competition objective). The Web Interface also allowed the organizers of the

[6] The term 'Web Interface' has been used to refer to the Frond End – the part that the users interact with - while the term 'Web Infrastructure' has been used to refer to the entire software infrastructure, built by TopCoder, MIT and AFS, as shown in Figure 8 along with the website, community forums, tutorials, team and project management as well as the tournament management tools

program, the administrators, to conduct tournaments and competitions with thousands of users and hundreds and thousands of simulations entirely online.

3.4.1. Programming Interface

Typically, programming the SPHERES satellites requires users to have access to the Texas Instrument compilers for the SPHERES processor and familiarity with the Guest Scientist Program. None of this is possible for a tournament meant for high school and middle school students. Instead, MIT and TopCoder have developed a web-based interface to program the satellites which makes use of the same SPHERES high-fidelity simulation that is used to develop flight software.

Users can program the SPHERES using a web-based GUI, which provides a simplified interface to the Guest Scientist API functions and enforces constraints that guarantee compatibility with the SPHERES compilers. Students have access to a text-based editor as well as a graphical editor, for those with little or no prior programming experience. A distributed computation engine, hosted on Amazon EC2 virtual machines, compiles the user code with the core SPHERES software, and performs a full simulation of the program. An Adobe Flash-based front-end visualization creates an animated representation of the results. The code programmed by the students via the web interface can be executed in the hardware. The flow of information in the ZR software infrastructure is shown in Figure 8. The user code is transmitted to the web application, which launches a simulation instance on the 'Farm', which on completion returns the results to the web app and finally the browser, then rendered in the form of an animation as shown in Figure 9. The ZR 'Farm' is the back-end engine, developed and troubleshot by a TopCoder member, to handle and implement compilation and simulation requests from the web app. The ZR projects are compiled/simulated in conjunction with the SPHERES embedded system (SPHERESCore) code and the ZR game code.

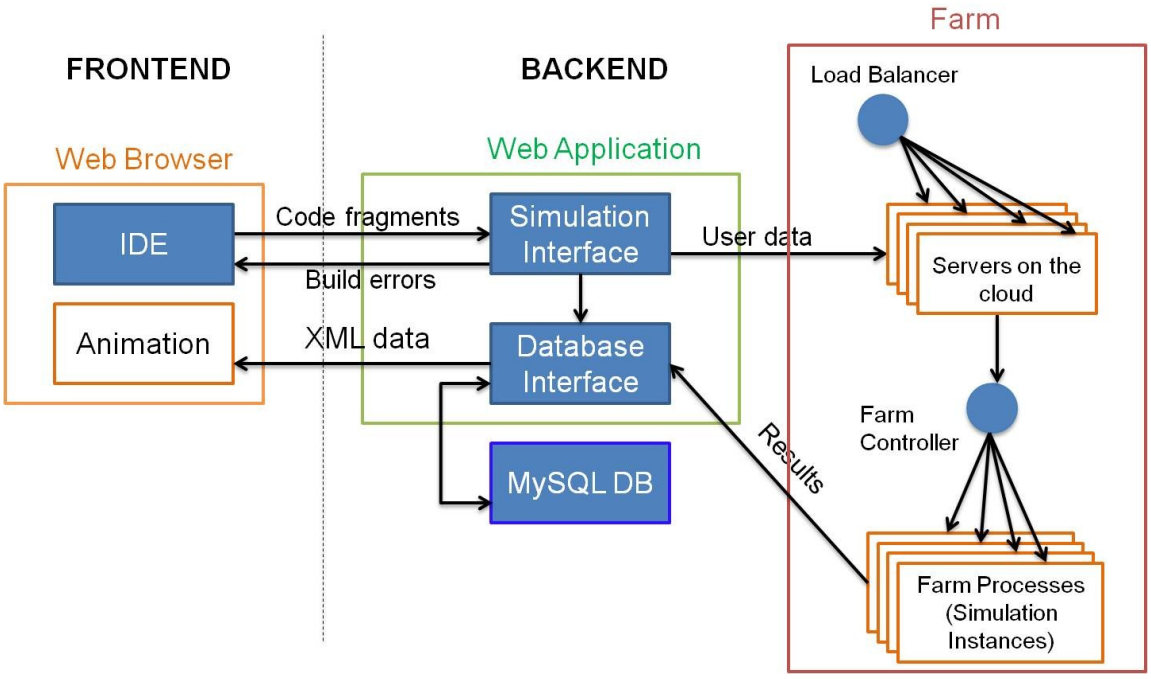


Figure 8: ZR Software Architecture

Users write their programs to control SPHERES inside the main function called *'ZRUser()'* available as a template in each project (see Figure 11, within the IDE). Users are not allowed to change its signature. *ZRUser()* is called at every iteration of the satellite control cycle (once per second). Users may also declare and define additional procedures, which are all called inside this main loop. The inputs to *ZRUser()*, available to be used by the users, are the SPHERES state (position, velocity, attitude and attitude rates) and the time since the game began. These inputs are obtained from the 'game code', which in turn gets it from the SPHERES embedded system code (explained later in Section 4.1.1, Figure 26). For running simulations, the code within *ZRUser()* is inserted into a pre-defined template and simulated by the SPHERES Simulator along with 'game code' and embedded system code. Fresh high school students take less than 3 weeks to learn how to use the IDE and write a fully capable program to play a ZR game.

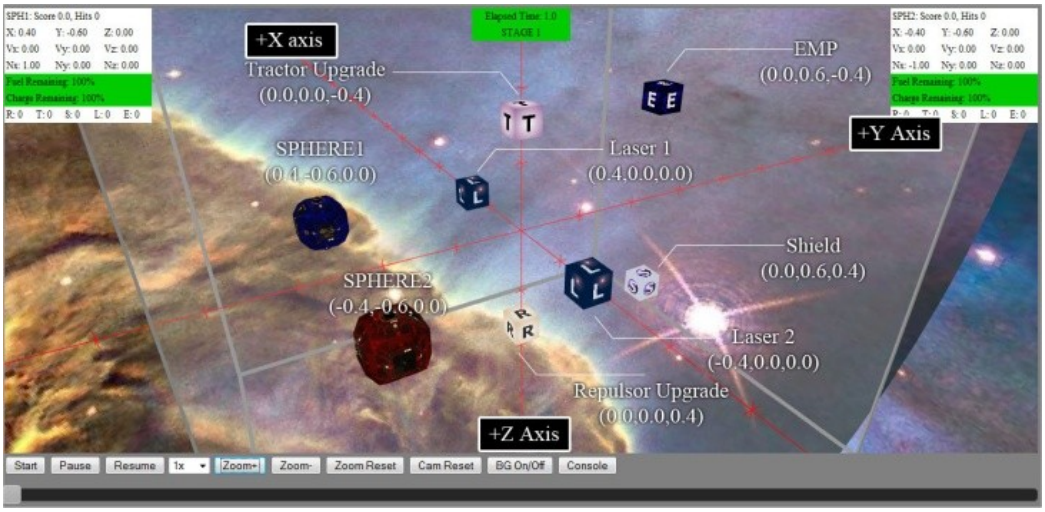


Figure 9: Example of a ZR Animation

Graphical Editor: The ZR graphical editor, as shown in Figure 10, allows users with little or no C experience to write code using drag-and-drop programming. It is currently possible to see and generate C-code from the diagram view so that users can initiate their code with diagrams but can move on to more complicated code using the C editor. The graphical editor uses standard procedural language constructs such as if/then/else calls, variable assignments, array iterators, range iterators, case-statements, etc. The Zero Robotics API procedures and functions as well as game specific API functions are integrated into the drag-drop programming icons. Furthermore, user-defined procedures/functions and variables are supported. The graphical editor is written in JavaScript and is derived from the Waterbear JavaScript editor (http://waterbearlang.com). The implementation uses a Model-View-Controller paradigm where the block diagram and "C" views are different renderings of the same underlying model. From past ZR experience (summer program of 2010), middle school students have typically taken less than 10 days to learn to use the graphical editor and submit a program.

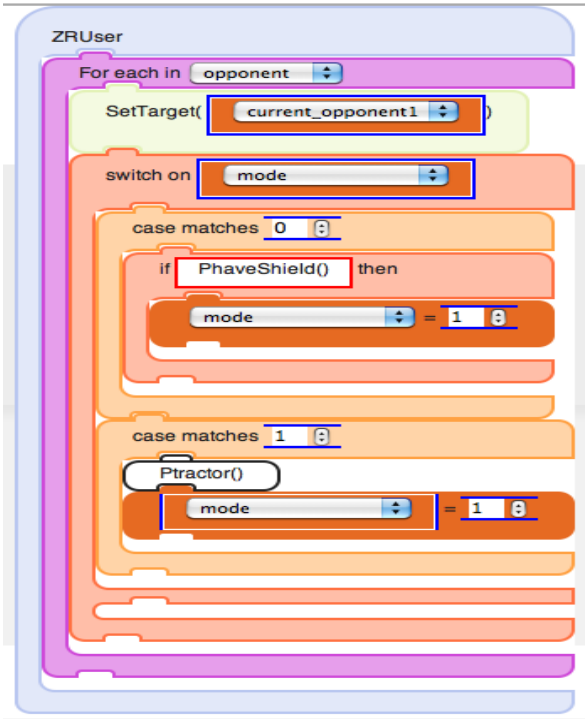


Figure 10: Example of code in the Graphical Editor

3.4.2. Team and Project Management Tools

Teams are organized into two types of members: team leads and team members. Users are required to create an account on the ZR website before submitting an application to a tournament. A tournament application for high schools typically comprises of entering school, potential team and mentor information. We also sought a commitment that the students have internet access and have found at least professional individual, affiliated with the school and capable of teaching CS-STEM, who will serve as their 'team mentor'. On acceptance, the user who submitted the application is designated as a team lead of a newly created online team (unique ZR ID assigned) or a previously formed one. A team lead can then invite other users to join the team and assign more team leads. Figure 11 shows the main programming editor, Figure 12 the project management tool and Figure 13 the simulation management tool, where the user may replay his past simulations and animations. The project management tool also allows users to navigate and edit projects that have been shared within his team. All users who share a common project have access to the 'project instant messaging' tool so they can chat with each other online while editing their shared projects. No chat logs were saved to protect user privacy.

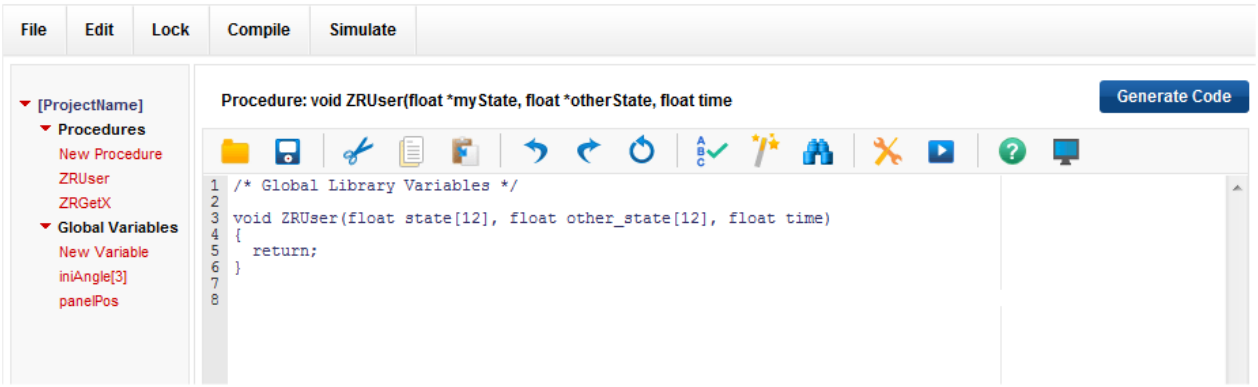


Figure 11: IDE Text Editor for programming projects to control the SPHERES. The procedure on screen is *ZRUser()* , the main function where all ZR API functions and other procedures are called

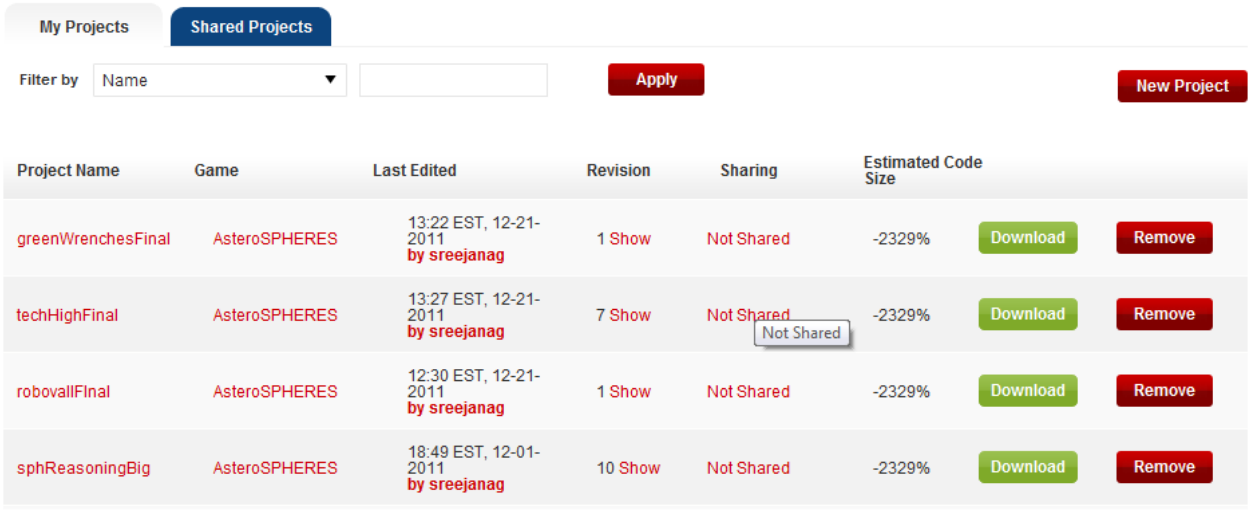


Figure 12: User Project Management tool

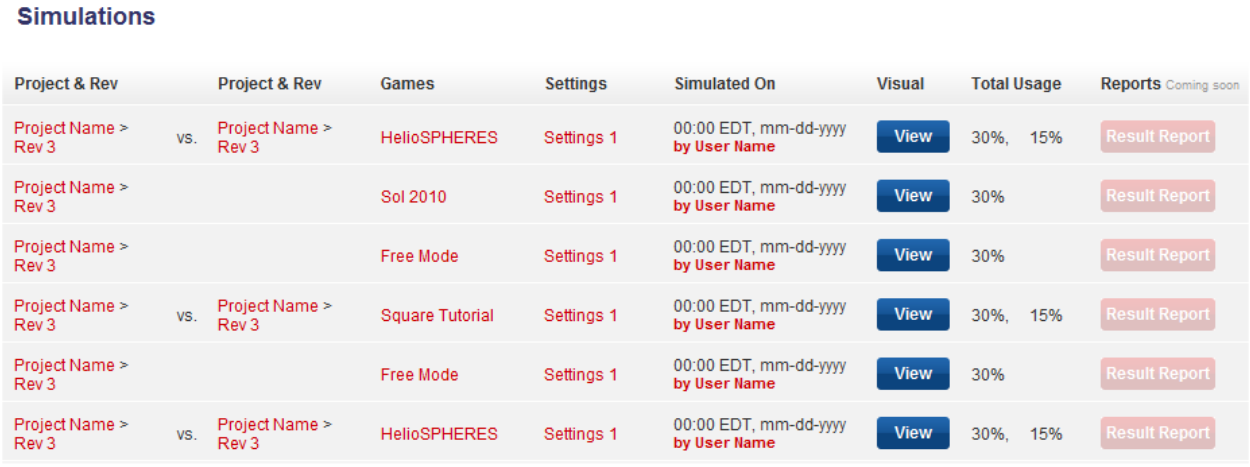


Figure 13: User Simulation Management tool

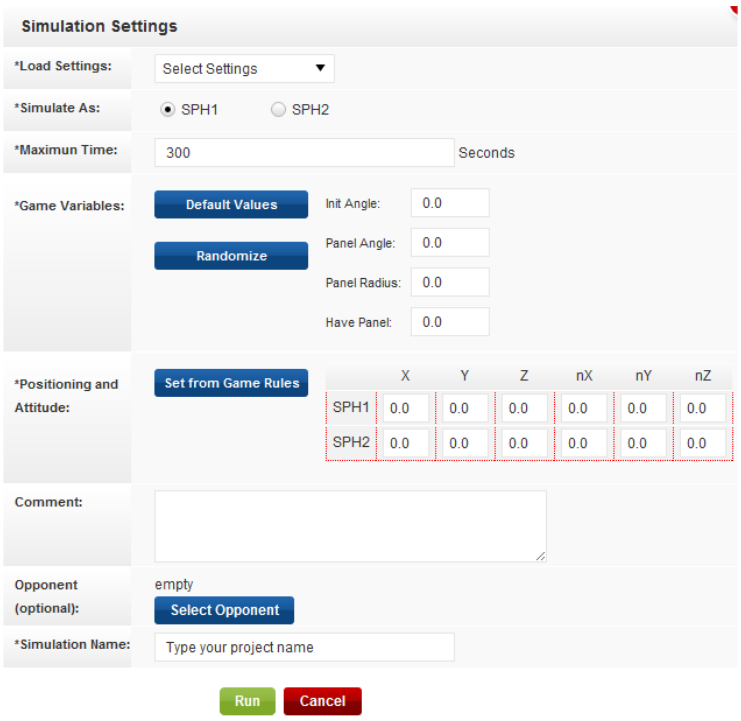


Figure 14: Simulation Settings Window to tweak game variables when practice programming

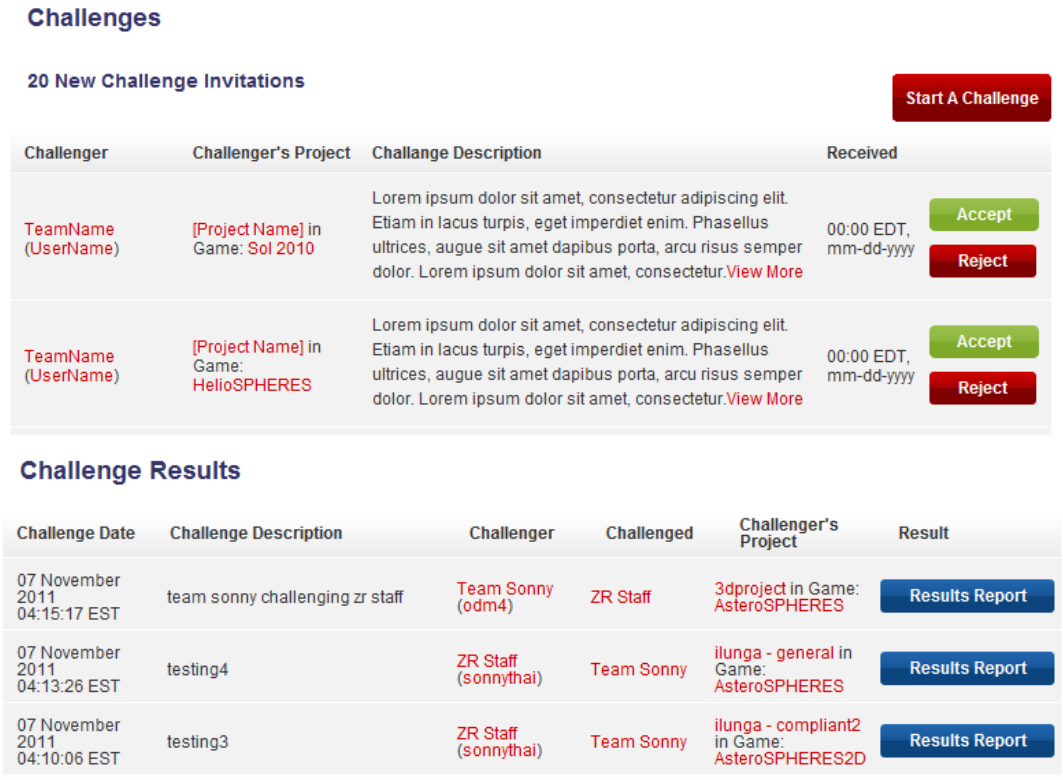


Figure 15: Tournament Challenges

Submissions

Competition Names	Project Name	Submitted By	Date of Submission	Submission Deadline
2D Simulation Competition	Project Name	User Name	00:00 EDT, mm-dd-yyyy	00:00 EDT, mm-dd-yyyy
3D Simulation Competition#1	Project Name	User Name	00:00 EDT, mm-dd-yyyy	00:00 EDT, mm-dd-yyyy
3D Simulation Competition#2	Project Name	User Name	00:00 EDT, mm-dd-yyyy	00:00 EDT, mm-dd-yyyy
ISS Competition	Project Name	User Name	00:00 EDT, mm-dd-yyyy	00:00 EDT, mm-dd-yyyy
ISS Competition#1	Submit Project			00:00 EDT, mm-dd-yyyy

Figure 16: Submissions for Formal Competitions

The ZR simulation allows users to tweak different game parameters and choose simulation settings – UI panel seen in Figure 14 - so that they can test different parts of their code independently. They can simulate an individual project, race against another member of their team or race against standard players (pre-coded projects to simulate against) provided by MIT. The simulation also allows students to control the speed of the game to show the motion in real time, or up to 10 times faster. In a formal competition, these settings are fixed by MIT, and the purpose of the simulation is to provide ample opportunities to test different versions of their strategies and finalize a robust submission. Users may simulate individual projects on the IDE itself, and therefore iterate to improve their projects.

All through the tournaments, teams are given the opportunity to challenge other teams for informal scrimmages. The website provides the ability to select a user project and invite other teams to race their projects against the selected one – called a 'challenge'. Teams can accept or reject challenges using the provided UI and view the results, animations and leader boards for each challenge that they participated in (Figure 15). The web infrastructure provides for the running of an automated simulation when the challenge has been accepted by a team, and makes the results available on the website. A simple interface is available to teams for submitting a project as an entry into a formal competition (Figure 16) – any team lead may select an existing team project and submit it for a competition.

3.4.3. Tournament Management Tools

Administrators need tournament management tools to manage competitions and tournaments on the web interface. They can create an application form for an upcoming tournament, be notified when a user submits and application, review and accept an application – after which the user is automatically emailed a URL which they can use to create their new team or use an existing team for the new tournament. Each tournament may have multiple competitions; each competition can be associated with only one game (See the introduction of Chapter 4 for details). They may upload game code for any number of games on the web interface, to be associated with specific competitions later or to be simply available on the IDE for users to practice programming. Administrators can create competitions for any tournament, edit its description, set a game to play and set deadlines for the competitions. Users who do not submit their projects by the deadline are disqualified from the competition. To simulate multiple projects, there is a batch simulation tool available to administrators. This tool is very useful for running simulation competitions by simply selecting the user projects to be simulated, the game they intend to play and the game specific parameters. The tool automatically runs thousands of simulations and outputs the results, which the administrator can then make available on the website for users to review and learn from. Administrators may also moderate community forums, make announcements or any changes to the website.

3.5. Crowdsourcing Methodology for Web Interface Development[7]

To develop the ZR web infrastructure, TopCoder and MIT conducted contests among members of TopCoder's worldwide technologist community to create software and technology solutions. The methodology is described below in the form of a case study, followed by results and conclusions in Section 3.6 and 3.7. Since TopCoder is in the commercial business of developing software for NASA, e.g. in the NASA Tournament Labs [71] presented in Section 2.1, the application of the methodology to ZR's development can be seen as a representative or typical case study, both

[7] Adapted from a peer-reviewed and published conference paper, available on IEEE Xplore and presented here as a case study: S. Nag, I. Heffan, A. Saenz-Otero, M. Lydon, "*SPHERES Zero Robotics software development: Lessons on crowdsourcing and collaborative competition*", IEEE Xplore 10.1109/AERO.2012.6187452, ISBN: 978-1-4577-0556-4, March 2012. [89]

descriptive and explanatory in nature [12]. Problems are posed in an "open call" for solution submissions of a specific type, size, and approximate complexity, and submissions are judged to determine the winner, typically with monetary prizes awarded for the best solutions. For each of these contests, a specification for the desired deliverables is published along with the price to be paid for the "best" solution that meets minimum criteria. In response developers submit the actual deliverables. Contestants can compete to develop the best algorithm to solve a particular problem, to develop a user interface design, the code for a software component, or to conceive of the best approach to a business or operational problem or opportunity using technology. Solution submissions can range from documents containing ideas, workflow, schematics to graphic design assets such as user interface designs, wireframes and story boards to files containing software code, test data and technical documentation. For many solutions, standard competition types and deliverables formats reduce the learning curve for participants.

3.5.1. Evaluation Criteria

Judging methods depend on the type of competition. For most types of deliverables that can be reviewed objectively, submissions are peer-reviewed by historically top-performing reviewers from within the community with a rigorous scorecard, and the winner is selected based on those scores. However, not all deliverables can be judged objectively. Some other examples are:

- Sponsor of the challenges selects the submission they believe to be most valuable and most closely meets the criteria set forth in the challenge.
- Client and reviewers select the winner based on their preferred submission (subjective); e.g. business requirements contests.
- Automated testing and scoring are used to evaluate; e.g. algorithm development contests can be judged based on the performance and/or accuracy of the algorithm using a specified test data and scoring method focused on the desired results.

In each of these scenarios, the evaluation method needs to be clear and objective, and the results transparent for all participants.

3.5.2. Incentive Structure

The TopCoder web site is designed to identify, promote, and reward the best participants in each category of competition. Cash prizes are awarded to winners and runners-up, and competitor results are posted on the site for public recognition of outstanding performance. A member's username is displayed on the site in a color that reflects their rating, so that their rating becomes a part of their online identity [90]. Detailed, publicly-available statistics are kept on the web site so that all participants can see how they compare to others. Such statistics include biography, TopCoder contest statistics, reliability rating, performance and scores from all categories of contests participated in. This allows each member to judge the level of competition in a potential contest and determine the amount of effort he will put in accordingly. For each contest type, there are both short-term prizes and long-term incentives. Competitions typically include prizes for 1st place and at least one runner-up. Some contests also include milestone prizes that are paid based on mid-competition deliverables. In addition, there may be incentives for submission reliability over time and for continued participation, like the "Digital Run" prize pool. These are all in addition to opportunities for additional participation as a reviewer or co-pilot based on historic competition success.

Incentive structures for crowdsourcing challenges in the form of prizes can achieve societal influence in seven different ways [91]: Identifying excellence, Influencing public perception, Focusing communities on specific problems, Mobilizing new talent, Strengthening problem-solving communities, Educating individuals and Mobilizing capital

3.5.3. Benefits of Competition in Development

The competition-based development model is successful for a number of reasons. Some of them are that:

- The development conducted through competitions does not depend on the knowledge or availability of any particular individual as a single point of failure.
- There are innovation benefits that come from reaching out to a global pool of solvers who have a diversity of skills and experience, and bring their creativity to a particular task at hand.

- The contest judging process inherently includes a detailed review process for assuring the quality of work.
- Individuals self-select the tasks on which they choose to perform, and for which they are motivated and believe that they have the ability to be successful.
- Winning submitters are paid a fixed price for the deliverables, and are paid only if their deliverables meet minimum criteria and are delivered by the deadline.

TopCoder's platform has hundreds of new registrants each week and thousands of active participants. The platform is therefore likely to have individuals with the necessary skills and willingness to participate in a given technology-related task. Of course, these significant benefits come with some requirements. Problems must be presented in a format that is suitable for competition. TopCoder has had to develop expertise in developing the formulation of problems and presenting them to the community so that they can be solved in a systematic manner. Also, development environments and test data must be provided in a way that is accessible to the community. A subset of these resources was used for ZR development.

3.5.4. Benefits of Collaborative Competition in Development

The collaboratively competitive development of Zero Robotics' platform, as per the TopCoder methodology, is based on competition, in that there are competitions for each design and development task. These competitions offer both monetary and non-monetary incentives for the participants. Participation in competitions is entirely voluntary and allows the participants complete flexibility and control over their choice of projects. While each challenge is inherently competitive, the overall effort also includes a significant amount of collaboration, both structured and unstructured.

I. ***Structured Collaboration***

Much of the collaboration in TopCoder is structured collaboration, i.e. the TopCoder process dictates how that collaboration takes place. Portions or all of the deliverables created in one competition (e.g., software architecture designs) are used as specifications for another competition. The deliverables are created in a predetermined format to make the communication of information as seamless as possible. In addition, the architects and reviewers in a competition work with the developers during the competition to answer

questions and to finalize the deliverables. A "final fix" stage of the competition requires a developer to make changes in response to minor errors or omissions identified by the reviewers. This is similar to code reviews conducted by many development organizations, but takes place at each stage of the software creation lifecycle, not just coding.

II. ***Unstructured Collaboration***

With respect to unstructured collaboration, discussion forums enable participants to ask questions and discuss the requirements with the architects, clients and each other. This discussion often adds additional detail or resolves ambiguity in the contest specification. It also provides a record of the reasoning for the design and implementation decisions that are discussed. Even while members compete against one another, their interests in algorithms and software bring them to common ground, and members are typically willing to help each other as well as teach and advise beginners. The general discussion forums are home to a very active level of interaction about topics of interest to this community.

The structured collaboration in the TopCoder model is important because it enables individuals with varied skill sets to address different parts of the problem to be solved and enables distributed development. In other words, it allows a "team" to form in order to solve a complex problem without requiring the team members to establish relationships with each other. It allows team members to pick their contribution based on their interests and skills. Additionally, the structure of the collaboration process makes each team member's contribution and interaction transparent to the other participants. The documentation developed at each stage is critical because the members of the team can keep changing, so the combined knowledge exists not in the experience of the individuals alone but in the documentation and process.

On the other hand, this collaboration structure does add overhead. Since communication is limited to the written documentation and the forums, interface definitions and documentation are required at every stage. Collaboration with other individuals requires at least some written specification of the task, and evaluation of results. At times, particularly when a small, fast change is needed, this overhead seems to take longer than it would if one could just call up the developer on a team and request the change. However, there is not just one developer who can make the change, and so the availability of 'the' developer on the team is not determinative of whether the change can be made.

3.5.5. Development of Complex Software through Crowdsourcing Contests

Crowdsourcing is not just for using a single contest to solve a single problem. Large problems can also be broken down into smaller sub-problems in a manner that each can be solved by a contest. For example, a computational problem might require an algorithm competition to obtain an algorithm that would solve a problem, and a software component design competition and a software component development competition after that to implement the result of the algorithm competition. On the TopCoder platform, development projects typically are planned out in "Game Plan" schedules that show the series of competitions scheduled and estimated costs for delivering them. The game plans do not have particular individuals associated with each task. Rather, the competitors decide whether to participate in the contest for each set of deliverables. Predictions about the likelihood of successful completion during the competition lifespan can be made based on past history and the competition parameters (e.g., competition type, pricing, timing, etc).

For a large, complex project such as the Zero Robotics competition and development environment, we divided the project into several modules and used the traditional Software Development Life Cycle (SDLC) for each module. Each phase of the SDLC loop is a crowdsourcing contest and its outputs are fed into the next phase as input to the next crowdsourcing contest (Figure 17). Parallel development is therefore possible and interface requirements are very strict to prevent misfits later. Definition of interface requirements and integration of developed modules to make a holistic software is done by the TC project manager (the only managerially hired position in the project) and the program's co-pilot (selected by the project manager through a crowdsourcing contest). Both the project manager and the co-pilot are preferably kept the same for the length of a project.

The top-level phases of the lifecycle (Figure 17) are:

1. Conceptualization and Specification
2. Architecture
3. Component Production
4. Application Assembly
5. Certification
6. Deployment

A large project is broken into multiple modules that need to be developed; each module is developed through the above phases and each phase has one or more contests. Conceptualization competitions develop Business Requirements documents and High-Level Use cases as solutions. These are then provided as inputs to Specification competitions, which develop Application Requirements Documents, Use Cases, Activity Diagrams, and Storyboard and/or Prototypes. These design specification deliverables are then used in Architecture competitions to develop Module and System Design Specifications, Sequence Diagrams, Interface Diagrams, and Component Design Specifications. Test cases also may be developed at this time, by conducting testing competitions. The Component Design Specifications are used in competitions to design and develop reusable software components that implement the design. In Application Assembly, the components are assembled and the deployment requirements documented. In Certification, the assembled software is thoroughly tested through testing competitions and the application is deployed on a staging server for a final integrated set of tests. After the completion of all the phases, the solution is ready for deployment.

Figure 17 does not show all of the competitions currently offered by TopCoder. Neither is this the only way that crowdsourcing can be used to develop large, complex systems. Other contests that have not been shown include algorithmic problem solving, graphic design, user interface design, idea generation, wireframes, prototyping, etc. that might be employed in the development of a technology solution. Zero Robotics development included many such contests.

Conducting a competition is much more involved than simply posting the challenge to a web site. Important elements of the collaborative competitive infrastructure provided by the TopCoder competition platform used to develop the ZR web interface are:

- A web interface to make competitions structured, organized, compelling and interesting. TopCoder performs these functions using its website: www.topcoder.com
- A web interface that allows easy problem disambiguation, formulation, communication, validation, recognition and rewards.
- Behind-the-scenes infrastructure for handling competition participants' paperwork and inquiries, generating and assuring assent with competition rules, and for legal compliance.
- Intellectual property rules and documents in place to enable the conduct of competitions to develop assets for enterprise or government clients.

- Infrastructure to allow customers to create and launch their own contests and follow a workflow to administer the challenge to completion and transfer of assets.
- A centralized web location for participants to obtain problems, submit solutions, judge submissions, view results, scores, statistics, and so on.
- A central web location for discussion and interaction, providing the community with a "town square" with discussion boards and a wiki to share information.
- Profiles of and information about the different competitors - all of a member's activities are tracked in real-time and statistics on performance made publicly available.
- Collaborative software development infrastructure such as source code control, wiki content management, etc. Quick fix mechanisms to make time critical and small corrections to software developed during regular contests. At TopCoder, short stint challenges called "Bug Hunt" and "Bug Race" competitions are specifically designed to elicit a working solution to small problems. These challenges are used to update content, to develop quick fixes to technology assets and documentation where the contest ends once a demonstrable solution is submitted, often in a matter of hours.

TopCoder's clients can identify the problem to solve and even contribute to picking and choosing what parts of the process to use. This approach is particularly well-suited for the development of new systems, where the integration points with existing systems are well-defined and can be tested by the community or accurately simulated. Bugs in existing systems can also be fixed using the same types of development environment made available to the community.

Figure 17: TopCoder Development Cycle for *each* software component

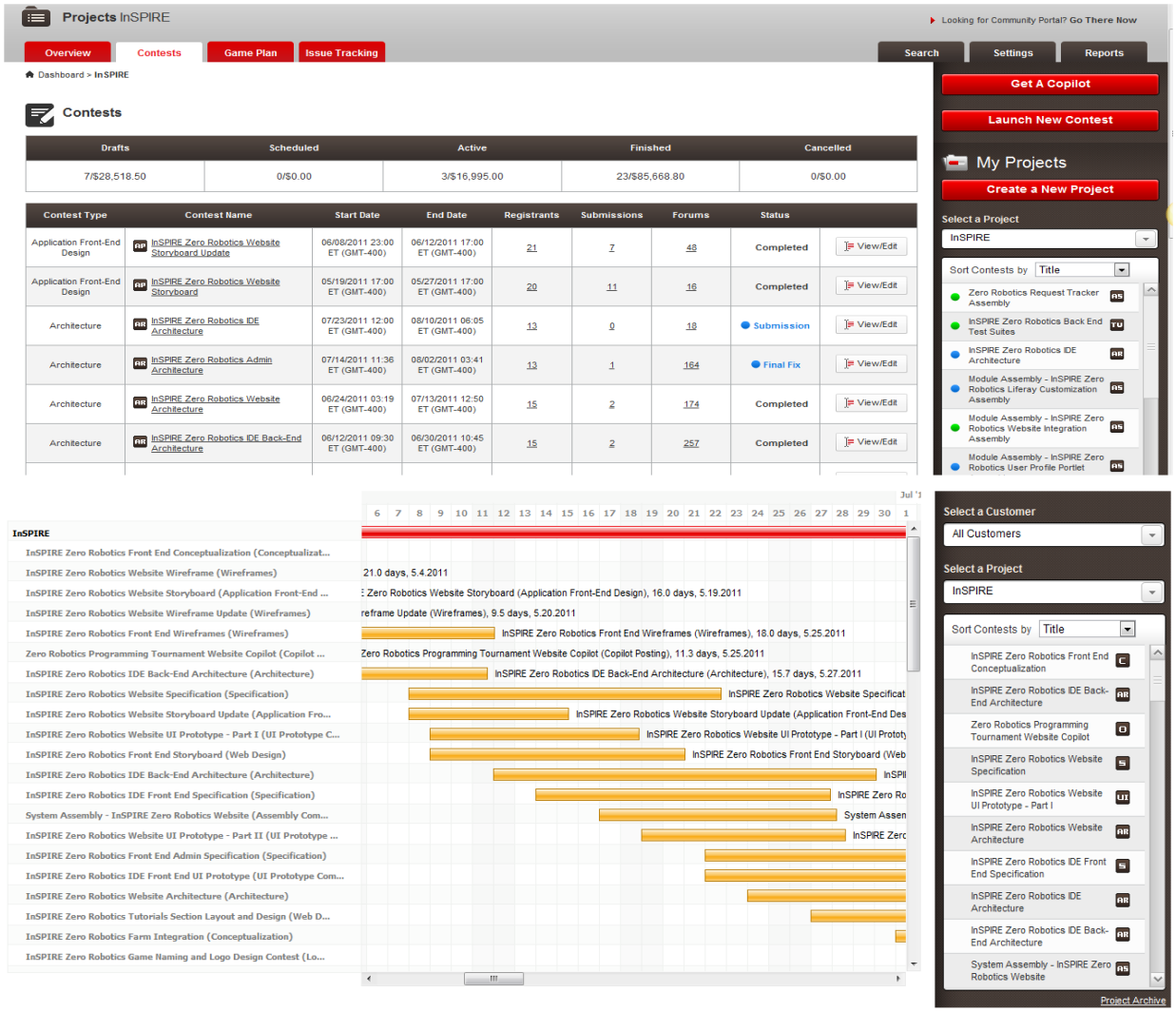


Figure 18: List of contest details and schedule of the InSPIRE program to develop the Zero Robotics Web Interface

Over the past three years, TopCoder has run over 4500 challenges with 91% completing successfully. Among other factors, TopCoder attributes the high rate of success to the methodology of breaking down a task and honing in on the key elements, the large size of the community covering a variety of technology disciplines, and the ability to use historical data to design and price the challenges in a way that they will be successful. Additionally, TopCoder has over the past ten years developed and refined these contests, attracting hundreds of thousands of technologists and the infrastructure to support them.

With the respect to the 9% of challenges that are not successful, TopCoder's view is that a number of factors contribute. Most typically, a competition does not complete successfully because the specification is unclear or is too complicated and is asking for more than is typically requested for that competition type. The main indicator of this is the activity – or lack thereof – in competition registration and in the discussion forums. Sometimes the market is changing, or TopCoder is testing the market, or the prize amounts are set too low to encourage sufficient participation on a particular problem. Usually, in these cases TopCoder can achieve a successful result by dividing the contest specification into multiple parts, and reposting as separate competitions, or by just raising the prizes. Of course, when TopCoder experiments with pricing, changes competition types or deliverables, or adds a new competition type, there is an expectation that some competitions may not complete successfully as the market adjusts to the change.

3.5.6. Crowdsourcing Contest Results[8]

The Zero Robotics infrastructure was built via TopCoder crowdsourcing contests, using the 2010 Zero Robotics web site as a prototype. The program has a TopCoder co-pilot who interacts regularly with TopCoder and MIT and provides technical support to the competition participants. MIT's role was to answer technical questions relating to the requirements in each of the contests and provide detailed feedback to the co-pilot and members. As mentioned in Section 3.4.5., there are online tools available to track the ongoing contests. Figure 18 shows a screenshot of the TopCoder Cockpit tool displaying the list of contests, present and past, statistics, and timeline. At a high level, the development tasks undertaken using collaborative competition were:

- Integration of the Graphical Editor being built separately by Aurora Flight Sciences
- Development of the Zero Robotics community website
- Development of the SPHERES integrated programming environment using the 2010 version as a prototype

[8] All the data and sources of eveidence presented in this section has been obtained through direct or indirect observation from the experience of personally running the TopCoder contests, querying the TopCoder/InSPIRE SQL database (with permission) and surveys from users who have used the final, developed product. Multiple sources have been used so that conclusions can be reconfirmed

- Integration of the SPHERES high-fidelity simulation into the TopCoder server compilation and testing 'Farm', which is the robust back-end that handles and implements the ZR simulation requests.

A Game Plan schedule was developed for each high-level task, divided into the following phases: Conceptualization, Wireframe (to design the look), Storyboard (to design the feel), Architecture, Assembly, Testing and Deployment. For each task and each phase, a list of required contests were made and recorded within the Game Plan. Part of the Game Plan for the front-end task is shown in Figure 19. The horizontal blocks represent each phase and the rows represent an individual contest. The columns are the timeline and the pink regions mark off the period when a specific contest is scheduled to take place.

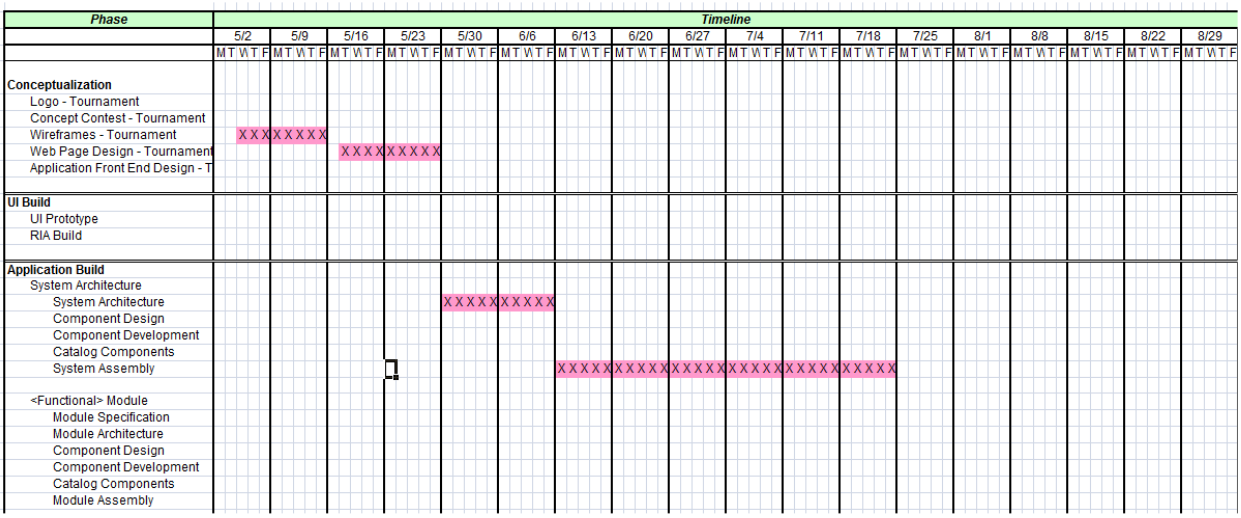


Figure 19: Front End game plan

Each individual contest lasted between 5-21 days and awarded prizes between $100-$2500 depending on the requirements and scope of the contest. The crowdsourcing contests included 3 types: graphic design studio contests (which have been described earlier; evaluated by MIT and TopCoder), software contests (which have the milestone and submission phases but are evaluated by reviewers selected from within the TopCoder community by the program manager) and bug race contests (where the first member of the TopCoder community to submit a solution wins).

Each Studio contest began with the release of a set of requirements and the inputs needed by the participants. Members of the community registered to participate in the contest during the 'Registration phase'. Once the contest launched, participants could review the requirements and work on the problem. For some competitions, such as the conceptualization and wireframe competitions, halfway through the contest participants were required to submit a "milestone" submission. Reviewers and/or the client team reviewed the milestone submissions and provided feedback to participants, awarding small prizes to up to five participants. Participants integrated the milestone feedback into their work, improved upon it and submitted their full solution by the contest deadline. All the entries were then evaluated and first and second place prizes were awarded. The winners were responsible for improving their submission according to the reviewer's final comments in the post-contest 'Final Fix' phase.

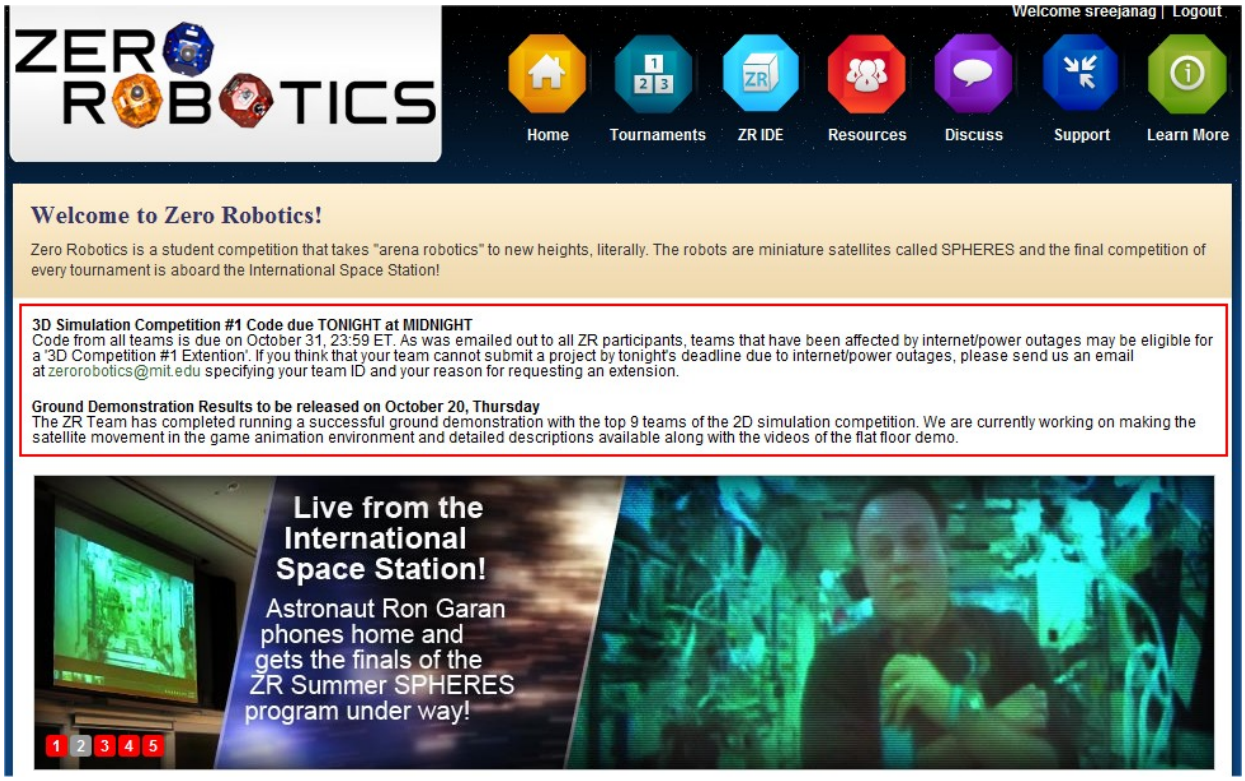


Figure 20: Zero Robotics Website, look designed by the storyboard contest

An example of such a contest is the Front End Storyboard Challenge. The purpose of this challenge was to generate ideas for a look and feel for the web-based integrated development environment to

be used by students to program satellites. The prizes for this competition were $1500 for first place and $500 for second place. There were 5 milestone prizes of $75 each. Participants were provided with a description of the solution needed, along with the conceptualization document and wireframes that had been developed in previous competitions. In response, the participants provided a series of graphic images that showed creative examples of how the screens might appear. The competition began on June 9, 2011 at 9 a.m. EDT. Milestone submissions were due June 12, 2011 at 9 a.m. EDT, and the final submissions due June 15, 2011. The winners were announced on June 21, 2011. The milestone submissions allowed the solvers to get feedback about their submissions, opening lines of communication. It also helped the competition sponsors determine whether there was sufficient participation in the competition. In this competition, there were 18 registrants, with 10 submissions at the milestone and 4 final submissions. The "best" storyboard, as determined by MIT and TopCoder (Figure 20), was selected from these 4 submissions and served as an input into the architecture group of contests for the website.

The contests to design the look and feel of the website (Website wireframe and storyboard contests) as well as contests to design the name and logo for the Zero Robotics games highlight the 'creative input' benefit of the crowdsourcing model. Evaluation was done and prizes were awarded based on MIT's judgment, with input from TopCoder. While the storyboard competition did very well, the design of the logo did not yield an integrated result satisfactory to MIT, in spite of 12 final submissions. MIT was able to finalize a logo by putting together contributions from 2 winning submissions. Had MIT not been able to do that, TopCoder could have run another logo contest using the winning submissions as inputs, and so conducted an interactive development cycle.

While the Studio and software development contests were the main development tools used to further development, Zero Robotics used Top Coder Bug Race contests to fix quick, time-critical bugs. A short problem statement and the appropriate section of design or code were released for each competition, and the first competitor to satisfactorily submit a fix was awarded a prize. The Bug-Race tracking system allows clients and reviewers to easily create requests in order to obtain the specific fixes required. These competitions typically range from about one day to a week, and by design have significantly less participation than the development contests. The 'Bug Race' competitions have takers, because the tasks are very specific and need quickly available, specific

skills. The participants work closely with the person who submitted the ticket and resolve the problem. This capability highlights the 'crowd production' benefit of the crowdsourcing model.

