Oppositionsprotokoll

Rapportförfattare

Jesper Bratt & Alexander Wilczek

Rapportens titel

USAR-Robot Interface Evaluation

Opponent

Mathias Lindblom

Förord

Trots att rapporten är på engelska har oppositionen valts att göra på svenska. Detta för att kommentarer inte ska misstolkas på grund av språkbrister. Kontroll har även gjorts för att säkerställa att rapportskrivarna talar flytande svenska.

Rapporten har fyllts med kommentarer efter ett färgschema. Det är totalt 4 färger.
Detta för att rapportskrivarna lättare ska kunna hitta högre prioriterade kommentarer. Följande är en förklaring av varje färg med start i den lägst prioriterade färgen:

- Grå: Kommentarer där det inte alls är säkert att något ens borde ändras. Kan även vara en positiv kommentar eller bara en fråga till rapportskrivarna att ta ställning till.
- Gul: Små fel eller mindre oklarheter. Kan till exempel vara stavfel, konstig meningsuppbyggnad eller att mindre prioriterade delar av rapporten som behöver förtydligas.
- Orange: Allvarliga fel eller större oklarheter. Berör ofta viktiga delar som mål eller så kan det vara meningar med syftningsfel som kan få läsaren att missförstå viktiga delar.
- Röd: Fel eller oklarheter som berör de viktigaste delarna av rapporten eller delar som berör hela rapporten. Kan vara hela stycken som borde placeras om eller brister som påverkar hur huvudmålet och huvudsyftet tolkas.

Dessa kommentarer går att läsa i den pdf som medföljer. Här nedan kommer en mer övergripande analys av rapporten. Denna övergripande del ska ses som den mer abstrakta sammanfattningen av rapporten medan detaljerna går att utläsa i kommentarerna. Dock kommer

vissa viktigare delar som även kommenterats att tas upp och förtydligas.

Rapportens olika delar

Syfte och mål

Rubriken visar tydligt vad syftet är. Man förstår vad projektet handlar om när man läser introduktionen. Däremot lyfts inte något huvudmål tillräckligt tydligt fram i introduktionen. Den blandas dessutom ihop med “Problem statement” och “Purpose” så även om man förstår vad som ska göras i rapporten så har man inte jätteklart för sig vad det exakta målet är.

Bakgrund

Det finns ingen ordentlig bakgrund. Det var ett intressant ämne så det hade varit kul om det stod något om nutida robotar (om det finns några) eller om hur utvecklingen har sett ut.

Metod

Denna del är, för det mesta, bra beskriven och det är lätt att följa med. Däremot är det ett par saker som är lite konstiga. Det ena är att samtliga moment inför användartesterna tas upp innan användartesterna utom iterationerna (modellerna). Dessutom beskrivs första, andra och tredje iterationerna bra men den sista, som valdes och ser helt annorlunda ut än föregående iterationer, saknas. Hade varit intressant att veta varför det var 2 extra hjul samt varför hjulen var så överdimentionerade jämfört med chassit. Vilka problem denna slutgiltiga modell led av hade dessutom varit intressant att läsa om (om den nu hade några problem alls).

Det som opponenten fann mest underligt var att det känns tidigt som att det ska komma en jämförelse mellan verktygen Unity och USARsim som verkade vara ett väldigt viktigt val. Trots att arbete har lagts ned på att ta reda på det mest lämpade verktyget så nämns inte detta fören i resultatet och diskussionen vilket känns lite sent.

Resultat

Resultatet är tydligt presenterad men det är konstigt att skriva resultat om Unity vs USARsim som tas upp först efteråt i diskussionen. Om det inte finns någon form av studie eller arbete i metoden ska det inte finnas något resultat. Annars är det bara ett par smådetaljer som står i kommentarerna.

Diskussion och slutsatser

I diskussionen angående testresultaten finns det utrymme för mer diskussion och felmarginaler. Det första opponenten tänkte på var att även då de simplare interfacen prestera bättre resultatmässigt så var alla testarna nybörjare. Ett avancerat system är **alltid** svårt i början men kan vara bättre då man kan systemet. För att exmeplifiera tankegången; En nybörjare som ska åka slalom får nybörjarskidor trots att det finns professionella tillgängliga. Detta för att nybörjaren kommer prestera bättre på de “simplare” nybörjarskidorna men ett proffs tar alltid de “avancerade” skidorna då denna presterar bättre på dessa.
Helt enkelt kommer rimligtvis de som använder USAR-robotar vara professionella och tränade i

systemen vilket gör att denna “förvirring”, som nybörjarna upplevde, inte behöver vara ett problem eller ens existera. Detta känns som en väldigt viktig och missad poäng. Visserligen nämns det i slutsatsen att simplare inte alltid är bättre men den, enligt opponenten, uppenbara poängen nämns inte.