It is worth mentioning that the crowdsourcing model used by TopCoder for Zero Robotics is different from other online staffing outsourcing resource sites that are available, such as oDesk or eLance, in that those sites allow their customers to hire a specific person for a job, follow up with him and pay him after completion. The focus is on selecting an individual, and the competition is in the candidate selection process rather than the solution selection process. Also, in those models every contractor typically gets paid rather than only the winners. The Bug Race competitions differ from the regular crowdsourcing model (as explained with Figure 17) in that they are a request for a deliverable, rather than for a specific person, even though the result is that a small number of individuals complete most of the tasks.

3.5.6.1. Contest Participation

The participation in the contests for the development of Zero Robotics was generally what would be expected – as predicted by TopCoder based on their experience from prior development efforts. There were 54 Studio and software contests in 12 broad categories held among members of the TopCoder community between April 2011 and December 2011. These contests cumulatively received 857 registrations (notice of intent to participate), 149 full submissions, and 57 prizes for these contests were awarded. There have been a total of 239 unique participants in the 54 contests.

Figure 21 shows data from the 54 contests. The contests have been sorted in the order of occurrence in the development cycle shown in Figure 17. Registration represents the amount of initial interest in the contest and submissions represent the final output from the contest, of which one is chosen to move forward per contest. Specification contests that include making wireframes, storyboards, web design and application front end design as well as the assembly contests attracted the highest number of registrants possibly due to the large number of people who possess the required design and software skills. Component production contests include prototyping tasks. On the submissions side, conceptualization is lowest, possibly due to the specificity of the task (abstraction of the given project required rather than execution of a defined task using pre-existing skills such as design). It will be shown later using Figure 23 that the submissions number and prize

values turn out to be correlated because the prize values are determined by the market, to induce the desired levels of participation.

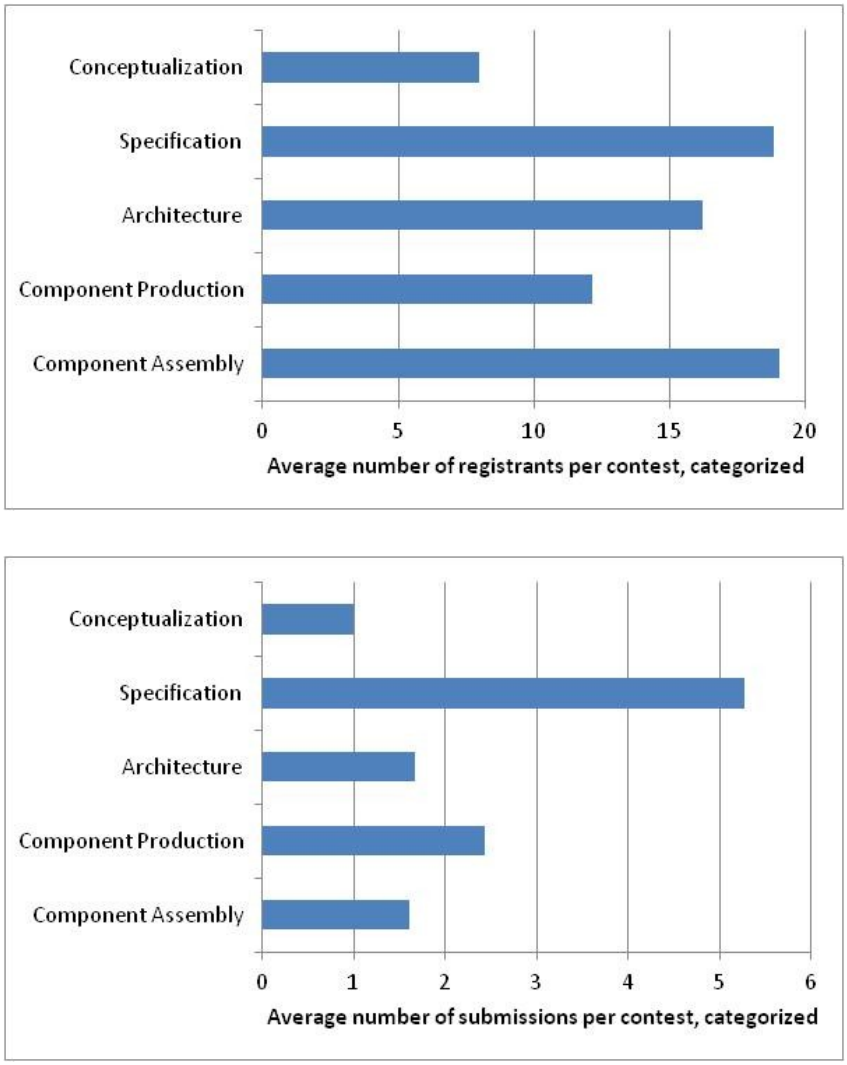


Figure 21: The average number of users that registered (top) and submitted valid solutions (bottom) per contest, arranged by broad contest category

Architecture contests, which involve discussing the software requirements with the client and reviewers, documenting them in detail and making test suites and test scenarios, had the most discussion threads on the forums. Architecture contests are also the critical point for technical design, and there were occasions where MIT rejected the winning entries because they did not meet the specifications. The back-end conceptualization and architecture contest was conducted 3 times,

and ultimately the community member who won the architecture contest not only designed but also assembled and supported the back-end all through.

It was noticed that component assembly had a skewed number of registrations vs. submissions. A disproportionately large number of people registered for these contests. It appears that they gauged their probability of winning by the discussion forum content, and only a small subset of the participants ultimately followed through to submit a solution. For example, the User Profile Portlet Assembly contest had 45 unique registrants but was dominated by the community member who won the most assembly contests in the InSPIRE project. This phenomenon was seen in multiple assembly contests – many registrants but eventually 3-4 submissions. As mentioned above, fixing bugs that are identified in the production software, small changes and integration tasks are performed using Bug Race competitions. The bugs are identified by MIT or the ZR website/IDE users and are documented in the TopCoder system in the form of an issue report. Unlike the Studio contests, there is no competition for the best solution of a Bug Race. Instead, community members contact the ZR TC program manager or co-pilot with the request to take up the Bug Race competition and the first acceptable solution is selected to fix the bug. The fixed piece of software is then merged into the existing framework. There were 163 Bug Race competitions between September 2011 and December 2011, all of which were solved by 32 winners.

3.5.6.2. Contest Prizes

Given that 239 unique members of the community participated in the contests and bug races, from one viewpoint, we were able to 'buy' diversity in participation at the rate of $800 per user over a period of about half a year. However, among the participants (counted as those who registered for a crowdsourcing contest or a reviewer), there were 90 individuals who won prize money. TopCoder therefore paid an average of $2000 per winning competition member over the 6 month period, although the payments were skewed toward larger amounts to a smaller group. Therefore, the number of people working on our problems was far greater than the number of people we paid. This does raise the concern of retention, since making any money is based on a probability of success. However, since all participants have access to the discussion forums and members' histories, they are expected to make educated predictions on the probability of their winning and

participate accordingly. As shown in previous literature, access to complete information actually encourages the participation of the strongest contenders.

Figure 22 captures the 54 Studio contests run over a period of 7 months in terms of the number of unique members who registered to participate, i.e. expressed interest to compete, and the number of complete solutions submitted at the end of the contest. The contests have been arranged in decreasing order of efficiency, defined as the ratio of submissions to registrants. Efficiency of a contest is defined as the ratio of number of submissions received to the number of registrants who expressed their interest to participate. The overall efficiency over all the contests was ~ 15% and the figure visually indicates a large number of contests that have an abnormally low efficiency, which can be due to a variety of reasons. The user profile portlet assembly contest and back-end architecture contests have low numbers because the pool of potential participants contained a member (different for each of the 2 contests) who was known to have a nearly 100% winning streak in Zero Robotics contests. As a result, the other participants backed out after gauging a lowered chance of winning. On the other hand, the highly efficient contests like the game name and logo design contest were very creative ones that did not require very specific skills, and none of the participants competing in the category had prior history with ZR. Low efficiency can be a source of concern since it potentially indicates failure to retain the captured interest in a contest and additional effort is required to increase active participation such as increasing the prize money, advertising on the TC website or actively reaching out to skilled members. This is especially required for contests where there are no strong competitors in the participant pool.

Figure 23 shows the prize money distributed for the development of products in each of the categories listed. The vertical blue line marks the average money paid per payment, which is $356 (525 payments were made, including co-pilot and reviewer payments). A total of $186,000 spent on payment as prizes and reviewer compensation (as of December 2011), not including payment of full-time staff at TopCoder. Contest categories such as conceptualization are rewarded much higher than the average prize money in order to attract members to participate in them, in a market based determination of awards. Contests that appeal to a broader skillset (as seen earlier by the number of registrants in Figure 22) such as prototyping i.e. component production and deployment did not require as high a prize for gaining potential interest. The number of contests for conceptualization and architecture is also far lesser than, say, assembly. Correlation with Figure 21 shows that contests

that had the lower number of submissions (e.g. Conceptualization) required the highest value of prizes and those that had higher number of submissions (e.g. specification and component production) had lower levels of prizes.

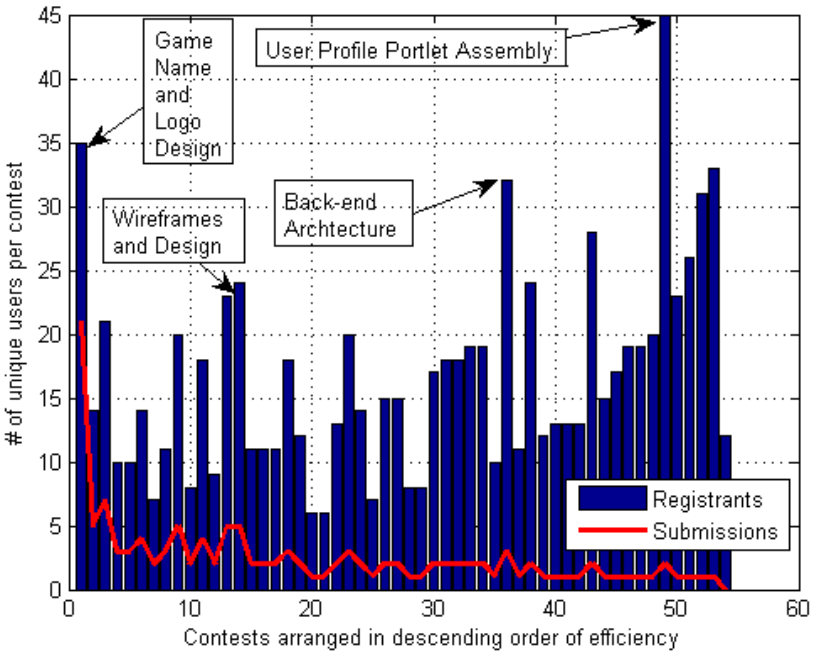


Figure 22: Number of users per contest for the Zero Robotics Development Program. The contests have been arranged in decreasing order of efficiency i.e. number of submissions (red line) to registrants (blue bars)

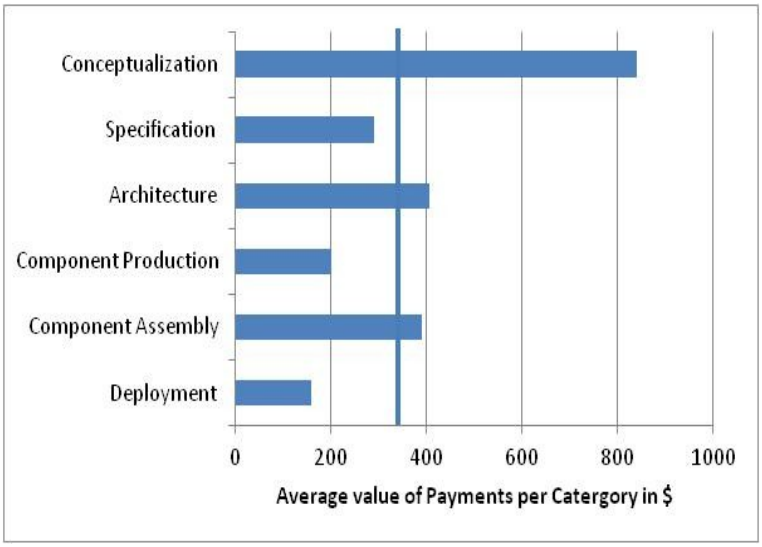


Figure 23: Dollars spent as prize money for each contest category *per contest*. The blue vertical line is the mean of all the contest prizes run through December 2012.

From the participants' point of view, a participant dominating the contests can find a good source of income, no matter which category he chooses to dominate in. This leads to loyalty that is very useful, because not only does it retain the good quality participants but also provides a field for Bug Race competitors. Figure 24 and Table 3 show the cumulative earnings of the top 11 earners in the ZR crowdsourcing contests. These 11 highest earners among the 90 total winners claimed 62% of the total money spent on all the payments. The individual who dominated the assembly contests (maximum in number and average in prizes) claimed nearly 26% of the total prize and reviewer money in assembly contests. Since the number of assembly contests is high, there was opportunity for other participants to compete for the dominating position and make significant prize money. Moreover, 4 of the top 11 winners are those who dominated the assembly contests, where the combined prize money of the lower 3 of the 4 amount to 5% of the assembly prize money. The member who won the initial architecture contest for designing the back-end of the IDE also went on to architect the entire back-end and, since the back-end is the heart of the ZR simulator, he monopolized all subsequent back-end contests as well. As a result, 100% of the back-end prizes were awarded to that individual. The individual who dominated the architecture contests claimed nearly 45% of the architecture prizes. This appears to be a direct result of the fact that architects need to clearly understand the client requirements and document them precisely in order to do well in the contests. Table 3 shows three of the highest earning categories (as established in Figure 23) and the percentage of the total earnings in that category that was claimed by the participant who claimed the highest in that category. Very obviously, it pays very well to be a "loyal" participant.

Category	% of total payment in category
Conceptualization	68%
Architecture	45%
Assembly	26%

Table 3: Percentage of prize money earned by a monopolistic player in each category

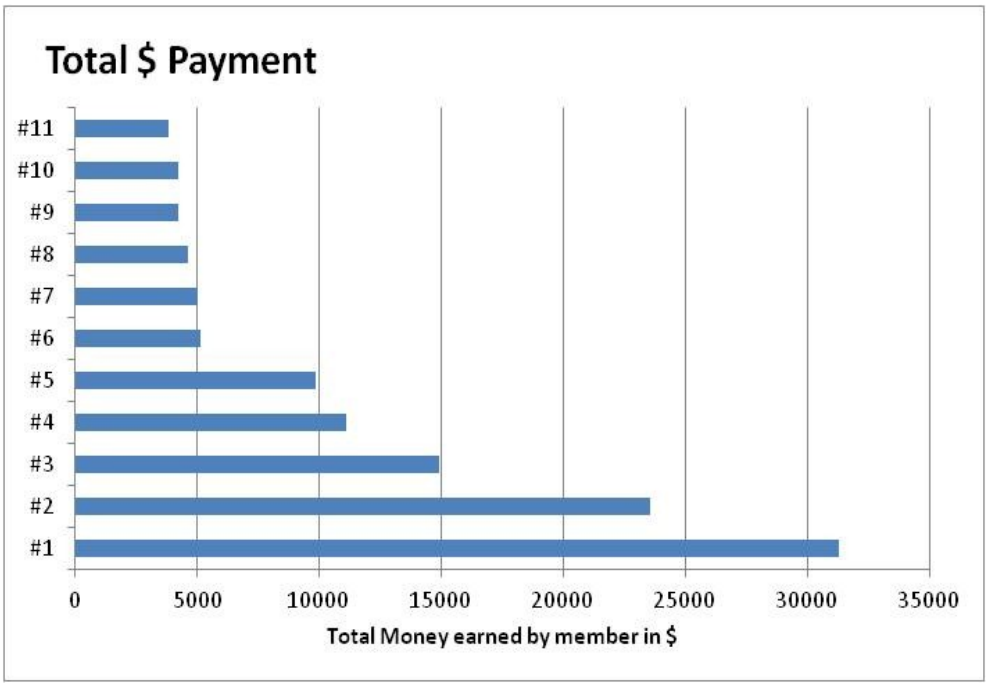


Figure 24: Prize money in $ of the top 12 community members in terms of total earnings

The "loyal" participants have been consistently conversing with MIT on the TopCoder forums over many contests and are well versed with the ZR framework, increasing their chances of winning contests due to their subject matter expertise. From the perspective of the customer, the phenomenon of "loyal" participants reduces the effort of educating new participants on the background of the ZR framework. For this reason, TopCoder provides incentives in order to encourage member loyalty. Apart from domination opportunities in contests as seen in the statistics above, loyal members (as evaluated by their 'reliability rating' and contest participation) are given extra payments in addition to the per-contest prize money. While this seems to favor partial monopolization of a market that is inherently supposed to be competitive in order to produce quality, the caveat is that the groups of people who dominate the contests are self-chosen from all around the globe, who have competitively established their position through the process of crowdsourcing. It would been much harder, if at all possible, to find such a match by looking locally for such a candidate, hiring him full-time and managerially requiring that he keep up his standards of work.

For specific problems where a large crowdsourcing contest is not required, past "loyal" members may be invited to solve it through a 'bug race' and receive a paycheck. Figure 25 is a pie chart

showing the distribution of Bug Race competition winners. The top 7 Bug Race competition winners are the same people as the top 12 members in all the contests put together, as shown in Figure 24. This is different from traditional managerial assignment in that members volunteer to participate in the races as and when they are available.

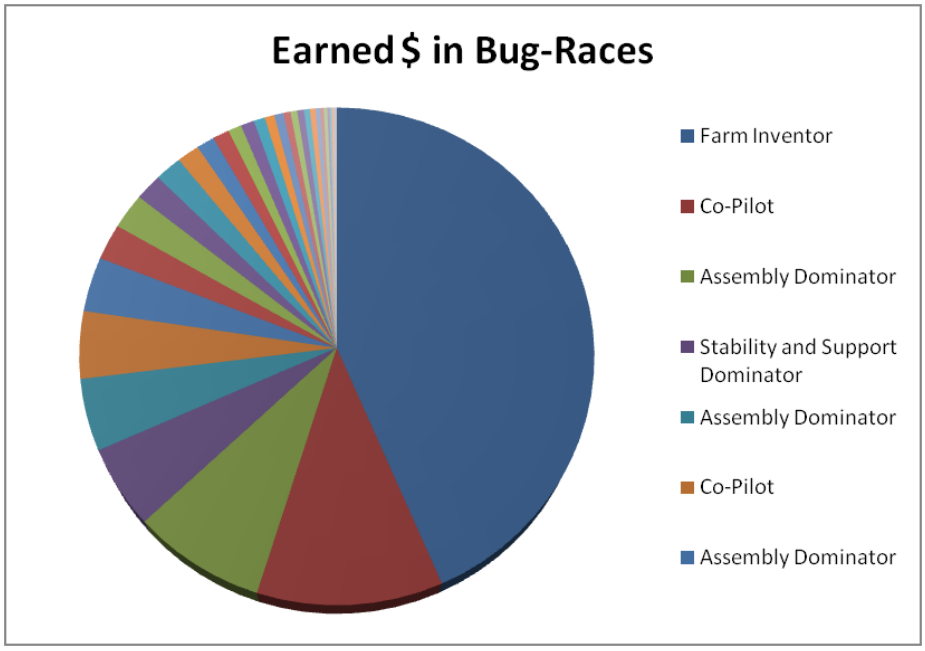


Figure 25: Prizes earned by members in the Bug Race contests. The earnings have been sorted in descending order before plotting and the top 7 highest earning members listed using their aliases

Overall, we saw participation of TC members decrease from the thousands available in the community to a few dozen that regularly submitted to the ZR contests. This trend of survival of the most powerful contestants in the presence of complete information is predicted in theoretical crowdsourcing literature [72]. Here it attracted the best of the pool to compete and also benefited the newcomers, although not quite as much as the winners. The incentive of very high rewards combined with detailed feedback is expected to motivate newcomers to climb the learning curve, if they think it possible, after which the TC loyalty benefits keep them engaged and involved in their area of expertise. The availability of detailed member performance records on the TopCoder website provides the community with the advantage of transparent information to make an informed decision on what works best for them.

3.5.6.3. Product Quality

The quality of the ZR web interface can be judged by its ability to perform load tests successfully and by the satisfaction of the students who are using it to participate in ZR tournaments. Since the first tournament on the new web interface launched in September 2011, the website has seen more than 480,000 page views with over 70% returning users. There are 1800 account holders who have among themselves created and saved more than 254,000 project revisions, run 100,000 simulations on the IDE and posted 5,150 messages on the community forums (all data as of January 2012). The 'Farm', designed and developed by a TopCoder member, is the back-end engine that manages all the simulation requests sent from the IDE and sends back the results after completion. It has yielded a robust framework for handling and managing multiple requests to the SPHERES simulator from clients simultaneously, which is key to managing crowds of users writing and simulating SPHERES software online. The 'Farm' (explained in Section 3.4.1) has the ability to relay the requests to 24 available processors currently, a number that can be scaled by adding more virtual machines on the cloud. The website initially had stability issues which irked the users (57% of 109 polled users called it their biggest complaint), but the issue was resolved within 3 weeks using Bug Race competitions and dedicated member support. Currently, the website is supported by multiple servers with the ability of adding more. Traffic can be directed to different servers by a load balancer. The numbers indicate that crowdsourcing has indeed yielded a stable web environment that successfully supported the tripling of ZR's web usage from 2010 to 2011.

An online survey was conducted at the end of the 2011 tournament season to access student and mentor feedback after they used the developed web interface. Of the 30 alumni students[9] who responded to the survey, 63% were more satisfied with the website in 2011 than 2010 and 75% were more satisfied with the IDE in 2011 than 2010. More specifically, alumni rated their website satisfaction in 2011 with an average of 3.64, standard deviation 1.24 on a 5-point Likert Scale. They rated their IDE satisfaction at 3.94, standard deviation of 0.8, again on a 5-point Likert Scale. This implies that, although only a small fraction of alumni responded to the survey, we can be more than 68% confident that the alumni population preferred the crowdsourced website. These numbers

[9] Alumni students refer to those who had participated in 2010 on the prototype web interface and returned to participate in 2011 on the web interface developed through crowdsourcing

already indicate improvement, and more improvement is expected in 2012 using the lessons learned from the 2011 pilot program.

3.6. Lessons from Apparatus Development as a Crowdsourcing Case Study

This chapter demonstrated the development of the SPHERES Zero Robotics program web infrastructure using TopCoder's (TC) crowdsourcing methodology, presented as a case study. Crowdsourcing was conducted along with established techniques of collaborative competition among TC's community of members. Members developed components of a large software system in stages, incentivized by prizes. Within this competitive framework, collaboration was mandatory for certain aspects such as supporting future contests and encouraged for other aspects such as helping fellow participants within community forums. Specifically for ZR, we introduced further collaboration by sometimes combining multiple winning entries of contests into one and working one-on-one with community members.

The case study (Section 3.5) uses qualitative (direct and indirect observation by personally supervising the contests) and quantitative data (by querying the TopCoder SQL database, with permission) to verify crowdsourcing theory trends and explain the results logically, but also serves to smoothen the design and operations of future contests. Key crowdsourcing benefits identified in literature were revisited and their pros and cons identified with respect to the lessons learned through the case study.

Important benefits observed and measured were that development does not depend on the knowledge or availability of any particular individual thus reducing single point failures, a large pool of contributors may be accessed for task diversity and loyalty in such a scenario is invaluable to since the best person for the job does it, and does it well. The total cost of prizes and reviewer payments for developing the ZR web infrastructure, enough for launching one tournament in 2011, was $186,000 (not including staff payments at TopCoder e.g. project manager). In a traditional set-up this is equivalent to hiring four software designers and developers, full-time. For the same money, we have leveraged the attention of over a thousand, the solutions of over 300 diverse-skilled individuals and tens of thousands of hours of effort – enabling a tremendous increase in benefit to cost ratio.

Frustrations were also observed: When individuals begin to dominate in particular tasks, they may become indispensible for critically related tasks (e.g. farm development, stability resolution issues and critical assemblies). Access to a global pool of solvers is possible providing diversity in innovation benefits; however, getting a point across to people from different cultures and languages can be time-consuming and carries a high risk of miscommunication and possibly faulty submissions. The judging process involves several experienced reviewers globally to ensure quality, but the process is long and makes it difficult to meet critical deadlines (e.g. the release of the IDE for the 2011 tournament was delayed by 2 weeks). Individuals self-select their tasks, so they are motivated; however, if the potential participants have more lucrative opportunities, participation will drop (e.g. work nearly stopped during the TopCoder Open, since the members were engaged there). The time and quality standards are best met when the number of active participants is at a healthy number. One of the key stumbling blocks learned from the entire TopCoder process of crowdsourcing the website development were that the time taken to finish tasks is much longer than if the task was managerially assigned to appointed software developers. The delays were primarily because of low participation, members taking much longer than expected to complete the tasks assigned to them (e.g. final fixes) and mistakes in merging the parallel or subsequent contest solutions.

Large crowds of amateur users, especially students, are currently using the developed web infrastructure to program real satellites on the International Space Station and contributing to developing spaceflight algorithms of use to MIT, NASA and DARPA as well as getting educated. Statistics from the web development effort, deliverables from the contests and the overall lessons have also contributed to helping us design ZR tournaments with the objective of crowdsourcing. The next chapter will describe the usage of the developed infrastructure in deploying a ZR game and running a ZR tournament.

Chapter 4 -
Tool and Metric Development: Zero Robotics Tournaments

Once the web interface for hosting Zero Robotics games and tournaments was ready, the next step was to design a game around a candidate spaceflight algorithm that MIT's research community would like to crowdsource and then make the game available in the form of a ZR *tournament,* as indicated pictorially in Figure 7.

Zero Robotics (ZR) is the umbrella program under which multiple *tournaments* are held. A tournament is a series of *competitions* which cater to the same group of participants (e.g. high school students or middle school students) and require one application to be submitted to participate in the tournament. Until 2011, all applications had to be from *teams* and not individuals. A *competition* is a bracketed set of matches among the participants (e.g. round robin, double elimination) at the end of which a ranked list can be declared. The participants play one game per competition, and games may be repeated over multiple competitions. Participants write programs online to play the pre-defined game and submit their program for the purpose of an automated competition. A *match* is a head-to-head run between two SPHERES satellites, in simulation or hardware, controlled autonomously by programs written by participants. Typically, opponent *players* control one SPHERE each and are each given an automatic score at the end of the match. A player is a computer program, i.e. a full ZR user project, written using the ZR web interface and capable of autonomously maneuvering a SPHERE in simulation or hardware when executed with the game code and the embedded system code.

After a tournament is kicked off by MIT, student teams can submit applications on the ZR website - http://zerorobotics.mit.edu/. Upon acceptance, they can create, edit, share, save, simulate, and submit code to play the games available in the tournament, all from the ZR website. The objectives of the tournaments are to further spaceflight algorithm research as well as educate the next generation workforce in STEM fields. By playing the game and competing in tournaments, the competitors create spaceflight algorithms. Additionally, by leveraging the excitement of the virtual gaming world and providing the reality of astronauts, ISS satellite control and a final showdown event, ZR successfully inspires crowds of students, the way only space can.

The 2011 high school tournament called the 'SPHERES Challenge 2011' was designed with the motivation to crowdsource a cluster flight problem as well as to stimulate students in STEM education, within a framework of competition and collaboration. The intent was to evaluate the research hypothesis that the right mix of competition and collaboration within the same tournament can improve both education and the quality of crowdsourced solutions. Various collaboration environments were introduced within the competitive ZR tournament structure, with the intent of improving the educational experience of participating teams and learning to design future tournaments better, with an appropriate and beneficial mix of collaboration and competition. As explained earlier in Section 3.2, it differed from the 2010 SPHERES Challenge high school tournament; however, it did draw upon a lot of the 2010 framework.

4.1. Components of the Zero Robotics Tournaments

As mentioned before, each tournament unveils one or more software games which the participants are expected to play by programming their robots, the SPHERES satellites, to achieve the game objectives within a predefined period of time. The game objectives are defined and programmed by the game developers at MIT, to elicit algorithms of research interest from the players (to achieve the crowdsourcing objective) and to make the game an educational one (to achieve the STEM objective). For teams to program the satellites, submit their programs for competitions and to interact with each other (to achieve the collaboration and competition objective), a programming interface and several other tools are provided on the ZR website, as described in the previous chapter. To complete the system described in Figure 7, which would serve students and output algorithms and education, games and tournaments are required. These components are described below.

4.1.1. The Zero Robotics Game

For each tournament, the Zero Robotics development team at MIT designs a different game. A ZR game is essentially a layer of software that interfaces the projects written by the users or students using the IDE described in Chapter 3 with the SPHERES low level code or the SPHERES embedded system, which is the computerized brain of the SPHERES satellite.

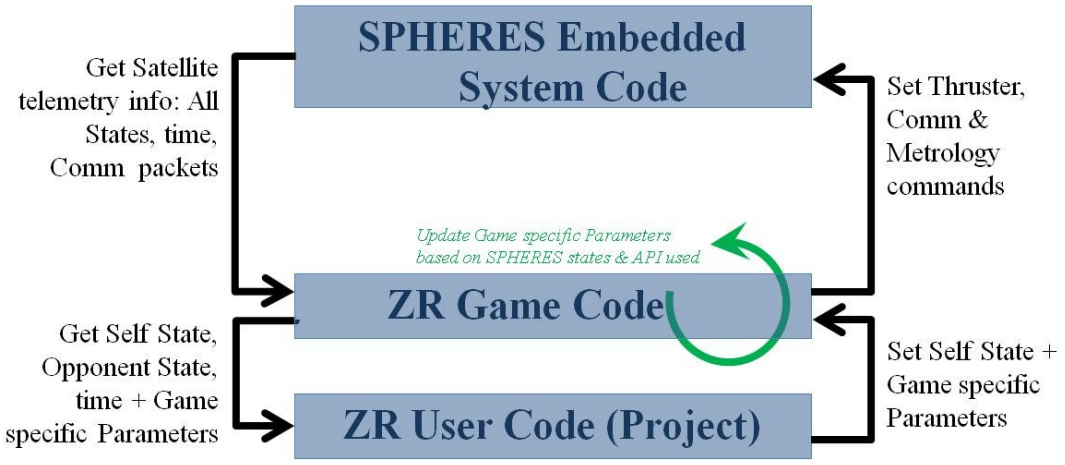


Figure 26: Block diagram of the flow of information between the three levels of code that make up the spaceflight software that operate *each* SPHERES satellite.

A simplified version of the software hierarchy is shown in Figure 26. The direction of the arrows indicates the direction of flow. In each autonomous SPHERE, for every control cycle, the SPHERES embedded system code sends the basic satellite telemetry information to the ZR based on its hardware and embedded system software. These comprise of the state (position, velocity, attitude quaternion and attitude rate) of all satellites operating inside the game volume, the absolute time of operations and the communication packets received from the other satellites. The ZR Game code software layer sends all of this information as well as game-specific parameters to the ZR User Code layer. The ZR User layer, which is essentially the projects programmed by students participating in ZR tournaments, uses the received information to play the game. To achieve the game objectives, the user code commands the SPHERE to set a specific state and/or sets game specific parameters using a library of API functions available for that specific game(see Appendix A for the API library provided for the 2011 game). The ZR Game Code layer receives this information from the ZR User layer and combines it with the information received from the SPHERES embedded system layer (states, time and comm. packets). Since the game code layer contains the definitions of all the API functions, the ZR Game code then updates the global game status i.e. game specific parameters. This process is indicated by the green circular arrow in Figure 26. Based on the updated game parameters and the user commands sent from the ZR User code, the ZR Game code sends commands to the SPHERES embedded system to command the satellite's thrusters to achieve the commanded state, broadcast communication packets containing the game

parameters and the self-state of the SPHERES to the other SPHERES and ping the metrology system to begin its estimation cycle. The SPHERES embedded system then initiates the physical motion of the SPHERES and the communication broadcast, in simulation or in hardware. This loop repeats itself at every control cycle of the satellite's software (set at 1 Hz frequency for the SPHERES). Additionally, the SPHERES states and state of health packets are broadcast to each other and the laptop that controls the SPHERES tests at 5 Hz.

The 'ZR Game Code' is a set of game-specific programs that are written to define the game objectives, time limits and area or volume of operation of the SPHERES satellites. Users play the game by programming their projects to achieve these objectives within the ZR User Code (as seen in the text editor in Figure 11 or the graphical editor in Figure 10). When the user projects are simulated, they are done so by the SPHERES simulator along with the game code libraries and the SPHERES low level libraries (embedded system code). For hardware operations, the executable file uploaded onto the SPHERES contains the user projects, game code and SPHERES embedded system code. For any given game, the users are provided with a library of API functions that they may use within their project (within the main function or other procedures) to make the SPHERES aware of the game state, communicate with the other SPHERES and command their SPHERE to perform particular actions. The 'game code' is therefore responsible for responding to the states of the SPHERES and the user projects and accordingly, command thrusters, broadcast communication packets and update the state of the game (scores, satellite fuel, etc.). It also contains the definitions of the API functions available to the participants to command the SPHERES satellites (to see examples of 'game code', please refer to Appendix B). Together, the game code and the user projects therefore command the SPHERES (via the embedded system) to behave entirely autonomously.

The ZR game, as programmed by the game code, must meet the following criteria, developed from the lessons learned during previous instantiations of Zero Robotics tournaments and constraints of the SPHERES hardware and software:

- A game with relevance to state-of-the-art research with SPHERES, so that the work of students can contribute to future research at MIT, NASA, and other research centers.
- Each player controls one SPHERES satellite during the game, which involves two players. Games of 3 players could be possible in the future, since there are 3 SPHERES aboard the ISS.

- Each live ISS event is constrained by available ISS crew time to approximately 3 hours. For effective use of resources this translates to approximately 3-5 minutes per match between players and approximately 15 matches per ISS session.
- The game must be easily played in 2D for ground contests on the Flat Floor Facilities at MIT or other NASA centers, but expandable to use the 3D nature of the ISS for the finals; both the 2D and 3D versions of the game must work correctly in simulation.
- Since it is not possible to manifest game pieces to the ISS for each tournament, all game items apart from the SPHERES are virtual. Games must be designed such that playing them results in SPHERES maneuvers and formation flight that are interesting to watch on the ISS.
- All matches must be bound within the physical playing area of an ISS lab
- Due to the dynamics of the satellites, games are slower than typical arena robotics games, and collisions are not allowed. Other approaches must be used to enhance the excitement of the competition.
- The game should be such that a large percentage of the participating teams are represented on the ISS. One method of implementing this is by requiring the finalist players to be composed of alliances of multiple teams. This will enable teams to work together for the finals aboard the ISS, increasing the number of teams that participate in the finals.
- Games should be both challenging and compact, so that the game code, player code and SPHERES satellite operating system code all fit in the highly constrained flash memory available on each satellite.
- After the end of a match, each participating satellite communicates an 8-bit integer to the onboard laptop. Game scores should be such that they can be returned within these 8 bits, so that scores of each ISS and ground match can be announced immediately after completion, rather than having to wait for all the test data to be downloaded from the ISS and analyzed.

A ZR game is therefore also a full gaming environment, where the SPHERES satellites behave as robots competing or collaborating to achieve the game objectives. They not only allow students to program the SPHERES embedded systems through an indirect interface but also serve as the basis to organize educationally engaging and video-game like tournaments, where the participants get to control real satellites through the video game interface.

4.1.2. Generic Tournament Structure

The 4 main phases in a ZR tournament are: 2D simulation phase, Flat Floor demonstration, 3D simulation phase and ISS final phase. Each phase may have one or more competitions. In each competition, students program their SPHERES satellite to play the game associated with that competition. Each competition ends with the formal submission of each team's project to control the SPHERES, following which MIT runs an automated batch simulation among all the submitted programs and declares the results. The ranks and scores may be used for elimination immediately or stored for seeding for later phases. The four phases, classified as simulation, flat floor and ISS competitions are described below.

4.1.2.1. Simulation Competitions

The Zero Robotics programming interface provides a simulation that interprets the programs written by the students in the same way as the programs will be used in the actual SPHERES hardware. In a simulation competition, MIT runs a complete round robin among all the submitted projects for that competition, where every team competes against every other team, providing useful results for the students. The web infrastructure of ZR has an automatic batch simulation tool that allows us to run thousands of simulations by just specifying the team numbers, their associated projects and the ID of the game that they are playing – as described in Chapter 3. Round robins are conducted such that for every two pairs of players or programs, one match is played where the players are allocated one SPHERE each to control during the match. It is assumed that the SPHERES are identical so each pair of players plays just once, instead of twice where each controls a different SPHERE. The simulation does not replicate every aspect of the hardware; therefore, there is still a need for ground-based testing. All results, reports and animations are made available on the website for users to review and improve their software. 2D simulation competitions precede the ground competition while 3D simulation competitions precede the ISS competitions.

4.1.2.2. Ground Competitions/Demonstrations

Teams have the opportunity to run their software on the SPHERES ground hardware available on the Flat Floor facility at the MIT SSL. Plans for expanding this event to NASA Centers (initially

Ames Research Center and the Jet Propulsion Laboratory) are underway. For flat floor operations, the satellites operate in 2D by floating on special air carriages that allow almost frictionless movement across the floor. The satellites can move autonomously using their thrusters, just like the ones aboard the ISS, and transmit data in real-time to the computers, which can display the motion of the satellites in the simulation environment, so that students can relate the hardware testing with their earlier simulation work. By watching the event webcast live, the teams have an opportunity to see the SPHERES satellites operating and learn differences between simulation and actual hardware. A flat floor competition in a ZR tournament can be seen in Figure 27.

Figure 27: A 3 Degree of Freedom (DOF) test using two SPHERES satellites on the MIT Flat Floor Facility. The onlookers are middle school students participating in the Zero Robotics Summer of Innovation Program 2010 for middle school students in the greater Boston area

Feedback from the 2010 participants strongly suggested that the importance of the ground competition scores be reduced in comparison to the simulation competitions because the facilities are not as well calibrated as the ISS. Extra mass, friction and the requirement of manual assistance to help the SPHERES move caused a lot of complaints. As a result, the 2011 ground competition was held as a demonstration event only and the video footage, telemetric data and scores were available for review online on the ZR website. The Flat Floor Facility at MIT is currenly being renovated so that it may be appropriate for ground competitions by 2013. Additionally, collaborations with NASA

Marshall Space Flight Center and NASA Ames Research Center are being finalized such that ZR may use their flat floor facilities for ground competitions in subsequent years.

4.1.2.3. ISS Competition

Teams that reach the final round have their programs run on the SPHERES satellites aboard the ISS with the help of astronauts. The astronauts run the final robotics game on the ISS, act as referees and interact with participating students via a live video broadcast. The final competition is a big event at MIT where all teams are invited to attend, interact with each other and watch the video broadcast from the ISS. The event will be webcast live to all participants so that teams which could not attend the event at MIT can see it remotely. Such a strong and strategic culminating event acts as an incentive to motivate students and the program therefore makes a positive impact on amateur participants. A photograph of the ISS finals of the 2011 high school tournament, hosted by Astronauts Greg Chamitoff, Richard Garriott, Leland Melville, John Grunsfeld and Jeff Hoffman at MIT can be seen in Figure 28. The competition aboard the ISS, which is seen being streamed live, was hosted by Astronauts Don Pettit and André Kuipers.

Figure 28: Live streaming of the ISS final competition of the ZR High School tournament 2011 in an MIT Auditorium where the mentors and students from the tournament had gathered on the day of the finals to watch the live telecast. The event was hosted by 5 astronauts at MIT and 2 astronauts in the ISS

4.2. Collaborative Gaming in Zero Robotics

In 2011, the Zero Robotics high school tournament was themed on collaboration, to evaluate the research hypothesis that collaboration among participants improved the educational benefits gained by participants as well as the quality of projects they submitted to control the SPHERES. Collaboration among participants was introduced in three ways:

4.2.1. Collaboration within Matches

The 2011 game is focused on the topic of collaboration within competition and strives to answer the question of how teams can collaborate to achieve mission objections (crowdsourcing) while also getting ahead to win the game (exciting education). The results of the 2010 game, HelioSPHERES, showed a lot of aggressive play, so much so that only 1 of the 10 finalist teams completed the game objectives on the ISS. All the other teams concentrated on trying to break the opponent's game and prevent them from achieving the game objectives. The 2011 game strongly incentivized communication and collaboration between the two players in the match such that playing 'together' got each more points than playing attack/defense. Moreover, matches were scored to exactly distinguish between the quality of formation flight algorithms so that even if perfect solutions were not achieved, the best could be calculated. Also, competition score of each team was the sum of points accumulated by that team over all its matches in that competition. This allowed us to identify the most robust players and also forced teams to think beyond simply winning a match, because they not only needed to win and get bonus points but also needed to get a large number of points to hike up their aggregate score.

The 2011 game was called 'AsteroSPHERES' [92]. The theme was asteroid mining, and it was based on the premise of NASA's future missions to explore near Earth objects. The fictional story released as a mission statement to the participants is:

"Time is running out! Our planet's energy sources are dwindling and we have little time left to save the situation! BUT, not all hope is lost. Scientists have detected the presence of Helium-3 ore on two Near-Earth Asteroids, Opulens and Indigens. MIT engineers have built SPHERES satellites that can mine the Helium-3 and collect it in mining stations for Earth-transfer. The SPHERES satellites can extract the ore by spinning on (drilling) or revolving

around (surface collection) the asteroids. More ore can be extracted if one satellite drills while the other collects from the surface of the same asteroid. The ore on Opulens is more enriched; however, it is protected by a layer of thick ice which has to be melted to mine it. Therefore, the mission to Opulens is much more difficult, but much more rewarding.

A large mining company has leased the SPHERES satellites and embarked upon a mission to maximize the collection and delivery of the Helium-3 ore from the asteroids before their orbits take them far from Earth. The satellites can collect tools that will help their mission, but if used maliciously, can disrupt the navigation of the other. Your mission, as a team of expert strategists to the company, is to devise and implement a plan to pick up the best items, extract the Helium-3 ore, deposit it at the mining station near the asteroids and signal your success back to Earth. You will be paired up with a variety of strategist teams. If you top the charts of total ore mined for the whole mission, you will emerge as the winning team and get a large percentage of the company's profits. While you do want to get ahead of the other teams and mine more ore, it is in your best interest to collaborate to maximize ore collection. The energy future of mankind depends on you and fame and glory await you!"

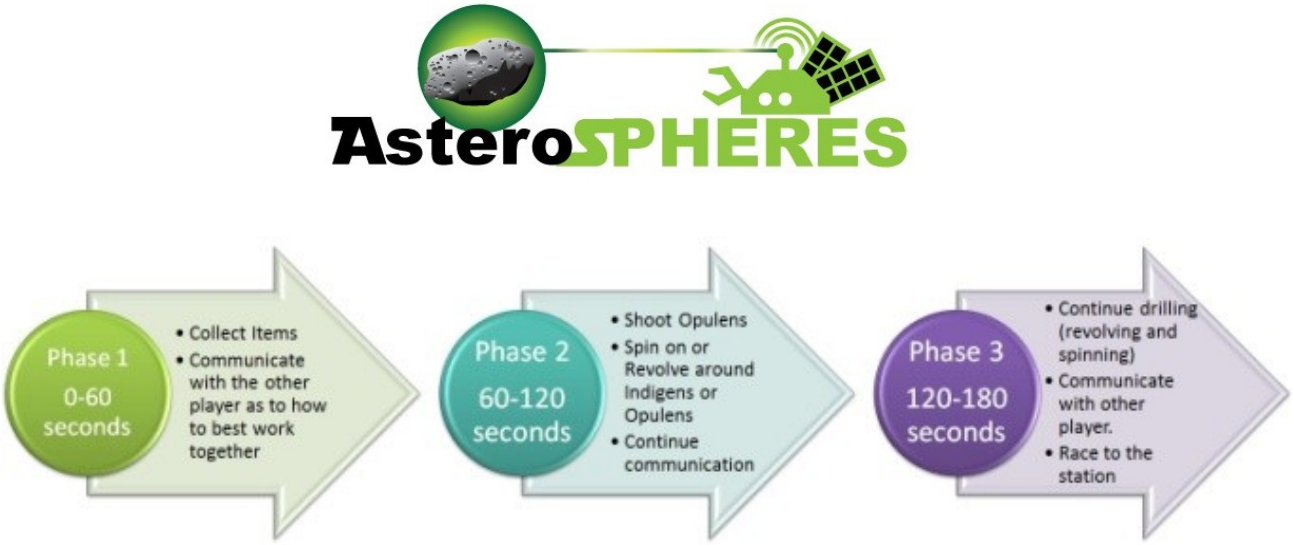


Figure 29: Game Logo and overall Game structure

In accordance with regular Zero Robotics games, each match was played by 2 SPHERES satellites controlled by opponent teams or alliances using a preloaded program (player), such that the behavior of the satellites in the matches was completely autonomous. Like the 2010 game, each player was constrained within finite resources of virtual fuel, virtual charge and code size. The virtual fuel allocation was a fixed percentage of the total SPHERES tank capacity, so virtual fuel use is

directly correlated to real satellite maneuvers. Similarly, the satellites had a finite amount of power to use the tools they collect, which was *not* correlated to real battery power of the SPHERES. The satellites were not allowed to collide with each other during a match. There was an underlying collision avoidance algorithm coded within the game such that if the satellites' trajectories intersected within 20 cm of center-to-center distance in the next 10 seconds, then all user control was disabled and the satellites were steered in perpendicular directions to their velocity till the collision was avoided.

AsteroSPHERES consisted of three stages of 60 seconds each. The game had two versions: a 2D version where all the game items, objectives and behaviors were spread on the X-Y plane *only*, and a 3D version. Each player possessed a weak repulsor and a weak tractor, which served to repel and attract the other player, respectively. These could be used to either help or obstruct the progress of the other player, depending on the strategy chosen by each team. Participants programmed the SPHERES to play AsteroSPHERES (as explained in Section 3.4) by using available ZR game API functions within their C code. See Appendix A for a full list of API functions for the 2011 game and their descriptions.

Phase One: Tool collection

Virtual tools were available to be picked up by the players: two lasers, a shield and a disruptor upgrade. A player could only pick one laser. To pick up the tools a satellite had to pass through within 5 cm of the tool's location at a velocity less than 5 cm/s. The disruptor upgrade doubled the force of the tractor and repulsor. The shield protected the satellite from the repulsor or tractor of the opponent. The laser could be used to melt the ice on Opulens (the asteroid to be mined), attack the shield of the opponent and signal mission completion back to Earth. To use any of the items, the SPHERES satellite had to be pointed in the direction of the target within a 5 degree error. The pointing direction was determined by the –X face of the satellite as shown in Figure 30. This phase did not earn points. The objective was to obtain the right tools for the strategy of Phase 2 and 3. Items that were not picked up in Phase 1 disappeared. Phase 1 in 3D is shown within the animation environment in Figure 31. To know the status of the items within a match, the participants could program appropriate API functions into their code.

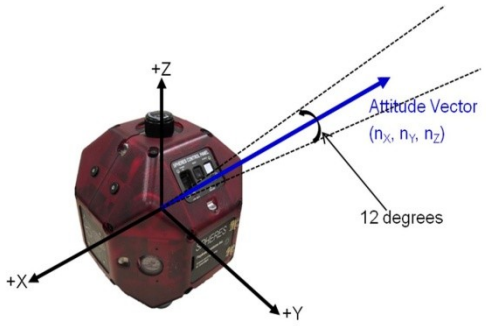


Figure 30: Virtual attitude vector of the SPHERES, must be pointing correctly for usage of items within the game

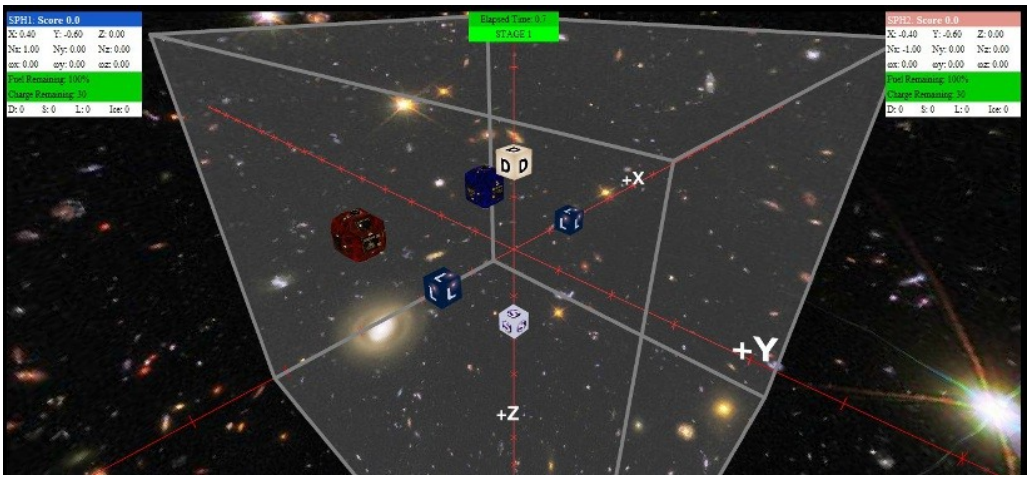


Figure 31: Stage 1 in AsteroSPHERES3D, where L=Laser, D=Disruptor, S=Shield. The red and blue satellites indicate the initial positions of the players at the start of the game. The location of the items was different in AsteroSPHERES2D. Positions of all items and initial locations were known to the participants through the game manual

Phase Two: Asteroid Mining

Two asteroids, called Opulens and Indigens, appeared. To extract Helium-3, the players could either spin on (drilling) or revolve around the asteroids (surface collection), both of which earned points. If they collaborated on extraction operations on the same asteroid, such that one spun and one revolved, both SPHERES earned *double* the points that would be earned if extraction were done individually. For an operation to be logged as 'spinning', the satellite had to hold position within 5 cm of the asteroid location at a linear velocity less than 5 cm/s and spin as per Figure 32. For an operation to be logged as 'revolving', the satellite had to be positioned within an annular shell of 20

cm to 40 cm within the asteroid location and revolve as per Figure 32. The orientation of the asteroid axis was a random vector that was randomly generated for each competition but remained the same for all matches in a competition. The players could determine the orientation real-time within a match by calling an API function (See Appendix A for the full list of API functions). The idea was to teach students rotation about a generalized 3D vector and solicit a robust algorithm that was capable of achieving the goals, irrespective of random environments. The simulation settings window (Section 4.3.2 and Figure 14) before running a simulation provided users with the ability to play with the start time of the game, phase # or items collected before any match so that they could test their algorithms within any phase of the match while programming. For formal competitions, these counters were set to zero.