Det var dessutom 25 testare så det borde nämnas att mer omfattande tester bör göras för att säkerställa resultatet eller något i den stilen (oavsett om det är självklart). Det är inget fel med få testare men mer kritik kring detta hade varit bra.

I slutsatsen används flera gånger ordet “comfort” konstigt och fritt utan att förklaras. Det står mer i kommentarerna kring detta men vill förtyliga att detta ord kommer lite från ingenstans och känns malplacerad.

Allmänt känns diskussionen och slutsatserna rätt vaga och inte helt genomtänkta.

Generella punkter

Starkare sidor

Resultatet är förhållandevis bra och tydlig. Metoden är, förutom de nämnda problemen, välskriven och lätt att förstå. Rapporten upplevs pedagogiskt skriven och bilderna hjälper läsaren oerhört. Samtliga bilder passar dessutom in väldigt bra till texten. Kunde dock varit till och med fler bilder. Till exempel en interface-bild där man verkligen får se vad kamerorna visar

Svagare sidor

Framförallt bakgrunden då den inte finns, förutom lite kort i introduktionen. Diskussionen är ganska vag och skulle kunna utvecklas betydligt mer. Dessutom känns den delen påskyndad (läs kommentarerna). Unity vs USARsim känns som en punkt som borde få betydligt mer fokus.

Språket

Rapporten var på engelska. I det stora hela var det lättläst och man hakade inte upp sig på själva språket men det finns några saker att påpeka. Till exempel flera långa meningsuppbyggnader och kommatecken som inte borde finnas. Förutom detta fanns det några fel som man direkt insåg inte var av rapportskrivarnas mening att de skulle finnas där (exempelvis helt malplacerat ord).

Litteraturslistan

Litteraturslistan ser bra ut, både utseendemässigt och innehållsmässigt. Det var 2 länkar ([2] och [6]) som inte visade något men då datum för inhämtning står så borde det inte vara något problem.

Slutord

Rapporten är intressant men ganska kort. För att förbättra rapporten bör rapportskrivarna främst fokusera på följande; En bakgrund, även om kort, bör läggas in. Unity vs USARsim borde verkligen få mer plats och om en jämförande analys har gjorts bör det stå något om det i metoden. Rensa i diskussionen och slutsatsen och fyll ut dessa delar efter antingen de förslag som angivits ovan samt i kommentarerna eller såklart efter egna passande sätt.

Det var för övrigt ett intressant ämne och något som opponenten inte hade någon förkunskap i. Opponenten har även varit väldigt nogrann, enligt han själv, och hoppas att det uppskattas av rapportskrivarna.

KTH Computer Science
and Communication

USAR-Robot Interface Evaluation

Creating and evaluating interfaces in Unity3D

JESPER BRATT
ALEXANDER WILCZEK

Bachelor's Thesis at NADA
Supervisor: Petter Ögren
Examiner: Mårten Björkman

Abstract

Urban Search And Rescue (USAR) is something that is utilized more frequently in modern society due to the advancements made in the robotics field. Being able to use a robot to search for survivors in urban environments after disasters is invaluable. The fact that robots are able to endure hazardous areas significantly reduces response times when searching for survivors, and, in turn increasing chances of finding and rescuing victims. A major contribution to why robots have yet to dominate USAR is the fact that the robot interfaces have not evolved at the same pace that the robots have. Being able to intuitively instruct the robots is a core functionality that could be improved.

Creating and implementing such an interface is not an easy task however, which is why research has to be conducted in order to find out what really makes a well functioning interface. In this case has a USAR robot interface not only been created and implemented in different varieties, but user testing sessions have also been conducted in order to properly determine the requirements of a well functioning interface. The results of these tests have shown that some factors appear to be more important than others, but also that appearances can be deceiving and that none of these factors are in fact without its own importance.

Referat

Utvärdering av USAR-Robot gränssnitt

Urban Search And Rescue (USAR) som på svenska kan översättas till sökning och räddning i stadsmiljö, är något som på senare tid allt oftare används i dagens samhälle på grund av framstegen gjorda inom robotiken. Att kunna använda en robot för att söka efter överlevande i stadsmiljöer efter katastrofer är ovärdeligt. Faktumet att robotar kan hantera farligare områden än vad människor kan kortar ner tiden för att hitta överlevande avsevärt, och ökar samtidigt chansen att hitta överlevande offer. En stor bidragande faktor till varför robotar inte används överallt inom detta område är faktumet att gränssnittet som används inte har utvecklats i samma takt som robotarna har. Att intuitivt kunna instruera robotarna vad de ska utföra är en kärnfunktion som väsentligt kan förbättras.

Att skapa och implementera ett gränssnitt är däremot inget lätt åtagande. Detta leder till att forskning behöver genomföras för att ta reda på vad ett funktionellt gränssnitt innebär. I detta fall har USAR-robot gränssnittet inte bara skapats, utan även implementerats i olika versioner. Användartester har använts för att kunna bestämma vad kraven för ett välfungerande gränssnitt är. Resultaten av dessa test har visat att vissa faktorer verkar vara viktigare än andra, men de påvisar även att allt inte alltid är som det verkar.