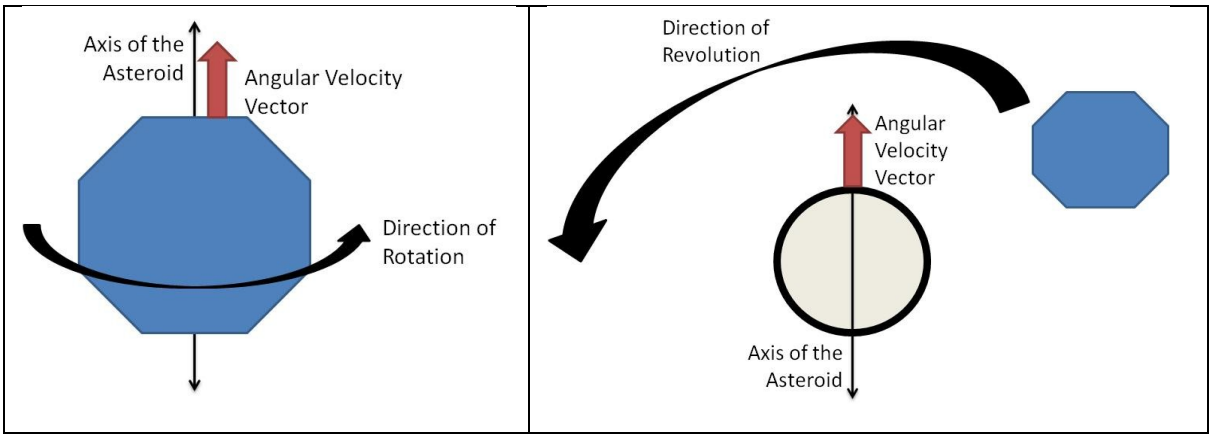


Figure 32: Concept of 'Mining' a virtual asteroid. The left panel shows the process by which a SPHERE should be programmed to spin on the virtual asteroid. To gain maximum points that angular velocity vector must be parallel to the axis of the asteroid (axis should be perpendicular to the direction of rotation). The right panel shows the process by which a SPHERE should be programmed to revolve around the virtual asteroid. To gain maximum points the angular velocity vector about the center of revolution must be parallel to the axis of the asteroid (i.e. axis should be perpendicular to the direction of revolution).

Opulens had more enriched ore, i.e. worth more points, but had a layer of ice that had to be melted by shooting a laser at it (by correctly pointing toward it and calling an API function) before any extraction. Shooting the ice layer together earned more points and melted it faster. SPHERES could begin mining Opulens as soon as the ice layer melted. Indigens could be mined from the beginning of Phase 2, but earned fewer points, as it had less enriched ore.

Phase Three: Deposit mined Ore

At the start of Phase 3 sunlight melted Opulens' ice, so both asteroids could be mined throughout this phase. In the last 10 seconds of the phase, two mining stations opened up. The first satellite to reach any station got that player points, but if collision avoidance was activated during this phase, both players were penalized, and the substantial avoidance maneuver disrupted their paths. 'Reaching the station' implied that the satellite held position within 5 cm of either station location at a linear velocity less than 5 cm/s. A match could end in four ways:

1. The first satellite to reach its station transmitted its "done" command, by firing a laser in a predefined direction, which ended the match.
2. Both satellites reached their stations (which earned points for *both* players, so the first one to reach the station had an incentive to let the opponent reach the station too).
3. Both satellites ran out of fuel without reaching the station.
4. 60 seconds elapsed in Phase 3.

The player with more points at the end of the match won and earned bonus points. The points due to the race, although also collaborative, were balanced in order to provide a competitive advantage in a largely collaborative game. Phase 3 in 3D has been shown in Figure 33.

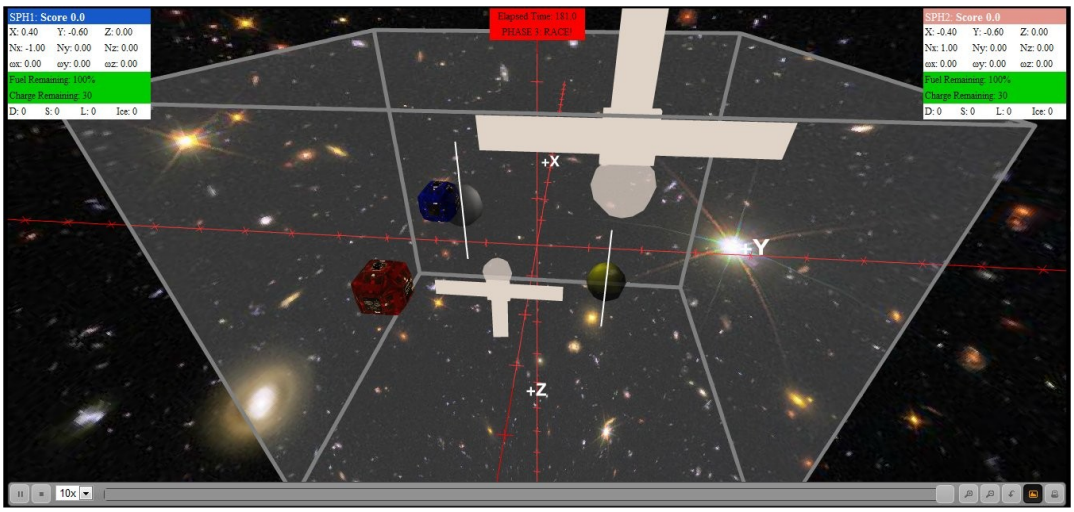


Figure 33: Stage 3 in AsteroSPHERES3D, where the yellow transparent shell marks the position of Indigens and the black transparent shell the position of Opulens. The asteroid positions were same through Phase 2 and 3. The white lines through the asteroids indicate the orientation of the axes – randomized per match. The white T-shaped structures with the shell indicate the position of stations. The location of the asteroids and stations was different in AsteroSPHERES2D, and all were known to the participants.

Match and Competition Scoring

Users could simulate their projects on any or both the satellites in a practice simulation as described in Section 3.4.1. A formal match during competitions was an automatically run simulation between programs submitted by any two competing teams. Batch simulations could be set up to run a fully automatic set of projects against each other (Section 3.4.2).

The match score of a player was calculated by summing the total number of points accumulated by that player in the 180 seconds of the match. Points in the game could be earned, as explained before, by shooting at Opulens' ice, mining, racing or winning the match and could be lost by going outside the game volume or activating an avoidance maneuver in Phase 3. The 2011 scoring was designed such that formation flight solutions submitted could be fine resolved in terms of their relative quality. For example, the number of points earned due to mining was a linear function of the angular velocity that the game required the SPHERES to spin at or revolve around a fixed point at. The mining points were prorated using Equation 1, where the CONSTANT was determined depending on whether the position of spinning or revolving was Indigens or Opulens (Opulens score = 1.3 * Indigens score), what the mining operation was (Revolve = 1.1 * Spin) and if the mining operation was done collaboratively (points doubled for both players if one spun while the other revolved). Further, mining points could be accumulated for spinning only if the SPHERE was held within 5 cm of the fixed asteroid location or for spinning if the SPHERE was within a 20 cm to 40 cm ring around the asteroid location. Finally, in Equation 1, ω for spinning is the angular velocity of the SPHERE calculated as the dot product with the asteroid normal, ω for revolving is the angular velocity that the SPHERE radii to the asteroid location sweeps calculated as the dot product with the asteroid normal, resonance ω is the above angular velocity at which maximum points can be gained and maximum ω is the maximum angular velocity at which any points can be gained (different for spinning and revolving).

$$\frac{points}{sec} = \frac{CONSTANT}{resonance\ \omega} * \omega$$

$$\frac{points}{sec} = \frac{CONSTANT}{resonance\ \omega} * (maximum\ \omega - \omega)$$

Equation 1

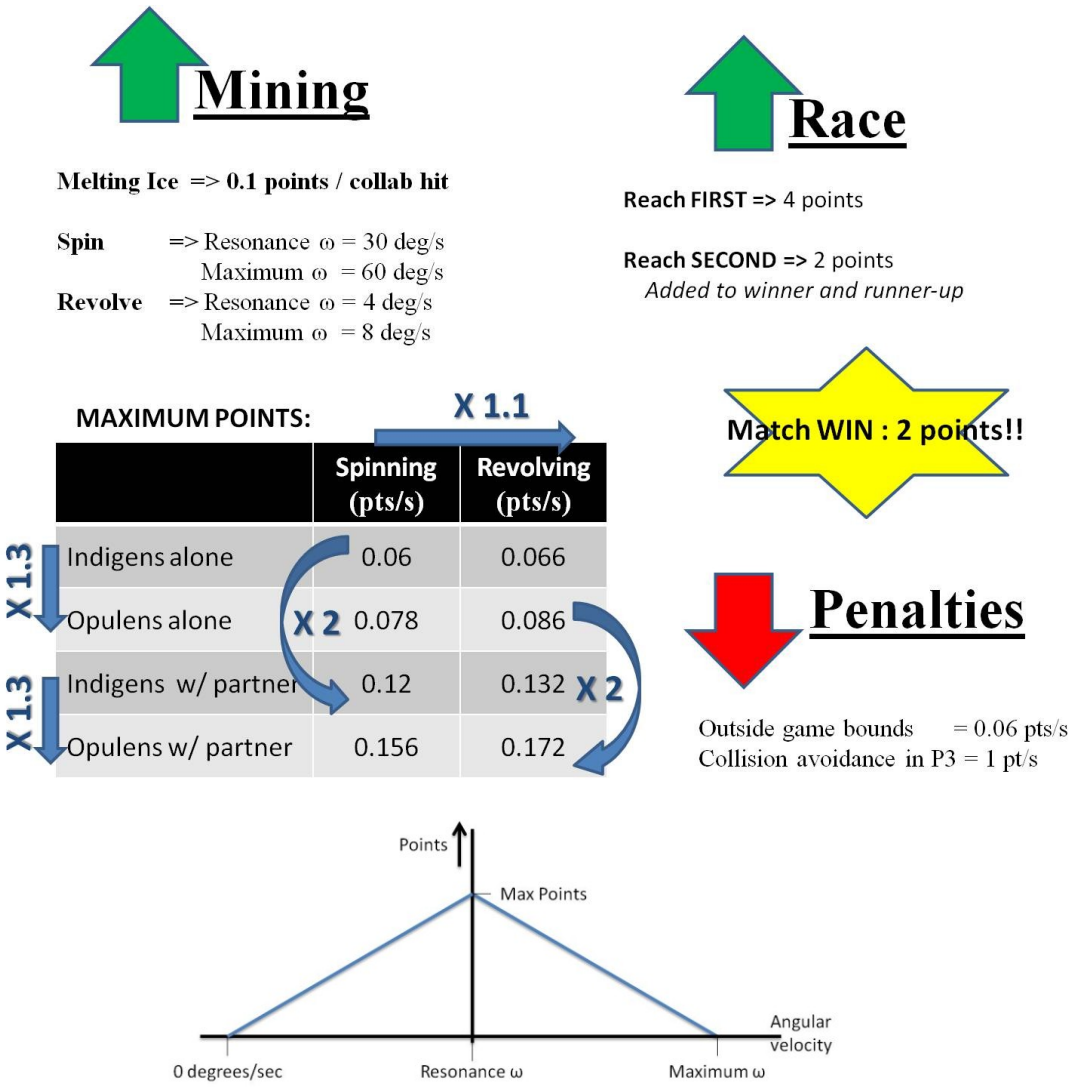


Figure 34: Scoring Summary of the AsteroSPHERES game. The numbers mentioned in the above figure were the numbers for those for the last 2 competitions in the tournament. Score numbers changed with every competition in the tournament, as detailed in the text. Refer to Appendix B for example Game Code written to respond to User Code such that the SPHERES behaved like autonomous robots to perform the maneuvers and gain points accordingly

The match scoring scheme for the last competition in the 2011 tournament is summarized in Figure 34. Figure 35 shows a typical player profile as he collects points through the duration of a match. The inclined phase is due to mining and the box phase is due to the lump sum of points received due to docking to the station. The lump sum is 4 or 6 points if docked individually or with the player's opponent. It is clear from the example above and the intricate and prorated scoring system

that the match scores are capable of differentiating between finely different formation flight maneuvers, as required in the game objectives. Therefore, students who program their SPHERES to achieve high scores in simulation and hardware have indeed achieved exact solutions to pre-defined problems, whose exactness can be calculated and sorted to find the 'best' solutions. The process of evaluating the results and more insight into the scoring equations is available in Section 5.1.2.3.

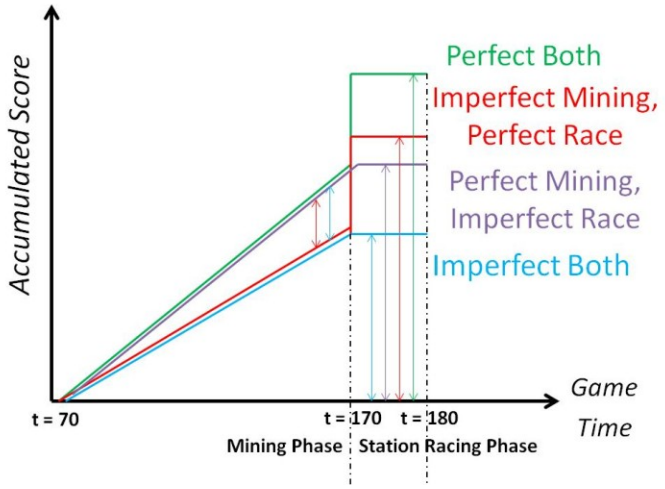


Figure 35: Typical score accumulation profile during a match, assuming uniform mining behavior. In Phase 2 and 3 (mining phase), teams can mine – spin or revolve - perfectly (purple, green) or imperfectly (red, blue) and points will be prorated accordingly. In the last 10 seconds of the match, teams may race successfully (green, red) and get a 4-6 points chunk or fail to dock to the station (blue, purple) and get no points. The final match score, indicated by the vertical arrows, is the sum of the prorated mining score (+ Opulens shooting score) and the score chunk for docking to the station. The distance between the inclined lines is the score difference due to mining efficiency.

The scoring system and some game rules changed with every competition, such that the ISS finals were the most competitive (e.g. relatively more race points than before). Since the game was inherently collaborative, each competition was in a round robin format such that every player played every other player (players were submitted by teams or alliances – explained in section 4.2.2). The competition score for any player was the sum total of the scores over all the matches played by that player. Thus, it was in the player's advantage to collaborate within each match to maximize his score rather than just beat the opponent. This also implies that the competitions were scored such that the

players which could achieve the match/game objectives and maneuvers, irrespective of opponent and environmental situations, emerged higher than those who were not so capable.

4.2.2. Collaboration within Alliances

An important lesson learned from ZR 2010 was that there was significant loss of interest from teams that fell back after the first elimination rounds. We tackled this problem by allowing more teams (27 in 2011 as opposed to 10 in 2010) to reach the ISS finals as 9 alliances of 3 teams each. The 2011 tournament required that the 54 semi-finalists, chosen from all participating teams after the elimination rounds, form groups of 3, called 'alliances', and work together to make a common project for submission. Alliances were formed by an automatic algorithm, taking into account preferences of partnering teams and the relative seeding of teams, as will be desribed at the end of this section. The intent is to encourage teams to review the performances of their peers, form alliances with those they find complementary to their skill set, and work collaboratively on common projects using our online tools.

The schedule of competitions in the tournament is shown in Figure 36: in the first two simulation competitions, one 2D (where participants played the 2D version of the game) and one 3D (where participants played the 3D version of the game), the participants competed as individual teams while in the last two competitions - one simulation and one on the ISS- they competed as alliances of three teams that submit one integrated project. As mentioned before, the website allowed each user to share his projects with other teams in the alliance such that multiple users could edit the same project, therefore making alliances with geographically separated teams possible. In fact, the EU alliances had teams that came from different countries.

The alliances were formed taking into consideration the preference of teams for partners as well as the tournament seeding of the teams. After the 3D Simulation Competition #1, the top 54 teams, ranked by the combined scores of the 2D and 3D simulation competitions, were divided into 3 tiers of 18 teams each. In the first phase, teams in the top tier ranked their preferences for alliance partners in the middle tier using a tool available on our website. Likewise, teams in the middle tier ranked their preferences for alliance partners in the bottom tier. In the second phase, MIT used this information to form the alliances. *Starting with the bottom seed* of the middle layer, each team was

partnered with their first remaining preference from the bottom tier. Therefore we had a partnership between each team of the middle tier and their corresponding selection from the bottom tier. Similarly, *starting with the top seed of the top layer*, each was partnered with their first remaining preference from the middle tier. This resulted in an alliance comprised of one team from the top tier, its partner from the middle tier and the middle tier's partner from the bottom tier. By dividing teams into tiers and enforcing a team from each tier in an alliance with preference to the lower seeds in the second phase, we prevented the strong teams from getting stronger by partnering with only the other strong teams. The weaker teams had a chance to join forces with the stronger teams and learn from them. While all teams in the alliance could share projects and chat online with anyone who was also editing the project, only the tier 1 teams were allowed to submit projects for formal competitions. The process has been summarized in Figure 37.

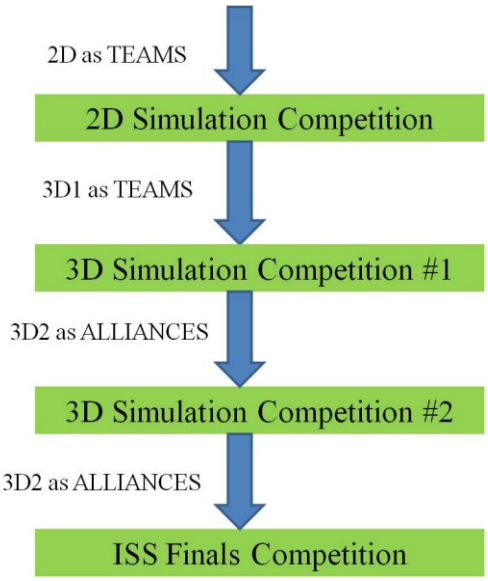


Figure 36: Schedule of competitions within the 2011 HS Tournament. 2D competitions required participants to play the 2D version of AsteroSPHERES as the game and 3D competitions required participants to play the 3D version of AsteroSPHERES as the game. There was ~ 3 weeks for teams to play the game associated with the competition and submit their projects via the website for the formal simulation competition (or finally, to send to the ISS to run on space SPHERES hardware) – each blue arrow in the diagram is ~ 3 weeks long. All simulation competitions were essentially batch simulations of all the submitted projects by teams or alliances, run by the web administrator in the RR format after being associated with the competition's game.

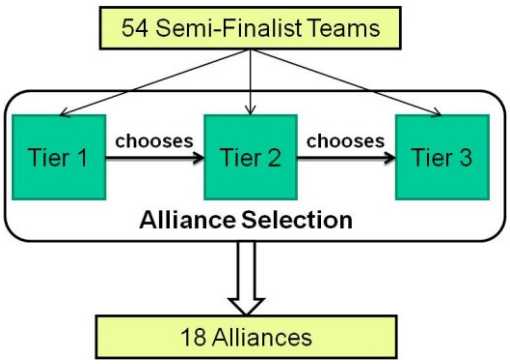


Figure 37: Alliance Selection of ZR 2011

4.2.3. Collaboration on the Community Forums

The Zero Robotics website provided discussion forums for teams to communicate with each other and the game developers on the topic of programming/educational materials, brainstorming for strategies of collaboration within the matches, debating communication protocols within the limited bandwidth of data transmission between the SPHERES satellites and many other competition related interests. The forums were used extensively, with some users posting hundreds of messages. For example, AsteroSPHERES allowed the players to transmit unsigned short typed messages to the opponent player and receive the opponent's messages once every second. Teams took advantage of this facility by collaboratively coming up with elaborate communication protocols and game strategies based on the protocols. Eventually, one protocol and strategy emerged as one that more than 50% of the participants took up and followed, thus exhibiting a truly collaborative gaming environment.

The challenges and project sharing tools also facilitated interaction among the teams on the website. Additionally, after every competition, MIT posted every simulated match played out in the competition on the website, in the regular animation environment so that teams could learn from their mistakes and others' exhibited behavior.

4.3. Design of Quasi-Experiments using the ZR Tournaments Tool

Chapter 3 explained the SPHERES Zero Robotics (ZR) Program and the development of the web interface required to run the program. This chapter so far explained the components of the ZR Tournaments that can be launched through the program, with special focus on the design of the 2011 tournament and its collaborative environments. This section will discuss the methodology to achieve the thesis objectives using the tournaments: To assess the impact of crowdsourcing on CS-STEM Education and to measure the effects of collaborative competition within this framework. In the first half of this section, I will highlight *how* the ZR Tournaments were used as a tool that impacted crowdsourcing of cluster flight software and education, and then propose a framework to evaluate the effects of collaboration among participants on both these objectives. In the second half of the section, I will list the metrics and sources of data that I used to make measurements of impact of ZR on the thesis objectives and discuss my methods in the context of reliabity, validity, significance and representativeness.

4.3.1. ZR Tournaments as a Tool

Crowdsourcing, in this thesis, is defined as the method to solve a hard problem by opening it up to crowds in the form of an open call. To achieve both crowdsourcing and education in the same program, the ZR Tournaments would have to be designed such that a hard problem is solved by students or potential students. Further, the tournaments should have the opportunity to evaluate the impact of the different collaboration environments introduced, as described in Section 4.2. The following sub-sections will describe how the ZR Tournament in 2011 was used as a tool to achieve the above aims and thereby justify the use of general ZR tournaments for the same.

4.3.1.1. A Crowdsourcing Tool

The ZR 2011 tournament was designed such that in writing computer code to achieve the game objectives, the students were writing code for formation flight maneuvers ('gaming of problems'). The scoring in the ZR tournaments was designed such that more robust algorithms, as per predefined metrics, scored higher points.

In the context of Figure 26, this means that in ZR 2011, the ZR Game code layer and the ZR User Code layer *together* commanded the SPHERES embedded system to make the SPHERES demonstrate formation flight maneuvers – first in simulation and then in microgravity inside the ISS. To play the game, as mentioned in Section 4.2.1, the participants had a library of API functions available to them which they could use within their projects to program the SPHERES to make simple movements, assess the state of the game (e.g. fuel remaining, time remaining, game score, etc.) and communicate with the other SPHERES. When the participant code was simulated or was run on the SPHERES hardware on the ISS, it was executed with the game code (which contained the definitions of the API functions) and the SPHERES embedded systems code. The user projects and game code complemented each other perfectly and formed an artificial intelligence (AI: Figure 38) program that controlled the SPHERES on the ISS – making them behave like autonomous robots.

This also demonstrated that by designing a hard cluster flight problem as 'game code', inviting participants to play the game by writing 'projects', combining the game code with the projects,and finally testing the combined AI software on embedded systems and hardware in space., it is possible for amateur crowds to develop new and improved algorithms for complex formation flight maneuvers. For example, if the formation flight problem to be solved is to develop a control algorithm that follows a zigzag path between two points, then the designed game objective could be to move between the points while navigating through a path of virtual obstacles placed in a zig-zag manner. The designed scoring could be a function of the time taken, fuel spent and penalties for hitting the obstacles. To play this game, students would program the SPHERES to move in a zig-zag fashion to avoid the obstacles by using the API functions available for this game, and the quality of their algorithm would be reflected in their score. Since the game code contains definitions of all API functions and is able to interface with the SPHERES embedded system code, the game and user code put together (Figure 38) form the software for zig-zag navigation of the satellite and hence the solution to the formation flight problem.

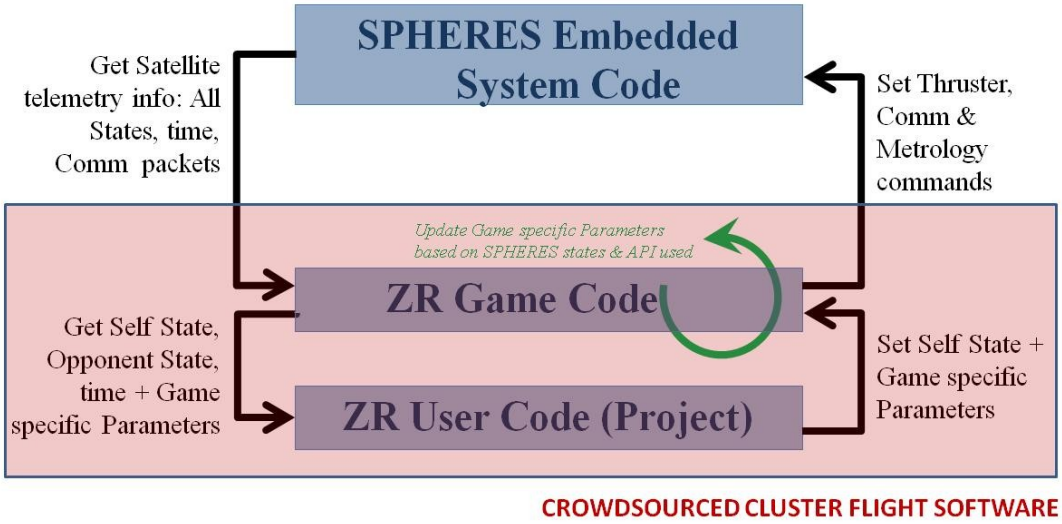


Figure 38: Block diagram of the three layers of the software that run the autonomous SPHERES satellites as represented in Figure 26. The red block now represents the formation flight software developed through ZR's crowdsourcing efforts, by combining the game code layer – where in I coded the problem – and the user code layer – where in the students code the solution to the game. Together, they command the SPHERES embedded system to achieve formation flight maneuvers

4.3.1.2. An Educational Tool

By allowing students to program real satellites using a high-fidelity simulator in an exciting video-game environment, the ZR program helps teach them math, physics, programming, strategy and communication i.e. 21st century skills, through engagement in real-world problems. ZR has successfully demonstrated tapping into the positive effects of games, as described in Section 2.4, in the following ways:

- Each ZR game has a fictional but feasible story [59] to provide participants with an epic mission. The youth likes to save worlds and learn from heroes. A 'Star Wars' inspired droid (SPHERES) racing for revolutionary goals goes a long way far in inspiring them.
- The flash animation environment provides a sense of virtual worlds like a video game which allows programming to be fun and play and not just writing code.
- ZR provides the opportunity of an epic win [60] in a race that is literally out of this world. The incentive of ISS participation and astronaut interaction serves to motivate students all

along. Also, since in the culminating event of ZR, all participants are invited to a common location to prepare for this 'epic win', the programming competition enters the real lives of people. Participants who had been corresponding and collaborating mostly over the internet can then meet each other and share the excitement.

- Games increase productivity by keeping up the sense of urgent optimism [60]. ZR allows racing among team members and scrimmaging against other teams. These online tools as well as closely spaced competitions, i.e. multiple short and long term goals, keep the pace of performance high through the tournament.
- ZR games aim to incentivize collaboration among opponents [63]. This is a valuable lesson for students because projects are increasingly becoming complex, and hardly any can be completed by an individual discipline, office or organization. Students work together as a team, outside of their teams in alliances and together with opponents to achieve game objectives. Collaboration in so many layers is expected to lead to exchange of knowledge and communal discovery. Students get a valuable primer that will help them in real world collaborative scenarios in the future.
- ZR games are strategy and mathematics intensive which encourages analytical thinking and pique the problem solving interest of many. It provides food for different skill sets within a team.
- Every ZR game has random variables and participants are expected to write players that can deal with the element of uncertainty. While the online tools give users the ability to tweak these variables, their random nature makes for unexpected and interesting twists in the competitions.
- The program is free of cost and completely web-based. It requires just mentor and student enthusiasm and very minimal resources, so it is easily accessible and quickly scalable.
- Each competition and challenge returns a large set of results. Consistent feedback of performance [63] allows teams to monitor their progress. Participants have the opportunity to review performances of all others and form alliances that are stronger than any of its individual parts, leading to more evolved players.

ZR taps into real world problems, frames an interesting game around it and therefore tries to promote project-based learning, guided by mentors.

4.3.1.3. Effects of Collaboration

The effect of collaboration on the objectives described in Sections 4.3.1.1 and 4.3.1.2 can be measured by first enumerating the design space. Table 4 shows the design space for evaluating the influence of collaboration in the ZR Tournaments on the dual objective of developing spaceflight algorithms through crowdsourcing and STEM education of students in an experiment design framework [10]. The three types of collaboration environments are arranged on the vertical axis and the objectives of the program arranged on the horizontal axis. The white colored rows and columns of the table represent the data sources that are used to explore the effect of a particular collaboration environment on the particular objective.

Exploring all six white blocks in Table 4, for all levels of the collaboration variables, would imply exploring the full design space. Even if only two levels per collaborative environment were to be considered, *exists* or *does not exist*, the total number of full-factorial experiments to be conducted to explore the design space of 3 variables (or factors), 2 attributes (or levels) each and 2 objectives is 64. If one were to do a reduced parameter study [93], i.e. do just enough experiments to compare the effect of a variable against a control but not capture the interaction between variables, = 1 + (2-1)*3 = 4 experiments would be required. Finally, if one were to take a Latin hypercubes approach [10] and randomly select a combination of levels for all the variables and objectives, 2 experiments would be required. However, the randomized set of experiments to explore the design space by any of the methods described above was not possible because ZR 2011 was primarily an outreach program where all the mentioned 'experiments' are passive collection and analysis of data, reported on an aggregate basis without disclosing anyone's identity. Dividing participants into control and experiment groups and subjecting them to different environments could negatively affect the fairness of the program and interest within it. Therefore, quasi-experimental techniques were used to understand the design space. Hence, the associated experimental framework and the analysis presented in Chapter 5 should be interpreted more as an ***observational study*** than an exact experiment.

DESIGN SPACE		VALUE DELIVERED TO	
		Spaceflight Algorithms by Crowdsourcing	**STEM Education**
COLLABORATION ENVIRONMENT:	**In-Game**	Competition results in 2010 and 2011, satellite telemetry of SPHERES Operations on the ISS	Surveys
	Alliances	Competition results	Surveys, Competition results and Participation statistics in 2011
	Forum	Competition results and Website usage statistics in 2011	Surveys, Competition results and Website usage statistics in 2011

Table 4: Design Space for deductively evaluating the research hypothesis of the benefits of collaboration on STEM Education and development of spaceflight algorithms, filled in with the data sources[10] used for the conclusions. The colors correspond to the colors on the thesis motivation image in Figure 2

Quasi-experimental analysis is a part of evaluative research [8]. Quasi-experiments are controlled enquiries somewhat resembling controlled experiments but lacking key elements such as pre- and post-testing and/or control groups. They are distinguished from true experiments primarily by the lack of random assignment to groups, not only because such groups do not exist but also because participants self-select their groups, by freedom of choice. Data collection is therefore passive. Quasi-experiments can be evaluated by (among others):

- Time series analysis i.e. using measurements over a certain period of time

[10] Data for analysis is obtained by querying the ZR SQL database for all the competition results, participation and website usage statistics, from NASA Marshall Spaceflight Center for satellite telemetry during SPHERES operations in the ISS and through SurveyMonkey databases for user feedback

- Non-equivalent control groups i.e. those that are similar to the experimental group but not created by random assignment and differing in terms of the key variables
- One-shot case studies i.e. compare a single experimental run with a well-established standard
- One-group pretest-posttest design i.e. compare before and after results
- Static group comparison i.e. compare different individuals who have been subject to different treatments

Quantitative evaluations can be done using univariate or multi-variate analysis i.e. the analysis of one or more variables simultaneously with the intent of determining an empirical relationship between them. Sometimes there may be controlling effects of a third or different variable, and the relationship with the third variable can be measured using the elaboration model [8]. Qualitative evaluations can be done by identifying patterns, case-oriented analysis and cross-case analysis and the grounded theory method i.e. an inductive approach to the study of social life that attempts to generate a theory from the constant comparing of unfolding observations. Quasi-experimental analysis is used to analyze passively studied/observed data, the results of which will be presented in Chapter 5.

4.3.2. Metrics and Sources of data

To analyze the effects of the ZR program and its collaborative environment on the objectives-spaceflight algorithm crowdsourcing and STEM education, metrics were determined to measure the value of the algorithms developed by crowdsourcing and the value of STEM Education imparted. Possible data collection methods and data sources were identified to quantify the metrics and the collaboration environments. These sources are listed in brief within Table 4. Both metrics and data sources as well as methods of collection have been described below.

The ZR game in 2011, AsteroSPHERES, was designed such that the scoring for each of the players (SPHERES satellites) was prorated proportional to the achievement of objectives, which in turn were framed such that the algorithms written to achieve them could be of interest to the research community. For example, one of the game objectives was to write a controller to make the SPHERE revolve around a pre-determined point, maintaining a specific distance and orientation

around it. Although this algorithm is a very well-researched one, with analytic solutions available, it was solicited within the ZR 2011 game as proof of concept - to show that a formation flight algorithm can be solicited by framing a game around it and opening up to students, therefore providing scientific and educational value. The scoring was prorated with objective achievement so the scores at the end of a match were a proxy for the quality of the algorithms developed as well as reflective of the performance of the participating teams.

The ***value*** of crowdsourcing and STEM Education through the Zero Robotics program and the effect of collaborative competition can be best measured in terms of ***costs and benefits*** of both the objectives. While the metrics of costs have been listed here, cost and effort numbers have not been researched upon a great detail in this thesis. We have constrained our scope to measuring value in terms of primarily benefits, described in Chapter 5. However, since the value-centric design in aerospace [94] and software engineering is defined by costs and benefits over product lifecycle, the structure of costs is qualitatively mentioned.

The ***costs incurred*** by the program are:

1) The capital resource costs of installing the program (non-recurring costs), i.e. making the software infrastructure as described in Chapter 3. These costs can be measured by the money spent on full time employees working on the program at MIT and TopCoder, money spent on TopCoder contests (prizes, payment to reviewers, co-pilot payment, etc.) and the man-hours put in by the TopCoder contest participants. It is expected that once the program is well established, the only costs to the organizers will be maintenance costs, as enumerated below. As an example, ~ $186,000 was spent on prizes and reviewer payments for developing the ZR web infrastructure from May through December 2011, not including staff and managerial payments. The developed infrastructure was used to conduct the HS tournament but was not complete. Completion and bug fixing i.e. 'well-establishment' was estimated to take at least another 6 months.
2) Tournament Maintenance costs include the time and resources required to:
    - Decide on an interesting and relevant research problem to be solved
    - Design a game around the research problem such the scoring system correctly reflects the quality of the algorithm in solving the research problem
    - Program the game code, game API libraries (example in Appendix A)

- Program an appropriate animation environment to view the simulation results as an exciting visualization
- Write a detailed game and tournament manual
- Create a new web tournament on the website, publicize the tournament to invite registrants from appropriate audiences
- Kickoff the tournament at a pre-announced date where the following will be available
  a) Game and tournament manual on the ZR website
  b) The game code as an executable on the IDE, so that participants can write their programs and compile them with the game code in order to play the game
  c) Access to discussion forums for the tournament
  d) Access to a Support Ticket system for the tournament
- Create competitions for the tournament, invite participants to submit entries for the competitions
- For each competition, run batch simulations using all the submissions per competition and make all results and animations available on the website
- During ground competitions, test and finally run the competitions on the Flat Floor Facility i.e. a full SPHERES hardware session in 3 DOF, at MIT as well as ensure good video recording of the event
- During the ISS Competition,
  a) Ensure a full testing and deployment cycle for a regular SPHERES ISS Test Session
  b) Organize a large event for hundreds of participants to visit MIT to see the test session streaming live from the ISS in an MIT Auditorium
- Answer participant queries and troubleshoot technical or game issues all through the tournament

These costs are measured by the money spent on hiring the organizers of the tournaments, support from NASA and astronauts and the man hours they put in on a per tournament basis.

3) The effort spent by all the participants within the tournaments is measured and extrapolated using survey responses (there is no monetary cost to them, since the program is free of charge – no funding/no support). At the end of the 2011 tournament, all tournament participants were invited to a program assessment survey in which one of the questions was

the number of hours the individual spent on Zero Robotics. Goodie packets containing individual participation certificates, SPHERES stickers and ZR calendars were provided as incentives to teams that had at least 5 members fill out the program evaluation survey and a complete team response survey (one per team). The average man hours per week was calculated from this data and has been discussed in Section 5.2.3.

The metrics of value delivered to STEM Education, in terms of ***benefits provided*** are:

1) Improvement of student skills in CS-STEM and other 21st century learning areas [3][50]
2) Increased inclination toward STEM fields
3) Overall program satisfaction
4) Performance in the competitions within ZR and improvement of performance through the tournament

Metrics such as student skill improvement, inclination and satisfaction are quantified using the response data from both the individual surveys sent to all participants as well as team surveys filled out by mentors for the entire team. Metrics such as performance in tournaments are quantified by the average match scores of teams in each of the competitions and compared across competitions to measure significant change. The results will be discussed in Section 5.2.

The metrics of value delivered to spaceflight software development through crowdsourcing using Zero Robotics in terms of the ***benefits provided*** were:

1) Competition and Match Scores (i.e. performance of participants) declared by the satellites at the end of each match; since match scores are designed to measure the goodness of solution to problem proposed in the competition.
2) All (not just the winning) programs submitted by the top-level participants, for a more qualitative analysis. This entails combing through programs submitted by the finalist teams to look for good pieces of software that for some reason failed to achieve perfect scores, but if improved and/or used in conjunction with other pieces of robust software could add value.

The next task was to list metrics to quantify the collaboration environments in Zero Robotics. The first two collaboration environments: in-game collaboration and alliance-based collaboration, were considered variables with binary attributes i.e. they were either implemented in a competition or not.

The third type of collaboration environment: discussion forums, has always been a part of Zero Robotics. The metric used to quantify this environment is the amount to which a team used the forums – quantified by the number of posts by that team in the discussion forums, projects shared and challenges proposed. This data is available through feedback surveys as well as website usage statistics, obtained by querying the ZR SQL database.

4.3.3. Concept of Reality

Like all research, facts have been perceived through the framework of experiments and therefore the conclusions drawn are prone to influence by the quality of the experimentation and instrumentation. The following section discusses the four possible issues that might arise – reliability, validity, representativeness and significance - and how ZR is prepared to deal with them.

4.3.3.1. Reliability

Reliability is that quality of a measurement method that suggests that the same data would have been collected each time in repeated observations of the same phenomenon. Reliability in the data collection process for ZR has been ensured by framing the survey questions in line with established question styles for educational programs. Other traditional methods of increasing reliability, such as the test-retest method and the split-half method [8], were not used because they could dissatisfy participants and overcomplicate the feedback process. Reliability in the data analysis process was ensured by applying tests for statistical significance to all the major conclusions. Note that reliability, or precision or repeatability, is different from accuracy in that accuracy is the degree of closeness of measurements of a quantity to that quantity's actual (true) value. Accuracy in social experiments is ensured by careful handling and analysis of collected data, to keep human errors as low as possible, which was done for ZR.

4.3.3.2. Validity

Validity is the measure that accurately describes the concept that it is intended to measure. While reliability asks the question “Did I do the thing right?”, validity asks, “Did I do the right thing?”.

Validity issues in experimental design arise due to internal invalidity – referring to the possibility that the conclusions drawn from experimental results may not accurately reflect what went on in the experiment – or external invalidity – referring to the possibility that the conclusions drawn from experimental results cannotbe generalized to the 'real' world. Described below are some known sources of internal and external invalidities known in social experiments [9] that may affect the proposed experiments in ways that are difficult to measure (only sources relevant to ZR have been listed). These concerns are discussed in Chapter 5, when discussing the results.

I. Internal Invalidity

- *Maturation of participants* – refers to the phenomenon that people are continuously changing. This is important because participant performance is measured at different points in time (i.e. time series analysis) and data is used to assess the effect of collaborative environments introduced at different times. Maturation can potentially be a source of distortion in the collaborative variable. A possible solution is to measure the maturation appropriately, either directly or using a control, and then mitigate its effects.
- *Effects of testing* – refers to the process where the perceived objective of the test itself influences people's behavior. In ZR 2011, the perceived nature of the tournament could have influenced the way participants played i.e. by doing what they think needs to be done. For example, some participants considered collaboration the main objective and considered any strategy aimed toward adversarial competition not in the spirit of the game or even cheating. A possible solution is to correctly and as clearly as possible communicate game and tournament intentions to the participants.
- *Statistical regression* – refers to a possible selection bias such that participants have no room to show results in the direction expected. This could be a cause of concern in ZR. For example, the game scoring may be designed in a way such that players gain maximum points even while there is still room for improvement to the algorithms; or worse, the resolution of scores is not good enough to separate the fine differences in algorithm quality. A possible solution is to carefully design and test the game several times before release. Another instance where this was observed was in the evaluation of top ranking players who had little scope of improving scores (since the

maximum was 23 points) compared to lower ranked players who had much more scope. The solution adopted was to correlate variables of interest with scores as well as relative ranks.

- *Demoralization* – refers to the possibility that feelings of deprivation in a group may cause them to give up. This is important for ZR, specifically for those student groups that still have a lot of room for improvement but drop out of the tournament when they feel that they will soon be eliminated (from 2010 feedback surveys). A possible solution is to try to give low performers a second chance and ensure maximum participation in the final rounds. While the 2011 tournament grouped high and low ranking players together to ensure diverse participation, the low ranking teams sometimes felt overshadowed by the higher ranking ones.

II. External Invalidity

- *Interaction/reactive effect of testing* - refers to a process by which individuals selected for an experiment tend to behave differently than if passively observed, not only in terms of perceived objectives (#2 of internal validity) but any other factor as well. This is not such a large concern because the program was largely outreach event with over a thousand participants and *not* conducted as an experiment, so this thesis is indeed more of an observational study.
- *Interaction of selection bias and testing* – refers to the process by which selected individuals who continuously participate in the experiments are biased due to relatively more testing. Since there had been a nationwide pilot program in 2010, there were 17 teams and many students in the 2011 tournament who had played ZR games and programmed the SPHERES before. Their responses and performance records could bias the overall conclusion. To counter this concern, one of the survey questions asked if the responder was an alumnus, and attempts were made to factor this information into the analysis.
- *Multi-treatment interference* – refers to the having the same participants in all experiments, such that their memory causes a bias. This concern is very important in the analysis of tournaments through survey responses because there was only one feedback survey for the entire tournament, where the participants were asked to rate the effects of multiple variables on multiple skills. Not only could this give rise to a

hindsight bias but also raises the risk of them making mistakes in assessment of the intensity of influences of various factors.

4.3.3.3. Representativeness and Significance

Representativeness indicates that the conclusions drawn from this thesis is generalizable enough that it fully answers the research question. Representativeness can be broken due to exceptional cases. Care has been taken to design the 2011 program so that it fits into the general patterns of crowdsourcing for cluster flight software, STEM Education and online collaboration as available in literature. Since the problem used for the crowdsourcing demonstration in 2011 was a proxy problem, there may be concerns on whether a scientific problem, baffling to space engineers, will be something students can solve. However, analogies drawn from the results as well as the student participation in a current full-fledged ZR crowdsourcing tournament (see Section 5.1 for details) dispel these concerns. The research question is certainly a very significant one as the gap analysis in Chapter 2 and the literature references in Chapter 6 indicate.

4.4. Tool and Metric Development Summary

This chapter describes the usage of the ZR web interface developed (as described in Chapter 3) to launch ZR tournaments and theZR games as well as the overall structure of the tournaments. Collaborative competition has been introduced in the ZR program starting in 2011 and the different environments of collaboration - through game design, alliances and discussion forums – have been enumerated in this chapter. The ZR web interface has been treated as an apparatus (Chapter 3) and the ZR tournaments with the ZR games as a tool (Chapter 4) to create the ZR program and learn lessons on combining crowdsourcing and STEM education into one program. The analysis of the impact of collaborative competition in ZR on crowdsourcing and STEM education has been proposed as a design of experiments framework and metrics and data sources for the analysis have been listed. Again, as explained elsewhere, since the ZR program is largely an outreach effort, the analysis is more of a quasi-experimental observational study than a full-fledged experiment.

Chapter 5 –
Analysis of Zero Robotics Tournament Results

The research hypothesis proposed in the thesis is that crowdsourcing for spaceflight software development and STEM education of students can be done through the same program. The effect of collaborative competition in the achievement of these objectives is measured and analyzed. Chapter 3 discussed the use of crowdsourcing to develop the web infrastructure for the Zero Robotics program. If ZR can achieve both crowdsourcing and education, this is a great case study of end-to-end development of a software system using crowdsourcing alone. Chapter 4 highlighted the process of conducting ZR tournaments, through which crowdsourcing and STEM education will be achieved, and the framework though which collaboration effects will be measured. It also discussed the metrics that will be used to quantify collaboration in the tournaments and the value delivered to education and cluster flight algorithms. This chapter discusses the qualitative and quantitative attributes of the metrics based on the data collected in the ZR Tournaments. It has been divided into two major sections representing the two main objectives of the research effort: Benefits to Crowdsourcing and Benefits to CS-STEM Education. The costs of running the program have been briefly discussed in Section 4.3.1. and have not been explored further in this chapter.

5.1. Benefits to Crowdsourcing Spaceflight Software

Crowdsourcing has been achieved through the ZR 2011 Tournament in the process described in Section 4.3.2.1. The primary sources of data are the scores of teams in simulation and hardware competitions which reflected their formation flight performance, the maneuvers they demonstrated and the software they submitted. It is important to note that this thesis does not demonstrate solutions to real, unsolved spaceflight problems, propose better solutions obtained through ZR crowdsourcing compared to existing literature or describe the process of technically integrating the crowdsourced modules with existing cluster flight software. There are experiments which are currently ongoing within the ZR program [95], with the primary objective of demonstrating crowdsouring and have also been briefly described in Section 5.1.3. The thesis does, however, claim that students are capable of solving hard formation flight problems, given the ZR tools and access to the SPHERES simulator, in short periods of time. It also demonstrates the process by which

difficult cluster flight problems can be 'gamed' such that students can contribute to solving them (Figure 38), game scoring designed such that students' solutions reflect the achievement of algorithm objectives and the solutions tested in simulation and on hardware to evaluate their robustness/efficiency quantitatively. The thesis proposes that the same process can be applied to real, cluster flight problems. The results highlighted in this section are important to gauge the benefits to STEM Education too, since performance of the teams in the ZR competitions was a metric for the value delivered by STEM Education, as detailed in Section 4.3.1.

5.1.1. Crowdsourcing Lessons learned from Pre-2011 Tournaments

The tournaments conducted in ZR in the years 2009 through 2010 were all primarily aimed at outreach and education with no effort to identify and 'game' a spaceflight algorithm to be solved through crowdsourcing. The games were designed such that the actions performed by the satellites to achieve the game objectives were similar to those observed in formation flight (FF). However, none of the algorithms sought through the competitions were unsolved ones and more importantly, the game scoring was not designed to finely resolve the quality of maneuvers, required to sort hundreds of submitted solutions.

For example, in the 2010 high school tournament, the game was called HelioSPHERES, and it was based on autonomous scanning and docking research. The game story read, "*Attention SPHERES flight engineers: Welcome to Falcon Solar Inc. As you may know, as the world's largest solar energy company we have undertaken many bold projects to advance the state of the art in solar power generation. Our most recent project has the potential to revolutionize energy production world-wide: the first space-based solar array. […] Our main competitor, SolTech Industries, […]has also begun construction on a space-based solar array, […] We must complete construction first to establish our rights to the technology. […] However, we have run into a problem. Just before starting its final orbital maneuver to approach the station, the launch vehicle transporting the panel experienced a catastrophic failure. […] our emergency systems jettisoned the panel before any damage could occur, but we do not have accurate knowledge of its location. We have deployed a SPHERES satellite near the last known location of the vehicle. Your mission is to configure the satellite to find the panel, dock with it, and bring it to the array. We don't know exactly where the panel is, so you will have to use the worker's radar to find the panel. This would be a relatively simple task if it weren't for one thing: SolTech Industries. We have reports that they have deployed their own satellites to the same*

vicinity to rendezvous with a vehicle launched just hours ago. They will do anything to prevent us from finishing before them, and you must make sure they do not succeed. […] we may employ a jamming signal that remotely alters another satellite's guidance and navigation system. Using this device we can push them away from the worksite. […] The fate of the company is in your hands."

Teams in the tournament thus competed as either "Falcon Solar" or "SolTech" spacecraft interchangeably. The tasks in a match included a search for a missing solar panel (a "lost in space" problem) using a scanning vector as shown in Figure 30. After the panel was found, i.e. the panel position lies within the cone of sight of the satellite, the position and orientation of the panel became known to the player. The next task was to 'dock' with it by aligning the attitude vector (Figure 30) of the SPHERE in the orientation of the panel at its position and finally complete the solar array by docking the panel to the home station using the same mechanism as the one used for the panel, with a different position and attitude. Since 2 SPHERES played the match together, the player to finish all the tasks first or finish more tasks before the time or fuel ran out won the match. Since the three tasks – 'panel finding', 'panel docking', 'station docking' - could be only performed sequentially, they represented increasing levels of objective achievement. In the event that none or one player completed all objectives, the player with higher level of achievement won the match.

The game was relevant to autonomous search and space robotic assembly research, however both have been demonstrated in simulation and on SPHERES hardware, ground and ISS, earlier [87]. The objects that the SPHERES were docking to were all virtual, and did not add much value to the existing literature, which has already documented the results of the SPHERES docking to chains of external structures such as flexible beams [96]. Moreover, the scores were able to distinguish between only three levels of achievement and the time each took, and there was no point-by-point resolution for the specific quality of algorithms. In the event of adversarial attacks where nearly no one completed the game goals, it was hard to determine which algorithms were the 'best', had they completed the goals. Therefore, while the 2010 game built on the existing SPHERES research problems, the intent was more to engage students in real-time research than to learn from their algorithmic solutions.

Since the 2010 game was adversarial in nature, and both players had access to a navigational disruptor to repel their opponent at no cost to themselves (i.e. no loss of their own game objectives),

it was observed that teams tended to beat out their opponent player in the match by pushing them off course or even out of the game volume rather than focusing on achieving the game objectives. Moreover, voluntarily or involuntarily (i.e. being repelled) leaving the game volume resulted in the dropping of the panel, which caused it to be repositioned within the volume, and the docking procedure had to be repeated to retrieve the panel. This caused even fewer teams to achieve the all game objectives, in spite of having the programming ability to do so.

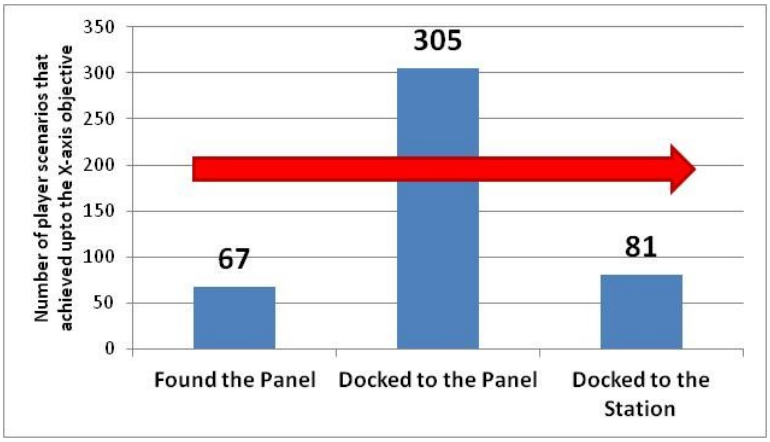


Figure 39: Histogram of the player scenarios in the RR Simulation Competition among 22 teams, that a player achieved *upto* the mentioned three levels of objectives. Since there are 2 players per match, a total of 462 player scenarios were possible. The red arrow indicates the direction of better performance.

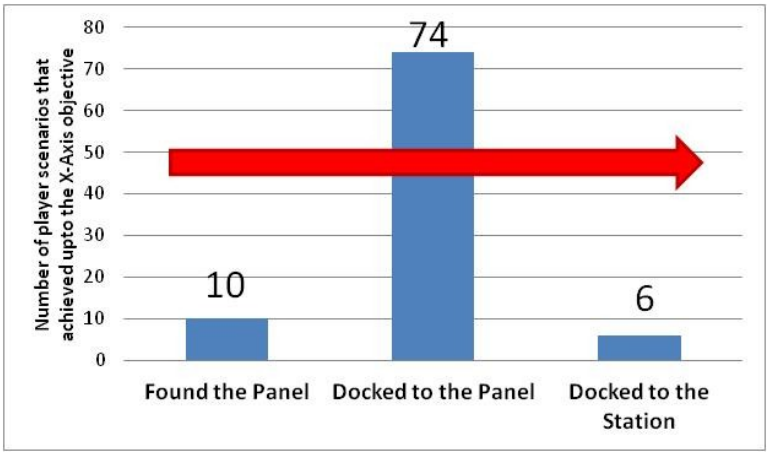


Figure 40: Histogram of the player scenarios in a RR simulation (not competition) among 10 submitted players by ISS finalists, where a player achieved *upto* the mentioned three levels of objectives. Since there are 2 players per match, a total of 90 player scenarios were possible. The red arrow indicates the direction of better performance.

Figure 39 shows the number of player scenarios in a round robin (RR) simulation competition that 22 teams played. The bars represent the total number of scenarios where a player achieved *upto* the mentioned level of objectives. Two scenarios can be possible per match since there are 2 players. Figure 40 shows the same statistic when the 10 projects submitted by 10 ISS finalist teams were simulated in an RR. In the simulation RR (Figure 39), the number of scenarios that achieved *upto* 'station docking', i.e. the highest level of achievement, was *greater* than those that achieved *upto* 'panel finding' i.e. the lowest level of achievement. This is in contrast to the ISS RR (Figure 40), where the number of scenarios that lowest objective was *greater* than those that achieved the highest. The final hardware competition aboard the ISS was a single elimination bracket where only 1 of 10 teams docked to the station i.e. only 1 scenario achieved the highest level of achievement. This reduction in the performance of teams in terms of the objective achievement was attributed to the fact that, as the tournament evolved, more teams concentrated their efforts on debilitating their opponent's ability to achieve the objectives by using attack strategies, thereby channeling their time and fuel in efforts different from achieving the objectives. The purpose of crowdsourcing solutions from dozens, hundreds or thousands of people is to identify the high performing outliers and top solutions i.e. players that 'Docked to the Station'. Reducing the number of top solutions (right tail of the histograms in Figure 39 and Figure 40) as the tournament proceeds effectively reduces the value of crowdsourcing.

An important lesson learned from the 2010 game, in spite of not being a directed crowdsourcing effort, was that in order to achieve crowdsourcing objectives for the development of cluster flight algorithms, it is best if the game is designed to allow both SPHERES to work with each other, *like a real cluster,* through direct or collaboration to truly demonstrate *formation flight*. This also helps maximize the use resources available on the ISS for a research-based test than a competitive demonstration where each SPHERE behaves independently. Not penalizing attack strategies also causes teams to get competitive in that they aim at another's loss, causing the overall gain to be reduced. In keeping with lessons from the FIRST Robotics experience [73], games require the right mix of competition and collaboration to maintain the excitement of elimination while achieving one's own objectives. Lastly, the game scoring should be designed such that the scores reflect accurately the quality of the crowdsourced solution so that there is fine and quantitative resolution between the hundreds of solutions submitted, as opposed to 3 buckets of performance in 2010.

5.1.2. Crowdsourcing in 2011

The objective of the 2011 high school tournament was also primarily STEM outreach and education, however the game was designed around a harder and more research-relevant problem than the previous years. Additionally, the game objectives were designed such that perfect scores were possible only if both SPHERES in a match collaborated to achieve the objectives together, physically and strategically as described in Section 4.2.1, so participants were incentivized to write collaborative players. The ideal flight demonstration and best scoring match would both be if SPHERES demonstrated synchronized formation flight. Game scores were designed so as to exactly prorate the quality of the solution to the complex maneuver sought. Therefore the 2011 tournament served as a good platform to evaluate the use of the ZR Program for crowdsourcing cluster flight software. All the maneuvers required in the match, such as position control, attitude control, controlled spinning and controlled rotation have been demonstrated in literature as well as the SPHERES testbed using different methods. The combined use of all the available maneuvers to solve a complex, formation flight maneuver – a proxy problem – was tested.
The results from the 2011 tournament will therefore be used to:

1. Prove the concept that crowdsourcing (demonstrated with proxy problems in 2011) for developing robust formation flight algorithms can be done using a STEM educational program, and that students were able to solve perfectly harder control problems e.g. trajectory tracking in 2011, than simply setting target position and/or velocity.
2. Analyze the distribution of scores of teams in the every competition, performance improvement with time and its dependence on the collaboration environment. The scoring of each match was prorated with fine resolution (as shown in Figure 34 and described in Section 4.2.1) such that the better the teams performed the formation flight maneuvers, the more their scores were. This allowed quantitative comparison of the algorithm and strategy quality across teams and across time.
3. Qualitatively and quantitatively gauge the spread of methodologies used to achieve the FF objectives proposed in the game on ISS flight hardware. Also, to calculate efficiency of algorithms compared to SPHERE research acceptable levels, robustness to noise on the SPHERES hardware in microgravity (from sources not modeled in the simulation e.g. from airflow) and quantitative comparison to simulation results.

4. Analyze the fuel optimality of the algorithms developed by students, contrasted against those developed in-house by MIT undergraduates
5. Document the lessons learned by introducing a hard formation flight problem in an educational tournament, to prepare for future such demonstrations; Gain insight into designing tournaments, open to all age groups and professions, purely for the development of spaceflight algorithms by crowdsourcing (Section 5.1.3).

The achievement of each of the above objectives will be discussed in the sections below.

5.1.2.1. Crowdsourcing Proof of Concept

The 2 main proxy problems to be solved by crowdsourcing, used in the 2011 game, were to write fuel-efficient algorithms for two activities as shown in Figure 32

- Spinning a SPHERE at a predefined orientation, angular velocity and position while another SPHERE revolves around it at very close proximity, without colliding.
- Revolution of a SPHERE about a fixed position spinning SPHERE, in a pre-defined plan, at a predefined velocity and within a predefined close proximity radii without colliding.

Such FF behavior is useful for close proximity inspection by an inspector satellite (the revolving one with controlled attitude) of a target satellite (the spinning one). Also, the spinning satellite may be considered analogous to a tumbling target and the revolving satellite analogous to a satellite demonstrating docking to a tumbling target [97]. The revolution problem, in particular, sought a precise trajectory tracking controller developed by high school students. The game aspect designed around this proxy problem was centered around the theme of mining virtual 'asteroids' whose position in the game volume was fixed and known to participants. Maximum points for mining could be obtained only if a team programmed the SPHERE to follow a precise and efficient trajectory around the asteroid location while the other (opponent) SPHERE spun at the asteroid location. 'Precise trajectory' meant moving at a specific angular velocity, within a specific annular ring and in a specific plane of revolution. The SPHERES were allocated a finite amount of virtual fuel, which was a predefined fraction of the real fuel, to perform all their tasks in a match, so the maneuver was additionally required to be fuel efficient. The points were prorated depending on which asteroid was mined, at what angular velocity, what orientation and whether alone or

collaboratively (Figure 34) and any integer between 0 and 23 points per match was possible. The scoring for *the first 3D simulation competition* was prorated such that the maximum score of 23 points at the end of a match was possible only if the SPHERE executed perfect attitude control to 'melt Opulens' ice' and spent the rest of the game time in perfect collaborative revolution with a spinning satellite on Opulens – the richer asteroid (Section 4.2.1).

The perfect score in a competition (not only a match) was therefore possible only if a player had:

1. A perfectly collaborating opponent to get the best out of all the available resources - Section 5.1.1
2. A perfectly optimized strategy of war-gaming
3. A perfect control algorithm for trajectory tracking of the SPHERE

War-gaming refers to a robust and autonomous decision making strategy that teams would code within their submitted player such that, during the match, their SPHERE would respond smartly to the opponent's SPHERE and achieve the game objectives, without getting in each other's way. AsteroSPHERES was designed such that every phase in the game presented the players with several choices for their course of action, all of which were not possible in the constraints of time and fuel. Participants were expected to come up with a war-gaming strategy that best supported the technical capabilities of their programmed SPHERE and expectations of their opponents' behavior – communicated or otherwise. For example, the first phase had a choice of 4 items, the second phase a choice of 2 mining behaviors and 2 asteroids and the third phase a choice of mining versus racing to any of the two stations, apart from making autonomous choices to react to dynamic behavior. Moreover, unequal choices were provided in the game to make decisions harder. For example, although it was obvious that *overall* collaboration was more favorable than not, the scores for spinning were *lower*[11] than that for revolving to make it a hard decision to resort to spinning in a match the sake of quiet cooperation.

War-gaming was not only called for but also possible to achieve. Since the teams were capable of online communication and the SPHERES were capable of communication within a match, they

[11] Spinning was worth lower points because it was considered an easier and less fuel consuming behavior than revolving (apart from the intent of making the decision to choose a mining behavior asymmetric and increasing the war-gaming, strategic approaches to play the game)

could make real-time decisions in a match based on pre-decided criteria. For example, the teams could unanimously decide to let the player with the most number of ice hits on Opulens and/or furthest distance from Opulens to revolve and then program their player to play a game in keeping with this decision. The 2011 tournament was aimed to indicate that high school students were capable of all three, and therefore capable of being very good crowds to solve hard strategic and FF problems.

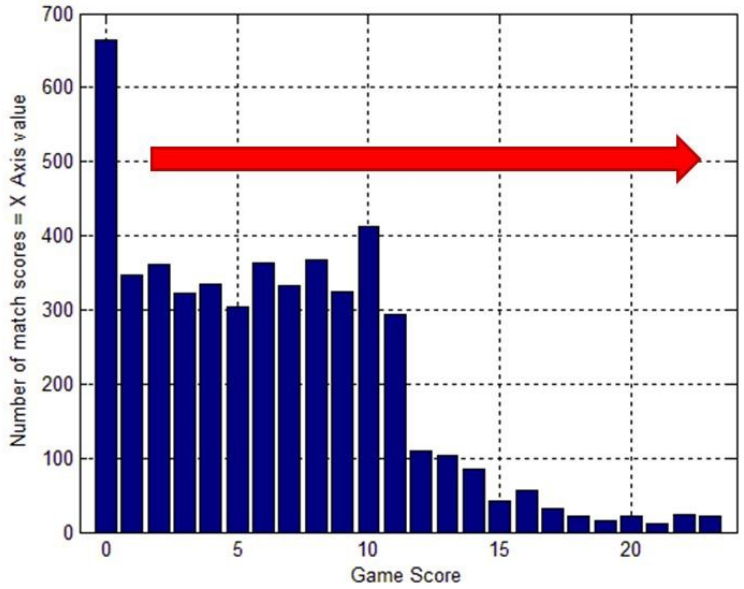


Figure 41: Histogram of the number of times a particular game/match score was achieved in a game scenario for the first 3D competition. There were 4095 RR matches played among 91 players (submissions) thus 8190 scores. The maximum score was 23 and the minimum is 0. The red arrow indicates the direction of better performance.

The students were able to come up with efficient trajectory tracking algorithms and war-gaming strategies for collaboration. They maintained large spreadsheets of calculations, as reported in the post-tournament surveys, to back up their decisions. They even pointed out flaws in the game inconsistencies during the tournament, that the ZR Team corrected when moving between one competition to the next. The teams thus demonstrated analytical and strategic abilities as well as collective intelligence. As shown in Figure 36, AsteroSPHERES2D was released at the tournament kickoff and teams were required to submit their projects for the 2D competition, 3 weeks after the release of the game. AstroSPHERES3D was released after the 2D competition was evaluated, and teams submitted their projects for the 3D competition #1, 3 weeks after its release. 91 projects

(from 91 teams since only one project per team was allowed) were received for the 2D competition and 88 projects for the 3D competition #1. Among the 88 teams that submitted for 3D, 4 unique teams in 10 different matches were able to achieve the perfect score of 23 points – within 3 weeks of the game's release and 6 weeks of the IDE's release. The histogram of scores of all the matches in the competition, i.e. each team's project against every other team's project, is shown in Figure 41. These high performing outliers (i.e. the rightmost tail of the histogram distribution) justify the value to crowdsourcing. Also note that there were 23 levels of objective achievement in 2011 as opposed to three levels in 2010 (Section 5.1.1). The forthcoming sections describe further improvement of results in the subsequent competitions, discuss the effect of collaboration, and provide detailed analysis of the ISS testing of the submitted programs.

5.1.2.2. Effect of Collaboration

This section revisits the effects of the collaboration environments mentioned in Section 4.3.1 and on Table 4 on the crowdsourcing objectives of the 2011 tournament. The proxy problem in 2011 was to program a formation flight maneuver such that one SPHERE spun at a fixed position (asteroid) and orientation while the other SPHERE revolved around it on a fixed plane. This maneuver was only achievable if *both* SPHERES, which in a match would be controlled by opponent teams, collaborated to achieve the respective objectives without any misunderstandings. Therefore, even amidst a match within a competition, the game design was able to optimize the resources and time available on the ISS to demonstrate a useful formation flight maneuver, thus the importance of collaboration environment #1 – in game collaboration.

Collaboration within Discussion Forums

The AsteroSPHERES game API library contained functions that users could use within their program to send and receive limited communication messages from the opponent SPHERE in a match. The message could only be an unsigned short value. Therefore teams would have to set up a protocol to assign understandable meanings to the available integers (1 through 655535 since 0 was the default state of the communication packet) so that their players/SPHERES could 'talk' to each other in a match. The game manual suggested 8 such integers with their associated meanings and we

called them standard messages. Teams could program war-gaming strategies to cooperate with opponent teams in a match by either programming their SPHERES to send standard messages or come up with their own protocol offline and program their SPHERES to follow it. See Section 3.1. for SPHERES communication details and Appendix B for implementation of the game communication in the Game Code.

Teams used the discussion forums on the website, i.e. collaboration environment #3 of Table 4, to come up with a global communication protocol beyond the standard messages we published to cooperate more efficiently and play the game better. A qualitative analysis of the proceedings on the forums showed that there were three main protocols that emerged during the course of the tournament, with varying levels of success:

- ***Y0b0tics! Protocol*** [12]– The most popular protocol, which called for SPHERE1 to perform revolution as the mining maneuver and SPHERE2 to perform spinning, both on Opulens. Over a very large number of matches, each team would be assumed to play as SPHERE1 and SPHERE2 nearly equal number of times and therefore average out the difference in the spin and revolve scores, which were different per match. The protocol set aside one message, by broadcasting which the SPHERE declared to its opponent that it was following this protocol, so that the opponent could respond accordingly. The protocol was very well received and worked well with hundreds of matches per player. It faced some skepticism later in the tournament when the number of players and therefore matches were reduced. The game rules assigned SPHERE numbers to players in a match randomly, and by following the y0b0tics! Protocol, the reduced number of matches favored the teams that randomly were assigned to SPHERE 2 and therefore got to revolve and get more points. Team y0b0tics!, the team that invented and popularized this protocol, was awarded a special recognition prize for coming up with it.
- ***Delta Protocol*** [13]– An extensive communication protocol where Team Delta published a list of integers and their associated meanings (e.g. 7 = "I will pick up laser 2 and spin") in an

[12] Original names used with permission from Richard Kopelow, the primary mentor of Team y0b0tics! From Montclair High School, NJ, the team that invented and popularized this protocol

[13] Original names used with permission from Salvatore Lorenzen, the primary mentor of Team Delta from Post Falls High School, ID, the team that invented and popularized this protocol. The protocol concept was led by Brett Menzies.

online document. Players following this protocol would broadcast the appropriate integer to their opponent player. Given the rising popularity of the y0b0tics! Protocol, this protocol was modified to be compatible with it.

- ***Secret Alliances*** – A secret group of alliances that contacted each other by email covertly a week before the 3D semi-final competition to boycott any player that broadcasted the y0b0tics! Protocol signal. While this protocol received loyalists who committed to be a part of the boycott, most alliances that joined the protocol ended up being eliminated in the semi finals (including the one that invented it).

There were other protocols that were suggested and emerged on the forums during the tournament, but the forum posts and team behavior showed the above three protocols to be the most popular. Students were collectively able to decide a protocol using a collaboration environment and results in the subsequent sections will indicate its usefulness in achieving the maximum scores possible in a tournament.

An important *qualitative* comparison here is the summer program of 2011, wherein 5 middle school programs from the greater Boston area played the game AsteroSPHERES first on simulation and then on the ISS hardware. This is not an experimental comparison between MS and HS scores because there are several sources of invalidity - the age of the subjects was different, preparation and programming time was different (3 months for HS vs. 4 weeks for MS) and the game API library for middle school students was far more exhaustive compared to HS (Appendix A). Knowing them to be non-equivalent control groups (Section 4.3.1.3), it is important to note that the quality of in-game collaboration among the HS students was much higher due to forum-based collaboration. The MS students did not have access to any forums to communicate with each other online and decide on a protocol the way the HS students did. They relied on the standard messages suggested in the manual, which were not enough to make an autonomous decision in case of conflict. As a result, several projects played on the ISS made both SPHERES in the same match simultaneously spin or simultaneously revolve, or even activate collision avoidance since they got in one another's way. Not only did this lower their points due to lack of collaboration, but also cost the teams more virtual fuel than they had budgeted for.

Figure 42 shows a plot of the satellite telemetric state of 2 SPHERES playing a match aboard the ISS. The players belong to the top 2 teams in the MS competition - they stood first and second in the competition. Note the squiggly behavior of the satellites as both try to revolve and, therefore, activate collision avoidance several times therefore scoring only 7 and 11 points each. The existence of forums in the HS tournament apparently brought out the collaborative nature of the game (Compare Figure 42 to Figure 46 and other associated metrics of collaborative success described later for the HS tournament). Therefore, collaboration environments #1 and #3 are incomplete without each other and together produce far better results than without one or the other.

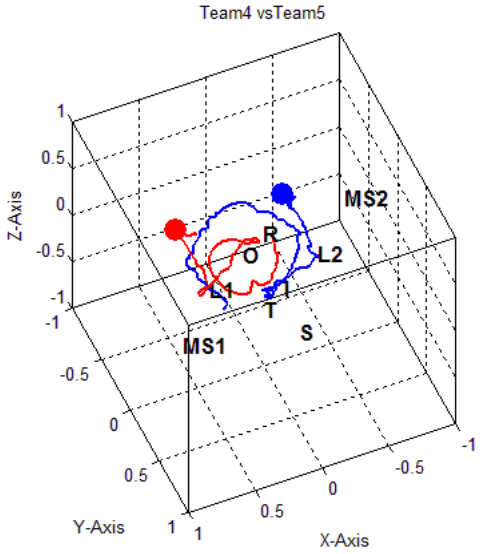


Figure 42: An ISS match between the top 2 middle school programs in the summer version of AsteroSPHERES (same mining phase but different item collection phase). The red and blue trajectories are the paths taken by the 2 SPHERES in the match, as parsed from satellite telemetry data in a post processing analysis. The red and blue spheres indicate the initial positioning of the two satellites. The alphabet indicates the position of the items, asteroids and mining stations in the game volume (L1=Laser1, L2=Laser2, R=Repulsor, T=Tractor, O=Opulens, I=Indigens, MS1=Mining Station 1, MS2=Mining Station 2)

Collaboration within Alliances

The results from the team-based and alliance-based results from the tournament indicate that alliances of teams showed higher average scores than individual teams, demonstrating the importance of collaboration environment #2. As seen in Figure 43, the mean score among all the

teams has improved significantly after grouping the teams as alliances. To mitigate the effect of selection bias on experimental validity, only teams that participated as alliances in the 3D #2 Competition were chosen for analysis in the 3D #1. See Figure 36 for the sequence of competitions. The mean score of the team competition (3D#1) was 9.1 (standard deviation of 5.6) and the mean score of the alliance competition (3D#2) was 14.6 (standard deviation of 4.6).

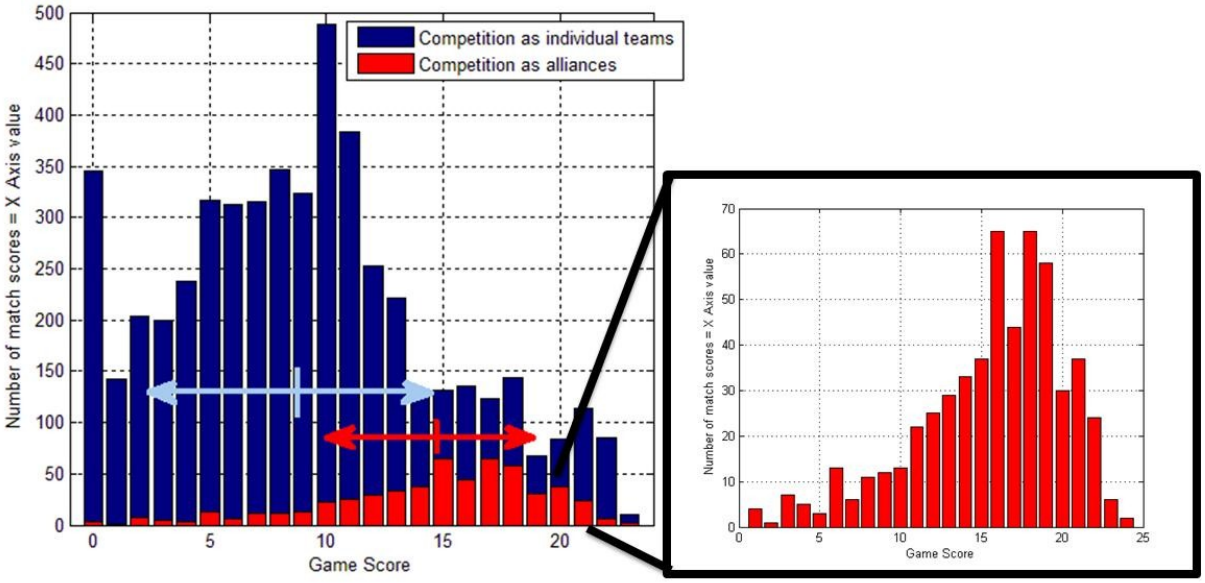


Figure 43: Comparison of score distributions with and without alliance-based collaboration. The blue bars are the scores of teams in 3D#1. The red bars are the scores for alliances in a separate competition, 3D#2. The blue histogram contains 4095 round robin match scores, played between every pair of the 91 teams. The 72 highest teams were formed into 24 alliances, of 3 teams each. Thus, there were 8190 (blue) and 276 (red) matches in 3D#2. The mean and standard deviation of each set is shown in the figure

The mean of the alliance scores is more than one standard deviation greater than the mean of the team scores. However, the scores are not normally distributed (by the Kolmogorov Smirnov test), hence a t-test could not be used to calculate the differences. The interpretation of this difference in scores is further complicated by possible learning over the three week interval as well as minor modifications in game rules between the competitions. For instance, while it was entirely possible to have a perfect score in the first competition by programming a perfectly collaborating, strategic revolve maneuver around Opulens, getting a perfect score in the second competition additionally required a perfectly timed trajectory and a perfect maneuver to dock to the mining station.

The score distributions of the 2D and 3D #1 Competitions were compared to find the effects of the learning period and game rule changes. Both competitions had a 3 week period of preparation/programming (schedule in Figure 36), team participation and modification of game rules. While the competitions received 88 and 91 submissions respectively, only the 70 teams that participated in 3D #1 were chosen for analysis in 2D. The mean 2D score was 6.2, standard deviation 4.78, and the mean 3D score was 7.83, standard deviation 5.6 (Figure 44). From the figure, it is easy to visually interpret that the improvement in the mean between 2D and 3D#1 is far less than the improvement in mean from 3D #1 to 3D #2. It is therefore reasonable to conclude that a larger share of improvement in game scores when alliances were introduced was due to the existence of the alliance variable rather than the learning and game rule modification variables. This conclusion, however, assumes that the combined effect of game rule modifications and participant maturation between the two sets of competitions is equivalent. This assumption is cannot be verified because neither set is quantifiable and both are unrelated changes. No other control was available for this observational study.

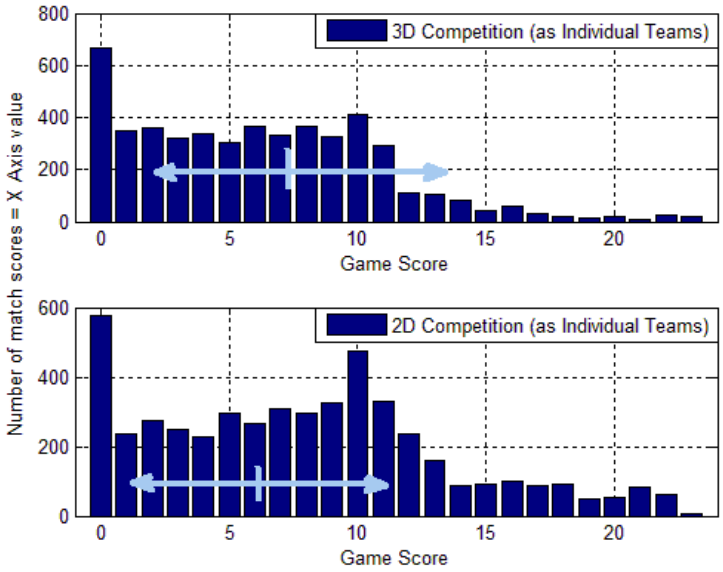


Figure 44: Comparison of the 3D #1 with the 2D scores (both played as teams). The 3D #1 competition is the same as that shown in the left panel of Figure 43, but only those (70) teams that played both 2D and 3D#1 were chosen for analysis. The mean and standard deviation of each set is shown in the figure

While alliances apparently affected the overall scoring of participants, as shown above, it was seen that it showed varying effects on the number of perfect solutions obtained i.e. the right tail of the histograms in Figure 43 and Figure 44. Table 5 shows the number of unique 23-scorers in the 2D competition, 3D #1 and 3D #2 and the number of matches they achieved the perfect score of 23 - normalized. The decrease in the number of unique players with the sequence of competitions indicates *no* change from 3D#2 to 3D#1, *over* the 3D#1 minus 2D control (Equation 2 where metric = number of unique players). However, the normalized number of matches that achieved a perfect score (as the metric in Equation 2) showed an increasing trend with the introduction of alliances. The decrease in the metric between individual competitions seems to indicate that the game changes were harder than what the participants could pick up in a three week learning period.

$$changeDueToAlliances = [metric(3D\#2) - metric(3D\#1)] - [metric(3D\#1) - metric(2D)]$$

Equation 2

Competition	Number of Unique perfect players	Normalized number of matches where the perfect score was accomplished
2D as Teams	6	21/87 = 0.27
3D#1 as Teams	4	10/91 = 0.11
3D#2 as Alliances	2	2/23 = 0.09

Table 5: Table comparing the perfect solutions obtained through 3 simulation competitions. The number of matches in the third column has been normalized by the number of matches each player played in the competition. For example, there were 88 submissions of projects by 88 teams in the 2D RR competition, so the number of matches each played was 87.

Note that the decrease in the number of unique perfect players from 3D#1 to 3D#1 could not have been because teams that made perfect players came together as an alliance because the alliance forming process used a tier-based system. No top performing teams could have joined together and

all could continue to further their own strategy. The tier-based alliance selection process therefore ensured that no perfect solutions could be eliminated in the process due to dilution with each other.

5.1.2.3. ISS Hardware Demonstration

The final competition of any ZR tournament is always the ISS finals. Since the ISS projects have passed a sequence of simulation elimination rounds, they are the best of players in the tournament i.e. the rightmost tail of the histogram distribution of scores (all of Figure 39 through Figure 44). The best user algorithms developed through online crowdsourcing are therefore finally tested on real satellites to select the most robust and efficient of them. The projects submitted by the finalist teams or alliances are uploaded on the SPHERES satellites in the ISS and the matches are played in microgravity, under the supervision of astronauts. Operations on the ISS have been described in detail in Appendix D. Results shown in this section falls in the category of "*One-shot case studies*" in quasi-experimental analysis i.e. comparison a single experimental run with a well-established standard [8]. This method is used to analyze and interpret data what is very difficult to get and ISS test session data fit the requirement, due to the hundreds of thousands of dollars of resources required per hour of operations.

The projects submitted for the ISS competition in 2011 were qualitatively analyzed to evaluate the strategies used by different teams to accomplish game objectives, and a full double round robin (RR) batch simulation using the projects was run internally at MIT. The double RR had 132 matches for the 12 submissions, since the players played a match each as SPHERE1 and SPHERE2, unlike all the formal simulation competitions where the SPHERE number allocation by player was random. The highest score achieved in the RR was 13.1 with a standard deviation of 4.85, which is lower than the average scores in the previous competition (2D, 3D #1, 3D#2 as shown in Figure 43 and Figure 44). Also, no perfect scores of 23 were observed, in keeping with the decreasing trend of unique players shown in Table 5. The maximum score was in the RR batch simulation was 22. Since the 2011 game scoring was prorated with respect to the different objectives and how well they were achieved, it was possible to break down the scores to analyze why the performance dropped. The match scores were broken down by the 2 major game objectives for the finals – mining and station

racing. The finals competition had increased weights for the race component compared to previous competitions. Qualitative analysis of project strategies showed that:

- Given the score distribution in the final game, a perfect miner could score anything between 16 to 21 points (depending on the strategy, e.g. revolving a few points higher than spinning, losing an attempted race, etc.). The perfect score of 23 was possible only by a perfect revolver and racer while 22 was possible only by a perfect spinner and racer. The total match score alone does not indicate errors in the mining maneuver because players could have chose not to race (strategy error) or attempted the race and failed (trajectory following implementation error or collaborative racing error)
- The lack of perfect 23 scores was due to no pair of players being able to perform a perfectly collaborative, trajectory-optimized race. However, these players were able to track perfect spinning and revolution trajectories i.e. crowdsourcing success of mining algorithm objectives
- All teams followed the y0b0tics! Mining Protocol i.e. if they played as SPHERE1 they performed revolution mining on Opulens and if they played as SPHERE2, they preformed spinning mining on Opulens i.e. crowdsourcing success of collaborative mining objective
- Players that won the game after the low-scoring spinning operation did so because they won the race to the mining station. This did not prove to be unfair to some teams since in the ISS RR bracket, since each player got to play once on each satellite in their 2 respective matches per RR (explained later in this section).
- There was one team that entirely went against the collaborative concept of the game and instead, used the disruptor to push players off course to win its matches. This team emerged as the lowest in the double RR, proving the success of the AsteroSPHERES scoring system in differentiating between an algorithmically useful player vs. a disruptive player.

Analysis of the batch simulation was useful to predict what would potentially happen in the ISS final competition. The final competition itself was a hardware-based competition. The submitted programs from the alliances were packed with the game code and SPHERES embedded system code, and converted into an executable. This executable formed part of the test session program (see Appendix D) and was sent to the ISS to be uploaded and played on the SPHERES testbed inside the ISS. ISS astronauts assisted in setting up the tests, deploying the satellites for the test, recording

the test result numbers as automatically reported by the satellite, maintaining the resources such as batteries and fuel in the satellites and calling down/commentating on the match results on the live link to the MIT auditorium where the competition was being aired live. ISS operations have been explained in Section 4.1.4.3 and Appendix D.

The US and EU champions in 2011 were determined through separate ISS competitions, each competition being a series of RR brackets because ISS time is precious, and there was not enough to conduct a full RR competition. Each RR bracket comprised of 3 players each, therefore 3 races, and the winner of the bracket was the one with the most total points in their two races. The European competition consisted of one RR bracket, with a total of 3 players. The US competition required four RR brackets to declare a champion out of 9 players. The 9 players were divided into 3 groups of 3 for the first 3 round robins. The winner of each initial RR bracket competed in a final "championship" round robin bracket. "Test Result" numbers were called down by the astronaut as soon as possible at the end of each match and the MIT ZR team calculated the score for each player for that match. MIT maintained the tally of all results and determined the champion (one for Europe and one for the USA) during the competition. Each "Player" in the competition represented an "Alliance" of three teams. Therefore, there were a total of 9 European teams (in 3 Alliances) and 27 USA teams (in 9 Alliances) represented in the finals.

Each match started with the primary and secondary satellites positioned in the same location, as shown in Figure 45. The items in the game were all virtual as mentioned (Section 4.2.1). Satellites moved to the items' known locations to perform game-related activities (Section 4.2.1). The asteroid axis orientation for AsteroSPHERES was a random vector, but was fixed for all the matches in the competition. It was available to players by calling an API function (see Appendix A) within their computer program. For the purpose of this competition, the axis of both the asteroids was oriented to [0.6820f; 0.7248f; 0.0975f], and the test results showed a lot of skewed spinning and revolving.

The satellite telemetry and state of health packets, as returned to the SSC (the JEM module laptop used to communicate with the SPHERES during a test session) have been parsed, and the real time positions of the satellites, both primary in red and secondary in blue, in 4 of the finals matches have been plotted in Figure 46. Note that the red SPHERE, which is SPHERE1, always revolves around Opulens and the blue SPHERE always spins on Opulens, in keeping with the y0b0tics! protocol.

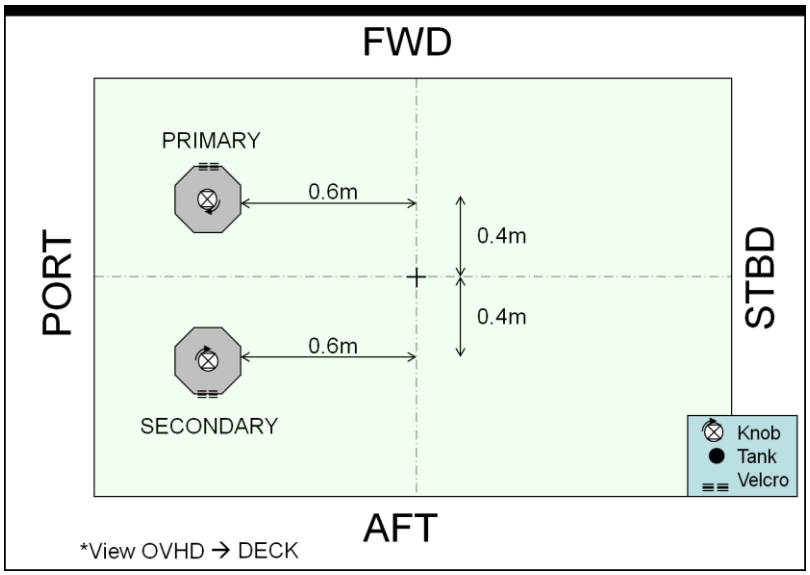


Figure 45: Initial positioning of the 2 SPHERES in each match, as submitted to the ISS astronauts as part of the Test Session files to help them deploy the SPHERES correctly. The view is as the astronaut would see the positioning if he were looking from the overhead to deck of the JEM module where the ISS competition was held. The co are in the ISS frame: port to starboard and forward to aft in the JEM module.

The 4 different strategies seen on the ISS competition for matches that completed successfully are shown in the Figure 46:

1. Spinning and revolving with racing **(top left: Team 5 vs. Team 7)** – Ended the highest score of 21 for the revolver and analysis of telemetric data indicates that both SPHERES docked successfully to the station. The lack of a perfect score is due to imperfect trajectory following, but perfect collaboration and strategy. This strategy, if implemented perfectly, would get 23 points for the revolver.
2. Spinning and revolving without racing **(top right: Team 3 vs. Team 1)** – Ended with a score of 19 for the revolver and 16 for the spinner, and clearly from the figure none tried to race to the station. In such a strategy, the maximum score possible would have been 21 and 18 respectively, which indicates an imperfect mining operation.

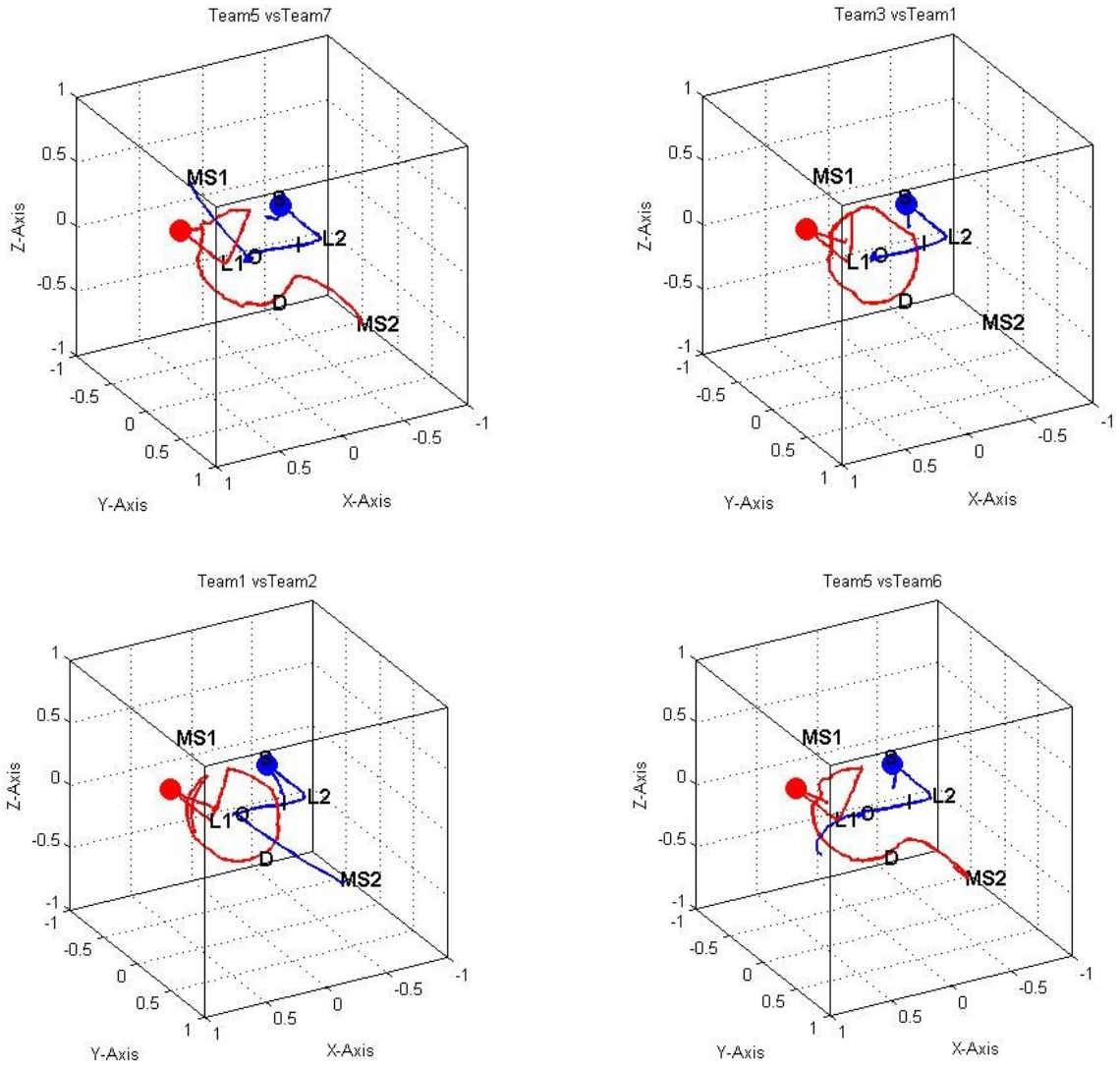


Figure 46: Trajectories of the PRIMARY (red) and SECONDARY (blue) for 4 matches in the US competition of the test session that demonstrate unique strategies of playing the AsteroSPHERES match. The red and blue spheres indicate the initial positioning of the two satellites. The tiny squiggles around them are due to the initial 30 seconds when the satellites position themselves autonomously (Appendix D) before the match begins and players' program control takes over. The alphabet indicates the position of the items, asteroids and mining stations in the game volume (L1=Laser1, L2=Laser2, R=Repulsor, T=Tractor, O=Opulens, I=Indigens, MS1=Mining Station 1, MS2=Mining Station 2). The game volume was enclosed in a cuboid of 2m X 1.7m X 1.7m length within the JEM module. These are only a subset of matches played on the ISS.

3. Spinning and revolving with only the spinner racing **(bottom left: Team 1 vs. Team 2)** – Ended with a score of 16 for the revolver and 18 for the spinner. Analysis of telemetric data indicates that the spinning SPHERE docked successfully to the station and hence won the match. The maximum points for such a scenario would be 17 and 20 respectively; however, the spinner left for the station a bit too early to gather enough mining points through its perfect mining algorithm and the revolver therefore missed out on collaboration points in spite of also having a perfect mining algorithm.
4. Spinning and revolving with only the revolver racing **(bottom right: Team 5 vs. Team 6) -** Ended with a score of 17 for the revolver and 12 for the spinner. Analysis of telemetric data indicates that the revolving SPHERE docked successfully to the station. Neither had a perfect trajectory-following maneuver, hence they fell short of the maximum points possible in this strategy.

To analyze the efficiency of the trajectory-tracking algorithms for revolution and spinning about the virtual asteroids on SPHERES ISS hardware, the satellite telemetry of SPHERES states in the mining phases of each match were processed. From the simulation analysis and the trajectories on the ISS (Figure 46), we know that all the players mined the stronger asteroid, Opulens, which can only be mined after the ice sheet has been melted using the virtual laser. The effective mining phase is therefore at least 10-20 seconds into Phase 2 i.e. *at least* 70 seconds after the start of the game. The mining phase may last for as long as the players' strategies permit, i.e. either till the end or until the SPHERES race to the mining stations.

Figure 47 shows angular velocity of revolution maneuver of a SPHERE (for this competition, by protocol, always SPHERE1) around the asteroid location while there was a spinning SPHERE at the asteroid location. The instantaneous angular velocity of the SPHERE about the asteroid normal – 'angVel' in Equation 3 – were calculated using SPHERE1's position and velocity at every time step (200 ms). This process was repeated for all the successful matches on the ISS and all curves were plotted together in Figure 47.

$$angVel = |< \omega, \hat{n} >| * 180/pi$$

Equation 3

$$\omega = \frac{[\hat{r}, vel]}{|r|}$$

Equation 4

$$\hat{r} = \frac{pos - asteroid}{\|pos - asteroid\|}$$

Equation 5

Where:

pos = instantaneous position of the SPHERE

vel = instantaneous velocity of the SPHERE

$\hat{n}$ = unit normal of the asteroid axis

|| || = norm of a vector

<,> = dot product of the vectors enclosed

[,] = cross product of the vectors enclosed

ω = angular velocity of the SPHERE about the asteroid location

r = radius vector from the SPHERE to the asteroid location.

Maximum points per second were awarded if *angVel* was 4 degrees per second, prorated as a linear ramp from 0 to 8 degrees per second, *and* if the revolving satellite was within 10 cm to 40 cm of the asteroid center (Section 4.3.1). In Figure 47, it can be seen that in the main mining phase in almost ALL the matches, the SPHERE is indeed correctly positioned (red, not pink) and revolving correctly (close to black arrow). The one match where the player on the SPHERE did not perform well has been indicated with an arrow. The corresponding trajectory figure is in the inset, and the team that controlled the revolving satellite mentioned. Team 6 got 9 points for this match, the lowest in the competition among successful tests, but the *only* one where the revolve maneuver was not near resonance. After 140 seconds of game time, Figure 47 shows that the angular velocities start dropping off and satellites start leaving the revolution radius (specifically marked in magenta). This is a transition phase where some players chose to stay revolving while others began to leave for the mining station.

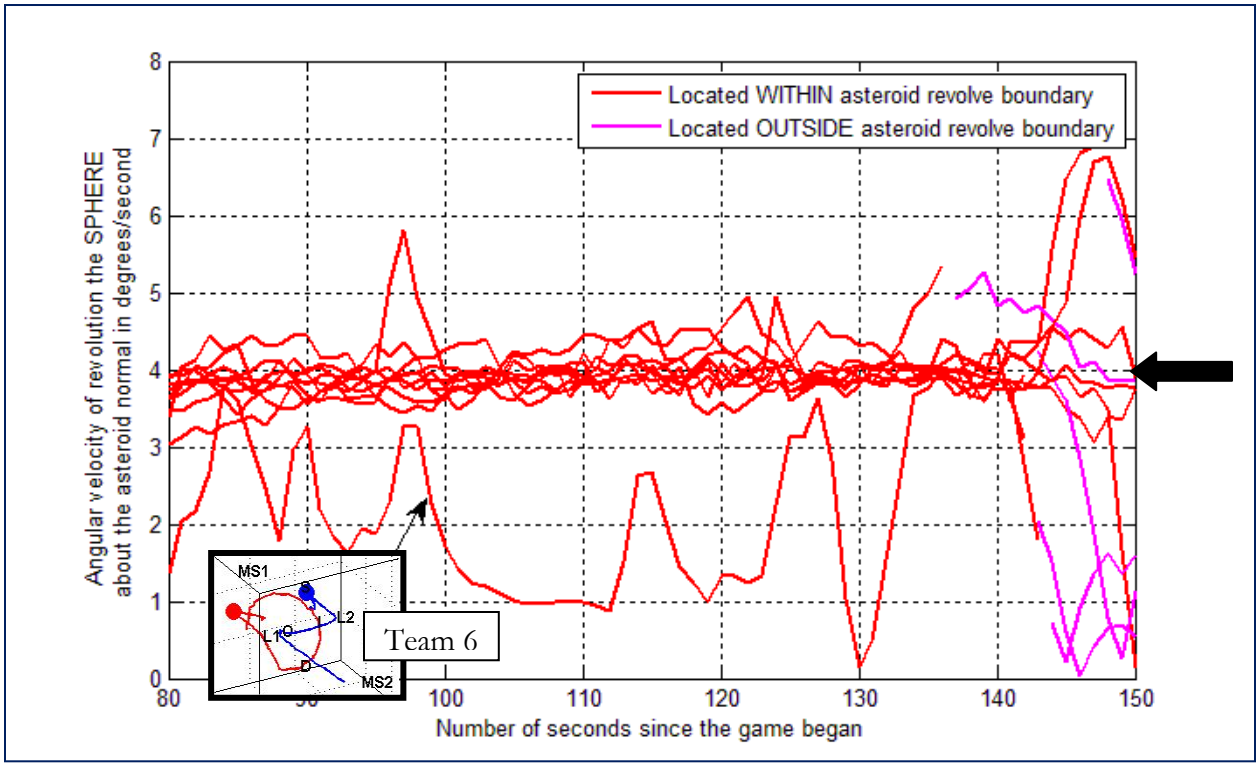


Figure 47: Plot of the main mining phase (80 to 140 seconds after the start of the match) behavior of SPHERE1 over all the ISS matches, in terms of the angular velocity at which the sphere revolved around the virtual asteroid on a plane perpendicular to its normal. SPHERE1 always revolved around the virtual asteroid Opulens. The resonance angular velocity, for which maximum points were awarded per second, was 4 degrees/second, marked with a thick black arrow. The inset figure indicates the match corresponding to the revolve behavior shown by the arrow. The plot color is the same as SPHERE1's trajectory (revolving behavior) in Figure 46 and the inset. The magenta sections indicate the angular velocity when the SPHERE was revolving, but out of the annular shell of point accumulation (outside 10cm-40 cm of asteroid location)

The proxy problem of precise revolution of a SPHERE about a spinning one, gamed in the form of mining asteroids, therefore yielded very robust algorithms from high school students that could perform efficiently even in microgravity – demonstrating the value of crowdsourcing through ZR. Equation 3, Equation 4 and Equation 5 are the same equations used to determine the dynamic scores of the satellites due to revolving from their state vector telemetry during the mining phase –

showing the efficiency of the game scoring mechanism to determine the 'best' crowdsourced solution.