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

The subject in question is Urban Search And Rescue (USAR) robot simulation. It revolves around the use of robots to aid in search and rescue missions around the world. In increasingly hostile environments, such as earthquake sites and cities affected by tsunamis, having the ability to utilize robots instead of humans could significantly increase the survival rate when disastrous events occur. Being able to design and implement a user friendly and easy-to-use interface through which these robots can be controlled could - if done well - make the technology accessible for more people.

There are many great advantages of using robots in USAR. These include, but are not limited to, reduced latency to enter hazardous zones, ability to augment field of vision and equip sensor-enhancing equipment such as infrared cameras and radiation detectors as well as being able to navigate areas inaccessible to humans. [3]

1.2 Problem statement

As of today, the human-robot interaction barrier is one of the main problems when it comes to why robots are not widely used in USAR. By being able to design and implement an interface that is easy to use, feedback from many users will be ensured by allowing both experienced as well as inexperienced users to test the product. The goal is to do this using Unity3D, henceforth referred to as "unity", or USARsim. The main difficulty of this project will probably be to get a grasp of the different environments that will be used and later use them to implement a variety of interfaces that capture the essence of usability and simplicity. Finally, user testing will be used in order to determine which interface is the better in that regard.

1.3 Purpose

Robot designs and interfaces today are commonly considered an open problem. This is probably due to a variety of reasons, however in this case the focus is directed mainly at design and usability problems.

In short, the purpose of this project is to use one of the game engines mentioned above in order to create, simulate, evaluate and finally rank a variety of robot interfaces for usage within the USAR field. In order to properly evaluate the resulting interfaces user testing sessions, within the simulated USAR environments, have been performed.

Furthermore, an additional goal and purpose is for the project to actually be put to use in some way, possibly contributing to the development of future technology within this field.

1.4 Statement of collaboration

Overall, familiarizing with, and gaining experience in, the field of 3D modeling and rendering could be of substantial use later in life and could be applied when being part of other projects. It is also exciting to try something that no one in the group has had any previous experience with. Striving to make interfaces usable and accessible to many has been an interest within the group for quite some time and it is definitely something that can be applied in other areas of computer science as well.

The work revolving around this essay consists of 4 main parts.

1. Gathering information, deciding environment.
2. Design and implementation.
3. User testing and iteration.
4. Finalizing essay.

While most things were done collaboratively, Jesper focused more on graphical implementation and design whereas Alexander spent more time researching involved scripts as well as keeping the report up to date between minor alterations. How the report was divided can be seen below.

Section	Name
Introduction	Jesper
Problem statement	Jesper
Purpose	Alexander
Statement of Cooperation	Jesper
Overview	Jesper
Development tools	Alexander
Simulations	Alexander
Robot design	Jesper
Interfaces	Jesper/Alexander
Reiteration	Alexander
Results	Jesper/Alexander
Discussion	Jesper/Alexander
Conclusion	Jesper/Alexander

1.5 Overview

After going through the introduction, the document covers how the task was completed as well as describes the final state of the implementation. To increase readability the second chapter is divided into a lot of sections. After presenting how the implementation has been done, chapter 3 focuses on the results obtained during the implementation and testing process whereas chapter 4 and 5 discusses these results and finally presents the conclusions made both over the course of the project and after analyzing the resulting data from the user tests that have been conducted.

Chapter 2

Approach

This chapter handles the various means with which the project has been approached as well as how the project has been designed. In short, a detailed description of both the work done and the actual end products can be found below.

2.1 Development tools

In order to properly execute this project a variety of tools is required. The first and main thing that is needed is a means of rendering and simulating potential disaster zones. In this particular case two options were available, the USARsim and unity rendering tools. In order to determine which the better of the two is a comparison between them has been performed.[6, 10]

However, merely having the ability to render zones is not going to be enough without any actual objects to render. Because of this the 3D-animating and modeling tool known as Maya has been used in order to design such objects.[5]

2.2 Simulations

The project is divided into two major time trial simulations, see figures 2.3 and 2.4. The first of these involve the user controlling the robot in order to find and rescue cube like objects within a larger city-like area at night, this has to be done within a time frame of 180 seconds. The second simulation is built on the same concept except now the user no longer has to complete the task within a set amount of time. Furthermore, the second simulation takes place in a smaller more mountain road-like environment at daytime.