Figure 48 shows the spin velocity of the spinning maneuver of a SPHERE (for this competition, by protocol, always SPHERE2) at the asteroid location while there was a SPHERE revolving around it. The instantaneous spin velocity of the SPHERE about the asteroid normal – 'spinVel' (Equation 6) – is calculated from the attitude rate and quaternion of SPHERE2 at every time step (200 ms). This process was repeated for all the successful matches on the ISS and all curves plotted in blue in Figure 48.

$$spinVel = |< \bar{\alpha}, \hat{n} >| * 180/pi$$

Equation 6

$$\bar{\alpha} = rotMat * \alpha$$

Equation 7

$$\begin{aligned}
rotMat[1,1] &= q4*q4 + q1*q1 - q2*q2 - q3*q3 \\
rotMat[1,2] &= 2*(q1*q2 - q3*q4) \\
rotMat[1,3] &= 2*(q1*q3 + q2*q4) \\
rotMat[2,1] &= 2*(q1*q2 + q3*q4) \\
rotMat[2,2] &= q4*q4 - q1*q1 + q2*q2 - q3*q3 \\
rotMat[2,3] &= 2*(q2*q3 - q1*q4) \\
rotMat[3,1] &= 2*(q1*q3 - q2*q4) \\
rotMat[3,2] &= 2*(q2*q3 + q1*q4) \\
rotMat[3,3] &= q4*q4 - q1*q1 - q2*q2 + q3*q3
\end{aligned}$$

Equation 8

Where:

α = instantaneous attitude rate of the SPHERE in its body coordinates

$\bar{\alpha}$ = instantaneous attitude rate of the SPHERE in global/ISS coordinates

[q1, q2, q3, q4] = instantaneous quaternion of the SPHERE

$\hat{n}$ = unit normal of the asteroid axis

|| || = norm of a vector

<,> = dot product of the vectors enclosed

[,] = cross product of the vectors enclosed

$rotMat$ = rotation matrix to transform SPHERE attitude from local to global coordinates using the instantaneous quaternion

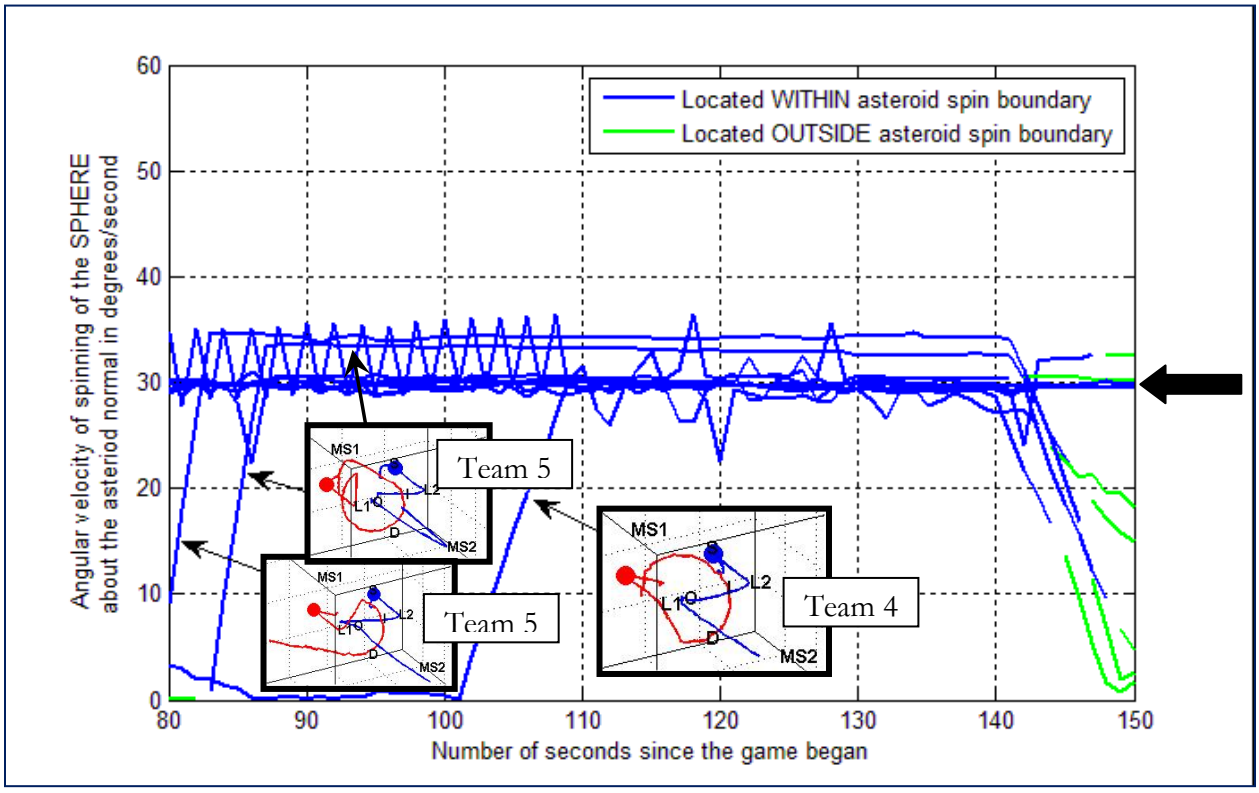


Figure 48: Plot of the main mining phase (80 to 140 seconds after the start of the match) behavior of SPHERE2 over all the ISS matches, in terms of the spin angular velocity at which the sphere spins about any body axis aligned with the asteroid normal. SPHERE1 always spun around the virtual asteroid Opulens. The resonance angular velocity, for which maximum points were awarded per second, was 30 degrees/second, marked with a thick black arrow. The inset figures indicates the match corresponding to the spin behaviors shown by the arrow. The plot color (blue) is the same as SPHERE1's trajectory (spinning behavior) in Figure 46 and the inset. The green sections indicate the angular velocity when the SPHERE was revolving, but out of the zone of point accumulation (outside 5 cm of asteroid location)

To be awarded maximum points, the spinning SPHERE had to spin at a spin angular velocity (*spinVel*) of 30 degrees/second. For all other values between 0 and 60 degrees/second, the score was

linearly prorated (Section 4.2.1). To score, the spinning SPHERE was also required to be positioned within 5 cm of the asteroid location. Figure 48 shows that the spin velocity curves are bunched up at the resonance spin velocity of 30 degrees/second (black arrow) for a significant amount of the mining time period. After 140 seconds, some players continue to spin while others drop off their spin and move away toward the mining station – green curves instead of blue - as they leave the asteroid mining zone. Three players started their spin maneuver late into the mining phase in spite of being correctly positioned (since the curves are all blue at time < 100s). The trajectories of the corresponding matches and the team that controlled the spinning satellite have been shown in the inset. Team 5 scored 8 and 9 points respectively in the two different matches shown. Team 4 scored 6 points in the match shown. This match (Team 6 vs. Team 4) was the lowest scoring match in the competition among all successful tests, since the revolving SPHERE, controlled by Team 6, also did not do well (Figure 47). However, the match was certainly an outlier among the 12 successful matches. Team 5 lost further points since they controlled their spin velocity, very accurately, at an angular rate ~ 4 degrees/sec higher than the resonance velocity. Team 5 reported later that this was due to a bug in their project, and evidently an outlier among all teams.

The proxy problem of spinning a SPHERE at an exact position, orientation and angular velocity while another SPHERE revolved around it, therefore yielded efficient solutions that performed well on space hardware too. This, again, demonstrate the value of crowdsourcing through ZR. Like the revolution case, Equation 6, Equation 7 and Equation 8 are the same equations used to determine the dynamic scores of the satellites due to spinning from their state vector telemetry during the mining phase – showing the efficiency of the game scoring mechanism to determine the 'best' crowdsourced solution.

Efficiency of both mining maneuvers, revolution and spinning, has been calculated as the percentage of time that the players spent within acceptable error levels of angular and spin velocity respectively, for each test/match of the ISS test session (Equation 9). For every test, only the time between 80 – 140 seconds since the match started is used to calculate efficiency, since that period is the main mining phase as described earlier, i.e. the summations in Equation 9 are for tStep ϵ [80, 140]. The acceptable error level for angular velocity (Equation 3) is assumed to be 1 cm/s about Opulens' location. This value is chosen because SPHERES velocity control within 1 cm/s is defined as a successful test within SPHERES research test sessions. Note that the same value is used as the

acceptable velocity error picking up items and docking to the stations. Similarly, the acceptable error level for spin velocity (Equation 6) is assumed to be 6 degrees/s, a typical value for a attitude rate control for a successful SPHERES research test. Note that the SPHERES acceptable attitude cone (Figure 30) for usage of items is also 2*6 = 12 degrees for the same reason.

$$efficiency(test) = \frac{\sum_{tStep}[resonanceVel - errVel \leq ang/spinVel(tStep) \leq resonanceVel + errVel]}{\sum_{tStep} 1}$$

Equation 9

The average efficiency of revolving and spinning players over the main mining phase of all successful ISS tests is 93.5% and 91.8% respectively when acceptable error levels are used (triangles in Figure 49). Seven of ten tests show 100% efficiency in revolution and the main outlier is Test 6 due to Team 6's imperfect control, also seen in Figure 47. Test 6 is also a spinning maneuver outlier, due to Team 4's late start, also seen in Figure 48. As mentioned before, this was the lowest scoring and exceptionally underperforming test among the ISS tests. The above analysis repeated for acceptable error levels *halved,* i.e. within 5 mm/s (about Opulens) of resonance for revolution and within 3 degrees of resonance for spinning, the average efficiency is 90% and 76.5%. The main outlier that causes the drop of spin efficiency is Test 8, due to the imperfect control of Team 5. Since they apparently controlled their attitude rate at 34 instead of 30 degrees/s, their control was very efficient but at the wrong resonance velocity, hence 100% efficiency at acceptable error levels but 0% at stricter levels. The average efficiency of spinning without Team 5, assuming stricter error levels is 83.5%. Student teams, therefore, demonstrated that they were capable of writing very efficient control algorithms for hardware demonstration in space which not only met research acceptable levels (91-93%) but also doubled standards (76-90%).

All the matches that were declared successful on the ISS were run in simulation with higher noise levels in the SPHERES simulator (e.g. with asymmetric thruster firing, random noise in thrust levels) to compare the ISS performance of the players with respect to what the teams programmed them to achieve. The results of the simulated matches were compared to the ISS results in terms of 3 major metrics: *Station docking results for each team in each match, mining robustness of teams* and *Match scores for both players in each match.*

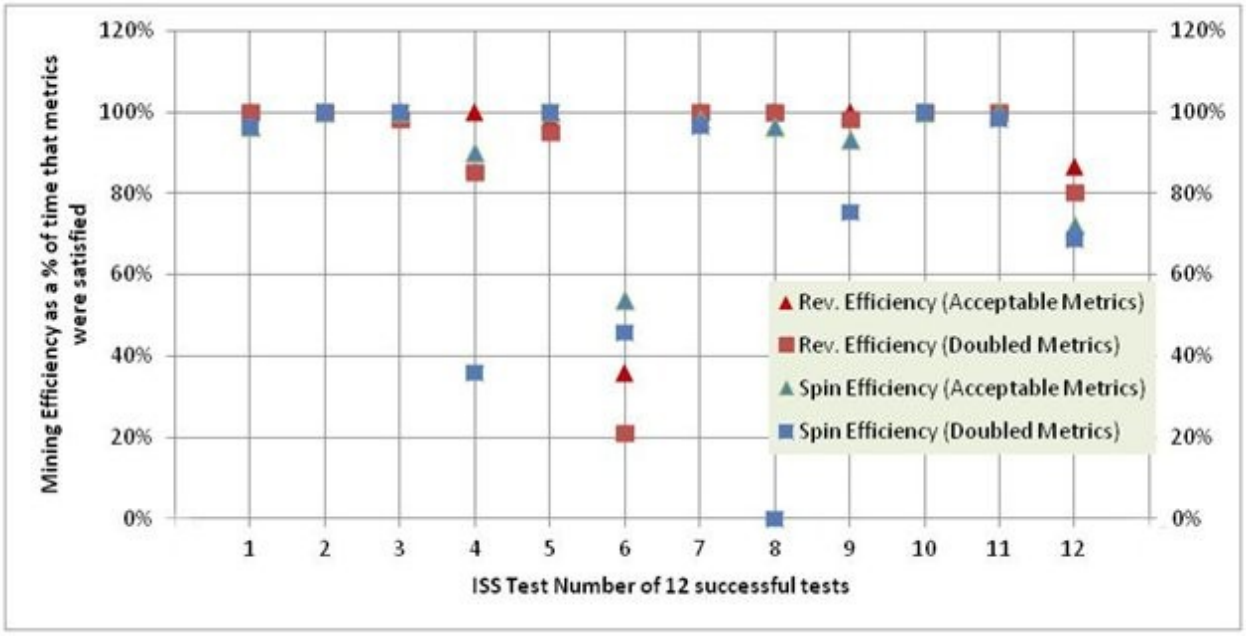


Figure 49: Efficiency of the revolve (blue) and spin (red) mining maneuvers with respect to regular metrics (triangles) and stricter metrics (boxes), over the time span of the main mining phase i.e. between 80 and 140 seconds since the match begins (X axis of Figure 47 and Figure 48)

Figure 50 shows a bar chart comparing the number of times over all its matches a team tried to dock its SPHERE to the station (measured by its success in simulation) to the number of times it successfully did so in the microgravity environment. The results show the teams tried to dock a total of 21 times, 14 of which were a success on the ISS. Analysis of Team 4's performance (the maximum defaulter in Figure 50) shows that the team was also slow in initiating its spinning behavior on station – indicative that their trajectory tracking not very robust. Statistically, removing Team 4 as an outlier, it can be seen that teams were able to come up with a solution to station docking at corner of the game volume within the last ten seconds of a match on the ISS efficiently,. Although this task was not one of the main proxy problems, its achievement is indicative of ZR's ability to interlace different problems within the same game and modularly separate the success in each problem using satellite telemetry. This result is also representative of crowdsourcing value achieved.

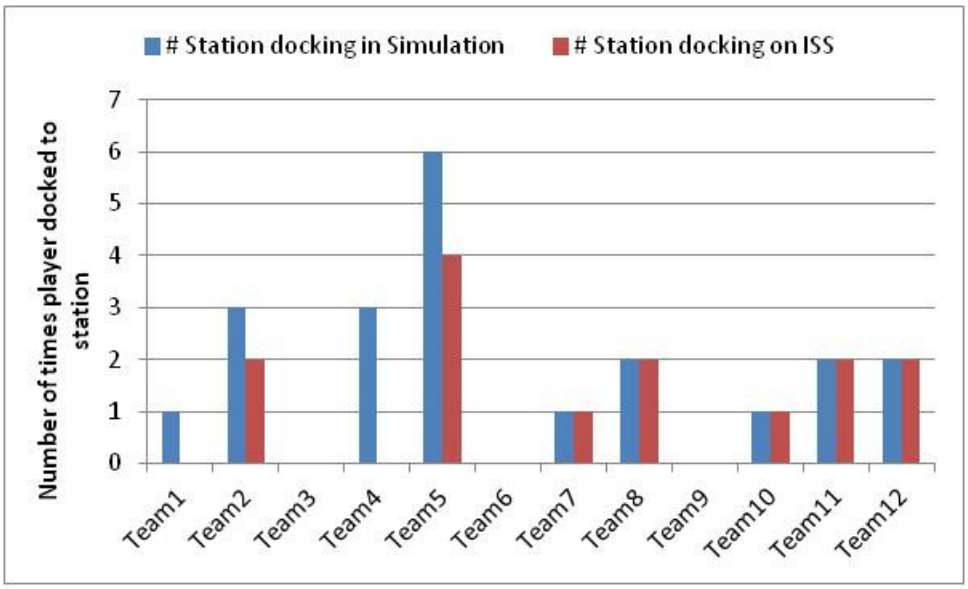


Figure 50: Number of times that a SPHERE controlled by a team docked to a mining station in simulation vs. on ISS hardware. A player could dock to a station only once per match. Teams played between 1-4 successful matches in the competition, depending on whether they qualified in the final RR bracket. 3 matches on the ISS were unsuccessful due to a hardware reset, due to which Team 9 has no recorded full ISS matches. Team 3 and Team 6 did not attempt station docking at all, even in simulation, as part of their strategy. Teams 1-9 are US teams and Teams 10-12 are EU teams.

Robustness of a team's mining algorithm to ISS conditions (*How similar are the ISS and simulation behaviors?*) can be calculated in terms of its mining score on the ISS versus in simulation. The mining score of a team is the total match score of a team minus points received for station docking. Therefore, robustness of mining is the difference of scores in ISS and simulation, after subtracting the difference in station docking scores (Equation 10), averaged over all matches.

$$miningRobustness(team) =$$
$$\sum_{match} [matchScore(match, Sim) - matchScore(match, ISS)] - stationDockPoints$$
$$* [stationDock(match, Sim) - stationDock(match, ISS)]/numMatches$$

Equation 10

Figure 51 shows the difference of average match scores on the Y-axis, total station docking successes on the X-axis (red and blue difference in Figure 50) and the slope of the dotted red line is the number of points received by a team per station docking (4 if alone, 6 of collaborative). Teams

with robust mining behaviors would lie on the dotted line because their difference in scores is entirely due to differences in station socking success on ISS vs. simulation (e.g. blue and red lines or green and purple lines in Figure 35). Teams plotted on the Y-axis achieved station docking whenever they attempted it so their y-intercept is the difference in mining behavior between ISS and simulation i.e. a measure of their mining algorithm robustness. For all other teams, the robustness is represented by their Y-coordinate with respect to the red dotted line.

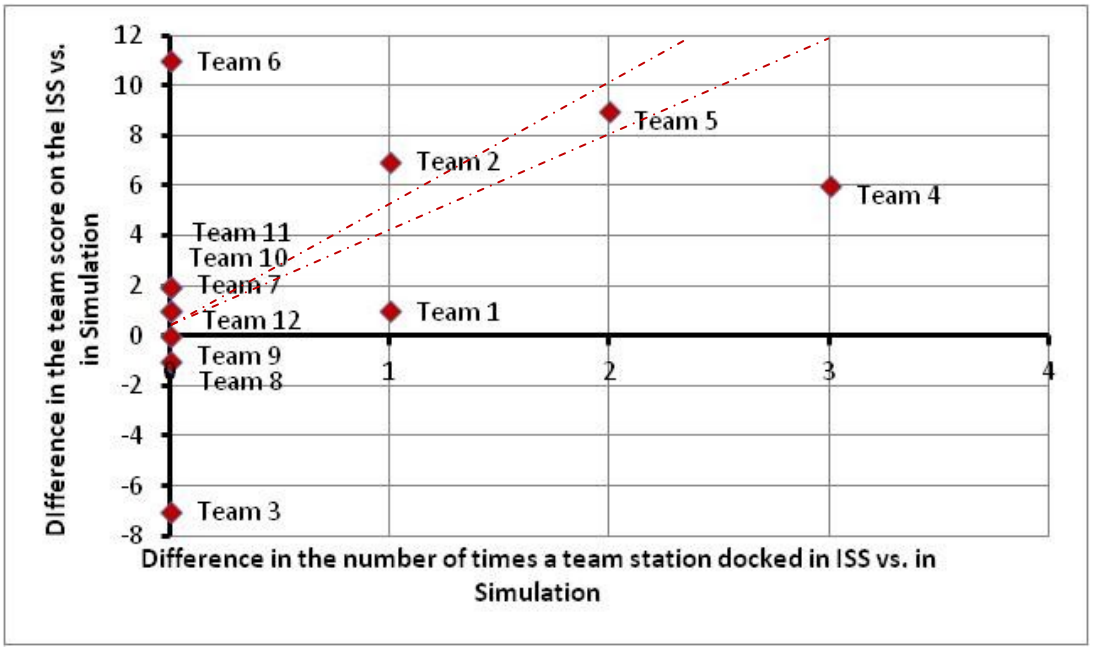


Figure 51: Scatter plot between the difference in ISS and simulation performance of terms (averaged over all ISS matches) of docking to a mining station and in terms of total score. The correlation of the two variables indicates the perfection of overall ISS mining behavior with respect to simulation behavior (r = 0.52). Teams 1-9 are US teams and Teams 10-12 are EU teams.

The analysis of the mining behavior of teams done in the previous paragraphs was used to identify the outliers in mining efficiency. As seen in Figure 47, Team 6 exhibited inefficient mining (revolution) – hence is far from the dotted lines in Figure 51. The average difference in points due to mining non robustness, calculated by averaging the vertical distances of all teams w.r.t the appropriate dotted line (depending on if they collaboratively docked) is 2.94. This is equivalent to missing out on 17 seconds of mining i.e. 80% efficiency as calculated by Equation 9. The bonus for winning the game was 2 points, so the difference in total scores could be because a different team

won the match by a hair's margin on the ISS vs. in simulation. Assuming an acceptable robustness error level of 2 points, 8 of 12 teams (67%) have research-robust algorithms for mining. Note that robustness is not equivalent to efficiency. For example, Team 3 in Figure 51 had a non-robust mining algorithm i.e. their player behaved *differently* in ISS and in simulation, however, the player performed *better* on ISS than simulation hence achieving highly efficient results on the ISS that met SPHERES research effiency standards (Equation 9).

Finally, it is important to establish that the analysis done and conclusions drawn using simulation competitions and scores (all other sections in Chapter 5 besides this one) is indeed applicable when the algorithms are demonstrated on space hardware. While Figure 51 and the associated discussion established parity in terms of mining maneuvers, the same is required for the overall objectives of the game. Figure 52 shows the comparison of all match scores in 12 successful matches conducted on the ISS test session; 10 of the 12 US matches and 2 of 3 EU matches were successful. For the unsuccessful matches, the simulation results were used to determine the match winners for the championship award. The average difference between the simulation and ISS scores is 2.75 (with a standard deviation of 2.52), which is equivalent to a third of the station docking points possible or less than a sixth of the mining points possible. Since the disparity between ISS and simulation is not significant, results from the analysis of simulation scores may be used to extrapolate the effects of collaboration on crowdsourcing of spaceflight software algorithms.

In playing the highest scoring match of the competition, which resulted in scores of 21 and 13 respectively, both SPHERES had ~23% of their virtual fuel remaining at the end of the match. Furthermore, in the next best match of the competition, with scores of 20 and 17 per SPHERE, the fuel remaining was 41% and 61% respectively. Both these matches were played by 4 unique players i.e. 4 alliances of 12 teams each. This clearly indicates that the participants were very capable of more challenging and resource-constraining scenarios which were not measured using the 2011 scores. The best players in the tournament could have been sorted further for fuel-efficiency, had the score been a function of fuel usage as well. For the middle school version of AsteroSPHERES, MIT undergraduates had written functions for the spin and revolve maneuvers as part of the game API library which MS students used to play the game. Analysis of the MS fuel usage *on simulation* reveals that the best performing revolver had at most 7% fuel remaining over all the matches that it played. Since the MS ISS event typically used about 4% extra fuel for SPHERES maneuvers compared to

simulation, a *qualitative* comparison with the HS ISS results shows that the HS students showed more fuel-optimal performance compared to randomly selected MIT undergraduate students. This finding strengthens the theory that crowdsourcing done in an appropriate way can be educationally powerful and produce resource-efficient algorithms.

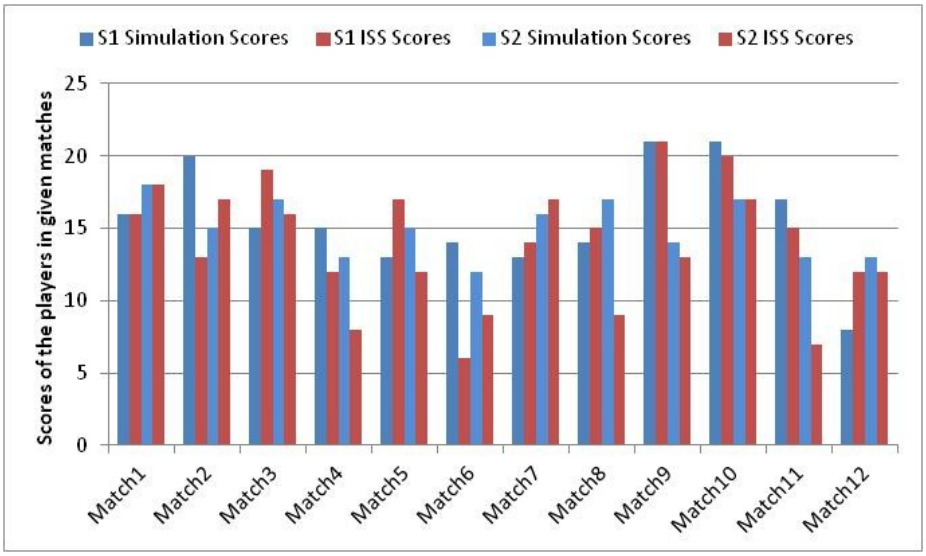


Figure 52: Comparison of the scores of both SPHERES in all the successful matches on the ISS with the scores when the corresponding matches were simulated on the SPHERES

5.1.3. Dedicated Crowdsourcing Tournaments in Zero Robotics

The 2011 HS tournament was primarily aimed at STEM education and outreach; however the research problems chosen for the game were formation flight problems, wherein participants were required to write closed-loop, precise trajectory tracking problems for correctly oriented spinning, rotation and coasting of the SPHERE. The tournament thus served as proof of concept that harder and unsolved formation flight problems may be introduced as a game within ZR and solved through tournament-based crowdsourcing. Furthermore, the scoring can be designed such that it is reflective of and prorated to the robustness and fuel efficiency of the required maneuver.

The lessons learned from the 2011 experience will be used to design tournaments chiefly for crowdsourcing autonomous spaceflight algorithms which can be tested on SPHERES as a small

satellite microgravity testbed. For example, an ongoing tournament within the program is the ZR Autonomous Space Capture Challenge [95] , whose goal is to develop an algorithm related to the recently announced DARPA Phoenix demonstration [98]. The objective of this specific challenge is to write a computer program to control a satellite (called a "Tender") to enable it to dock with a space object (or POD) that may be tumbling through space. The best algorithm submissions from simulation competitions, conducted every week for four weeks, will be tested in zero gravity on real SPHERES satellites aboard the ISS. The top performing program from each week's competition will be published on the website, so that other participants can learn from it and build on existing know-how rather than re-invent the wheel. The tournament is open to all age groups and all nationalities, unlike the HS tournaments which for US and EU students.

Another method of performing dedicated crowdsourcing through ZR Tournaments is to use newly developed algorithms as part of the ZR game code, which will be simulated every time plays the game i.e. simulates his project (Section 4.1.1 and Figure 26). Since tens or even hundreds of thousands of simulations are run for each tournament, this will be an opportunity to verify and validate (V&V) the robustness of the newly developed algorithm on the SPHERES simulator by subjecting it to hundreds of programs written by a random sample set of people. For example, the collision avoidance algorithm (as described in Section 4.3.1), that was an inherent part of the ZR game in 2010 and 2011 to prevent 2 SPHERES in a match from running into each other, was written by an MIT graduate student. It has been activated in many of over the 150,000 simulations run as part of the 2010-2011 competitions and therefore has provided data for verification and validation (V&V) of the efficiency of the algorithm. Details about the algorithm and V&V results are beyond the scope of this thesis.

The intent of this section was thus to point out that ZR Tournaments are great platforms where new algorithms can be developed by designing a game around specific problems and inviting crowds to play it and developed algorithms can be tested by subjecting them to thousands of human-designed simulations.

5.2. Benefits to CS-STEM Education

The high level goals for education and outreach using ZR [88] are to:

- *Engage students*, especially from schools that do not have funding for expensive robotics programs, in STEM activities by giving them hands-on experience with the SPHERES hardware and software
- *Create educational materials for students* to be used both during the season and the school year for extended learning and sustained engagement
- *Increase educator capacity and comfort in teaching STEM subject matter* by working collaboratively with certified in-school and out-of-school educators from participating schools, school districts and/or community based organizations
- *Build critical engineering skills for students,* such as problem solving, design thought process, operations training, and team work. Ultimately we hope to inspire future scientists and engineers so that they will view working in space as "normal", and will grow up pushing the limits of engineering and space exploration

MIT uses the unique CDIO Initiative for Engineering Education. CDIO stands for *"Conceive Design Implement Operate"* and offers an education stressing engineering fundamentals in order to create systems and products. By hands-on engagement, CDIO teaches students to appreciate the engineering process, contribute to the development of engineering products, and do so while working with an engineering organization. ZR follows the CDIO Initiative where students will *conceive* of a strategy to win the game, *design* a program using the SPHERES programming interface to demonstrate the brainstormed strategy, *implement* their projects using SPHERES hardware on the Flat Floor facilities and, using the feedback from the 3DOF environment, finally *operate* the SPHERES satellites using their projects aboard the ISS.

The different components of the educational experience delivered through Zero Robotics have been evaluated below. The intent of the following sections is to understand the observations made during the ZR program, specifically the 2011 high school season, and to design future tournaments such that educational benefits are maximized. Additionally, the effect of collaboration on educational

benefits has been deduced to evaluate the hypothesis of collaborative competition is helpful to students and mentors.

Note that a few results contain references to the ZR summer season conducted in both 2010 and 2011, where middle school students from handpicked schools from the Greater Boston area participated in a 5-week Zero Robotics tournament. The program was much smaller in scale than the high school one and is organized in collaboration with the Massachusetts Afterschool Partnership. The intent behind mentioning the middle school program in a few upcoming sections is to highlight the benefits of the STEM-Education side of the program, in terms of registration demographics and educational quality and to serve as a control for no collaboration environments. Since the middle school program does not demonstrate efforts in the direction of simultaneous crowdsourcing and STEM education or collaborative competition, which is the objective of this thesis, it has not been mentioned in much detail.

5.2.1. Registration Status

The 2011 high school tournament received applications from 123 teams in 30 USA states. Figure 53 shows the spread of participating schools in the US (Hawaii is not shown in the figure). Of the 24 teams that participated in 2010, 16 returned to participate in 2011, including all the 10 ISS finalists from the year. The ZR Program also expanded internationally in 2011. A select group of 22 schools from Italy, UK and Germany, handpicked under the supervision of the European Space Agency, played AsteroSPHERES on the same web platform as US schools. These schools have been geographically are shown in Figure 54. All the school teams participating together in the simulation competitions, however the finalists for the EU schools were selected separately. The ISS final competition was conducted separately for US and European teams and a separate champion alliance was declared for each.

Figure 53: Map of 123 registered US schools (2010 returning participants have been marked in blue pins)

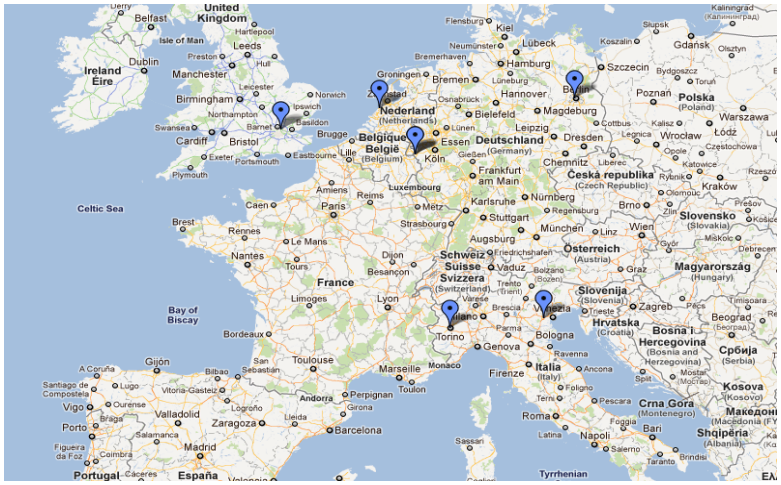


Figure 54: 22 registered EU schools in 5 geographic locations in 3 countries

5.2.2. Demographics

To evaluate if ZR had met its intended demographic objectives, surveys were sent to the 2010 and 2011 high school participants. In ZR 2010, 20 of the 24 participating schools (83.33%) completed the survey. There were 182 participating students with 62 mentors. The average number of students per high school was 9.1 – the maximum student number was 20 and minimum was 3. The average number of mentors per team was ~ 3. Of the 182 students, 82.2% were male, 20.9% came from low income families, 3.1% had disabilities, and 12.15% of them had English as a second language. In ZR

2011, 47 of the 145 participating schools (31.72%) completed the survey. 90% of the students were male, 9.18% came from low income families, 3.43% had disabilities, and 13.4% of them had English as a second language. 2010 had lower responses and more minorities, as seen in Figure 55 by the blue and red bars, because the schools were handpicked for a nationwide pilot, with special attention given to diversity, while 2011 was an open registration event. To compare the ZR demographics to the national demographics, data from the 2010 Census Bureau Report was used, as seen in Figure 55 by the green bars. To measure the representativeness of minorities in ZR, we averaged the ethnic distributions of all the U.S. states and multiplied them with the fraction of ZR schools from that state in 2011 (purple bars in Figure 55). High-school demographics of participating schools were not available. Comparison with both the national and state weighted average shows that in ZR the minorities are under-represented while Asians are over-represented by orders of magnitude. The ZR demographic numbers agree with the general trend of participation in STEM high school programs. The average national/weighted demographics need not represent the demographics of the participating districts or distributions at ages 13-17 years, which is the target age group. Also, publicity for the open registration event was primarily through NASA channels, so awareness within low-performing student districts and attractiveness of a primarily self-mentored program could have been lacking. For future years, the viral advertisement of the tournaments is planned so that more minorities can take advantage of the free, easily scalable program with extensive online tools.

A brief comparison of the performance of ZR 2011 HS teams from the U.S. with the demographic information collected through team surveys shows that performance had hardly any correlation with the female fraction in the teams, the average age of the team as determined from the distribution of students over different grades and minority fraction in the teams ($r <= 0.05$ for all). All responses in the team survey (for demographics) were sought anonymously and no identifying information was collected from user computers. All data has been reported only on an aggregate basis with no link to any one's personal identity. As a result, no individual responses or performance trends could be studied with respect to age, gender or race.

The program participation grew by 241% from 2010 to 2011. Demographic growth is calculated by the number of participants who reported to have completed the program (through feedback surveys) in 2011 vs. 2010 i.e. participated until they were eliminated by performance through a competition. The growth increase was 218% if calculated by the numbers who committed to

participate in 2011 vs. 2010, as reported in their application forms. In 2010, only 51% of the applicants were chosen to participate, since the program was intended for a nationwide pilot. In 2011, all applicants were allowed to participate if they committed to the requirements of the program. To avoid selection bias, growth should be calculated using total eligible applications instead of total participants. Participants who completed the program is more reflective of the program's success than those who committed to participate, hence a growth of 241% is more representative than 218%.

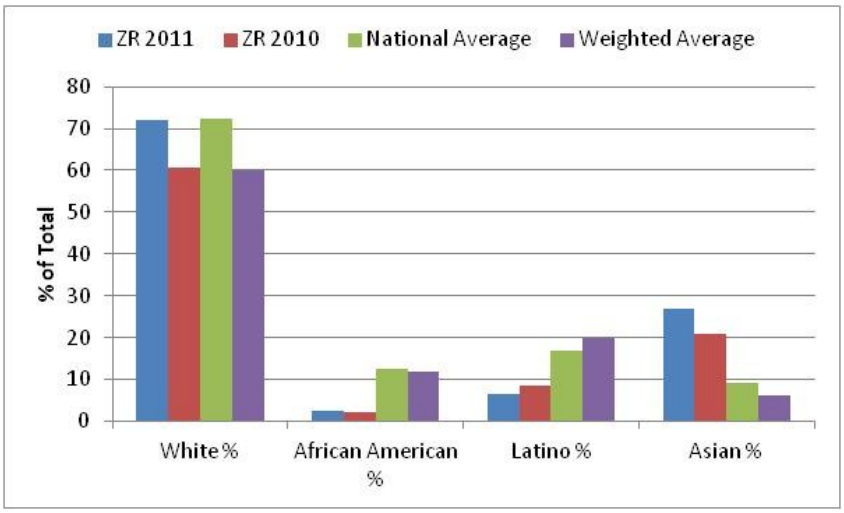


Figure 55: Ethnic distribution of ZR 2010 and ZR 2011 HS participants, national average of the ethnic distribution of all ages in the U.S.A. and the weighted average of ethnicities in 2011 based on the statewide breakdown of demographics

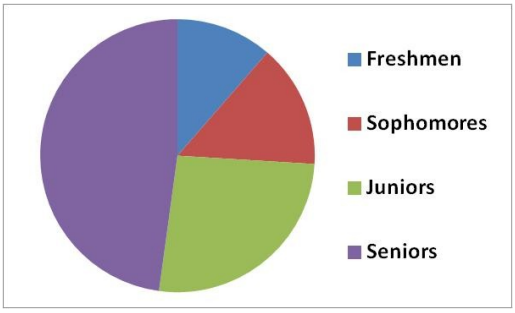


Figure 56: Distribution of students among the 4 HS classes based on the sample of students who responded to the survey among the population that participated

In September 2010, the President's Council of Advisors on Science and Technology released a report [3] that recommends that the federal government can and should create opportunities for inspiration through individual and group experiences *outside* the classroom (recommendation #5). Recognizing the need of afterschool programs, we have partnered with the Massachusetts Afterschool Partnership (MAP) for all our middle school programs. ZR engages students through the CDIO technique and establishes STEM institutional capacity through after school programs. Unlike the HS tournaments, the MS tournaments are open to only those schools selected by MAP and funded to hire teachers to guide the students – who serve as 'Team Mentors'. Additionally, each school has been exclusively supported by one MIT undergraduate mentor. The MIT mentors ensured that the students individually and as a team made sufficient progress with programming the SPHERES in order to complete the game successfully.

From the feedback surveys conducted after the middle school program in 2010, the statistics show success in achievement of our goals. There were over 200 middle school participants from 10 schools in the greater Boston area. 84% came from low-income families, 81% from ethnic minorities, 54% were female and 75% from low-performing school districts. The youngest participant was a rising 4th grader! All ten programs had a retention rate of 88% or greater and a daily attendance rate of 90% or greater. Due to funding limitations, the middle school program scaled down in 2011 and only 5 of the best 2010 schools participated. Of the 68 students in 2011, 31% were female, 79% came from low-performing school districts, 9% were diagnosed with learning disabilities and 10% were English Language Learners. The MS teams showed far more diversity than the HS teams because MAP selected the schools to uphold its objective of education for all.

5.2.3. Educational Quality

The quality of STEM education delivered by ZR has been measured in two ways: by analyzing improvements in game scores as the tournament progressed, and by using post-tournament surveys to obtain firsthand participant feedback regarding the educational impact of the program. The 2010 surveys were significantly qualitative since it was a pilot, and the descriptive feedback was intended to help design the open registration website for 2011 and further. In contrast, the 2011 surveys were

quantitative in nature with text space for providing optional written feedback. The findings are presented below.

There were two surveys- one team and one individual- both entirely online and available to all participants. Each team was requested to submit one response to the team survey, filled out preferably by a team mentor (See section 4.3.2 for definition). Each student participant was requested to submit one response to the individual survey. 240 mentors and students responded to the individual survey (out of 1274 open-registration participants from the US and ~100 from EU) and 47 teams responded to the team survey (out of the 145 teams whose applications were accepted and 110 teams who participated in at least one competition in the tournament). In the rest of this section, individual or student survey responses refer to participants *speaking about themselves*. Mentor or team responses refer to mentors speaking about *their overall team*. While all participants were reminded multiple times to respond to the surveys and incentives in the form of merchandise goodies were provided, potential for 'adverse selection' of respondents was ever present, especially since the sampled number was less than 50% of those who were potentially affected. The possibility that this may have biased conclusions drawn from the surveys to some extent cannot be completely ruled out.

The distribution of the participating grades in the HS program, shown in Figure 56, indicates that nearly 50% of the students are rising college freshmen i.e. only about half the current participants will be able to continue the program next year. The 2011 individual survey asked the students to rate the improvement of their skills in five different target areas. These target areas were chosen in a way as to measure as fully as possible the educational impact of the ZR program along the six primary objectives for federal STEM investment (as articulated in the 2010 Federal STEM Education Portfolio Report [99]): Engagement, institutional capacity, learning, leadership, STEM degrees and careers.

Figure 57 shows that the participants found their leadership, team-building and strategy-making skills the most improved, followed by programming, math and physics. The 2011 game was strategy-intensive and designed with the intent of incentivizing learning and achievement through collaboration and strategy. The idea was to get the students excited through peer-based learning techniques. This would potentially provide impetus to even the least STEM inclined to start off on

improving their basic CS-STEM and teamwork/leadership skills. The survey results (through ordinal data analysis) also show that more than 75% of the participants reported math, physics and programming improvements (Figure 57) and more than 90% reported leadership and strategy improvements (ratified by calculating the 90th percentile in the Figure 57 data).

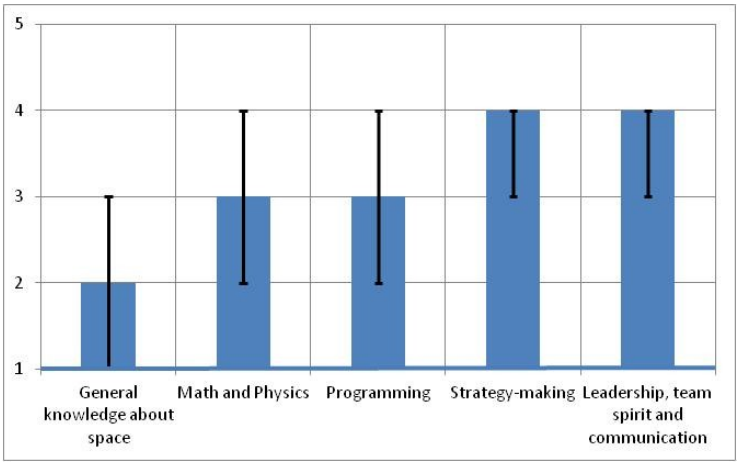


Figure 57: Median of responses to "*On a scale of 1 (no improvement) to 5 (significant improvement), please rate how the ZR Spheres Challenge improved your skills in the 5 mentioned areas*". (Error bars indicate the inter-quartile range) The horizontal blue line marks the neutral level.

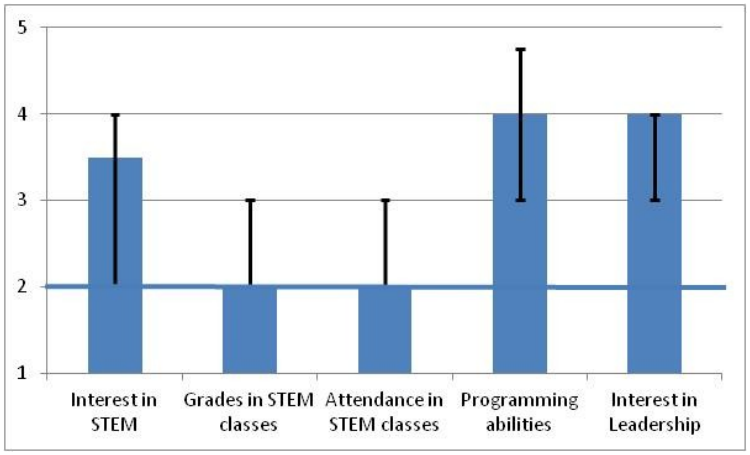


Figure 58: Median of responses to "*Please rate the students in the team on the following academic/education indicators compared to before the SPHERES Challenge 2011 where 0=Have no information, 1=Decrease, 2=No change, 3=Small but noticeable change, 4=Satisfactory increase, 5=Very significant increase*". Error bars indicate the inter-quartile range of responses. The horizontal blue line marks the neutral level.

Students were also asked the question *"How much has your inclination towards STEM increased due to the program?"* on a Likert scale (1=Not increased at all, 2=Not much, 3=A noticeable amount, 4=Significantly, 5=I am now certain of a career in STEM) to which their median response was 3. 89% of the participants in 2011 reported a measurable increase in STEM interest due to the program based on this question, and 15% declared, "*I am now certain of a career in STEM!".* The increase in STEM inclination yields a weak correlation (Pearson's correlation coefficient 'r' = 0.26) with the average number of hours that the participant reportedly spent on the program.

To guard against self-assessment bias, the responses of mentors that were relevant to their team's STEM improvement metric were taken into account (full results shown in Figure 58). Mentor assessment shows 'satisfactory increase' in programming and leadership abilities, with low range, and a nearly satisfactory increase in STEM inclination. Since 75% of the mentor responses lie above the neutral line (indicating 'No Change' due to the program), ZR is concluded to have significantly met the federal primary STEM objectives.

Mentors gave ratings of 'improvement in programming' that were about the same as those the students gave, but their ratings of improvement in leadership and interest in STEM fields were lower. The comparison of responses is shown in Figure 59. 85% of the mentors (speaking about their team) and 86% of the students (speaking about themselves) reported a positive increase in their programming skills. 89% of the students but only 77% of the mentors reported a positive increase in the team's leadership skills. Similarly, 88% of the students but only 73% of the mentors reported a positive improvement in team's STEM inclination. The neutral response along the Likert scale was at 1 for the individual evaluations and 2 for the team evaluations, which rates program improvement evaluation on a 5-point scale for individual surveys but only a 4 point scale for the team surveys.

The bottom, right panel of Figure 59 evaluates the math and physics skills improvement measured in two *different* ways. The individual survey asked by how much students perceived their skills to have improved, while the team survey asked by how much the mentors knew student grades to have improved. The figure shows that students reported a high individual improvement in math and physics skills (86% of them reported positive results). Their grades in those classes, however, showed only a small improvement (<25% showed positive results). Assuming that the students are not greatly exaggerating their improvements - a fair assumption given the correlation in the other 3

panels in Figure 59 – the net effect of these interventions is not immediately apparent in their school work. However, such indicators should be measured over the arc of their educational careers, perhaps every few years, to truly assess the long-term benefits of ZR.

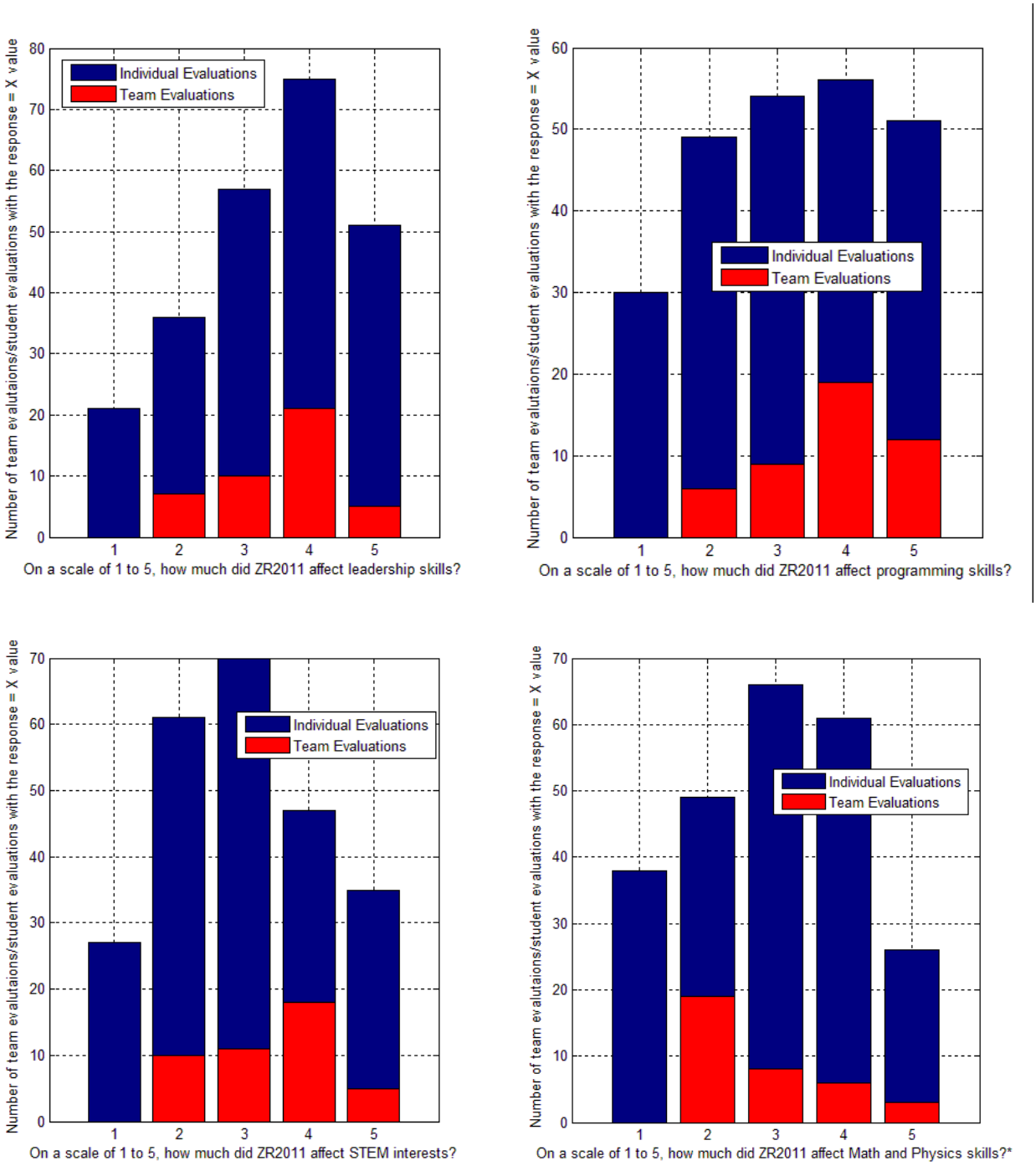


Figure 59: Histograms of responses to team (red) and individual (blue) surveys, on the effect of ZR 2011 on (roughly) the same 21st century skills[50]. The neutral response (indicating "No Change due to the ZR Program") was (1, 2) for the (individual, team) survey respectively.