2.3 Robot design

The robot, henceforth referred to as “the drone”, is a six-wheeled robot built with simplicity in mind. As the intention of the drone is to navigate semi-rough terrain

and zones with debris, a design-choice was made to make the wheels protrude from both sides of the chassi. This way, the likelihood of the drone getting its chassi stuck on various protrusions in the terrain was minimized. One of the reasons behind settling with a simplistic design was that uneven meshes often caused the earlier robots to become unstable. To integrate the robot in unity, the basic meshes were created using Maya. A mesh is a grid of points connected by lines forming triangular areas which, in turn, shapes the entire model as can be seen in figure 2.1 below.[1, 5]

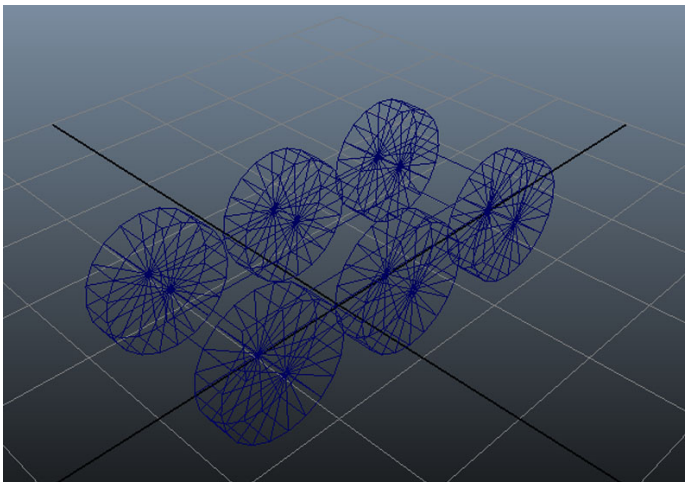

Figure 2.1. Mesh of the fourth, and final, iteration, the drone.

To interact properly with the environment in unity different colliders are used. These colliders define where the boundaries of the mesh are in a physical sense. This is needed to make sure collisions are handled correctly. Lights are utilized in multiple ways, in regards to the robot there is a spotlight lighting up the area in front of it to aid in visibility. The final iteration can be seen below in figure 2.2.

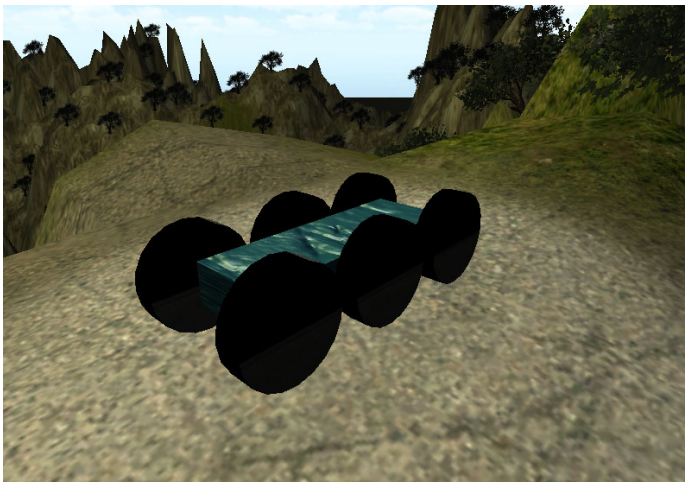

Figure 2.2. Fourth, and final, iteration, the drone.

2.4 Level design

The levels were designed using the unity terrain tool to sculpt the landscape, making sure the robot has got the possibility to navigate something other than flat ground. This, in conjunction with basic textured shapes with corresponding colliders acting as buildings and objects, forms the basic levels used for the simulations. The first level, seen in figure 2.3 below, was designed to resemble a metropolitan city at night. Tight corners and semi-hidden cubes aims to ultimately test how the robot maneuvers in the environment it is mainly intended for. The second level, seen in figure 2.4 below, was designed with the intention to test how well the robot maneuvers at full speed during the day. The reason for the fundamental difference between these two levels is to investigate the differences and possible discrepancies between how the users utilize the interfaces depending on the level that is run. [4]

Figure 2.3. Overview of the cube searching simulation.

Figure 2.4. Overview of the time trial simulation.

2.5 Interfaces

Each simulation comes equipped with three interfaces of different varieties, see figures 2.5, 2.6 and 2.7. These interfaces are all essentially based on the concept that proper vision of the surrounding environment probably is desired when maneuvering a USAR robot in the actual field. Preferably without having to add any overly complicated controls or sacrificing functionality.

In order to create interfaces that meet these criteria unitys GameObjects have been implemented and utilized in various forms and methods. A GameObject is essentially the base class for any unit or object existing within its designated unity scene.[9, 8] In regards of these interfaces mainly camera GameObjects have been used other than the drone itself. A camera provides the user with a view of the virtual world taking the position and angle with which it is placed into account. Each camera object projects its own view in its own camera window. Several cameras can display different points of view on the same screen, provided that the windows are configured not to overlap. Proper use of these functions proved sufficient enough to create the three different interfaces found within the project. [7, 12]

2.5.1 Camera views

Both the drone and the interfaces are designed taking the general concept behind the interfaces into account, meaning that several different points of views are available depending on which of the interfaces is used. In regards of the drone, two different kinds of camera objects have been implemented in two different ways each.

First person camera

A first person camera view has been achieved by placing and locking a camera object onto an empty GameObject which subsequently is locked onto the front of the drone. This provides the user with a first person view without having to modify the actual drone. In order to create varying interfaces this was implemented both with and without full screen capabilities by adding a second, identical camera with different monitor dimensions.

Third person camera

The third person camera behaves similarly to the first person camera. However, instead of being located and locked onto an empty GameObject in front of the drone it is locked onto the actual drone chassi and located behind and slightly above it. This ensures a third person perspective of both the drone and it's surrounding environment. Again, this has been implemented both with and without full screen capabilities. [12]

2.5.2 Switching between interfaces

To facilitate an easy switch between the different interface variations, CameraSwitch was written, see Appendix A.6: CameraSwitch.js. The main functionality of the script is changing the size and depth of the rendered camera. Altering the depth, which is a value between ranging from -100 to 100, changes the order of which the cameras are rendered. By constantly keeping the OrbitCam at a higher depth value than the other cameras it will always render on top of the other camera monitors once a simulation is run. This way, it is possible to, when pressing the appropriate key, change the depth, and thus, render the other cameras differently.

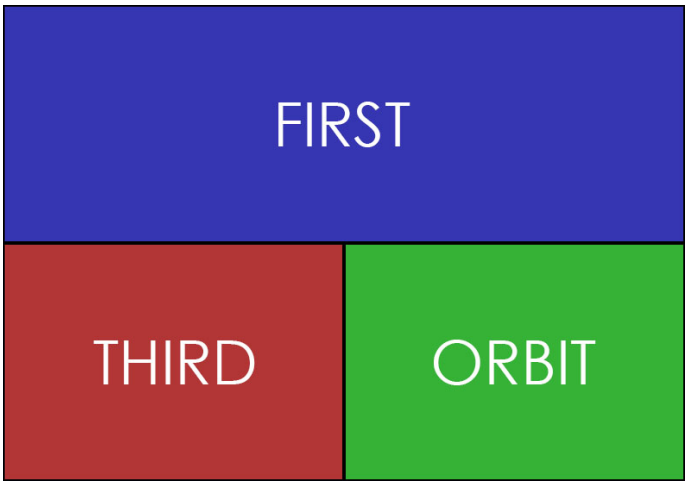

Figure 2.5. First interface.

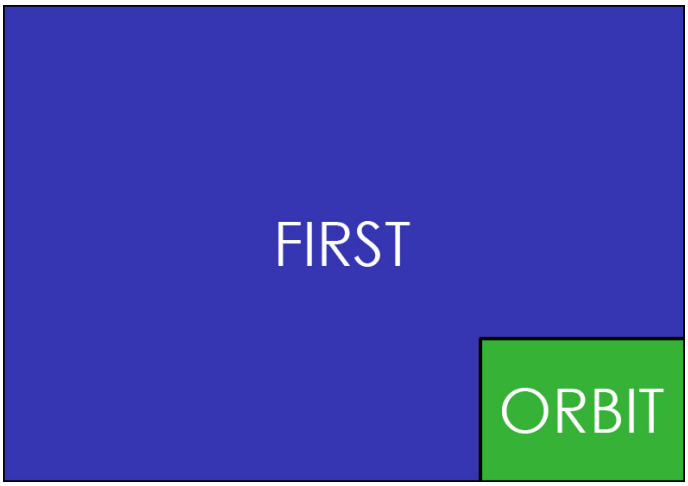

Figure 2.6. Second interface.

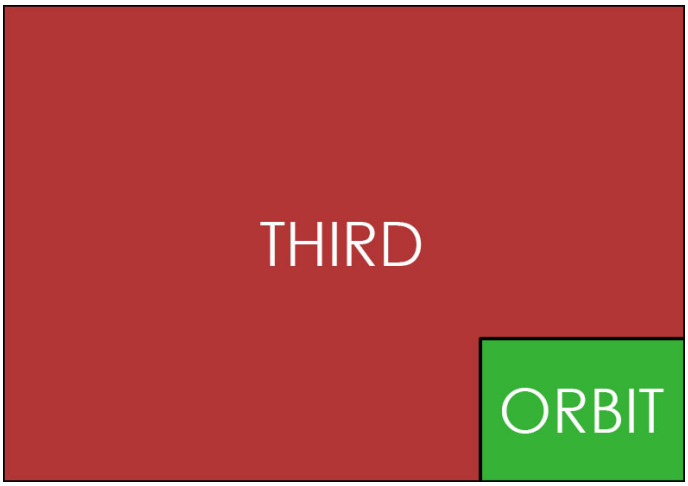

Figure 2.7. Third interface.