One important lesson learned in the analysis of mentor and student feedback is that the scale of comparable responses in a survey should be kept the same, so that comparison across questions is possible; the language and description of questions that seek the same answers should as nearly as possible be identical. The 2011 survey was a pilot evaluation of a new observational study and the process has proven to be a valuable learning experience for us in how to ask the right questions in the right way.

Mentors of alumni teams were asked the following question: "*If you participated in last year's SPHERES Challenge 2010, please check the year (2010 or 2011) that you felt contributed more to the improving education indicators below…*" 55% of the alumni teams reported that 2011 contributed more (between the two years of participation) towards increasing interest in STEM and Leadership and most of the others reported that they contributed equally. Also, 65% of alumni team members reported that 2011 contributed more to their programming abilities, and again, most of the others reported that the two years contributed equally. Overall, this suggests s that ZR's has contribution has improved educationally. Furthermore, 89% of the teams that responded to the survey said that they would participate again in 2012.

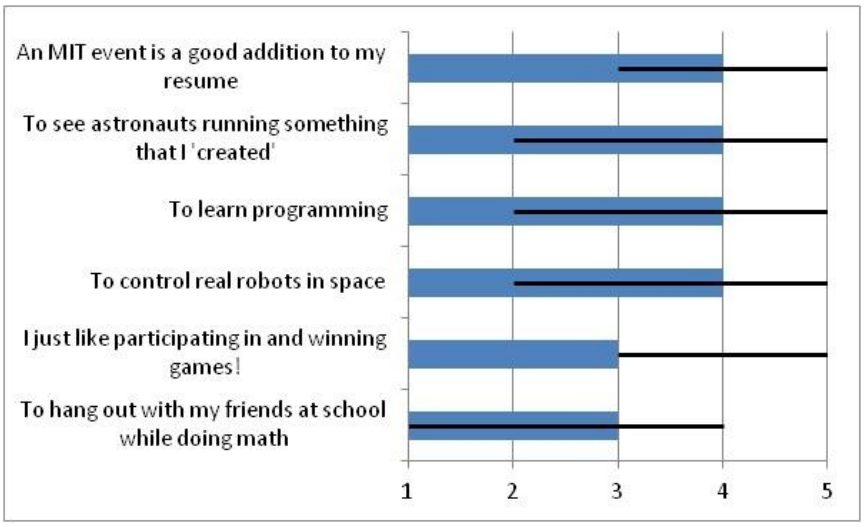


Figure 60: Median of responses to "Why did you participate in the SPHERES Challenge? On a scale of 1 (hardly a motivator) to 5 (significant motivator) please rate how much the following served as reasons". Error bars indicate the inter-quartile range of responses

To evaluate program satisfaction, participation motivation was studied and the program evaluated on grounds of achievement of its perceived motivational factors. Participants were asked to rate the reasons why they participated in the ZR tournament on a scale of 1 to 5 and the top 6 results are shown in Figure 60. MIT's name, the intricate engagement of the program with space, and programming skill-seeking emerged at the top of the motivational factors. The choice of the factor options for the survey was based on the vision for ZR's foundation and feedback from participants in the 2009 and 2010 pilot programs. For example, a 2010 mentor reported, *"We do not have a computer programming class at our school so this was a great activity and teachable time for students that were interested in programming to gain experience and accomplish a goal".* Another mentor had said "*This was one of the coolest projects I've been involved with. The fact that we were working on code that might eventually fly on the ISS was a very compelling motivator for the kids.* "

Significant amount of effort has been invested in ensuring that the value returned to participants for each motivational factor is high to maximize participant satisfaction. To allow teams the full MIT experience, the 2011 ISS finals event was held in a large MIT auditorium where ALL teams were invited. The event was hosted by 5 astronauts in attendance while the competition streamed in live from the ISS, hosted by 2 astronauts in space. Teams were able to meet their competitors and collaborators and interact with the MIT staff, all of whom they had met only over the ZR web interface. Attendance surveys showed that 245 participants (including 16 non-finalists) attended the event from 19 teams. For remote participants, the event was webcast live and screened live on NASA TV for over 6 hours on January 23rd, 2012. The 12 finalist alliances, comprising 36 teams, had their programs sent up to the ISS. Students saw 'astronauts run something they created' and successfully 'controlled robots in space'. 36 teams of the 145 that submitted an application (25%) and of the 91 teams that submitted a project to the tournament (40%) saw their motivational factors met. Programming knowledge objectives were met, as shown in Figure 58, wherein mentors indicated it to be the highest skill gained due to the program. Finally, to understand the value returned to the factor 'I like playing games', the students were asked an independent question: *"Compared to other video games/programming games you have played, how hard did you find AsteroSPHERES?*" where the response options were: 1=Fairly easy, I'd have liked harder challenges, 2=Difficult in the beginning, but was got a bit boring toward the end, 3=Challenging and engaging all through, 4=Too difficult for me to compete confidently. The median and mode of the responses peaked at 3 (Challenging and engaging all through). Overall, the program met its motivational objectives

satisfactorily and we now have a baseline in place to measure subsequent changes to the motivation and its achievement in the future years.

Optional descriptive feedback submitted by mentors and students indicated that apart from the factors in Figure 60, ZR appealed to them because it was a practical hands-on application of HS math and physics. A mentor provided the following feedback, "*I normally mentor programming contests with the students and this was different. The problem was more "real-world" and involved more strategy than just problem solving."* This tied in very well with ZR's founding principles and the thesis objective which is to provide accessible, real-world CS-STEM education to students.

Lastly, the performance of teams and alliances in competitions within the tournament and how they improved over time is an important metric of the educational quality of the program. This has been discussed in Section 5.1, since the performance metric is important for assessing crowdsourcing value as well. In future years, to improve value to students, the projects of the top performing teams in a competition may be published on the ZR website (with permission from the authors) so that other teams may learn from them and build on existing know-how.

Pre- and post-tests were administered during the Zero Robotics Summer Program 2011 for middle school students from the greater Boston area. Quantitative evaluation of results show that students' interest and engagement in the STEM fields increased as a result of participating in the program. Please see Appendix C for full results.

5.2.4. Effect of Collaboration

One of the objectives of the 2011 tournament structure and game was to introduce various elements of collaboration and understand their effects on STEM Education (research objective 2 in Section 2.5). It was hard to measure the independent effects of each collaborative factor without a tightly constrained human experiment. Since ZR is primarily an educational effort in which participants are meant to enjoy a fair game, the effect of collaboration on STEM education is measured using multivariate quasi-experimental analysis on passively observed/studied data. Quasi-experiments [8]

are distinguished from true experiments primarily by the lack of random assignment of subjects to experimental and control groups and sometimes, the lack of control groups (as is the ZR case).

One method of analyzing quasi-experiments is by using time-series analysis, which is the analysis of the changes of a variable in time, sometimes with the use of another time series to counter the effect of a third possibly confusing variable. Time series analysis was applied to the variable *competition scores* to assess the effects of collaboration in alliances on the scores of teams. There were 4 competitions in the 2011 tournament (Figure 36). Registered teams participated in the 2D Competition (2D) and then the 3D#1 Competition (3D#1). Team scores from both competitions were weighted at a pre-declared ratio of 1:3 and the 72 highest scoring teams were eligible for the 3D#2 Competition (3D#2). 3D#2 required teams to compete as alliances of 3 teams each chosen from 3 different "tiers" of performance as described in Section 4.2.2. Each alliance thus had a set of teams that had performed very diversely in the 2D and 3D#1 competitions. For the purpose of time series analysis, only 54 teams that participated in ALL 3 competitions above were considered. Each team played once against every other team in both 2D and 3D#1. Thus, every team played 53 matches in each of the two competitions.

Each alliance played against each of the 17 other alliances in 3D#2. We calculated the average match score of each original team (grouped vertically by the alliances they would later join) in the 2D and 3D#1 competitions using Equation 11 and plotted them in Figure 61 (red and blue error bars respectively). The mean score calculated by averaging over their future alliance, i.e. Equation 12, is plotted in blue and red asterisks. The average match score of each alliance in the 3D#2 competition, i.e. using Equation 11 with alliances instead of teams, is plotted as black asterisks.

$$averageMatchScore(team, competition) = \frac{\sum_{match}^{matches\ in\ comp} matchScore(team)}{number\_of\_Matches\_in\_Comp}$$

Equation 11

$$meanAllianceScoreOverTeams(alliance, competition) = \frac{\sum_{team}^{3\ teams\ in\ alliance} averageMatchScore(team, competition)}{3}$$

Equation 12

The mean score per match over all matches in each competition is plotted using a broken line – calculated by summing over Equation 12, by competition. As discussed in Section 5.1.2.2, the overall average increased from 3D#1 when there were no alliances to 3D#2 where teams played as alliances. Moreover, ALL teams showed an improvement by participating as alliances as seen in Figure 61. Since there was a 3 week learning gap and modifications in game rules between the two competitions, those changes could have contributed to the improvement.

The competitions 2D, 3D#1 and 3D#2 address the same problem in satellite (SPHERES) programming among the same subjects, on a game with the same structure in each case. The competitions occur three weeks apart and the placement of virtual objects in them is slightly different which affects the optimal strategy. Because of those (incomplete) similarities we use the difference between 2D and 3D#1 scores as a partial control for the difference between 3D#1 and 3D#2 scores. That difference was 3.4 points per original team over all the alliances (Equation 13). The same calculation was done in Figure 43 and Figure 44 in Section 5.1.2.2. Unlike crowdsourcing where we care about the topmost solutions i.e. the *right tail of the histograms*, for educational objectives, we care to maximize the students to learn and get motivated toward CS-STEM i.e. shift the *average of the histogram distribution.*

$$scoreImprovement(team) = averageMatchScore(team'sAlliance, 3D\#2) - 2 * averageMatchScore(team, 3D\#1) + averageMatchScore(team, 2D)$$

Equation 13

Figure 61 shows the broad range of 2D and 3D#1 scores of teams that came together as alliances – this is due to the tier system of alliance selection. As explained above, each alliance had one team from each of the three tiers of performance. Tier 3 teams showed the greatest improvement in performance from 3D#1 to 3D#2. These teams had the maximum opportunity to improve and it appears, the higher performing teams in their alliance helped them learn and improve quite rapidly.

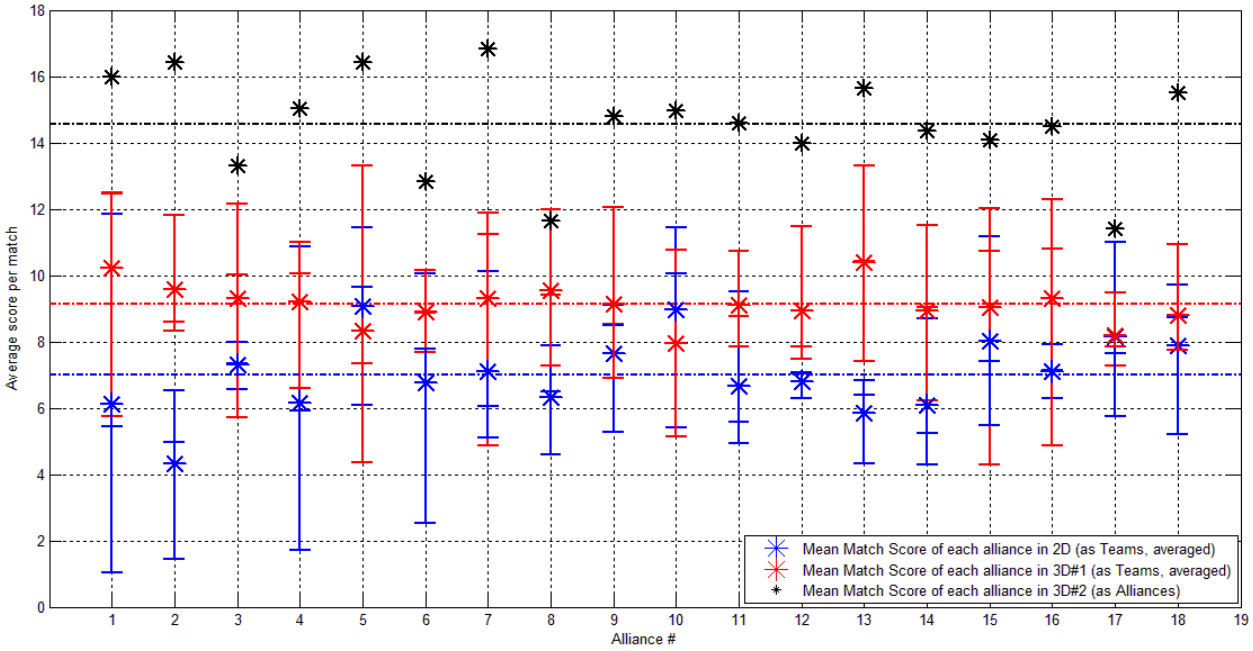


Figure 61: Alliances whose component teams participated in all 3 competitions (2D as Teams, 3D#1 as Teams and 3D#2 as Alliances) plotted against the average score of the alliance per match in the 3 competitions. For each asterisk, the error bar's horizontal line indicates the mean score of the component teams of the alliance for that competition; for 2D and 3D#1. The horizontal dotted line indicates the average score for that competition over all teams and alliances.

A similar time-series analysis was done using the relative ranks of the 54 teams, based on their total score in all the matches in that competition. Using ranks is convenient because Tier 1 teams have a smaller opportunity to improve their scores compared to others (their scores were already closer to the theoretical maximum of 23). Using ranks somewhat mitigated the risk of statistical regression (Section 4.3.3.2). As explained in Section 4.2.1, in any competition for the 2011 tournament, the team that received the maximum *total* score in a competition (i.e the summation of all its scores over all its matches in the round robin competition) received the topmost rank. Therefore, the average match score of a team/alliance as plotted in Figure 61 determined their rank in the competition. Figure 62 shows the improvement in rank for each original team from 3D#1 to 3D#2. In the 3D#2, the team is given the rank of its alliance's performance. Performance range is calculated as the 1-norm , the weighted range of each original team's score with respect to the average score its alliance. This was calculated using Equation 14 where averageMatchScore and meanAllianceScoreOverTeams is given by Equation 11 and Equation 12.

$$1 - norm(team) =$$
$$0.75 * |averageMatchScore(team, 3D\#1) - meanAllianceScoreOverTeams(alliance, 3D\#1)|$$
$$+ 0.25 * |averageMatchScore(team, 2D) - meanAllianceScoreOverTeams(alliance, 2D)|$$

Equation 14

The scatter plot in Figure 62 is U-shaped because we use the absolute value in Equation 14. Tier 2 teams have average match scores closest to the mean alliance score and therefore the lowest difference under Equation 14. By contrast, Tier 1 teams and Tier3, the extreme performers in their alliance have the largest distances from their alliance's mean score. The plot affirms that Tier 3 showed the maximum rank improvement and a Pearson correlation of 86% with the 1-norm range. This supports the general conclusion that the tier-based system of alliance selection used in the program was effective in bringing the competition spotlight on Tier 3 teams. On the other hand, negative correlations of Tier 1 and Tier 2 teams show that the diversity in recruiting fostered by the alliance formation protocol did not help them climb in rank.

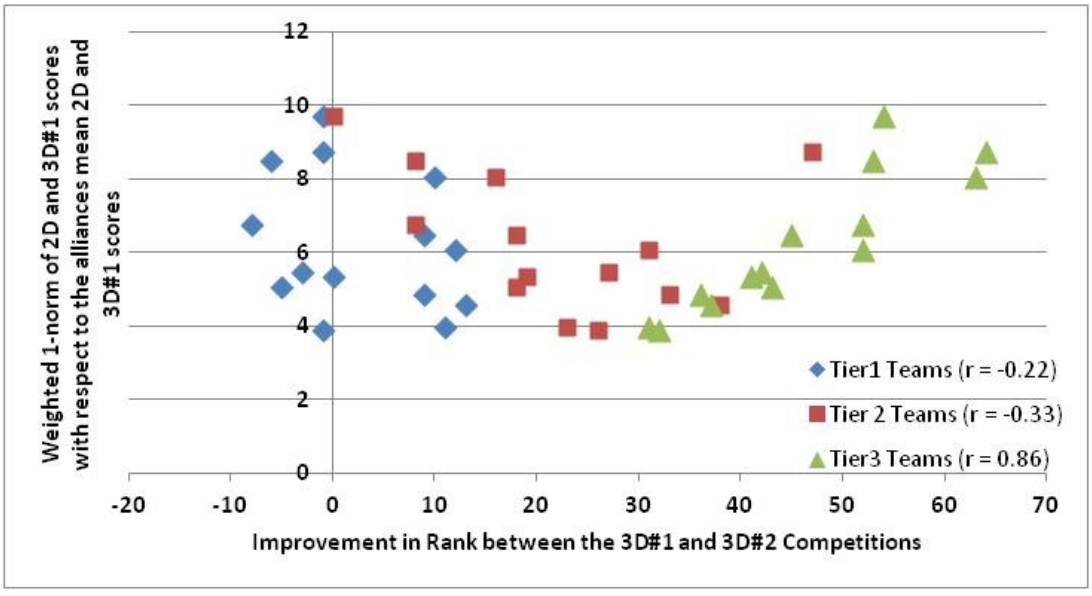


Figure 62: Scatter plot of the drop in rank (i.e. performance improvement) of 54 teams between the 3D#1 and 3D#2 competitions vs. the absolute range in their 2D and 3D#1 scores with respect to their alliance's mean, grouped by their alliance Tier Number. The Pearson's correlation coefficient for each tier's scatter plot is indicated in parentheses.

While Figure 62 showed that the *performance* of Tier 3 teams improved, it is important to examine whether this improved performance as an alliance was reflected their individual learning as a team. After the tournament, teams that competed as an alliance in the semi-finals and later were asked, *"How much of the alliance code did your team contribute?"* with the options of 5=Our team did all the alliance work, 4=Most of the contribution was ours, 3=Almost exactly 1/3rd of the work, 2=Much less than 1/3rd of the work, 1=Our team did not contribute to any alliance work. The range of responses of the teams (1 to 5) was normalized to (0 to 1). The average self-assessed contribution to the 3D#2 project of the Tier (1, 2, 3) was (0.909, 0.361 0.477) respectively. Stronger teams evidently felt that they contributed far more to alliance software and performance than the weaker teams did.

Figure 63 shows the (second level) difference between two improvements in an alliance's average score per match (Equation 15); the improvements between 3D#2 and 3D#1, less the improvement between 3D#1 and 2D. It is a rough measure of how much more they improved above their performance as separate teams before the alliances were formed. The red squares in Figure 63 indicate the averageImprovement by alliance, calculated by Equation 15.

$$
\begin{aligned}
&AverageImprovment(alliance)\\
&= averageMatchScore(alliance, 3D\#2) - meanAllianceScoreOverTeams(alliance, 3D\#1)\\
&- [meanAllianceScoreOverTeams(alliance, 3D\#1)\\
&\qquad - meanAllianceScoreOverTeams(alliance, 2D)]
\end{aligned}
$$

Equation 15

Figure 63 also plots the self-assessed contribution of each team in the alliance on a scale of 0-1 beside the actual improvements in score. The averaged self-assessed contribution over all team responses in an alliance is marked by a green triangle. Since some alliances had no responses from any of their teams, there are alliances without green triangles in Figure 63. The individual team responses (if received) are marked by horizontal bars about the triangles.

The ideal average contribution per alliance on a 0-1 scale should have been 0.33, irrespective of the spread of individual team contribution. The average of the self-assessed contribution was higher than that. This indicates that among the alliances in which all teams responded, many teams must have assessed their contribution to be greater than it may have been. Figure 63 shows a large overall

variation in the estimated contribution by teams to their alliance. The variation was not correlated (r = -0.01) with the improvement in alliance performance (Equation 15) and weakly and negatively correlated (r = -0.3) with the improvement in team scores (Equation 13). The latter correlation was not what the program intended – we had hoped the better scores would reflect a uniform contribution Instead, Tier 3 teams improved the most but claimed to have contributed the least. Conversely, Tier 1 teams improved least and claimed to have contributed most.

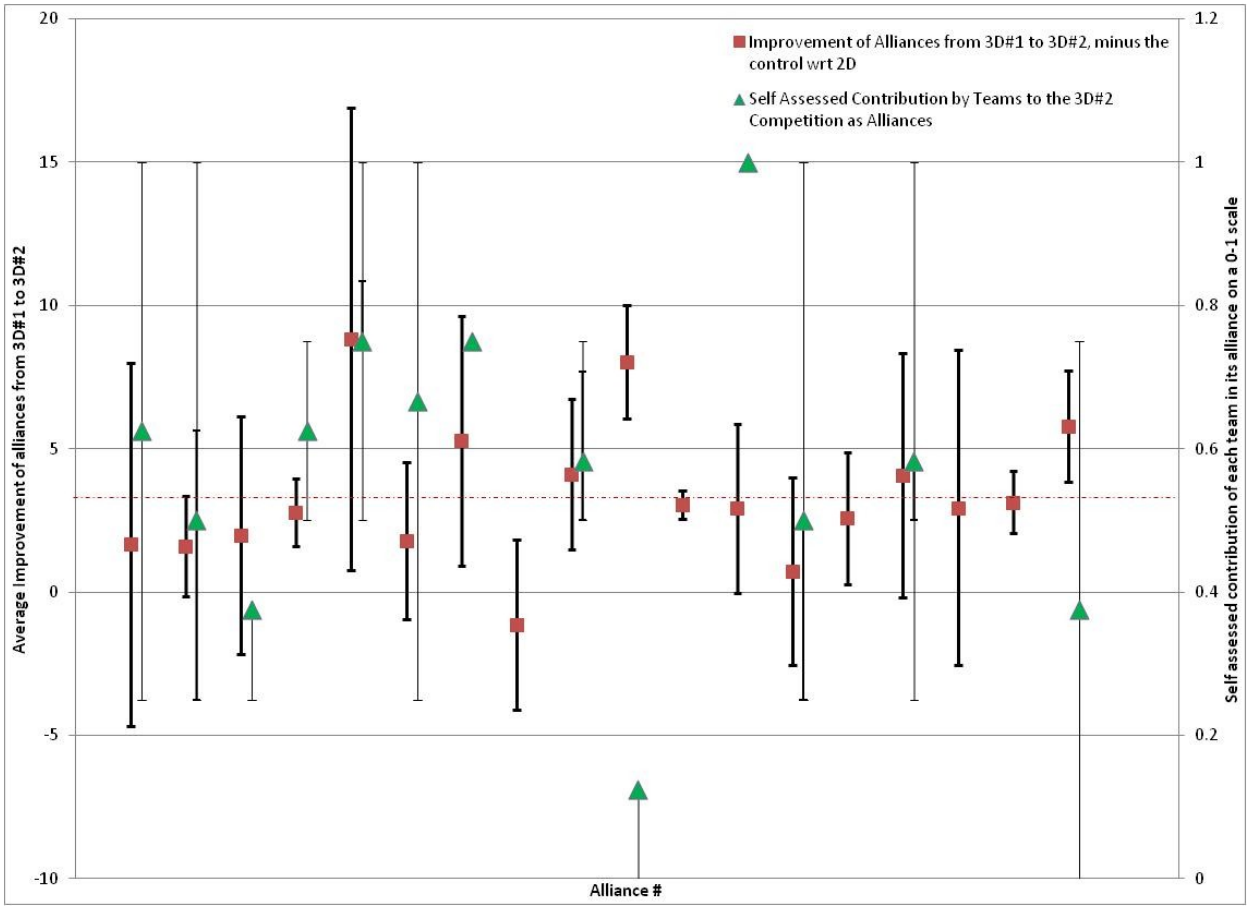


Figure 63: Change in average score per match from the 3D#1 competition (as Teams) to the 3D#2 competition (as Alliances) minus the control [2D minus 3D#1], to account for student learning and game change between the 2 competitions. The error bars indicate the standard deviation of the team averages with respect to the alliance averages (red squares) of match scores. The self assessed contribution of teams to their alliance's project (mean marked as green triangles, individual team responses, as bars) has been plotted on the secondary axis. The overall increase in the mean score, over the control was 3.4 points (horizontal red dotted line)

The analysis shows that performance scores of alliances alone are not enough to assess the educational value delivered to the teams. From survey responses, it appears that alliance formation limited the contribution of the weaker teams and slightly reduced the relative ranks of the stronger teams. This observation could potentially be attributed to demoralization bias – weaker teams may have been assigned less interesting or more menial work within their alliance and, as a consequence, felt they did not learn or contribute enough.

To investigate the effect that the special tier-based alliance selection method (Section 4.2.2), had on the improvement of teams' performances, team performance diversity (Equation 11 divided by averageMatchScore(team,3D#1) from Equation 11) was correlated against % improvement of match scores after alliances were formed - Figure 64. The correlation between the two variables is weak and positive, indicating that although the large difference in capabilities of the alliance teams correlated positively with the improvement of the teams' performance, it was weak (r=0.34) and tells nothing about causes. On the other hand, improvement in team ranks (X axis in **Figure 62**) correlated moderately and *negatively* with their average weighted 2D+3D#1 scores. In fact the correlation coefficient for Tier 1 teams alone was -0.4 and lower than the overall coefficient of -0.22. The individual capability of teams was not enough to help them improve. A finely balanced technique of inducing diversity in alliances without diluting their performances is therefore important.

While some respondents found the diplomatic collaborations within alliances interesting, found good ideas through them and agreed that they increased the overall STEM participation, others found it disappointing to remotely get in touch with teams that they had not worked with before and resolve differences of opinion. A mentor from the tournament said, "*It has been very interesting to work with another team, but I think the third team cannot add a significant value to the alliance. Moreover, a team that has performed badly in the qualifying phase can access to the finals if allied with a skillful team.*"

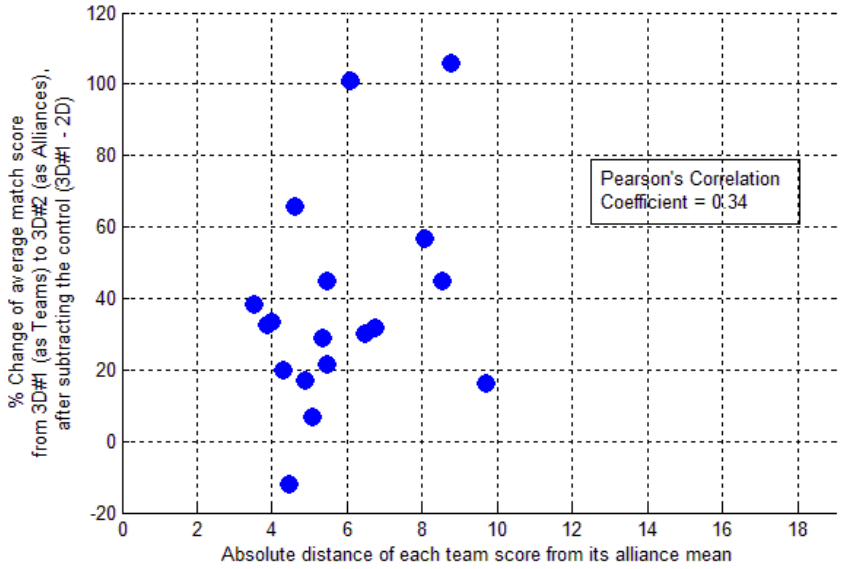


Figure 64: Scatter plot of the 1-norm range (Equation 14) of 2D and 3D#1 scores of each team in an alliance about the alliance mean score vs. the fractional improvement in average match score of each team from 3D#1 to 3D#2.

In future years, ZR will seek to design games that evolve significantly between the non-alliance and alliance competitions (e.g. 3D#1 and 3D#2 in 2011)so that the three teams in an alliance have enough remaining work to divide efficiently among themselves. No team should feel left out. The web infrastructure will require tools to promote equal contribution by all teams in an alliance. In the 2011 web interface, while all teams within an alliance could share projects with each other and use ZR's instant messaging tool to chat with each other when editing projects, the submissions tool allowed only Tier 1 teams to submit the alliance's project to a formal competition. Small details like these have been known to create a sense of alienation in Tier 2 and Tier 3 teams, as revealed in some of the essay surveys. Additionally, a re-evaluation of the alliance selection mechanism may also be needed. Tier 1 teams stronger expressed the desire to partner another Tier 1 team in an alliance, especially since discussion forums had forged friendships between already motivated teams.

The negative correlation of rank improvement of Tier 1 and Tier 2 teams as indicated in Figure 62 and the low correlation between a team's performance variability as part of an alliance and the same team's overall performance improvement provides an incentive to reconsider the idea of teaming diverse performing groups. While the formation of alliances has apparently increased the overall

performance of the group and received wide approval among the participants, there is room for improvement through the revision of the *methodology* of grouping teams into alliances.

While 63% of the survey respondents found the collaborative game challenging and engaging, and even intimidating, essay responses seem to indicate the way collaboration was implemented in the game and tournament was partially the reason why 33.8% found the game "*Difficult in the beginning, but got a bit boring toward the end*". Some students wanted a more adversarial game and many participants wanted more substance in the game after teams grouped up as alliances, so that each team would have something extra to do. It is important to note all respondents unanimously expressed that the collaborative nature of the tournaments should be retained to some capacity. From this feedback, the lessons learned for in-game collaboration are that while collaboration was well received as an objective, the game should have more adversarial components than just a finale race.

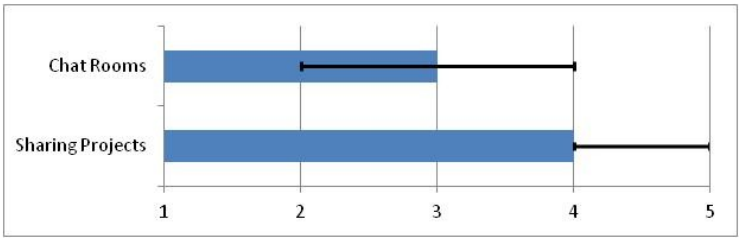


Figure 65: Median of responses to the individual survey question: "Please check that which applies to each of the collaborative features below". The response options were 1= Found it annoying, 2=Had problems using it, 3=Didn't notice/use this at all, 4=Used a lot but would like this improved, 5=Found this extremely helpful. Error bars indicate the inter-quartile range of responses

There were several web-based collaboration tools available to the participants in 2011 such as project sharing tools, an project instant messaging (IM) system among all users among who a project is shared, informal challenges such that teams could play matches against each other outside of formal competitions and a discussion forum. The participants were asked to rate the project sharing feature and IM chat features on a 5-point Likert scale and the results are shown in Figure 65. While project sharing was very well received and chat room feedback shows that up to 50% of the population might not have known about the chat application. This is because the chat application, like challenges, was released well halfway into the tournament and did not receive attention during

the kickoff introductions. Since the website is now well developed, it is expected that the features will be well advertised next year.

To understand the usage of discussion forums, the scores of teams in the 3D#1 competition was correlated with the number of forum posts by users of the team. This competition was chosen because it was the last one before forming alliances and we wanted to make any conclusions drawn independent of the alliance variable. The Pearson coefficient (r) was 0.37 which indicates a moderate positive correlation. The scatter plot of the data values is shown in Figure 66. It must be noted that, visually, the data seems to be bounded in a quadratic curve and is not linearly arranged i.e. very low performers hardly participated in the forums, very intense forum participants had high scores but there were high performers with relatively low participation. This analysis is a correlation and does not imply causation. The observed trend implies that students who were participatory and frequent at the forums tended to do well – more collaboration, better results. The trend could have been further strengthened due to the collaborative nature of the game. More forthcoming teams had the strategic advantage of interfacing with other teams to make a block of successful collaborators (e.g. protocols described in Section 5.1.2.2) while the quieter teams either efficiently programmed the strategies being discussed or did not invest effort in strategizing or programming. Overall, the message board system was very educationally popular (as indicated by the essay-type feedback) and logged a total of 5150 messages by 164 unique users in the entire tournament period.

To measure the value the participants felt they gained through the various features provided by the ZR program, the individual survey asked: "*On a scale of 1 (no contribution) to 5 (significant contribution) please rate the contribution of the following ZR features to your educational experience"*. The results are shown in Figure 67. Features that are specific to inter-teamcollaboration are marked in green. Among the purely collaborative features, 75% reported gaining from in-game collaboration (Collaboration Environment #1 in Section 4.2 and 4.3.1.3) and forums and challenges (Collaboration Environment #3). Less than 70% reported gaining positively from alliance-based collaboration (Collaboration Environment #2). It can be argued that the survey responses in Figure 67 are heavily influenced by hindsight bias – since all evaluations are based on a post-tournament survey - and interference bias - too many collaboration variables were being evaluated at the same time. (People are likely to make errors in judging the individual impact of each factor.) Future editions of the ZR program can achieve more precise evaluations by having participants fill out a short questionnaire between each

evaluative phase of the tournament. This will allow the factors that would produce bias to be better isolated. Too many questionnaires may also irk the participants, so a balance must be struck.

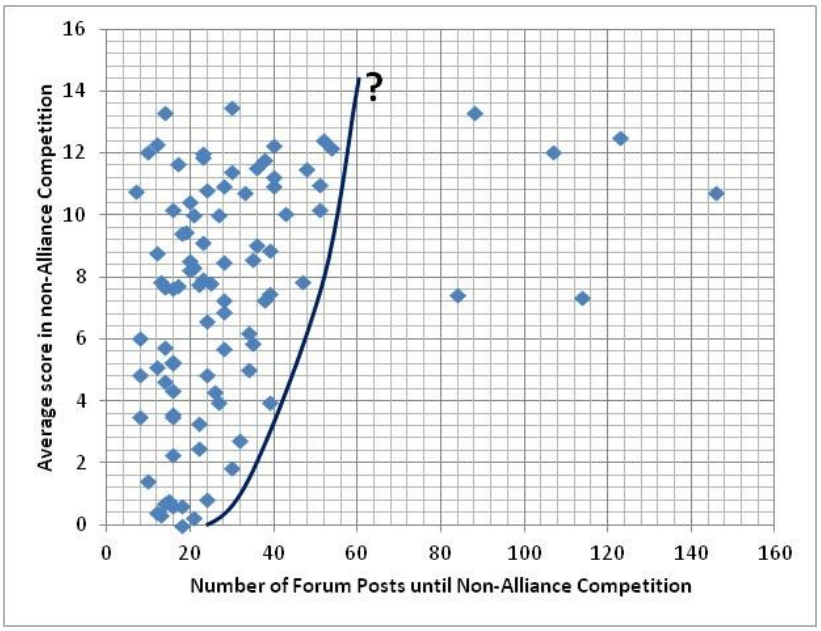


Figure 66: Scatter plot of the number of posts made by a team on the website discussion forums of before the submission deadline of a competition versus the average match score obtained by that team in the same competition. Correlation coefficient (r) =0.37, quadratic trend seen.

A more detailed analysis of 201 responses (only alliance participants among 246 total responses received) compared the relative significance of the ZR Features listed in Figure 67 in terms of their educational benefits to users. The relative preference of each individual for a specific ZR Feature was calculated by subtracting two corresponding responses. This was repeated for all 201 responses and the histograms of differential preferences plotted in Figure 68. The histograms were found to be normally distributed so calculating the mean differential preference (Figure 69 as a color map) between every two ZR Features were enough to specify the pattern of preferences.

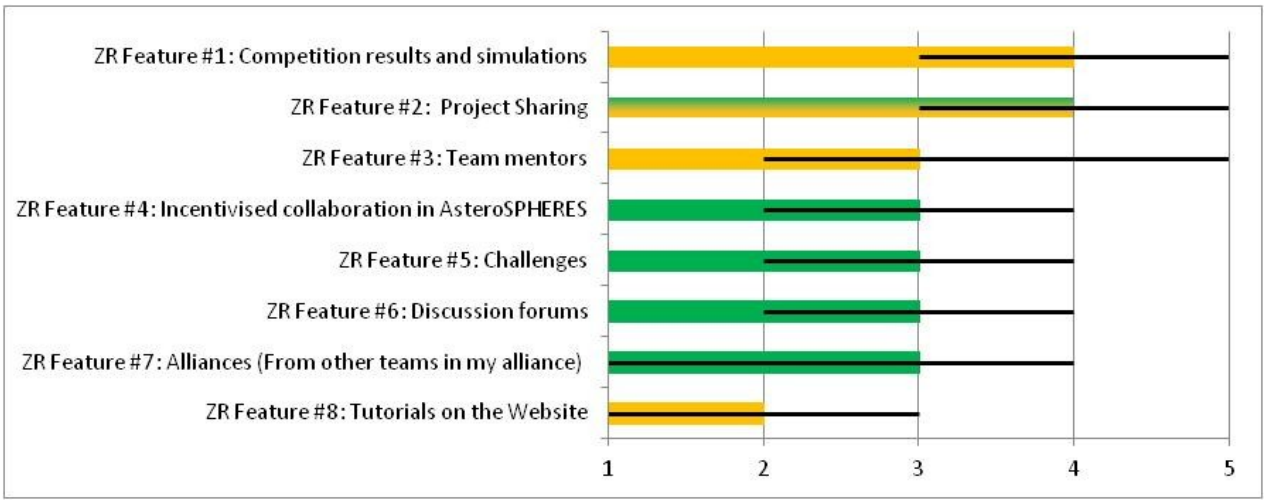


Figure 67: Median of responses to :"On a scale of 1 (no contribution) to 5 (significant contribution) please rate the contribution of the following ZR features to your educational experience." Error bars indicate the inter-quartile range of responses. 'Green' indicates the inter-team collaborative tools. Project Sharing is marked half in green because it may be used to promote collaboration within a team or outside of a team, within an alliance.

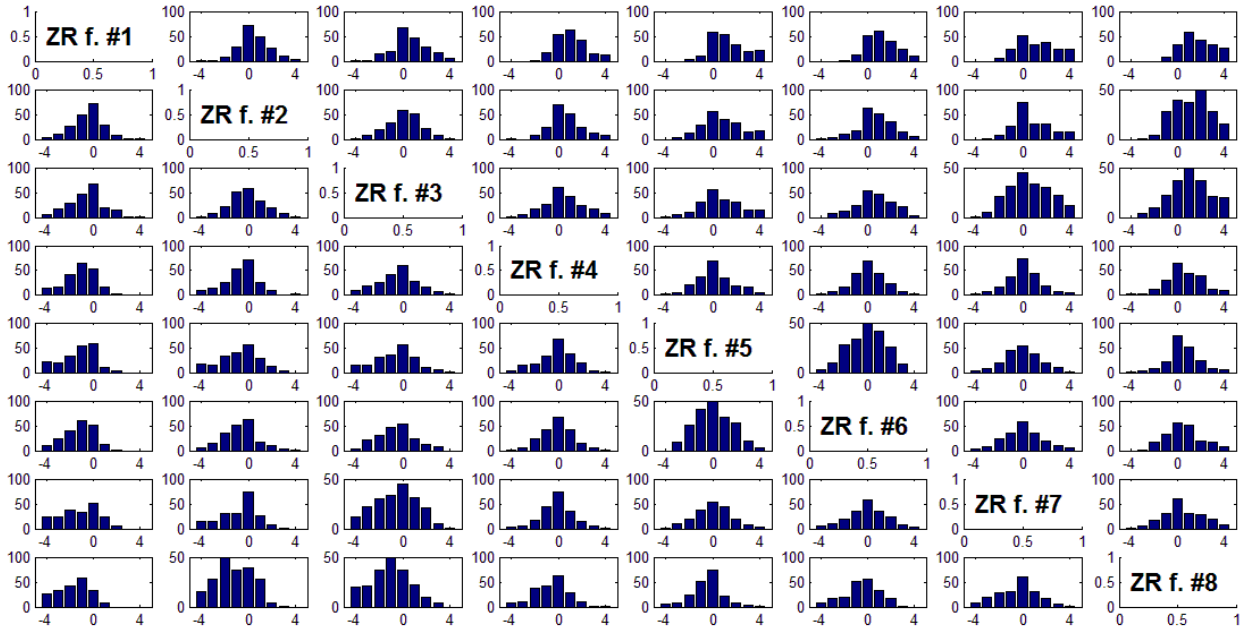


Figure 68: Histograms of differences between pairs of preferences for the 8 ZR Features in Figure 67. The histogram in the Square X-Y represents the distribution of those differences for all subjects. For example, the histogram in the (X,Y)=(1,8) shows the distribution of the differences between the responses i.e. the preference for 1 over 8 ZR Feature #1 and ZR Feature #8. The range of preferences is -4 through +4.

There are few significant differences in relative preference among ZR Features #4 to #7 i.e. the inter-team collaborative tools – marked green in Figure 67 - and little relative preference between ZR Feature #2 and #3 i.e. intra-team tools - instruction by mentors and project sharing. This inference was made using Figure 69 which shows two squares of green that indicate near-zero relative preference between the rows and columns it represents in the color map. Intra-team tools however proved to be more beneficial to the students than the inter-team collaborative tools. This is noted from the yellowish patch in rows 2-3 (intra-team features) and columns 4-7 (inter-team collaborative features). The website tutorials (#8) were of lowest value and the competition results and simulations published on the website (#1) of highest. In those, teams could learn from their mistakes and from others' strategies.

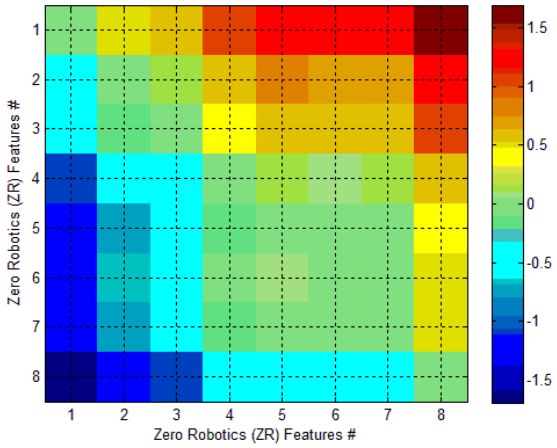


Figure 69: Color map representing the average preference between the 8 ZR Features in Figure 67, taken two at a time. The average for each 8X8 block is obtained by finding the mean of the histogram distribution for that block from Figure 68. For example, the top right-most corner of the color map indicating 1.5 in crimson is the average difference of response values to ZR Feature #1 and ZR Feature #8. Note that the color map is anti-symmetric about the main diagonal, because mean(X-Y) = - mean(Y-X)

The nonparametric Friedman test was conducted on the survey responses to rate the ZR Features #4-#7 (inter-team collaborative tools). The similarity in their relative values as seen in Figure 69 was found to be not significant. The subjects did not agree on the relative ranks of the tested tools. Results of the pair-wise Friedman test run (Table 6) on the responses to the 4 inter-team collaborative tools show high p-values for all the entries i.e. no statistically significant difference between the responses. On the other hand, the low p-value between ZR Features #2 and #3 (Table

6) and the color map in Figure 69 shows that the subjects agreed that the two intra-team collaborative features were equally important.

Responses to Feature #	Responses to Feature #	Statistic	p Value
Pairwise Comparison of Inter-Team Collaboration Features			
7	6	0.066	0.947
7	5	0.110	0.912
7	4	0.795	0.427
6	5	0.044	0.965
6	4	0.861	0.389
5	4	0.905	0.366
Pairwise Comparison of Intra-Team Collaboration Features			
3	2	1.844	0.067

Table 6: Multiple Comparisons Test Table indicated the results of the nonparametric, pair-wise Friedman Test conducted on the responses collected about the inter-team collaboration tools numbered ZR Feature #4 through ZR Feature #7 in Figure 67.

Participants who participated in both 2010 and 2011 tournaments were asked to rate the learning through 2011's collaboration and more than 60% reported positive results on a 5-point Likert scale as seen in Figure 70. The 2010 web interface had no collaboration features apart from external discussion forum which logged 142 posts (the website had 144 users). This is a very low number compared to the 2011 web interface which logged 5150 posts (the website had 1689 users), even when the post to user ratio is considered. Figure 70 also shows that, on average, students who had participated in 2010 found C programming easier in 2011, indicating the CS-STEM value delivered by the program.

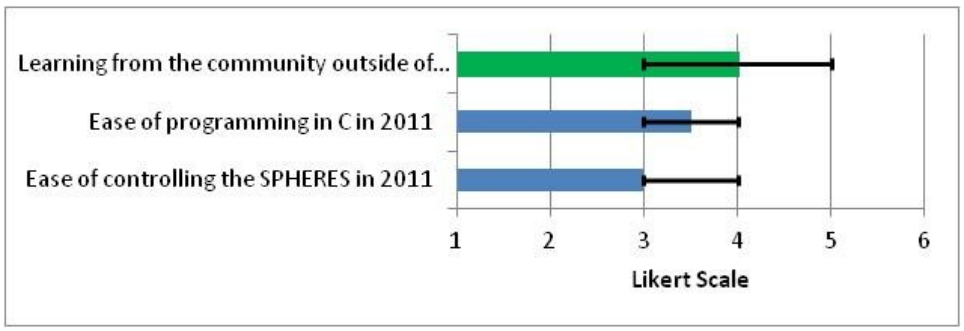


Figure 70: Response of alumni from the 2010 tournament to "Please rate the following in the 2011 tournament with respect to your experiences in 2010" where 5=Significantly more, 4=A little more, 3=Felt the same, 2=A little less, 1=Significantly less. Error bars indicate the inter-quartile range of 'Green' indicates a question targeted to evaluate inter-team collaboration.

5.3. Zero Robotics Tournament Results Summary

This chapter has attempted to assess the usefulness of collaborative games and competition in space education and development of useful algorithms by crowdsourcing, as assessed through the Zero Robotics Program.

The crowdsourcing impact of the program is measured from the performance of participants in the ZR tournament in terms of their scores in the simulation competitions and final performance on ISS SPHERES hardware. The game was designed such that by playing it, participants would submit algorithms to implement a few proxy formation flight maneuvers, and the match scoring was designed such that it would reflect the quality of these algorithms. Therefore, by analysis of the scores and performance trends of the teams, it has been possible to measure the impact of crowdsourcing and its dependence on the collaboration environments introduced in the 2011 tournament. Student teams were able to achieve perfect solutions to the crowdsourced problem in simulation and the results improved by more than one standard deviation about the mean due to all types of collaboration. Results of the ISS test session were evaluated from analysis of satellite telemetry recorded from the SPHERES during each test/match. These results certified the efficiency (>90% of players were within SPHERES acceptable research levels) and robustness (>80% of players within acceptable error levels) of the algorithms submitted on real nanosatellites in

microgravity as well as compared them to simulation results. Moreover, the top players achieved the game objectives within 23% of allocated fuel and 80-90% of the players were certified efficient even when calculated with acceptable error levels twice as strict as usually acceptable in SPHERES research. Finally, the lessons learned through this attempt at crowdsourcing proxy formation flight problems in 2011 have been important in designing a full-fledged crowdsourcing tournament, ongoing within ZR currently.

The educational impact is based on existing theory that games are motivational learning tools and kids and young adults are very fascinated by space. The utility of adding the collaborative and competitive factor to games based in space is evaluated by the development of a hands-on educational robotics program and conducting tournaments on it. Data collected over the last two years in the form of the performance in the competitions, usage of the web interface, hardware operations on the ISS and feedback about the program is used to measure the utility of the tournaments. The data analysis and the experience of running the program has taught us valuable lessons for better tournament design for efficient educational outreach within the ZR framework. Overall, the program has shown success in less than 2 years of nation-wide operation, demographically by growth percentage (241%), quality of STEM education (80%-88% definitive positive response) and retention rate (89%). Additionally, building on the existing theory that collaborative gaming is becoming a very powerful tool for learning and solving, we have introduced collaboration environments within ZR and attempted to assess the effect of these environments on the educational experience of the participants. Although the results obtained do not show conclusively positive results for all the collaboration environments, noticeable improvements due to collaboration have been observed. More importantly, the feedback has shown us ways in which the collaboration implementation within ZR can be improved to deliver better quality education and we have a framework in place for measuring the effects on our objectives.