2.5.3 Minimap

Each interface comes complete with a minimap, as portrayed by the images above, see the orbit area in figures 2.5, 2.6 and 2.7. The minimap consists of a camera, OrbitCam, that has been placed and directed at the world from a birds eye view. This has been implemented by writing and attaching a javascript file, see Appendix A.1: OrbitCam.js, to the aforementioned camera. The script is what places and locks the camera on top of the drone, thus ensuring that the drone remains within the cameras view range at all times. Furthermore, the OrbitCam window has been assigned to the lower right corner of each interface in order to avoid covering the primary camera and reducing visibility for the user.[2]

2.5.4 Simulation progress

Means of monitoring simulation progress (time passed and remaining cubes) is created and implemented by attaching javascripts, see Appendix A.2,3,4: ScoreCounter.js, ScoreTime.js, ScoreCountDown.js, to three empty GameObjects which are launched and displayed with each simulation. The ScoreTime script launches a simple timer that monitors the time passed since the mountain road simulation was initiated, the timer stops once the drone reaches the simulations final destination. ScoreCountDown is used instead of ScoreTime in the city simulation. It functions in a similar fashion, however the timer now starts at 180 and decreases by the second until the time is up instead. ScoreCounter is the script designed in order to keep track of the remaining amount of cubes in either simulation. The script is linked to all of the cube GameObjects, when a cube is rescued it disappears and the remaining number of cubes decreases. There is a total of ten cubes in each simulation.

2.5.5 Controls

A javascript, see Appendix A.7: ControlScript.js, has been designed in order to allow the robot to move as well as rotate. Moving forwards and backwards is done by applying either positive or negative virtual force directly to the drone chassi, pushing it in either direction. Rotating is done by invoking the drone chassis own rotate function, allowing it to turn to either side even when standing still. The controls to maneuver the drone are the standard arrow keys or W, A, S, D buttons.[11]

2.6 User testing

The user testing sessions have been conducted in such a way that random users have been offered a chance to participate in the aforementioned simulations. As previously stated there are a total of two different simulations and three different interfaces to try out.

Each user is allowed to try both simulations with just one of the interfaces (not of their choosing), this is to maximize efficiency and save time when conducting the tests. By conducting the tests in this fashion every user is always given the opportunity to utilize a new interface rather than get the chance to adapt and become accustomed to the controls as well as the simulated levels by attempting the same simulation more than once. Once a test is complete the user has to provide various bits of information regarding the experience as a whole. The factors that are taken into consideration are the users prior experience with simulations such as these, the users performance throughout the tests and finally the difficulty perceived from different perspectives on scales ranging from one to ten. The answers are stored in a separate google form for each interface used and the results from each test are finally all combined into a diagram for comparison.

2.7 Reiteration

A variety of robot models were tested in order to find the ideal design for these particular simulations. The first iteration, see figure 2.8, failed due to wheel malfunctions resulting in the robot performing so called “barrel rolls” when reaching high velocities.

Figure 2.8. First iteration.

The reason for the second iteration, see figure 2.9, failing was due to asymmetry as well as virtual overweight. Simply put, the robot was too large, too uneven and too heavy which made it difficult for unity to establish a stable weight point, this led to the robot falling over face first more often than not.

Figure 2.9. Second iteration.

The previous iterations made it apparent that factors such a wheel dimensions, size, weight and symmetry play a crucial part when designing robots such as these. In light of the failures of the previous, more complex iterations, a general "less is more" mentality was applied when designing iteration three, see figure 2.10. However, this iteration proved unsuccessful as well since the chassi covered too much space in proportion to the size of the wheels. This sometimes led to the robot being

locked in position on uneven surfaces due to the wheels not being able to reach the ground.[1]

Figure 2.10. Third iteration.

Chapter 3

Results

This chapter simply presents the projects results without discussing what so ever.

3.1 Unity vs USARsim

After having evaluated and compared both USARsim and unity, mainly by reading and watching various tutorials it was decided that unity was the tool best suited for this project. The reasoning behind this is presented and discussed further on in the discussion section.

3.2 Most successful simulation

The user testing sessions were successful and it is now apparent which of the three interfaces is easier to use. See figures 3.1, 3.2, 3.3, 3.4, 3.1 for diagrams containing the test results for the simulation in question. See Appendix B and C for the rest of the test results.

3.2.1 Experience with simulations

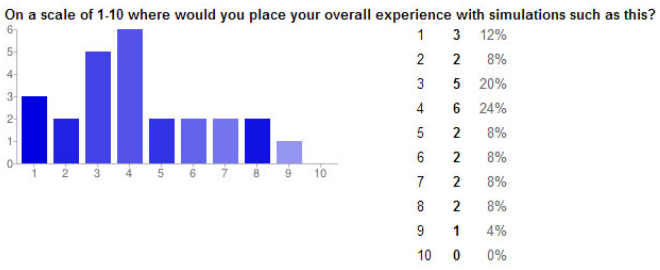

Figure 3.1. Interface 3: Experience with simulations.

3.2.2 Time trial time

Times: 72, 70, 64, 64, 70, 64, 59, 62, 60, 62, 70, 69, 64, 65, 62, 63, 61, 61, 65, 66, 59, 61, 66, 71 and 63 seconds. **Mean:** 63.32 seconds.

3.2.3 Time trial difficulty

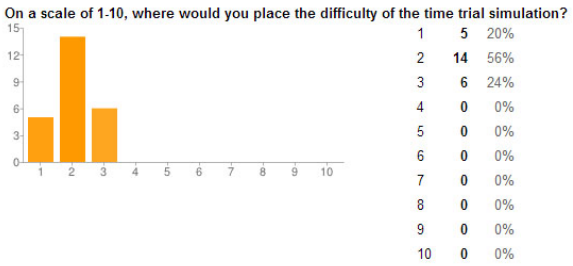

Figure 3.2. Interface 3: Time trial difficulty.

3.2.4 Remaining cubes

Cubes remaining: 5, 3, 3, 3, 4, 1, 1, 3, 2, 2, 4, 4, 2, 4, 2, 3, 3, 3, 3, 4, 1, 2, 4, 4, and 4 cubes. **Mean:** 2.96 cubes.

3.2.5 Find cubes difficulty

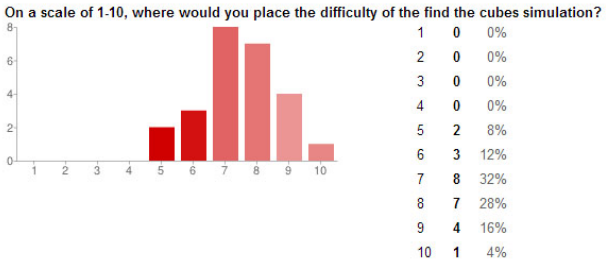

Figure 3.3. Interface 3: Find cubes difficulty.

3.2.6 Maneuverability

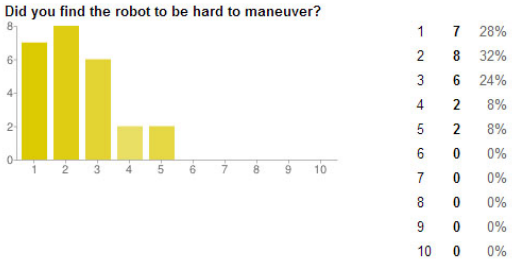

Figure 3.4. Interface 3: Maneuverability.

Chapter 4

Discussion

This section of the report brings up the various strong- and weak points of the many aspects of the project. Only by discussing, analyzing and comparing the good and the bad can an optimal solution to a potential problem be reached.

4.1 Why unity?

There are two main reasons why unity was chosen. The first one is the fact that there appears to exist a lot more information regarding learning unity rather than USARsim, this simplifies the learning process quite a bit considering that it is hard to educate oneself without the proper tools to do so. The second reason simply is that unity appears to be more widely used within the field, making it both more useful and fun to learn. However, it is also important to note that unity did in fact seem like a slightly harder tool to implement the project in. Still, it was decided that the amount of easily accessible information as well as its diversity more than made up for the increased difficulty. [2, 4, 6, 8, 9, 10, 12]

4.2 User testing results

By analyzing the test results from each simulation it becomes apparent that some interfaces performed better than others. The can be several reasons as to why this is the case. Earlier in the report it was stated that all three of the interfaces were designed with the general idea of maximizing the users and thereby also the drones awareness in terms of its immediate surroundings. There are probably multiple ways to implement drones that meet these criterias, and truth be told the implementation in regards of this particular project is probably one of the simpler methods of designing such a drone.

However, simplicity is not necessarily a bad thing. By taking the test results into account it soon becomes clear that an interface containing a lot of information as once risks confusing the user which affects performance. There is a difference in

regards of the two simpler interfaces compared to the third slightly more complex interface. When asked to rate the difficulty of the interface usage as well as the drone maneuverability it became clear that simpler interfaces were perceived as more easily used which subsequently made it easier to maneuver the drone. In short, the first interface which contains three camera monitors did indeed confuse the users at times, having three different perspectives on the same screen sometimes resulted in uncertainties in regards of where to look. It is however interesting to note that this did not necessarily always affect performance, which becomes apparent when looking at the results from the find the cubes simulation.

Finally, it is also worth bearing in mind that the human margin for error can affect the test results to a certain extent as well rather than it being a result of the interface being hard to use or inefficient.