To conclude this chapter, it must be stressed that there is a difference in the way performances are evaluated in achieving the dual objectives of crowdsourcing and STEM education. In crowdsourcing, one cares only about the very best of solutions, i.e. for the rightmost tail of the histogram distribution of performances in any competition (Figure 39, Figure 40, Figure 41, Figure 43, Figure 44). The purpose of sourcing solutions from dozens, hundreds or thousands of people is to identify the outliers that are most novel and high performing. In CS-STEM on the other hand,

one cares to get maximum number of students involved and influenced i.e. shift the average of the histogram distribution for any competition toward the right or raise the average score (Figure 61, Figure 63). The ZR program has proven that it is successfully able to achieve both simultaneously, apart from efficient and robust hardware test runs as well as positive user reviews of satisfaction and STEM inclination.

Chapter 6 –
Management Policy Implications

Chapters 3 through 5 established that crowdsourcing of spaceflight software and educating the participants is potentially feasible through the same program, and that real problems can be solved, to some degree, while letting students reap educational benefits through working on real-world projects. This chapter introduces a System Dynamics model for the collaborative crowdsourcing and STEM education effort. It uses the dynamic loops within the model to make high-level recommendations (Section 6.1). Backed by literature and lessons learned through designing, developing, operating and analyzing the ZR Program, plausible management policy with the implementation of the recommendations have been suggested in Section 6.2.

6.1. System Dynamics Model of Collaborative Crowdsourcing and Education

System Dynamics is a methodology to understand the behavior of a complex system over time [100]. It classifies all the variables affecting the system into stocks and flows or exogenous and endogenous. A stock variable is measured at one specific time, and represents a quantity existing at that point in time (which may have accumulated to that amount over time). On the other hand, a flow variable is measured over an interval of time. Stocks are connected to each other using flows alone. In fact, stocks are the integration over time of the net flow into them. Exogenous variables are independent variables which affect other variables in the system, which in turn are called endogenous. Endogenous variables and flows are related to each other and the stocks using causal links, each associated with an equation to calculate how one variable is related to another. An example of a simple systems dynamics model generated using the VensimPLE software is shown in Figure 71. The causality of the links has been marked in the figure; a positive causal link is one where the partial differential of the dependent variable with respect to the independent variable has a positive sign. The overall effect of the closed loops i.e. multiplication of all the causal links, has also been marked: 'R' indicates a reinforcing (overall positive) loop while 'B' indicates a balancing (overall negative) loop. Reinforcing loops cause exponential growth (or fall) dynamics for all the stock variables in the loop while balancing loops cause S-curve dynamics (goal-seeking) dynamics for all the stock variables in the loop.

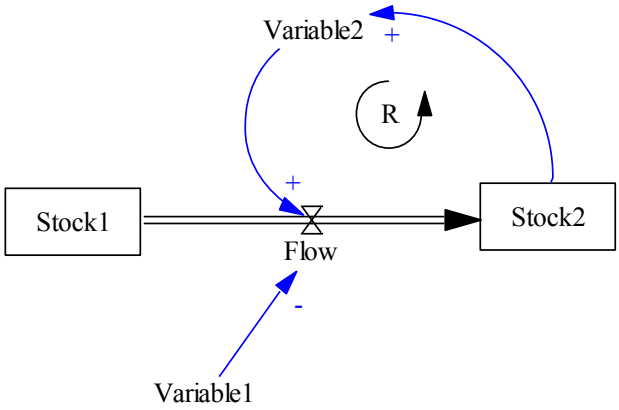


Figure 71: A simple systems dynamics model showing a flow variable connecting two stocks. Variable1 is an exogenous variable which affects the flow variable *negatively* (as shown by the minus sign). Variable2 is influenced by the stock and influences the flow, both positively. Stock2 is the integral of the flow and Stock1 the negative integral. Since the variable2, flow and stock2 variables form a closed loop whose overall effect is positive, the loop is called reinforcing.

The simultaneous crowdsourcing of formation flight algorithms and STEM education model of Zero Robotics has been represented as a system dynamics model in Figure 72.

The major **stocks** in the model are:

- *Formation Flight Unsolved problems* i.e. algorithms needed for the growing need of guidance, navigation and control of distributed space systems
- *STEM Disinterested students* i.e. students who are not interested in or do not want to pursue CS-STEM further
- *STEM Interested students* i.e. students who are enthralled by CS-STEM topics to some extent
- *User Engagement Software Required* i.e. web tools required for participants to develop algorithms for formation flight
- *User Engagement Software Available* i.e. web tools available to participants for developing algorithms. This is critical for them to contribute

The major **flows** in the model are:

- *Formation Flight (FF) Problem Discovery Rate* i.e. the rate at which the need for efficient FF algorithms is changing, which directly depends on the growth of the distributed space systems infrastructure

- *Formation Flight (FF) Problem Solving Rate* i.e. the rate at which the above problems are being solved which depends on either STEM interested students solving them directly in the future or the extent to which user engagement software (itself crowdsourced) allows programs such as Zero Robotics to solve the problems through crowdsourcing
- *Successful Outreach Rate* i.e. the rate at which students' interest in STEM topics changes.This is influenced by standalone outreach programs, crowdsourcing combined with outreach programs and user engagement software that allows students to engage in real-world projects.
- *Software Development Rate* i.e. the rate at which crowdsourcing software gets developed which depends upon the supply of STEM interested students capable of building such software and the demand of such software by crowdsourcing programs such as Zero Robotics.

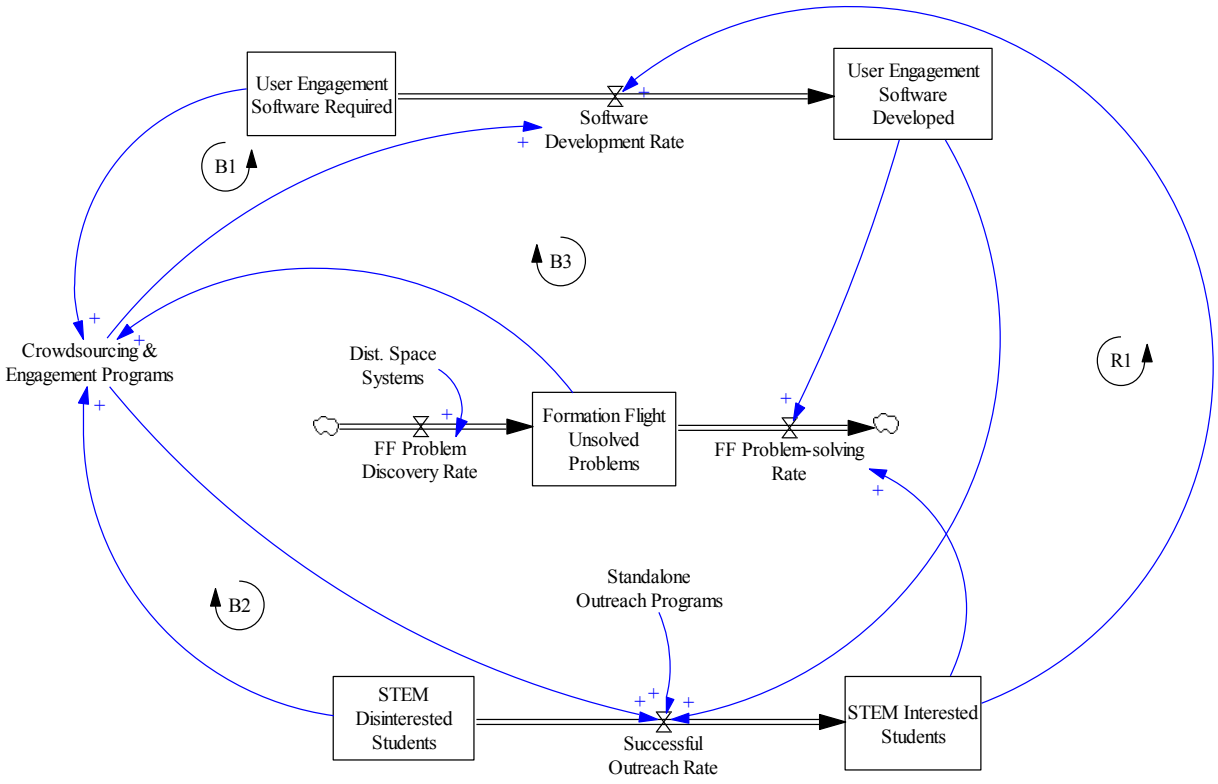


Figure 72: Simplified System Dynamics model for simultaneous crowdsourcing of formation flight algorithms and STEM education within the Zero Robotics Program. The reinforcing loop (R1) and balancing loops (B1-B3) have been marked in the direction of their flow. The clouds that some flows lead into or out of indicate infinite sinks or sources of stocks. No cash flows have been considered.

There is one reinforcing (R1) and three balancing loops (B1, B2, B3) in the model. The reinforcing loop mathematically shows that the net increase of any variable in the loop will lead to exponential increase in all the variables, in the absence of a countering decrease in any other variable due to something external to the loop [100]. R1 i.e. the loop containing Software Development Rate, Software Developed, Successful Outreach Rate and STEM Interested Students, implies that increasing the user engagement software will exponentially increase the number of STEM interested students and vice versa. Since balancing loops follow S-shaped dynamics, the B1 and B2 loops indicate that with increased crowdsourcing and STEM engagement programs, the number of disinterested students and the amount of user engagement software required will decrease and saturate at a minima. B3 is the main loop that can cause R1 to activate positively, thus causing the exponential *increase* effects described earlier. The B3 loop contains Crowdsourcing & Engagement Programs, Software Development Rate, User Software Available, FF Problem-Solving Rate and FF Unsolved problems. Therefore, *increasing* the Crowdsourcing & Engagement programs will cause everything until the problem solving rate to increase (due to a chain of positive causality) followed by a decrease in the number of unsolved FF problems and then a *decrease* in the programs required to address them. This balancing loop is in keeping with the supply-demand balancing loop observed in business where the stock of FF problems to be solved corresponds to demand and the stock of user engagement software available corresponds to the supply. This means that the former should decrease and the latter should increase till they balance each other out in an S-shaped curve of fall and growth respectively. However, the stock of unsolved problems is also partially the integral of FF problem Discovery Rate, a flow variable external to this loop. Since this flow is likely to remain constant or increase (due to growing distributed systems), the stock of unsolved problems is not likely to decrease or saturate out. The B3 loop will always be dynamic, and the increase in 'User Engagement Software Available' will cause the R3 loop to trigger, causing increased number of STEM interested students.

From the systems dynamics behavior of the variables of interest discussed above, the broad policy direction to take in order to increase the rate of solving formation flight problems for distributed satellites and increase the number of students interested in CS-STEM topics would be the:

1. Development of programs where crowdsourcing and student education are done simultaneously i.e. increase the variable "Crowdsourcing & Engagement Programs"

2. Open up modular sections of satellite software development for students to play with through the above programs i.e. activate the causal link between "Crowdsourcing & Engagement Programs" and "Formation Flight Unsolved Problems"
3. Introduction of hands-on real world projects within school curriculum and through after-school partnerships i.e. activate the causal link between "User Engagement Software Available" and "Successful Outreach Rate"
4. Development of online interaction tools for learning and creative development i.e. activate the causal link between "Crowdsourcing & Engagement Programs" and "Software Development Rate"

The above policy directions, however, are high-level and theoretical. The next section discusses their potential implementation and the associated concerns, based on literature review and lessons learned through the development, operation and analysis of the Zero Robotics program.

6.2. Management and Policy Concerns for Crowdsourcing and STEM Education

The management and policy concerns in implementing the recommendations proposed in Section 6.1 can be categorized as those centered abound crowdsourcing spaceflight software and those centered around CS-STEM Education using hands-on, real world projects. As demonstrated in Chapter 5 and its conclusions, the performance objectives for crowdsourcing and education are different: for the former, it is the best solutions that are reflective of value while for the latter, it is the mean and overall distribution of solutions that determine value. Collaboration serves the purpose of keeping the best solutions intact and improving while bringing more students onboard with the best and inspire them further. The following sections discuss the two objectives separately, on the basis of existing literature, implemented projects and lessons learned through Zero Robotics.

6.2.1. Collaborative Crowdsourcing

Crowdsourcing comes with the associated baggage of ***setting up software infrastructure*** such that crowds are able to simultaneously work toward the given problem and submit solutions to it. For

ZR, this was the development of the web infrastructure (Chapter 3) which took 6 months of concentrated effort, followed by a year of improvement at a total contract cost of over $600,000. ZR was a DARPA-sponsored effort to demonstrate crowdsourcing [85]. For commercial spaceflight companies or agencies to justify this capital investment versus hiring staff to solve the problems managerially will require life-cycle assessment of the benefit to cost ratios over the program's lifetime and proof that the net present value of the system will be positive at the end of life.

During the operations phase of crowdsourcing, even in the presence of a very attractive interface, the ***incentive structure*** for participation has to strong enough to attract a healthy crowd to compete and collaborate at solving the given problem. For ZR, the incentives provided to the students was the opportunity to run their creations on real SPHERES hardware on the ISS, visit MIT to watch the games live from the ISS, meet astronauts in person (at no cost to MIT) and SPHERES merchandise as goodies at the end of the tournament (if they completed the surveys) as discussed in Section 5.2 and Figure 60. Note that programs that crowdsource spaceflight software with the facility to test the best algorithms in space have the advantage of conducting the full robotics software development process, right from conceptualization to hardware verification and validation (Section 6.2.3 and 6.2.4 for ZR), through crowdsourcing itself. Additionally, the unique advantage of combining a crowdsourcing program with a STEM education one is that students and educators have the added incentive of wanting to learn through the program, which has been observed to be a very strong one (Figure 60).

Open source encyclopedia development is an important example where crowds of people have successfully created large databases of information without having an intuitive incentive to do so. For example, Wikipedia succeeded in the creation of an online, open-sourced encyclopedia on a worldwide scale while all others who had tried to achieve the same since the early 1990s, such as Interpedia, the Distributed Encyclopedia, h2g2, the Info Network and GNUpedia, had failed. The reasons behind this success have been researched [101] to be that Wikipedia attracted contributors because it was built around a familiar product — the encyclopedia, focused on substantive content development instead of technology and offered low transaction costs to participation. Following the same footsteps for crowdsourcing spaceflight software, ***to attract contributors***, the materials provided on the crowdsourcing website should be such that a participant can come up to speed with the problem easily i.e. build on familiar skills, be open access, provide an appropriate mix of

collaboration and competition and provide real-time performance feedback such as leaderboards to keep the enthusiasm of the competitors. As explained for the ZR web interface development effort in Section 3.5.6.1 and Section 3.5.6.2, the ratio of submissions to registrations for our commercial crowdsourcing effort was ~15% (Figure 22). As shown in Section 5.2, the ratio of submissions to registrations for the 2D competition in the ZR Tournament operations was 72% and for the 3D competition #1 was 63%. Therefore, it is important not only to attract contributors but also to ***engage*** them enough to climb the learning curve and submit a valid solution. Also, the efficiencies less than50% should not be interpreted as failure. While all efforts for retention should be made, crowdsourcing is a tool to identify people who are genuinely interested in and capable of solving the problems i.e. self-selection tool. Therefore, it is expected that as long as the number of registrants is healthy and the problem has gotten attention, self-selection will ensure that at least some solutions submitted are expected to be quality ones. This allocation of appropriately interested solvers to specific problems would be difficult through managerial assignment. ZR (like TopCoder) leaves it to the crowd of participants to find the best man for the job.

As pointed out in the summary of Chapter 3, the ***management overhead*** is significant for software development through crowdsourcing. In context of the NASA Systems Engineering development process as shown in Figure 71[102], crowdsourcing is very helpful for some of the green boxes i.e. the Technical Development Processes, but adds a large overhead for the orange box i.e. the Technical Management Processes and the project integration processes. Figure 71 shows the development processes only for a single phase, the NASA system model describes the same structure for every phase through Phase D. An analogy may be drawn between the green blocks of

Figure 73 with the software development cycle in Figure 17. The modular nature allows crowds to work on a specific module through a contest and pass on the results to the next block in the sequence for the next crowdsourcing contest, to be addressed by a differently specialized crowd. However, for each contest, the 'technical management processes' include detailed pre-planning of exactly what gets developed though the contest, drafting detailed documents of requirements for the contest so that crowds do not dissipate, make previously developed modules available with strict interface requirements so that newly developed solutions can complement existing ones, assess the submitted solutions, decide on the best ones to push forward, configure the solutions to integrate with the existing infrastructure and manage the large amounts of data associated with crowds of

participants and their solutions. The biggest drawbacks of crowdsourcing, as mentioned in Chapter 3 summary, are schedule delays and integration errors due to the above management overhead. An important example of such inefficiency was seen in the Boeing 787 Dreamliner aircraft development, which was assembled from parts contracted out to scores of international companies. The delays went on to affect costs and were eventually categorized as "unk-unks", aerospace jargon for "unknown unknowns" [102]. While the Dreamliner was essentially hardware development and this thesis deals with software development only, the example serves to illustrate that programmatic efficiency is as important to a quality product as technical efficiency. Therefore, before resorting to crowdsourcing methods for problem solving, it is important to assess whether the value gained through the crowd creative design process is enough to justify the overhead.

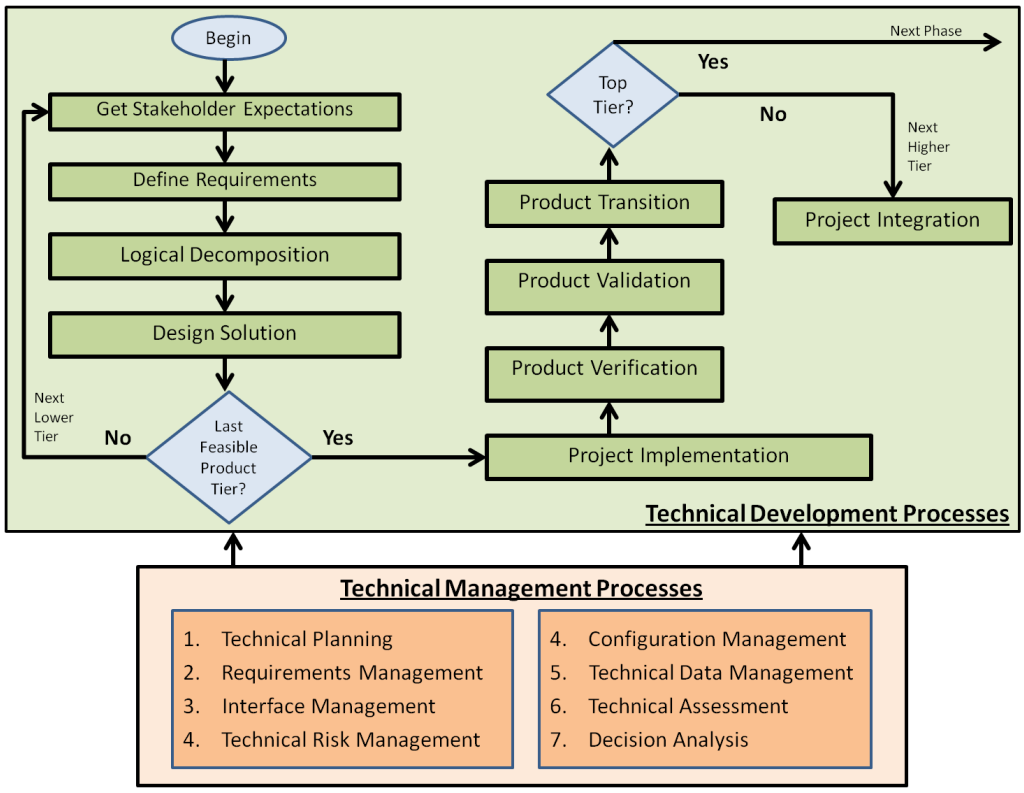


Figure 73: Development and Management Processes of a phase of the full NASA system life-cycle for a mission. These processes are applicable for Pre-Phase A: Concept Studies, Phase A: Concept and Technology Development, Phase B: Preliminary Design and Technology Completion, Phase C: Final Design and Fabrication, Phase D: System Assembly, Integration, Test and Launch but not for Phase E: Operations and Phase F: Closeout Adapted from the Systems Engineering View of the NASA Project Life Cycle Process Flow for Flight and Ground Systems [103]

Since a large section of space development efforts are protected by the International Traffic in Arms Regulations (ITAR) 2011 [104] and the Arms Export Control Act [105], one of the critical showstoppers for the ZR program,especially as it involves testing outside software on real flight hardware, is ***government security regulations***. For ZR, the participants were allowed to write code only within a given template, shown in Figure 11, which was compiled and simulated on the cloud and results sent back to participants in the form of an animation shown in Figure 9. At no point in the tournament were participants allowed access to the SPHERES embedded system code, low level algorithms or the programmable game code – these were a "blackbox" available to the MATLAB simulation as *mexed* (MATLAB executables) files, as seen in Figure 8. Moreover, SPHERES is a facility owned by NASA Ames Research Center but operated by MIT (an educational institution which does not conduct protected research), open access to the SPHERES software and permission to run code on the SPHERES hardware is already possible by an established program called the Guest Scientist Program [86]. To operate under the regulation constraints, spaceflight companies or agencies will either have to obtain open access licenses for modular sections of their code that they wish to open for crowdsourcing and be careful about keeping the protected parts of the code within a software "blackbox". The regulations are even tighter if the crowdsourcing competitions are open to foreign nationals.

6.2.2. Collaborative CS-STEM Education

This section will explore the management policy issues associated with using real-world, hands-on spaceflight software projects at a level of difficulty that cannot be easily solved by adults in the software field.

Studies by the Lifelong Kindergarten (LLK) group at MIT, one of the most influential educational research groups in the world (inventors of the programmable Lego brick and Scratch), have shown that students are largely motivated by peer pressure within a team [5]. Literature review in Section 2.3 endorses this view. Therefore, CS-STEM learning is most effective when students collaborate within teams. LLK studies have shown that participants in projects like to collaborate not only within their team but also outside of it. The top response to the question of what motivates students in projects was that they could not let their group down. Students want to join respectable

groups and work toward establishing a social status in the community. Similarly, when students were asked why they joined a company, many answers mention "fame," "credit" or "reputation". In an intriguing parallel, for the scientists studied, reputation is the prime motivator. In the ZR framework, Figure 67 and Figure 65 show how important collaboration *within* a team was to the educational experience. Also, a prime reasons participants offer for dropping out of open registration ZR events currently underway (where registration as individuals is allowed) is the lack of team pressure. Therefore, ***team-based learning*** where extra effort is invested in reinforcing the ***social status of CS-STEM topics*** is invaluable to its education. In ZR, this was accomplished by allowing exposingparticipants to astronauts and techie celebrities, who strongly conveyed to the participants how "cool" they find their work and profession in general. Development of educational programs based on existing spaceflight programs can benefit from playing up the "coolness" aspect to reinforce peer social status.

In the context of team work and collaboration, it is very important to note that new research [106][107] shows that many educational activities such as creative writing, reflection and generation of ideas is best possible at some degree of isolation with brainstorming used for refinement. Quoting David Brooks from the NY Times [108], "*The most important and paradoxical fact shaping the future of online learning is this: A brain is not a computer. We are not blank hard drives waiting to be filled with data. People learn from people they love and remember the things that arouse emotion. If you think about how learning actually happens, you can discern many different processes. There is absorbing information. There is reflecting upon information as you reread it and think about it. There is scrambling information as you test it in discussion or try to mesh it with contradictory information. Finally there is synthesis, as you try to organize what you have learned into an argument or on paper.*" While online educational programs (e.g. MIT and Harvard's EdX) help the first step of absorbing information and the team dynamics help the third step of scrambling information, it is isolation and self-study that helps the second and fourth steps of reflecting upon and synthesizing new information. Therefore, while promoting team work, CS-STEM education programs should adopt a ***spiral model of isolation and human friction*** and incorporate both team-based and individual components of learning[84].

The ZR experience has shown that the ***team mentors*** played a critical role in the educational success and performance of their teams, as explained in Section 5.2.3 and shown in Figure 67 and Figure 69. Note that the 2011 tournament introduced proxy crowdsourcing problems. As real and

harder problems are introduced to students, they will rely largely on mentors to help them come up to speed from basic math and physics to concepts of satellite control engineering. Absence of a guiding hand to help climb the learning curve is expected to result in loss of interest of the team, as indicated by some teams in their feedback.

While the ease of access to thousands of online learning tools may suggest that technology-proficient students can self-manage with minimal support from teachers, educational research suggests that this approach is not sufficient and that support should begin with ***the teacher*** [109]. In fact, additional technology support and learning opportunities directed to teachers resulted in greater technology integration in their teaching practice and enhanced effectiveness of teaching. Teachers have the pedagogical experience necessary for meaningful integration, which may be lacking in students [110] and favorable teacher attitude toward technology increases the likelihood of its adoption by students [111], which further emphasizes the need for initiatives to make teachers comfortable with technology. Most importantly, collaboration among teachers has proven to be one of the most effective methods of teacher integration. A blend of online and face-to-face interactions between teachers serves to mutually reinforce the development of relationships, understanding of practice and building of capacity among teachers [112]. Since ZR was a standalone program with no support provided from MIT to the participating teams, some of the ways we tried to ensure 'teacher support' was to ask for the team's commitment in the application form that they have identified at least one mentor who would meet with and help them through the tournament. The mentor was required to be a professional affiliated with the team's school with some background in programming. The ZR website had discussion forums for all mentors or students (discussed in Section 5.2.4) to congregate and discuss anything they wished to about the game and program. Finally, all participants were invited to the ISS Finals event at MIT, as described in Section 4.1.2.3 so that they could meet each other and reinforce the relationships built online.

Based on the above lessons and literature, in order to foster real world project based learning and programs, government, NGO or individual efforts for the ***professional development of teachers*** is extremely important. The questions to be thinking about, in this context, are: How can we enhance the ability of teachers to provide STEM education? How can the design of communities, gatherings, and resources enable teachers to understand and employ design-based approaches to the cultivation of computational thinking? Some suggested solutions are rigorous documentation of the

programming environment and context of the program, an online community for teachers working on or interested in the program, and workshops and face-to-face gatherings where teachers can gain a deeper inclination for ZR-like programs.. Teacher engagement may be implemented through federally funded initiatives, regular school initiatives or after school programs. For example, the Massachusetts Afterschool Partnership (MAP: http://www.massafterschool.org/about.html) is an NGO that provides teacher-training workshops and improves lives of youth through statewide policy development, local grassroots networks, education and advocacy and strategic public-private partnerships. MAP is also helping develop a ZR handbook such that middle school educators can learn individually and collaboratively and be able to teach students project-based programming without MIT's help. On the government side, there is a need for support from the Department of Education for math and computer science teachers such that we can encourage more qualified professionals to become information and communications technology (ICT) teachers and offer a national program of continuing professional development (CPD) to enhance the teachers' skills.

ZR demographics, in Section 5.2.2, showed that the female fraction among participants is ~10% and racial minorities fraction is in keeping with their participation in US STEM fields as well. The numbers imply that the program, *statistically,* is not gathering STEM interest from ground zero upward but *is* inspiring those who already are somewhat interested *much further*. When building a CS-STEM education program, it is therefore necessary to step back and ask what the end goal is.

If the goal is to improve the quality of students already interested in CS-STEM, then after school programs and voluntary participation, open registration programs with real-world, hands-on hard problems supported such as ZR by strong teachers is a great idea. In fact, a recent report released by the Technology and Innovation Foundation on fresh approaches to CSTEM Education [74] stated that, "*Getting 5 percent of the workforce to be STEM proficient does not require STEM education for everyone, everywhere, all the time.* ***Focusing on fewer individuals*** *allows the luxury of building a "new and improved" pipeline that emphasizes mass customization of content, development of innovation-era (rather than production-era) skill sets, and frequent industry engagement with the application and practice of those skills."*

On the other hand, if the goal is to convert more students from being disinterested in STEM to interested in any capacity, then ***change in the school curriculum*** is required. Computer science, i.e. the basic knowledge of computing and the use of computers to solve problems more complicated than math on paper or calculators, should be included in the basic middle and high

school curriculum along with math, science, English, social sciences, health and physical education instead of an elective or an AP course. Students should be taught computer science as a language in which the modern, digital world works – a language necessary to contribute and communicate to a technologically advanced society [1]. Quoting a manifesto submitted to the Secretary of State for Education in the United Kingdom [113], "*We teach elementary physics to every child, not primarily to train physicists but because each of them lives in a world governed by physical systems. In the same way, every child should learn some computer science from an early age because they live in a world in which computation is ubiquitous*." It is only through learning the basics of the subject at a young age, mandatorily, that students will grow their interest through participation in hands-on, project based programs with friends and peers creating a positive feedback loop. Girls and minorities will then grow up to relate to CS-STEM as a basic part of their curriculum rather than something that only the studious students *elect* to do among themselves. In fact, in one of the more famous papers on women and computers [114], Sherry Turkle claimed that computers can provide people with the power to achieve their potential to a degree beyond what cultural norms dictate - "*The practice of computing provides support for a <epistemological> pluralism that is denied by its social construction*". There needs to be just the right push in the right direction to learn the language to make this realization possible.

In launching CS-STEM programs based on real-world projects, the management team should ensure that the projects they make available are not just grunt work or number crunching. While such projects may have a low learning curve, they do not ***contribute to the students' 21st century skills*** [50][51] as introduced in Section 2.3 in Chapter 2. The unique advantage of combining crowdsourcing efforts with STEM Education is that the students get to work on real 21st century projects which has the ability to tap into "principles of effective learning" [115]:

- Authentic learning - learning from real world problems and questions
- Mental model building - using physical and virtual models to refine understanding
- Internal motivation - identifying and employing positive emotional connections in learning
- Multi-modal learning - applying multiple learning methods for diverse learning styles
- Social learning - using the power of social interaction to improve learning impact
- International learning - using the world around you to improve teaching and learning skills.

It is important that the management opens up enough of the problem to students such that they get a holistic picture and contribute meaningfully, instead of only writing code that solves a single specific isolated problem. Programs that teach 21st century skills, allow for social creativity by

fostering low barrier to entry and socio-emotional communication and play, provide students enough time to define their area of contribution and provide flexibility for students to develop their own computational modules within the program will go a long way in creative education.

Finally, the more immediate question to address is this: what are some of the ***regulatory restrictions*** that can help open up real-world spaceflight problems to teenage students in a way that they can contribute applicable solutions? ZR in 2011 could not be organized as a fully open program since only US schools were allowed to apply for the main program and schools from 3 countries in Europe handpicked by the European Space Agency were allowed to apply for the EU Pilot (since the funding sources were different) – DARPA sponsored the US part of the program and the European Space Agency (ESA) sponsored the EU part. The ISS finals were conducted separately (as separate round robin brackets as detailed in Section 5.1.3) for the US and EU groups since astronaut time was individual responsibility of NASA and ESA respectively. It is expected that as ZR matures into a standalone program with industry sponsors, open registration may be allowed for the simulation competitions. Astronaut time for ISS operations is something would still be required to work out. Since a lot of spaceflight development is sponsored by government agencies, programs that open it up for STEM Education will have to consider the respective policy implications of doing so nationally and internationally.

6.3. Management Policy Implications Summary

This chapter reinforces the benefits of developing programs that simultaneously crowdsource cluster flight software and foster CS-STEM education by opening up crowdsourcing competitions to students and allowing them to learn through engagement with real problems. Previous chapters have established that by developing the appropriate web framework and designing games and tournaments around real-world problems, high school students can be made to contribute satellite software for complex navigation and control algorithms. Feedback from and performance analysis of the tournament has shown that this framework has also helped students derive positive educational value. Additionally, this chapter introduced a system dynamics model that justifies how the framework is beneficial to crowdsourcers, students and educators.

Furthermore, using the model, four high-level recommendations have been made for spaceflight software development and the education community. Based on past literature and lessons learned through ZR operations, potential implementations of the recommendations have been discussed. Some of the important spaceflight crowdsourcing concerns are capital investment required to create a web infrastructure framework so that crowds can contribute to the problem statement, an appropriate incentive structure such that a significant number of people participate in the contests, the management overhead associated with solving problems using crowdsourcing versus using managerial assignment and regulation restrictions. Some important CS-STEM education concerns are development of real-world project problems that promote team-based and individual efforts, professional development of teachers and mentors who can help the teams, revision of the school curriculum to include computer science mandatorily so that more high school students are motivated to solve real-world project-based problems, promotion of 21st century skills and regulatory constraints.

As an end note, selecting the right problems for such dual objective programs is fundamental to achieving the objectives. The temptation to solicit solutions to very difficult problems that require highly specific skills has to be actively curtailed by the crowdsourcers because they will have too high a learning curve to be motivating enough for a crowd, especially amateurs. Furthermore, they will not be holistic enough to be educationally valuable to participating students. Similarly, problems that have a very low learning curve but need crunching of large data sets or simply trial and error attempts might also be of use to the crowdsourcer but will not be able to capture or retain the students' interest in the problem or in STEM. Therefore, problem selection has to be very judicious, and while this thesis has shown that it is possible, the onus lies of the program designer to ensure that it is indeed so.

Chapter 7 –
Conclusions

This thesis demonstrated, using a proxy cluster flight problem, that it may be possible to crowdsource a real spaceflight software problem by designing an appropriate game around it, scoring it correctly and making it available to crowds using a robust web infrastructure. The best solutions have been tested on flight hardware in space. When the crowds are students and the problem and game chosen to engage them in CS-STEM subjects and 21st century skills, positive educational effects are seen. Further, if a framework of collaborative competition is introduced in the right manner, not only do the solutions improve in quality and number, the student teams get the opportunity to cooperate with and learn from the best, and therefore are capable of achieving an average that is more than the sum of individual teams.

7.1. Research Statements Revisited

This section will revisit the research objectives introduced in Chapter 2 and summarize the findings of the thesis in their context:

1. ***Proof of concept that crowdsourcing of cluster flight problems as well as CS-STEM Education is possible using the same program***

We used the ZR program in 2011 to demonstrate end to end crowdsourcing capabilities by first building a web infrastructure through commercial crowdsourcing contests and then using the tournaments hosted on the infrastructure where thousands of students can participate to contribute to developing cluster flight algorithms (Figure 7).

The web infrastructure included the programming interface/user integrated development environment – text and graphical editor, team and project management tools, administrator tools for organizing tournaments and competitions, the website, tutorials, online support and discussion forums. The commercial crowdsourcing effort led by TopCoder that produced the ZR program infrastructure was studied as a case study. The methodology of breaking development up into

parallel and sequential contests, their integration, incentive structure, evaluation criteria and collaboration types in the contests were highlighted. Over 6 months until December 2011, i.e. at the end of one tournament run on the new web infrastructure, the contests received 857 registrations, 149 full submissions and 57 prizes were awarded. There have been a total of 239 unique participants in the 54 contests in 6 months, for a total cost of ~$186,000. In a traditional office set-up, only 4-5 people could have been hired at this cost, crowdsourcing buys diversity and multiple times the number of work-hours at a fraction of the cost. While a low submission to registration ratio was observed for some contests and special effort was executed (e.g. by increasing prizes or advertising on TC forums) to motivate submissions by strong competitors, the low ratio of ~15% in itself should not be perceived as failure. Crowdsourcing is designed to attract attention from dozens of interested people, incentivize submission of solutions from the truly motivated and to select the top solutions from the very best. Therefore, it is important to attract enough attention (#registrations) and retain the loyalty of strong members. As seen in Figure 24 and Table 3, the most loyal members from 4 contest categories claimed between 25%-100% of the prizes in that category. In fact, the 11 highest earners among the 90 total winners in all contests claimed 62% of the total money spent on all the payments. While this seems to favor partial monopolization of a market that is inherently supposed to be competitive in order to produce quality, the caveat is that the groups of people who dominate the contests are self-chosen from all around the globe, who have competitively established their position through the process of crowdsourcing. It would been much harder, if at all possible, to find such a match by looking locally for such a candidate, hiring him full-time and managerially requiring that he keep up his standards of work. For the same reason, creative and abstract tasks are awarded more than direct skill-based tasks (Figure 23) – to tap into a global, diverse pool of creativity hard to find locally. Survey feedback from alumni showed they preferred the newly developed website (63% positive response) and IDE (75% positive response). Users had complaints regarding instability and delays in bug fixes. On the administrative side at MIT, the time taken to finish tasks was much longer than if the task was managerially assigned to appointed software developers – a frustrating experience for a critical path schedule. One of the most important lessons learned through the case study was that crowdsourcing worked much better for creatively architecting and designing solutions than for simple, time-critical implementations, where a managerial approach would work better.

ZR Tournaments are used as a crowdsourcing tool by 'gaming' a cluster flight problem i.e. writing a game code around the problem. The game code contains game API function definitions (Figure 26). Users can play the game by using the SPI functions within their C programs. The game code and user code together form the autonomous software that controls the SPHERES by interfacing with its embedded system code. Users can therefore solve flight related problems by playing a game. The 2010 tournament, although not specifically designed for crowdsourcing, showed that the game scoring should be designed such that the scores reflect accurately the quality of the crowdsourced solution and there is fine and quantitative resolution between the hundreds of solutions submitted. There were only 3 levels of resolution in scoring in 2010 (Figure 39, Figure 40). The 2011 tournament was designed around a much harder, formation flight problem with a wide range of score distributions that prorated the efficiency of the solutions (Figure 41; scoring described in Section 4.2.1). Perfect solutions to the proposed problem were received from student teams in simulation (Table 5) and aboard ISS hardware. Satellite telemetry analysis of the ISS tests showed that more than 90% of the submitted algorithms performed within acceptable efficiency levels of SPHERES scientific research (Figure 47, Figure 48,), greater than 80% were within acceptable levels of robustness i.e. performed similarly in ISS and simulation (Figure 51, Figure 52). Moreover, the top players achieved the game objectives within 23% of allocated fuel and 76%-90% of the algorithms were within efficiency levels when calculated at acceptable errors twice as strict as acceptable SPHERES research (Figure 49).

In terms of educational objectives, the student users in the ZR tournaments found games and competition exciting and therefore learned math, science, strategy and programming while solving real-world problems by playing games (Section 4.3.1). The program has seen participation grow by 241% over the previous year (Section 5.2.2). Above 85% mentors and students have reported significantly positive improvement in CS-STEM and leadership skills, with moderate to strong correlation in opinions (Figure 59). ZR 2011 has fulfilled the motivations of the participants (Figure 60) and the predicted retention rate is approximately 89%. The program has therefore established that it is possible to successfully solicit precise formation flight solutions from a crowd of students while at the same time educating them in CS-STEM using games.

2. ***Analyze the effects of participant collaboration on both crowdsourcing and CS-STEM Education***

Three collaboration environments were introduced in the ZR 2011 tournament and their impact on crowdsourcing FF solutions and CS-STEM determined (Table 4). Since all effects were interpreted using quasi-experimental analysis of passively observed data, and not an active experiment, special care was taken to guard against sources of invalidity and unreliability (Section 4.3.3). The three collaboration environments introduced were in-game collaboration between opponent players during a match, inter-team collaboration within mandatorily formed alliances and inter-team collaboration on online discussion forums. Crowdsourcing benefits were measured in terms of match scores in competitions during the tournament, specifically the best algorithms submitted and their efficiency and robustness of performance on the ISS testbed. Education benefits were measured in terms of reported program satisfaction, increased STEM inclination, improvement of 21st century skills [50] and match scores in competitions, specifically the improvement of individual team performance and the overall average.

The 2010 tournament showed that the right tail of the histogram distribution of performance *reduced* compared to the left tail during the course of the tournament (Figure 39 compared with Figure 40) when the game was adversarial i.e. decrease in crowdsourcing value. The 2011 game was therefore designed to be collaborative in nature – that was the *only* way to maximize scores. The game was designed around a hard and relevant trajectory tracking problem of spinning or revolving a SPHERE at a particular angular velocity, position and orientation (Section 4.2.1). The perfect score in a competition (not only a match) was possible only if a player had a perfectly collaborating opponent to get the best out of all the available resources, perfectly optimized strategy of war-gaming and perfect control algorithm for trajectory tracking of the SPHERE. Participants utilized the discussion forums very efficiently to come up global communication protocols that resulted in multiple perfect solutions over multiple matches (Section 5.2.2.2). Introduction of alliance-based collaboration showed an improvement of the mean score by greater than one standard deviation (Figure 43, Figure 44), no change in the number of perfect solution but an improved number of demonstrations of the perfect solution (Table 5).

On the education side, the introduction of alliance-based collaboration in 2011 increased the overall performance of teams (Figure 61 and Figure 63) by 3.4 points on a 0-23 point scale. However, contribution of teams to their alliance project varied greatly depending on which tier they were

selected from during the alliance formation process. The average self-assessed contribution on a scale of 0 to 1 to the alliance projects of the Tier (1, 2, 3) was (0.909, 0.361 0.477) respectively, based on post-tournament surveys. Correlating contribution with improvement of team ranks after the introduction of alliances shows a very weak correlation (Figure 62). Tier 3 teams improved the most but claimed to have contributed the least and conversely, Tier 1 teams improved least and claimed to have contributed most. Moreover, the variance in performance within alliances showed nearly no correlation with performance improvement of the alliance (Figure 64). We leant several lessons about improving the implementation of the collaborative infrastructure to improve performance and learning. In future editions of the program, the game will likely be notched up in difficulty after introducing alliances and tools for equal contribution will be available. This will create more opportunity for each team in an alliance to contribute. Moreover, the method of forming alliances will be revisited such that teams of similar capabilities may be able to work together and no team feels left out. While the formation of alliances has apparently increased the overall performance of the participants and received approval among the participants, there is room for improvement through the revision of the *methodology* of grouping teams into alliances. Greater than 63% of the student respondents in the post-tournament survey found the game '*challenging and exciting all through*'. The discussion forums were educationally popular and logged a total of 5150 messages by 164 unique users in the entire tournament period; students who were participatory and frequent at the forums tended to do well (Figure 66). Participants attributed positive educational influence to *all* the collaborative features in ZR (Figure 67). Intra-team collaborative features were better received than inter-team features as indicated by their differential preferences (Figure 68, Figure 69), albeit at varying degrees of statistical significance (Table 6). Collaboration therefore proved to be beneficial overall to expand the outreach of the program and improve results, however revisions in implementation of some environments is necessary.

3. Recommend management policies for Spaceflight Software Development efforts combined with Education efforts

The framework proposed in ZR was modeled using Systems Dynamics with three levels of stocks and flows (Figure 72) – for user software design to enable crowdsourcing, for crowdsourcing of cluster flight software and for CS-STEM education of students. The model inferred that it is beneficial to both the scientific and educational communities if web infrastructure that serves the

needs of both communities is developed and deployed. The main challenges to STEM Education have been identified as building teacher capacity to support such programs, integration into the school curriculum so that participation from amongst the less STEM inclined students is also possible, an appropriate mix of scope for team and individual efforts, launching problems of appropriate difficulty levels and regulatory constraints. The biggest challenges to crowdsourcing are justification for capital investment and management overhead, and devising appropriate incentive structures for the framework to work. The performance objectives for crowdsourcing and education are different: For the former, it is the best solutions that are reflective of value (the right tail of the histograms in Figure 39, Figure 40, Figure 41, Figure 43, Figure 44) while for the latter, it is the mean and overall distribution of solutions that determine value (Figure 61, Figure 63). Problem selection and the method of 'gaming' around it should be very judicious. Students should not be treated as data crunchers, the learning curve to find the perfect solution should not be so high that the mean of the performance distribution falls – degrading education. Scoring should be appropriate enough to judge the quality of the top solutions incrementally in terms of crowdsourcing performance metrics and with fine enough resolution that important differences do not slip through the cracks (Figure 40 versus Figure 41). Collaboration serves the purpose of keeping the best solutions intact or improving them while bringing more students onboard with the best to inspire them further. Correct implementation of collaboration can therefore exponentially kick off the reinforcing loops in the systems dynamics model (Figure 72) and improve the experience of participants (feedback surveys in Section 5.2.3).

7.2. Limitations and Future Work

Studies in this thesis indicate that there is a case for combining collaborative competition, crowdsourcing and STEM education.

The essential ingredient is to find a balance between the needs of both stakeholders: scientific community and educators/students. The scientific community would want crowdsourcing contests to target specific skills, which might be too focused an approach for educators who want to introduce students to holistic 21st century skills. The education community might want something exciting, dynamic and high level which may not be of research interest to the scientific community.

From the "loyalty in crowds" lesson learned in Chapter 3, it is important to remember that from the crowdsourcer's perspective, everyone in the crowd need not solve the problem. In fact, it is sufficient if a large group of people is interested enough to try and solve the multiple sub-problems till an overall solution emerges. This allows for selection of talent from a large pool but retention of only the truly motivated and capable. An educator's perspective would want maximum retention of students. Problem definition and game design hence needs to be formulated with both stakeholder interests in mind so that middle ground can be achieved. Collaboration helped the dual objectives by improving overall performance of teams (Figure 43, Figure 61), number of perfect demonstrations in matches (Table 5), reported educational gain (survey results: Figure 67) and ability to solve a difficult formation flight problem in simulation and hardware (Section 5.1.2.3).

Can students really solve problems that scientists cannot solve?

This is a common question that arises when justifying the basis of the thesis. The first thing to note is that crowdsourcing is not being pitched for just those problems which scientists *cannot* solve. The process may be used if the potential crowdsourcer does not have time and/or resources within this organization to solve a problem, a subset of a problem or even help with solving a problem. It is a method to find the right candidates to address an issue by using an open call and then select the right solution from the submissions pool.

HS Students have demonstrated through ZR that, when mentored appropriately and with the right software tools available, they can even outperform MIT undergraduate students. Their solutions have demonstrated efficiency (>90% of submitted players, Figure 49) and robustness of control (>80% of players; Figure 51) for precise formation flight even in random noise levels aboard the ISS. In fact, the top solutions achieved the cluster flight game objectives using less than a quarter of the fuel allocated to them (~23%) and 80-90% of the players were certified efficient even when calculated using acceptable error levels twice as strict as usually acceptable in SPHERES research (Figure 47, Figure 48, Figure 49). This indicated that a much tougher problem could have been solved by students within this year's program.

Finally, a major feedback we received for alliance-based collaboration was that teams found there wasn't enough work to distribute among all three collaborating teams. This was the primary reason why high performers dominated the project finalization and lower performers felt abandoned. While

this is a lesson learnt in how best to formulate alliances and games for future years, the important point here is that many teams felt they could have contributed *much more* than what they did. Hence, given the alliance environment, more difficult problems are expected to be welcomed as a challenge by participants.

Does crowdsourcing not entail a waste of resources if only one solution is used?

The ethics of crowdsourcing is an important concern raised in literature [72] because crowds put in time and effort into submitting solutions to problems after which only a small subset (sometimes only one) solution is finally used. Effort is not wasted if crowdsourcing programs are additionally used for educational purposes. All participants learn and gain from the experience of solving real-world problems. Introducing collaboration into the framework further reduces wasted effort because it entails combining many good solutions into an integrated one. Therefore, the concept of simultaneous collaborative crowdsourcing and education for cluster flight algorithm development mitigates one of the chief concerns associated with standalone crowdsourcing.

Going forward, participation in ZR's dedicated crowdsourcing tournaments can be closely monitored to calculate the percentage of students and educators among participants and their relative performance. If the numbers are high, it implies that once ZR (as a program and infrastructure) is introduced to young minds, they are capable of participating as crowds in full-fledged non-educational, crowdsourcing competitions and submit competitive solutions to real scientific problems. As a corollary, if tournaments were designed for both crowdsourcing and education, such bright sparks in the crowds of students could help solve problems they pick for themselves while the others learn from the experience and perform better the next time. The right implementation of collaboration will only strengthen the learning and crowdsourced solutions.

As a concluding note, I would like to stress on the importance of iterative evaluation in the development of any such program as described in this thesis. In ZR 2011, the objective and descriptive surveys taught us many lessons about how the program could be further improved. These surveys combined with performance trends and participation statistics were invaluable in devising modifications to the program to make it more effective in the coming years. The scientific and education community are equal stakeholders in the process and hence pre-program input and post-program feedback from both is vital.

Appendix A – Example of a ZR User Library of Game API functions

To play any ZR Game, apart from the ability to program in C within the IDE, students were provided a library of API functions to make their SPHERES perform the activities required for the game. Below is the library provided for the game 'AsteroSPHERES'. They have been categorized as per the different operations required to play the game.

1. **Zero Robotics Basic Functions**

This is the basic library required to make any SPHERE move within the game volume and can be called within the *ZRUser()* template, which is the main programming template available to the participants. They are available for any game, not just AsteroSPHERES.

- ***void ZRUser (float myState[12], float otherState[12], float time)*** - The main user code loop called at every iteration of gspControl (once per second). This function will be the main function available in each project and you will not be allowed to change its signature. The inputs, all in SI (MKS) units, are-
    - myState is a float array of length 12 which is the state vector of the user satellite [position x, position y, position x, velocity x, velocity y, velocity z, att_vector x, att_vector y, att_vector z, att_rates x, att_rates y, att_rates z]. The attitude and attitude rates are in radians and radians per second and NOT degrees.
    - otherState is a float array of length 12 which is the state vector of the other satellite [position x, position y, position x, velocity x, velocity y, velocity z, att_vector x, att_vector y, att_vector z, att_rates x, att_rates y, att_rates z]. The attitude and attitude rates are in radians and radians per second and NOT degrees.
    - time since the user code was activated (in seconds)
- ***ZRSetPositionTarget (float posTarget[3])*** : Sets an x, y, and z position target for closed loop PD position control. Cannot be combined with velocity control. Input: posTarget is a float array of length = 3 containing the position targets x, y and z.
- ***void ZRSetVelocityTarget (float velTarget[3])*** - Sets an x, y, and z linear velocity target for closed loop velocity control. Cannot be combined with position control. Input- velTarget

is a float array of length = 3 containing the velocity targets velocity x, velocity y and velocity z.

- ***void ZRSetAttitudeTarget (float attTarget[3])*** - Specifies a unit vector (called the attitude vector) for the satellite to point toward. The satellite will move its -X face (beacon/Velcro face) to point at the specified (length 3) unit vector. Note that the satellites are limited to a maximum angular speed of 60 degrees per second. Input- attTarget length 3 unit vector to point toward
- ***void ZRSetForces (float forces[3])*** - Sets the x, y, and z forces to be applied to the satellite. These forces will be added to any closed loop control forces commanded by ZRSetPositionTarget or ZRSetVelocityTarget so using both together may result in unexpected actuation. Input- forces length 3 float array of forces.
- ***void ZRSetTorques (float torques[3])*** - Sets torques around the x, y, and z body axes to be applied to the satellite. These torques will be added to any closed loop control forces commanded by the satelliteZRSetPositionAttitudeTarget so using both together may result in unexpected actuation. Note that the state vector does not provide any information about the Y and Z body axes and this may limit the usefulness of this API function. Input- torques is a float array of length = 3 containing the torques to be applied about the x, y, z axes.

2. Math Functions

This was the library provided to perform basic math operations within the SPHERE's program. They are available for any game, not just AsteroSPHERES.

- ***float mathVecInner(float *a, float *b, int n)*** - Returns the inner (dot) two vectors a and b each of size n
- ***void mathVecNormalize(float *a, int n)-*** makes the supplied vector ('a') a unit vector, where 'n' is the number of elements in the vector
- ***float mathVecMagnitude(float *a, int n)*** - returns the magnitude of the supplied vector ('a'), where 'n' is the number of elements in the vector
- ***void mathVecCross(float vout[3], float a[3], float b[3])*** - returns the cross product of a and b in the supplied vector vout
- ***void mathVecAdd(float *c, float *a, float *b, int n)*** - adds vector a to vector b and returns the result in vector c and 'n' is the number of elements in the vector

- ***void mathVecSubtract(float *c, float *a, float *b, int n)*** - subtracts vector b from vector a and returns the result in vector c and 'n' is the number of elements in the vector
- ***float mathSquare(float a)*** returns $a \cdot a$
- All the standard C math functions were be available

3. AsteroSPHERES General Functions

This library was available specifically to play the AsteroSPHERES game, in order to find out the status of the game (e.g. amount of fuel or charge remaining) and hence help achieve the objectives in the game (e.g. sending and receiving communication messages)

- ***unsigned char PgetPhase()*** - Returns the stage the game is in -- PHASE1 (1) for the first stage, PHASE2 (2) for the second stage, and PHASE3 (3) for the third stage.
- ***unsigned short PgetCharge()*** - Returns the remaining charge
- ***float PgetPercentFuelRemaining()*** - Returns the remaining fuel as a percentage of the total (0-100)
- ***void PsendMessage(unsigned short message)*** - Sends a message token to the other player with the specified message. If the input is a value outside the range of 1 to 65535, no message token will be sent.
- ***unsigned short PgetMessage()*** - Checks if a message token has been received. Returns the value of the last message sent by the other satellite, if any, or 0 otherwise.
- ***float PgetScore()*** - Returns the active player's current score.
- ***float PgetOtherScore()*** - Returns the opponent's current score
- ***unsigned char PoutsideBoundary(float position[3])*** - Returns true (1) if the satellite has exited the interactions zone, returns false (0) otherwise Inputs: position 3-element satellite position vector
- ***unsigned char PisAvoidingCollision()*** - Returns (1) if the internal collision avoidance algorithm activated within the satellite in the previous second. Collision avoidance takes over user controls for 3 seconds since the time it activates.

- ***unsigned char PatMiningStation ()*** - Checks if the satellite has reached any mining station. Returns 1 if station1 has been reached, 2 if station 2 has been reached and 0 if none have been reached.

4. AsteroSPHERES Asteroid Functions

This library was available specifically to play the AsteroSPHERES game, in order to find out about the states of the virtual asteroids in the game and their ability to be 'mined' as per the game rules available in the manual [92]

- ***void PgetAsteroidNormal(float asteroidNormal[3])*** - Returns the normal to the asteroid plane in the supplied vector asteroidNormal[3]
- ***unsigned char PinAsteroid(float state[12])*** - Checks if the satellite state is valid to collect points on an asteroid. Both position and velocity requirements must be met. Returns OPULENS (or 1) if the satellite is Opulens, INDIGENS (or 2) if the satellite is on Indigens, 0 otherwise. Parameter 'state' is the state of the SPHERE for which you are checking the condition
- ***unsigned char PisRevolving(float *state[12])*** - Checks if the satellite position is valid to collect points by revolving around an asteroid. Only position requirements must be met. Returns OPULENS (or 1) if the satellite is around Opulens, INDIGENS (or 2) if the satellite is around Indigens, 0 otherwise. Parameter 'state' is the state of the SPHERE for which you are checking the condition
- ***unsigned char PiceMelted()*** - Returns true (1) if Opulens' ore is no longer protected by a layer of ice, false (0) if the ice layer still exists
- ***unsigned short PiceHits()*** - Returns the number of hits on Opulens' ice sheet by the active player
- ***unsigned short PotherIceHits()*** - Returns the number of hits on Opulens' ice sheet by the other player

5. AsteroSPHERES Item Functions

This library was available specifically to play the AsteroSPHERES game, in order to find out about the states of the virtual items in Phase 1 of the game and whether they have collected as per the game rules available in the manual [92]

- ***unsigned char PdisruptorUpgraded()*** - Returns true (1) if the active satellite picked up the disruptor upgrade, returns false (0) otherwise.
- ***unsigned char PotherDisruptorUpgraded()*** - Returns true (1) if the other satellite picked up the disruptor upgrade, returns false (0) otherwise.
- ***unsigned char PhaveShield()*** - Returns true (1) if the active satellite picked up the shield, returns false (0) otherwise.
- ***unsigned char PotherHasShield()*** - Returns true (1) if the other satellite picked up the shield, returns false (0) otherwise.
- ***unsigned char PhaveLaser()*** - Call this to see whether the active satellite picked up a laser. Returns 1 for the first laser, 2 for the second laser, and 0 if a laser has not been picked up.
- ***unsigned char PotherHasLaser()*** - Call this to see whether the other satellite picked up a laser. Returns 1 for the first laser, 2 for the second laser, and 0 if a laser has not been picked up.
- ***void Prepulsor()*** - Activates the repulsor; pushes the other satellite by a particular amount in the direction in which you are facing it. Consumes charge.
- ***unsigned char PotherRepulsor()*** - Call this to find out if the active satellite is being repelled by the other player. Returns true (1) if you are being repelled, and false (0) if otherwise.
- ***void Ptractor()*** - Activates the tractor; pushes the other satellite by a particular amount in the direction in which you are facing it. Consumes charge.
- ***unsigned char PotherTractor()*** - Call this to find out if the active satellite is being attracted by the other player. Returns true(1) if you are being attracted and false (0) if otherwise.
- ***void Plaser()*** - Activates the laser if it has been acquired.
- ***unsigned char PotherLaser()*** - Call this to find out if your opponent is using their laser on you or on Opulens (i.e. either you or Opulens is within its laser attack cone). Returns 1 if they are using Laser 1, 2 if they are using Laser 2 and 0 if they are not using a laser.
- ***unsigned char PgetShieldStrength()*** - Returns shield charge remaining for whoever has the shield, and returns void if no one has it

Appendix B – Examples of Game Code from ZR 2011

The AsteroSPHERES game code was more than 1000 lines of code, and therefore too long to copy and explain within the limited constraints of this Appendix. The intent here is to demonstrate how a game is programmed so as to interface with the SPHERES embedded system code/software as well as respond to the user programs (ZR User Code) and commands, as shown in Figure 26 by means of examples from the AsteroSPHERES code. Two examples will be provided here, one for scoring the mining behavior of the satellites i.e. to demonstrate how the SPHERES movements are converted into scores, and the other to show the process of sending and receiving communication packets, i.e. to demonstrate how the SPHERES synchronize the state of the game and the others' states. There are several other such functions within the game code, such as to initialize the game variables, command the states of the SPHERES, calculate the pointing direction of a SPHERE, calculate the number of laser hits at Opulens or the Earth, check if items have been picked up, check when and which mining stations the SPHERES have reached, check the fuel remaining, calculate the total score and many more.

Mining Behavior Scoring

The following lines of code demonstrate how the game code uses the state of the SPHERE it is being run on and the state of the opponent SPHERE in the match (as available through state of health packets being broadcast) to calculate its game state. In this specific example the code uses ctrlStateMe (self state) and ctrlStateOther (opponent state), to calculate the position, orientation and angular velocity of both around the two virtual asteroids in the game and therefore calculate the score accumulated in every control cycle. The code uses many scoring constants, written below as macros.

```
//spinning angular velocity in degrees/s
getGlblAngVel(glblAngVel); //Convert to global frame
spin = (fabsf(mathVecInner(glblAngVel,gameInfo.normAster,3)))*180/PI;

if (gameInfo.phase >= PHASE2 && !gameInfo.me.collisionActive)
{//Spinning case
	if (PinAsteroid(ctrlStateMe) == INDIGENS)
	{
		basePoints = calcAngularPts(SPIN_BASE,RESW,MAXW,spin);
		if (PisRevolving(ctrlStateOther) == INDIGENS) basePoints *= COLLAB_FACT;
		newPoints += basePoints;
	}
```

```
	else if ((PinAsteroid(ctrlStateMe)==OPULENS) && PiceMelted())
	{
		basePoints = calcAngularPts(SPIN_BASE*WEAK2STRONG,RESW,MAXW,spin);
		if (PisRevolving(ctrlStateOther) == OPULENS) basePoints *= COLLAB_FACT;
		newPoints += basePoints;
	}
	//Revolving case
	if (PisRevolving(ctrlStateMe)==INDIGENS)
	{
		revolve = calcRevolve(ctrlStateMe,((float*)INDIGENS_LOC));
		basePoints = calcAngularPts(REVOLVE_BASE,RESR,MAXR,revolve);
		if  (PinAsteroid(ctrlStateOther) == INDIGENS) basePoints *= COLLAB_FACT;
		newPoints += basePoints;
	}
	else if ((PisRevolving(ctrlStateMe)==OPULENS) && PiceMelted())
	{
		revolve = calcRevolve(ctrlStateMe,((float*)OPULENS_LOC));
		basePoints = calcAngularPts(REVOLVE_BASE*WEAK2STRONG,RESR,MAXR,revolve);
		if  (PinAsteroid(ctrlStateOther) == OPULENS)  basePoints *= COLLAB_FACT;
		newPoints += basePoints;
	}
}

//Check if the opponent SPHERE is spinning on the same asteroid
ASTEROID PinAsteroid(float *state)
{
	if (reachedItem(state,(float*)INDIGENS_LOC))
	{
		return INDIGENS;
	}
	else if (reachedItem(state,(float*)OPULENS_LOC))
	{
		return OPULENS;
	}
	return NONE;
}

//Check if the opponent SPHERE is revolving around the same asteroid
ASTEROID PisRevolving(float *state)
{
	//1=revolving around Opulens,2=Revolving around Indigens,0=not revolving
	float indigensDist;
	float opulensDist;

	indigensDist = dist3d(state, (float*)INDIGENS_LOC);
	opulensDist = dist3d(state, (float*)OPULENS_LOC);

	if ((opulensDist<=REVOLVE_RADIUS2) && (opulensDist>=REVOLVE_RADIUS1)){
		return OPULENS;
	}
	if ((indigensDist <= REVOLVE_RADIUS2) && (indigensDist >= REVOLVE_RADIUS1)){
		return INDIGENS;
	}
	return NONE;
}

//Return incremental scores due to spinning and revolving
float calcAngularPts(float realPts,float targetVel,float maxVel,float move)
```

```
{
	float score = 0.0f;

	if (move>0 && move<=targetVel)
	{
		score = realPts*move/targetVel;
	}
	else if (move>targetVel && move<maxVel)
	{
		score = realPts*(maxVel-move)/targetVel;
	}
	return score;
}

//Calculates the angular velocity about asteroid location, specified using
//omega = (r x v)/||r||^2.
float calcRevolve(float *state, float *asteroid)
{
	int i;
	float radius[3],angVel[3],dist;
	for (i=0;i<3;i++)
	{
		radius[i] = state[i] - asteroid[i];
	}
	dist = mathVecMagnitude(radius,3);
	mathVecNormalize(radius,3);
	mathVecCross(angVel,radius,&state[VEL_X]);
	for (i=0;i<3;i++)
	{
		angVel[i] /= dist;
	}
	return (fabsf(mathVecInner(angVel,gameInfo.normAster,3)))*180/PI;
}
```

Communication Packet Transfer

The following lines of code show the process by which game specific parameters, closed and open loop commands and communication messages is packed into a set of vectors in every SPHERE and broadcast once every control cycle (1Hz), to be received by the other SPHERES in the match and the laptop. This information is used for logging as well as to make game specific information available to the users of the opponent SPHERE. It is this communication between the SPHERES along with the API functions available in Appendix A that allows the different game variables to be synced between the programs playing the game – as seen in the lower loop of Figure 26. Note that the actual state of the SPHERES is not sent in this communication packet – that is sent as a state of health packet at 5Hz. The SPHERES communicate with each other using an 868 MHz channel.

```
//These vectors are used for feedback to the other SPHERE
DebugVecShort[0] = (short) (test_time / 100 );
for(ii=0; ii<3; ii++)
```

```
{
	DebugVecShort[ii+1] = (short) (1000.0f * ctrlStateTarget[ii]);
	DebugVecShort[ii+4] = (short) (ctrlControl[ii]*10000);
	DebugVecShort[ii+7] = (short) (userAtt[ii]*1000);
	DebugVecShort[ii+10] = (short) (ctrlStateError[ii]*1000.0f);
	DebugVecShort[ii+13] = (short) (gameInfo->normAster[ii]*1000.0f);
}

//First element always has test time
DebugVecFloat[0] = (test_time/1000.0f);
DebugVecFloat[1] = PgetScore();
DebugVecFloat[2] = ctrlControl[TORQUE_Y];
DebugVecFloat[3] = ctrlControl[TORQUE_Z];
DebugVecFloat[4] = PgetPercentFuelRemaining();
DebugVecFloat[5] = gameInfo->me.chargeUsed; //Send both chargeUsed (from weapons)
DebugVecFloat[6] = PgetCharge();   //current charge (weapons + shield hits) for ease of
	debugging
DebugVecFloat[7] = (float)getMaxFuelUse();
DebugVecUShort[0] = (unsigned short)(test_time/100);
DebugVecUShort[1] = PinAsteroid(statePosVel) + PisRevolving(statePosVel);
DebugVecUShort[2] = (unsigned short)PhaveShield();
DebugVecUShort[3] = (unsigned short)gameInfo->me.station;
DebugVecUShort[4] = (unsigned short)PdisruptorUpgraded();
DebugVecUShort[5] = (unsigned short)(PhaveLaser());
DebugVecUShort[6] = (unsigned short)gameInfo->me.repulsor.active;
DebugVecUShort[7] = (unsigned short)gameInfo->me.magnet.active;
DebugVecUShort[8] = (unsigned short)gameInfo->me.stationTime;
DebugVecUShort[9] = (unsigned short)gameInfo->me.laser.active;
DebugVecUShort[10] = (unsigned short)PgetPhase() + (gameInfo->gameTime > 170); //During
the final race, add 1 to the phase so animation can display stations
DebugVecUShort[11] = (unsigned short)gameInfo->other.shield.shieldHits;
DebugVecUShort[12] = gameInfo->me.truceValue;
DebugVecUShort[13] = gameInfo->me.collisionActive;
DebugVecUShort[14] = (unsigned short)gameInfo->me.iceHits;
DebugVecUShort[15] = teamId+1;
commSendPacket(COMM_CHANNEL_STL, GROUND, sysIdentityGet(), COMM_CMD_DBG_SHORT_SIGNED,
(unsigned char *) DebugVecShort,0);
commSendPacket(COMM_CHANNEL_STL, BROADCAST, sysIdentityGet(), COMM_CMD_DBG_FLOAT,
(unsigned char *) DebugVecFloat,0);
commSendPacket(COMM_CHANNEL_STL, BROADCAST, sysIdentityGet(),
COMM_CMD_DBG_SHORT_UNSIGNED, (unsigned char *) DebugVecUShort,0);
```

The following lines of code show the process by which the communication packets broadcast by every SPHERE is received and decoded by every other SPHERE in the match and the laptop. As mentioned before, it is this communication that allows the SPHERES satellites to sync with each other and play the game (Figure 26). Once these variables are available to the receiving SPHERE, the user may access any of them by calling the provided library of API functions (Appendix A).

```
void processGameComm(default_rfm_packet packet)
{
	if (packet[PKT_CM] == COMM_CMD_DBG_SHORT_UNSIGNED)
```

```
	{
		dbg_ushort_packet DebugVecUShort;
		memcpy(DebugVecUShort, &packet[PKT_DATA], sizeof(dbg_ushort_packet));

		gameInfo.other.shield.acquired =   (unsigned char)DebugVecUShort[2];
		gameInfo.other.station      =      (unsigned char)DebugVecUShort[3];
		gameInfo.other.repulsor.acquired = (unsigned char)DebugVecUShort[4];
		gameInfo.other.laser.acquired =    (unsigned char)DebugVecUShort[5];
		gameInfo.other.repulsor.active =   (unsigned char)DebugVecUShort[6];
		gameInfo.other.magnet.active =     (unsigned char)DebugVecUShort[7];
		gameInfo.other.stationTime =       (unsigned char)DebugVecUShort[8];
		gameInfo.other.laser.active =      (unsigned char)DebugVecUShort[9];
		gameInfo.me.shield.shieldHits =    DebugVecUShort[11];
		gameInfo.other.truceValue =        DebugVecUShort[12];
		gameInfo.other.collisionActive =   (unsigned char)DebugVecUShort[13];
		gameInfo.other.iceHits =           DebugVecUShort[14];


	}
	if (packet[PKT_CM] == COMM_CMD_DBG_FLOAT)
	{
		dbg_float_packet DebugVecFloat;
		memcpy(DebugVecFloat, &packet[PKT_DATA], sizeof(dbg_float_packet));
		gameInfo.opponentScore =           DebugVecFloat[1];
		gameInfo.other.chargeUsed =        DebugVecFloat[5];
		gameInfo.other.fuelFlag =          (unsigned char)(DebugVecFloat[4] > 0);
	}
}
```

Appendix C – Quantitative Evaluation of the ZR Summer Program for Middle School students

To evaluate the ZR Summer Program, the 5 middle school programs were asked to assess their interested in STEM fields by the Massachusetts Afterschool Partnership (MAP), MIT's partner in the Summer of Innovation Program 2010 and the Summer Middle School program 2011.

On a scale of 1 ("strongly disagree") to 5 ("strongly agree"), students, on average, **more strongly agreed** with the following statements after participating in the Zero Robotics Summer Program 2011:

	Average Pre-Test Answer	Average Change in Post- Test
"I like math"	3.71	+0.132
"Math is useful in everyday life"	4.42	+0.026
"I am good at math"	4.08	+0.132
"I try to do well in math"	4.11	+0.184
"Science is useful in everyday life"	3.87	+0.135
"I am good at science"	4.11	+0.105
"Doing well in science is important"	4.29	+0.079
"Doing well in engineering is important"	3.97	+0.054
"I like engineering"	3.92	+0.132
"Engineering is useful in everyday life"	3.71	+0.29
"I am good at engineering"	3.24	+0.263
"I try to do well in engineering"	3.86	+0.167

Conversely, using the same scale, students, on average, ***more strongly disagreed*** with the following statements after participating in the Zero Robotics Program:

	Average Pre-Test Answer	Average Change in Post- Test
"Math is boring"	2.42	-0.132
"Science is boring"	2.05	-0.053
"Engineering is boring"	2.05	-0.278

On a scale of 1 ("never") and 5 ("very often"), students, on average, expressed that they thought about performing the following jobs ***more often*** after participating in the Zero Robotics program:

	Average Pre-Test Answer	Average Change in Post- Test
"Working with computers"	3.95	+0.026
"Being a doctor"	2.37	+0.316
"Doing science experiments"	3.39	+0.108
"Building robots"	2.92	+0.351
"Being an astronaut"	2.27	+0.286
"Turning ideas into drawings"	2.95	+0.351
"Being an engineer"	3	+0.447

On a scale of 1 ("never") and 5 ("very often"), students, on average, expressed that they would perform the following activities ***more often*** after participating in the Zero Robotics program

	Average Pre-Test Answer	Average Change in Post- Test
"Talk about science with my family"	2.58	+0.162
"Do science activities on my own for fun"	2.66	+0.27
"Talk about math with my friends"	2.58	+0.083
"Talk about math with my family"	2.47	+0.25
"Do math activities on my own for fun"	2.32	+0.514
"Talk about engineering with my friends"	2.29	+0.189
"Talk about engineering with my family"	2.24	+0.306
"Do engineering activities on my own for fun"	2.47	+0.27

On a scale of 1 ("really don't like") and 5 ("like a lot"), students, on average, expressed that they ***more greatly enjoyed*** performing the following activities after participating in the Zero Robotics program:

	Average Pre-Test Answer	Average Change in Post-Test
"[Inventing] things"	4	+0.083
"[Helping] others fix things"	4.03	+0.027
"[Solving math] problems"	3.5	+0.135

Comments from the educators and teachers from the 5 participating middle school programs from the greater Boston area were collected and 2 most representative ones have been highlighted below:

"One of our female students had never been exposed to this level of science before or any program like this. From the first week she just took it with such enthusiasm and vigor. She would actually complain the challenge was waning. Having such an inquisitive mind, she probably gets bored and loses interest in the science performed at school. Towards the end of the program she was asking about other projects and programs for the school year. She's a student who was literally inspired right before our eyes. 'When are we going to build a satellite, and launch it? I want to be first on the list.'"

– STEM Curriculum Specialist, Salem CyberSpace

"I would recommend for all schools to create the opportunity for students to work in programs such as Zero Robotics. It allows the students to do really interesting and hard problems and apply the knowledge to their coursework such as physics and mathematics. We will showcase the programs results to the school on the first week and hopefully it will encourage other students to learn more and apply for the next summer."

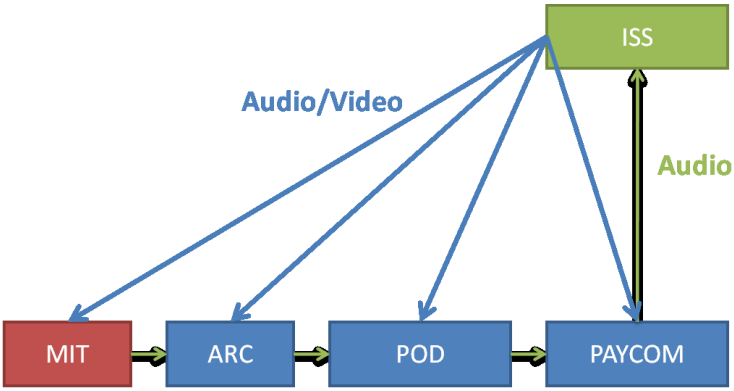
– Middle School Mathematics Teacher, James P. Timilty Middle School

Appendix D – SPHERES International Space Station Operations

This section will describe the generic SPHERES operations on the International Space Station. A test session is a predefined period of time that NASA Marshall Spaceflight Center allocates for one or more astronauts aboard the ISS to run the SPHERES experiments in one of the ISS modules. A Test Plan is sent to the astronaut at least two weeks before the session, highlighting the objectives of the session and describing each test in particular. At least one week before a test session, the program for the session is sent to the ISS via NASA. A program comprises of several SPHERES tests whose code is packaged in the form of an executable file along with html files containing instructions to run each test – all of which is readable using the SPHERES Graphical User Interface (GUI), loaded on the ISS laptop. At the start of every test session, the astronaut sets up the metrology system by positioning the beacons and synchronizing their locations with the GUI, loads the satellites with tanks and batteries, uploads the session program onto the satellites and runs a checkout test to ensure that the hardware is in good health. It is only after this regimen that the SPHERES tests begin. A SPHERES test is a 3-10 minute autonomous operations of the satellite which is programmed to demonstrate a particular research objective, is started by the astronaut manually using the ISS laptop's GUI and ends automatically or is ended manually by the astronaut using the GUI again. Each SPHERES test comprises of several maneuvers that switch sequentially and autonomously. The first maneuver, lasting about 10 seconds is usually the estimator convergence maneuver where in the SPHERES sounds the ISS metrology system (5 beacons as described in Section 3.1) and estimates its initial state. The second maneuver is usually initial positioning for about 20 seconds, following which the satellite executes its GNC maneuvers as programmed. During the course of the test, the SPHERES continually communicate with each other and the laptop using the STS and STL links (Section 3.1) and at the end of a test, transmits an 8-bit number to the laptop. Within a week of the test session, NASA makes the detailed logs of the satellite telemetry (as broadcast by each SPHERE during each test) available to MIT for data analysis.

The communications flow during a SPHERES test session involves several NASA-affiliated groups all over the country. Each can see and hear the crew through audio and video downlink. PAYCOM at NASA Marshall Space Flight Center (MSFC) in Huntsville, AL speaks directly to the crew. PAYCOM is managed by the Payload Operations Director (POD), also at MSFC. POD approves

everything before it is relayed to the crew. MIT's connection to POD is through NASA Ames Research Center (ARC) in Moffett Field, CA. MIT speaks to ARC, who speaks to POD and PAYCOM, and PAYCOM speaks to the crew. Therefore, at any point during a test, MIT can tell the astronauts if the test a match is not proceeding as expected from simulations and if a re-run is required.



Where the above mentioned acronyms mean the following:

POD	Payload Operations Director
PAYCOM	Payload Communications Officer
"SPHERES"	Call sign of the SPHERES research team at Ames
"MIT"	Call sign of the SPHERES research team at MIT
GUI	Graphical User Interface
Huntsville	Location of Payload Operations, at NASA MSFC
Space to Ground	Communication loop that NASA uses to talk to ISS
SSC	Standard Station Computer (laptop)
LOS	Loss of Signal
AOS	Acquisition of Signal
JPM	Japanese Pressurized Module, a.k.a Kibo

Zero Robotics test sessions are special test sessions where in the ISS competitions are held. As described in Section 4.1.2.3, the entire downlink from the ISS and all the communication links in between are broadcast live in an MIT auditorium. Each test is a match between two projects submitted by different teams or alliances, both of which play the same game. The test number, selected by the astronaut to initiate the test, corresponds to the team ID of the first team and the

opponent team is selected by pressing a key on the laptop keyboard (and therefore detected by the SPHERES software). At the end of the test, both SPHERES return a test result value to the ISS laptop which is a function of the team ID of the playing team and its corresponding match score. After every match, the astronaut calls down the test result numbers to ground and MIT calculates and announces the result and scores of the match. These scores decide which teams will proceed along the competition bracket and the final champion. Specifically, in 2011 for AsteroSPHERES, the result numbers and the equation to calculate the score is given in the table below. Therefore, a ZR test session comprises of many tests which essentially make up a structured and bracketed competition of many matches. The scores are autonomously available and called down and all the matches are viewable live by all participants.

0-9	Standard Errors such as forced reset or other hardware errors
7	Test Error; Other satellite reset/stopped
255	Satellite Reset
10-249	The game ended normally: the score of the team playing on this satellite can be calculated as follows: Score = (TestResult – teamNumber – 1)/10 - 1
254	Opponent player was not selected

The hardware required to run a typical SPHERES test session are:

1. The satellites, up to 3 available, and the laptop computer
2. A communication box (<70 g, 10 cm X 6 cm X 3 cm) using which the laptop establishes the STL link with the SPHERES, at 868 MHz or 916 Hz.
3. One customized paintball tank (length 21 cm, diameter 6 cm) per SPHERE filled with single propellant, liquid carbon dioxide weighing 612 g when full and 584 g when empty.
4. 2 battery packs per SPHERE made of 8 AA alkaline batteries each. They weigh 255 g and measure 6 cm X 6 cm X 3.5 cm
5. Five ultrasound beacons, for metrology, and 1 beacon tester
6. Test volume for running the test e.g. US Lab or the KIBO module inside the ISS. Currently SPHERES tests are run in KIBO/JEM which measures 2m X 1.7m X 1.7m

References

[1] C. Solomon, M. Minsky, B. Harvey, *Introduction to LogoWorks*. Byte Books, New York: McGraw-Hill, 1986.

[2] Jeff Howe, "The Rise of Crowdsourcing," *Wired*, Jun-2006.

[3] President's Council of Advisors on Science and Technology (PCAST), "Prepare and InspireL K-12 Education in Science, Technology, Engineering and Math (STEM) for Aamerica's Future," Executive Office of the President, Washington, DC, Sep. 2010.

[4] G. Pickard, A. Pentland, "Time-Critical Social Mobilization," *Science*, vol. 334, Oct. 2011.

[5] C.R. Aragon, S.S. Poon, A. Monroy-Hernández, "A Tale of Two Online Communities: Fostering Collaboration and Creativity in Scientists and Children," in *Proceeding of the Seventh ACM Conference on Creativity and Cognition*, Berkeley, California, U.S.A., 2009.

[6] S. Nag, J.G. Katz, A. Saenz-Otero, "The SPHERES Zero Robotics Program: Education using Games," in *Proceedings of the 62nd Annual International Astronautical Congress*, Cape Town, South Africa, 2011.

[7] Richard de Neufville, Frank Field, "Thesis Definition and Preparation: Some General Guidelines." MIT Technnology and Policy Program, Internal Release, 10-Sep-2010.

[8] Earl Babbie, *The Science of Social Research*, 12th ed. Wadsworth Cengage Learning, 2010.

[9] D.T. Campbell, J.C. Stanley, *Experimental and Quasi-Experimental Methods for Research*. Johns Hopkins University: Houghton Mifflin Company, 1965.

[10] Karen Willcox, Olivier de Weck, "ESD.77/16.888: Design Space Exploration," Massachusetts Institute of Technology, 2005.

[11] Walter Wallace, *The Logic of Science in Sociology*. New York:Aldine deGruyter, 1971.

[12] Robert K. Yin, *Case Study Research: Design and Methods*, 4th ed., vol. 5. SAGE Publications, Inc., 2009.

[13] Alvar Saenz-Otero, "Design Principles for the Development of Space Technology Maturation Laboratories Aboard the International Space Station," PhD, Massachusetts Institute of Technology, Cambridge, Massachusetts, U.S.A., 2005.

[14] J.G. Walker, "Satellite Constellations," *British Interplanetary Society Journal (Space Technology)*, vol. 37, pp. 559–572.

[15] Rodger W. Bybee, "What is STEM Education?" Science AAAS, Vol 329, 27-Aug-2010.

[16] Piers Harding-Rolls, "Subscription MMOGs: Life beyond World of Warcraft," 30-Mar-2009.

[17] F. Kleemann, G. Vob, K. Reider, "Un(der)paid Innovators: The Commercial Utilization of Consumer Work through Crowdsourcing," *Science, Technology and Innovation Studies*, vol. 4, no. 1, pp. 5–26, 2008.

[18] Dava Sobel, *Longitude: The True Story of a Lone Genius Who Solved the Greatest Scientific Problem of his Time*. United States of America: Walker Publishing Company, Inc., 1995.

[19] F. Aftalion, *A History of the International Chemical Industry: From the "Early Days" to 2000*. Chemical Heritage Foundation, 2005.

[20] C.A. Lindburgh, *The Spirit of St. Louis*. New York, U.S.A.: Scribner, 1953.

[21] R. Laubacher, G. Olson, T. Malone, "The Climate CoLab: Large scale model-based collaborative planning," in *IEEE Xplore*, 2011.

[22] M. Klein, "The MIT deliberatorium: Enabling large-scale deliberation about complex systemic problems," in *IEEE Xplore*, 2011.

[23] Seth Cooper, Firas Khatib, Adrien Treuille, Janos Barbero, Jeehyung Lee, Michael Beenen, Andrew Leaver-Fay, David Baker, Zoran Popović, Foldit Players, "Predicting protein structures with a multiplayer online game," *Nature*, vol. 466, no. 7307, pp. 756–760, 2010.
[24] "X-Prize Foundation - The Ansari X-Prize." [Online]. Available: http://space.xprize.org/ansari-x-prize.
[25] Office of the Press Secretary, "A Strategy for American Innovation: Driving Towards Sustainable Growth and Quality Jobs." Office of Science and Technology Policy, Sep-2009.
[26] P. Orszag, "Memorandum for the Heads of Executive Departments and Agencies," The White House, Washington, DC.
[27] *COMPETES Act: America Creating Opportunities to Meaningfully Promote Excellence in Technology, Education, and Science*. 2010.
[28] "NASA Tournament Labs on TopCoder Inc. website," *TopCoder Inc.* [Online]. Available: http://www.topcoder.com/nasa. [Accessed: 28-Apr-2012].
[29] "TopCoder Inc. Problem Statement Page," *TopCoder Inc. NASA Tournament Labs winner*. [Online]. Available: http://community.topcoder.com/longcontest/?module=ViewProblemStatement&rd=14481&pm=11313. [Accessed: 28-Apr-2012].
[30] "TopCoder Inc. Forum Posting Page," *TopCoder Inc. NASA Tournament Labs winner*, 24-Jun-2010. [Online]. Available: http://apps.topcoder.com/forums/?module=Thread&threadID=678649&mc=8&view=threaded. [Accessed: 27-Apr-2012].
[31] J. Ford, "Interview with NTL Winner," *TopCoder Inc. NASA Tournament Labs winner*, 13-Jul-2011. [Online]. Available: http://community.topcoder.com/ntl/?p=493.
[32] O. Brown, P Eremenko, "The Value Proposition for Fractionated Space Architectures," in *AIAA-2006-7506*, San Jose, California, 2006.
[33] D. M. LoBosco, G. E. Cameron, R. A. Golding, T. M. Wong, "The Pleiades fractionated space system architecture and the future of national security space," presented at the AIAA Space, Anaheim, California, 2008.
[34] Charlotte Mathieu, "Assessing the fractionated spacecraft concept," Massachusetts Institute of Technology, Cambridge, Massachusetts, U.S.A., 2006.
[35] "Proba Mission Page on the European Space Agency website." [Online]. Available: http://www.esa.int/esaMI/Proba/SEMXZ5ZVNUF_0.html.
[36] L. David, "Space Debris: A Growing Challenge," *American Institute of Aeronautics and Astronautics, Aerospace America*, pp. 30–36, Oct-2009.
[37] L. Grego, "Short History of US and Soviet ASAT Programs," *Union of Concerned Scientists*, Apr-2003.
[38] A. Richards, T. Schouwenaars, J.P. How, Eric Feron, "Spacecraft Trajectory Planning with Avoidance Constraints using Mixed-Integer Linear Programming," *Journal of Guidance, Control and Dynamics*, vol. 25, no. 4, 2002.
[39] Y. Kim, M. Mesbahi, F.Y. Hadaegh, "Multiple-Spacecraft Reconfiguration through Collision Avoidance, Bouncing, and Stalemate," *Journal of Optimization Theory and Applications*, vol. 122, pp. 323–343, Aug. 2004.
[40] J. Mueller, R. Larsson, "Collision Avoidance Maneuver Planning with Robust Optimization," presented at the ESA Guidance, Navigation and Control Conference, Tralee, Ireland, 2008.
[41] R. Larsson, J. Mueller, S. Thomas, B. Jakobsson, P. Bodin, "Orbit Constellation Safety on the PRISMA In-Orbit Formation Flying Testbed," presented at the ESA Formation Flying Symposium, The Netherlands, 2008.

[42] G.L. Slater, S.M. Byram, T.W. Williams, "Collision Avoidance for Satellites in Formation Flight," *Journal of Guidance, Control and Dynamics*, vol. 29, no. 5, 2006.
[43] J. B. Mueller, "A Multiple-Team Organization for Decentralized Guidance and Control of Formation Flying Spacecraft," presented at the AIAA 1st Intelligent Systems Technical Conference, Chicago, Illinois, 2004.
[44] J. G. Katz, A. Saenz-Otero, and D. W. Miller, "Development and demonstration of an autonomous collision avoidance algorithm aboard the ISS," in *Aerospace Conference, 2011 IEEE*, 2011, pp. 1–6.
[45] Alvar Saenz-Otero et. al., "Distributed Satellite Systems Algorithm Maturation with SPHERES Aboard the ISS," presented at the International Astronautical Congress, Glasgow, Scotland, 2008.
[46] S. Nag, L. Summerer, "Evolutionary Behavior based, Autonomous and Distributed Scatter Manoeuvres for Satellite Swarms," *Acta Astronautica Special Edition*, no. accepted for publication, 2012.
[47] P. Gonzales et. al., "Highlights From TIMSS 2007: Mathematics and Science Achievement of U.S. Fourth- and Eighth-Grade Students in an International Context," National Center for Education Statistics, Institute of Education Sciences, U.S. Department of Education, Washington, DC, NCES 2009–001 Revised, 2008.
[48] "OECD Report of Performance Ranks per Country." [Online]. Available: http://geographic.org/country_ranks/educational_score_performance_country_ranks_2009_oecd.html.
[49] "CIA World Factbook." [Online]. Available: https://www.cia.gov/library/publications/the-world-factbook/index.html.
[50] B. Trilling, C. Fadel, *21st century skills: learning for life in our times*. San Francisco, CA: Jossey-Bass, 2009.
[51] B. Trilling, "From Libraries to Learning 'Libratories' : The New ABC's of 21st-Century School Libraries," *School Library Monthly*, vol. 27, no. 1, pp. 43–46, 2010.
[52] Cathy N. Davidson, *Now You See It: How the Brain Science of Attention Will Transform the Way We Live, Work, and Learn*. Viking Penguin Books, 2011.
[53] "NASA Strategic Plan 2011." 24-Apr-2012.
[54] Matthew Allner, et al, "NASA's explorer school and spaceward bound programs: Insights into two education programs designed to heighten public support for space science initiatives," *Acta Astronautica*, vol. 66, pp. 1280–1284, 2010.
[55] A. R. Johnson, et al, "International Space Station National Laboratory Concept Development Report," NASA Headquarters, NP 2007 03 459 HQ, Dec. 2006.
[56] Mark Craig, "Letter to NASA Headquarters: Massive Public Engagement in Space Exploration," 16-Nov-1998.
[57] National Academies Press, "America's Future in Space: Aligning the Civil Program with National Needs," National Research Council, Washington, DC, 2009.
[58] National Aeronautics and Space Administration, "Augustine Commission Report," JSC-Houston, Texas, Jul. 2009.
[59] Stuart L. Brown, Christopher C. Vaughan, *Play: how it shapes the brain, opens the imagination, and invigorates the soul*. Penguin Publications, 2010.
[60] Jane McGonigal, *Reality Is Broken: Why Games Make Us Better and How They Can Change the World*. 2011.
[61] Jesse Schell, *The Art of Game Design: A Book of Lenses*. Elsevier/Morgan Kaufmann, 2008.

[62] Edward Castronova, "A Test of the Law of Demand in a Virtual World: Exploring the Petri Dish Approach to Social Science," Jul-2008.
[63] Tom Chatfield, "7 ways games reward the brain," 2010.
[64] Nick Yee, "The Psychology of MMORPGs: Emotional Investment, Motivations, Relationship Formation, and Problematic Usage," in *Avatars at Work and Play: Collaboration and Interaction in Shared Virtual Environments*, vol. 34, London: Springer-Verlag, pp. 187–207.
[65] Jeffrey Kim, Jonathan P. Allen, Elan Lee, "Alternate reality gaming," *Communications of the ACM*, vol. 51, no. 2, pp. 36–42, 2008.
[66] Helena Cole, Mark Griffiths, "Who plays, how much, and why? Debunking the stereotypical gamer profile," *Journal of Computer-Mediated Communication*, vol. 13, no. 4, pp. 993–1018, 2008.
[67] Dmitri Williams, Nick Yee, Scott Caplan, Ching–I Teng, "Personality differences between online game players and nonplayers in a student sample," *CyberPsychology and Behavior*, vol. 11, no. 2, pp. 232–234, Apr. 2008.
[68] Helena Cole, Mark Griffiths, "Social interactions in massively multiplayer online role–playing gamers," *CyberPsychology and Behavior*, vol. 10, no. 4, 583 575.
[69] J.S.S. van 't Woud, "The Mars Crowdsourcing Experiment," presented at the 16 th International Conference on Computer Games, Wolverhampton, England, 2011.
[70] S. T. Ishikawa, V.C. Gulick, "Clickworkers Interactive: Towards a Robust Crowdsourcing Tool for Collecting Scientific Data," presented at the 43rd Lunar and Planetary Science Conference, The Woodlands, Texas, 2012.
[71] K.J. Boudreau, K.R. Lakhani, "The Confederacy of Software Production: Field Experimental Evidence on Heterogeneous Developers, Tastes for Institutions and Effort," presented at the NBER 50th Anniversary Conference on the Rate and Direction of Inventive Activity, 2010.
[72] G. V. Ranade, L. R. Varshney, "To Crowdsource or Not to Crowdsource?," presented at the Collective Intelligence Conference, Cambridge, Massachusetts, U.S.A., 2012.
[73] Alan Melchoir et. al, "More than Robots: An Evaluation of the FIRST Robotics Competition Participant and Institutional Impacts," Heller School for Social Policy and Management, Brandeis University, Waltham, Massachusetts, U.S.A, Apr. 2005.
[74] R.D. Atkinson, M.J. Mayo, "Refueling the U.S. Innovation Economy: Fresh Approaches to Science, Technology, Engineering and Mathematics (STEM) Education," Technology and Innovation Foundation, Jul. 2010.
[75] J.S. Brown, P. Duguid, "The Social Life of Information," Harvard Business School Press, Boston, Massachusetts, U.S.A., 2000.
[76] Mizuko Iko et. al., "Living and Learning with New Media: Summary of Findings from the Digital Youth Project," MacArthur Foundation, Chicago, Illinois, Nov. 2008.
[77] C. Goodman, "Engineering ingenuity at iGEM," *Nature: Chemical Biology*, 2008.
[78] J.C. Bradley et. al, "The Spectral Game: leveraging Open Data and crowdsourcing for education," *Journal of Cheminformatics*, 2009.
[79] Gerhard Fischer, "Cultures of Participation and Social Computing: Rethinking and Reinventing Learning and Education," presented at the Ninth IEEE International Conference on Advanced Learning Technologies, Riga, Latvia, 2009.
[80] Chris Dahlen, "Surviving a World Without Oil," *PitchFork Media*, 13-Apr-2007.
[81] Eric Klofper, *Augmented Learning: Research and Design of Mobile Educational Games*. Cambridge, Massachusetts, U.S.A.: MIT Press, 2008.
[82] Mitch Resnick, "Technologies for Lifelong Kindergarten," in *Educational technology research and development*, Springer, 1996.
[83] N. Rusk et. al., "New Pathways into Robotics: Strategies for Broadening Participation," *Journal of Science Education and Technology*, vol. 17, no. 1, pp. 59–68, 2008.

[84] Mitch Resnick, "All I Really Need to Know (About Creative Thinking) I Learned (By Studying How Children Learn) in Kindergarten," in *Proceedings of the ACM Creativity & Cognition Conference*, Washington, DC, 2007.
[85] "DARPA Call for Crowdsourcing Ideas for Cluster Flight Software," *Federal Business Opportunities*, 11-Feb-2010. [Online]. Available: https://www.fbo.gov/index?s=opportunity&mode=form&id=def9ae35c237359f0bc75dd730e8ed54&tab=core&_cview=0. [Accessed: 27-Apr-2012].
[86] J. Enright, M. Hilstad, A. Saenz-Otero, and D. Miller, "The SPHERES Guest Scientist Program: collaborative science on the ISS," in *Aerospace Conference, 2004. Proceedings. 2004 IEEE*, 2004, vol. 1, p. 46 Vol.1.
[87] Swati Mohan, "Quantitative Selection and Design of Model Generation Architectures for On-Orbit Autonomous Assembly," PhD, Massachusetts Institute of Technology, Cambridge, Massachusetts, U.S.A., 2010.
[88] A. Saenz-Otero, J. Katz, S. Mohan, D. W. Miller, and G. E. Chamitoff, "ZERO-Robotics: A student competition aboard the International Space Station," in *Aerospace Conference, 2010 IEEE*, 2010, pp. 1–11.
[89] S. Nag, I. Heffan, A. Saenz-Otero, M. Lydon, "SPHERES Zero Robotics software development: Lessons on crowdsourcing and collaborative competition," in *Aerospace Conference, 2010 IEEE*, Big Sky, Montana, U.S.A., 2012.
[90] "TopCoder Inc. Reliability Rating Reference," *TopCoder Inc.* [Online]. Available: TopCoder Reliability Rating Reference. [Accessed: 28-Apr-2012].
[91] McKinsey&Company, "'And the winner is ...': Capturing the promise of philanthropic prizes," McKinsey&Company, Mar. 2009.
[92] MIT ZR Team, "Zero Robotics 2011 Game Manual," *Zero Robotics*. [Online]. Available: http://www.zerorobotics.org/web/zero-robotics/tournament-details?tournamentId=1. [Accessed: 28-Apr-2012].
[93] M. S. Phadke, *Quality Engineering Using Robust Design*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1995.
[94] G.G. Richardson, J.P. Penn and P.D. Collopy, "Value-Centric Analysis and Value-Centric Design," in *Proceedings of the American Institute of Aeronautics and Astronautics*, Reston, VA, U.S.A., 2010, vol. AIAA Paper 2010–8799.
[95] "Zero Robotics Autonomous Space Capture Challenge webpage." .
[96] J. G. Katz, "Estimation and Control of Flexible Space Structures for Autonomous On-Orbit Assembly," S.M., Massachusetts Institute of Technology, Cambridge, Massachusetts, U.S.A., 2009.
[97] S. Nolet, E.M.C. Kong, D.W. Miller, "Design of an Algorithm for Autonomous Docking with a Freely Tumbling Target," in *Proceedings of the SPIE Defense and Security Symposium 2005*, Orlando, Florida, U.S.A, 2005, vol. 5799–16.
[98] "DARPA Phoenix Mission Webpage." [Online]. Available: http://www.darpa.mil/Our_Work/TTO/Programs/Phoenix.aspx. [Accessed: 27-Apr-2012].
[99] Federal Inventory of STEM Education Fast-track Action Committee on STEM Education National Science and Technology Council, "The Federal Science, Technology, Engineering and Mathematics (STEM) Education Portfolio," Washington, DC, Dec. 2011.
[100] John D. Sterman, *Business Dynamics: Systems Thinking and Modeling for a Complex World*, 5th ed. Jeffrey J. Shelstad / McGraw-Hill Higher Education, 2000.
[101] Phoebe Ayers, Charles Matthews, Ben Yates, *How Wikipedia Works: And How You Can Be a Part of It*. No Starch Press Inc., 2008.

[102] David Greising, Julie Johnsson, "Behind Boeing's 787 delays," *Chicago Tribune*, 08-Dec-2007.
[103] "NASA Systems Engineering Handbook." National Aeronautics and Space Administration Headquarters, Dec-2007.
[104] "International Traffic in Arms Regulations 2011," *U.S. Department of State: Directorate of Defense Trade Controls*. [Online]. Available: http://pmddtc.state.gov/regulations_laws/itar_official.html. [Accessed: 29-Apr-2012].
[105] "The Arms Export Control Act," *U.S. Department of State: Directorate of Defense Trade Controls*. [Online]. Available: http://pmddtc.state.gov/regulations_laws/aeca.html. [Accessed: 29-Apr-2012].
[106] Susan Cain, *Quiet: The Power of Introverts in a World That Can't Stop Talking*, 1st ed. New York, U.S.A.: Crown Publishers, Random House Inc., 2012.
[107] Jonah Lehrer, "Groupthink," *The New Yorker*, 30-Jan-2012.
[108] David Brooks, "The Campus Tsunami," *The New York Times*, 03-May-2012.
[109] Dorothy Fuller et. al., "Internet Teaching By Style: Profiling the On-line Professor," *Educational Technology & Society*, vol. 3, no. 2, 2000.
[110] A. Cook-Sather, "Authorizing Students' Perspectives: Toward Trust, Dialogue, and Change in Education," *Educational Researcher*, vol. 31, no. 4, pp. 3 –14, May 2002.
[111] Damon L. Bahr et. al., "Preparing tomorrow's teachers to use technology: Attitudinal impacts of technology-supported field experience on pre-service teacher candidates," *Journal of Instructional Psychology*, vol. 31, no. 2, Jun. 2004.
[112] Paul A. Kirschnera, Kwok-Wing Lai, "Online communities of practice in education," vol. 16, no. 2, 2007.
[113] John Naughton, "A manifesto for teaching computer science in the 21st century; To: Michael Gove MP, Secretary of State for Education Subject: Proposals for rebooting the ICT curriculum," *The Guardian; The Observer*, 31-Mar-2012.
[114] Sherry Turkle, Seymour Papert, "Epistemological Pluralism and the Revaluation of the Concrete," *Journal of Mathematical Behavior*, vol. 11, no. 1, pp. 3–33, Mar. 1992.
[115] M. A. Snow, M. Met, F. Genesee, "A conceptual framework for the integration of Language and Content in Second/Foreign Language Instruction," *TESOL Quarterly*, vol. 23, no. 2, Jun-1989.