4.3 Possible optimizations

One possible way of optimizing the drone would be to provide it with several so called modes. What this essentially means is that the drone acts differently depending on the mode chosen by the user. For example, should the drone be equipped with a "safe mode", then this could probably make the drone avoid collisions with people or perhaps large objects. Seeing as the primary goal of USAR robots is to help when navigating through disaster zones this would probably prove to be quite useful in regards of keeping both disaster victims and the robot out of harms way. Another example could be to equip the drone with a so called "autonomous" mode. Once selected, the autonomous mode essentially allows the robot to control itself within certain limits. Instead of steering the drone the user now merely has to provide it with information regarding its destination and designated actions. Should the case be that a person requires extraction, simply provide the drone with an extraction location and command and it will then attempt to complete the task to its best ability. This could probably simplify the interface usage process a great deal if done properly considering how the user would no longer need to actually steer the drone.

To further optimize the scene it could be reasonable to construct a script that disables the cameras that are not currently active. The current solution renders superfluous cameras in the backround, but, since Unity renders everything specified, even if it is not shown, in order from lowest to highest depth value, it could lead to an excess of CPU-power being used for cameras that the user cannot see not to mention memory leakage.

Chapter 5

Conclusions

A couple of conclusions can be drawn when considering how the project was developed as well as the test results that were obtained. Firstly, given the abundance of easily accessible information regarding Unity in comparison to USARsim it soon became obvious that Unity had the required tools for developing this project and that the internet provided enough references with information on how to properly make use of said tools.

Secondly, as previously mentioned, the test results show that users found the simpler interfaces to be more comfortable and easy to use. However, this does not necessarily mean that the simpler interfaces were better given that the increase in comfort was not always reflected in regards of user performance. In short, it will take more than simply providing comfort for the users in order to create the perfect interface. While comfort proved to be an important factor in regards of how the users reacted to the simulation it did not in fact affect performance as much as expected. Furthermore, considering how the design of the virtual drone also affected performance in the form of it either falling over or getting stuck it becomes apparent that one needs to bear the stability factor in mind when designing units of this sort.

Given all of the above the conclusion can be drawn that when designing a USAR robot interface it is important to consider more than just any one factor. The tests showed that simplicity is not everything, neither is awareness in regards of the surroundings. What is required to create a well functioning USAR robot interface is balance, simply put. Balance between factors such as the ones mentioned earlier, but also for example design stability and overall efficiency. Only when taking factors such as these into account can a well functioning and efficient robot interface be created.

Bibliography

[1] James Arndt. *Creating a driveable vehicle in Unity 3D.* http://www.youtube.com/watch?v=21zuMIsy2GM Last updated: 2011-10-04. Accessed: 2013-02-08.

[2] Feillyne. *How to make a minimap.* http://www.moddb.com/groups/unity-devs/tutorials/how-to-make-a-minimap Last updated: 2010-10-17. Accessed: 2013-04-09.

[3] Illah Nourbakhsh. *Robotic Technology for USAR.* http://www.cs.cmu.edu/ illah/CLASSDOCS/Week4Class1USaRabxpdf.pdf Last updated: 2005-10-03. Accessed: 2013-02-1.

[4] Max Stuttard Parker. *Getting Started with Unity.* http://active.tutsplus.com/tutorials/unity/getting-started-with-unity/ Last updated: 2011-03-21. Accessed: 2013-02-18.

[5] Autodesk Maya team. *Maya homepage* . http://www.autodesk.com/products/autodesk-maya/overview Last updated: 2013-04-12. Accessed: 2013-02-08.

[6] SourceForge team. *USARSim wiki.* http://sourceforge.net/apps/mediawiki/usarsim/index.php?title=Main$_{P}age$ $Lastupdated : 2013 - 02 - 07.Accessed : 2013 - 02 - 01.$

[7] Unity team. *Unity - Camera documentation.* http://docs.unity3d.com/Documentation/Components/class-Camera.html Last updated: 2013-03-07. Accessed: 2013-02-18.

[8] Unity team. *Unity - GameObject.* http://docs.unity3d.com/Documentation/Components/class-GameObject.html Last updated: 2013-01-31. Accessed: 2013-02-18.

[9] Unity team. *Unity - GameObjects.* http://docs.unity3d.com/Documentation/Manual/GameObjects.html Last updated: 2010-09-14. Accessed: 2013-02-18.

[10] Unity team. *Unity homepage.*
http://unity3d.com
Last updated: 2013-04-02. Accessed: 2013-02-08.

[11] Gabriel Williams. *Unity: Creating a game in 30 mins.*
http://cgcookie.com/unity/2011/11/30/unity-creating-a-game-in-30mins/
Last updated: 2011-12-07. Accessed: 2013-04-10.

[12] Ian Zamojic. *Unity3D: Third Person Cameras.*
http://mobile.tutsplus.com/tutorials/game-engine/unity3d-third-person-cameras/
Last updated: 2012-06-21. Accessed: 2013-04-08.

Appendix A

Unity Scripts

A.1 OrbitCam.js

```
#pragma strict

var target : Transform;

function LateUpdate() {
  if (target) {
    transform.position.x = target.position.x;
    transform.position.y = target.position.y + 90;
    transform.position.z = target.position.z;
  }
}

function Start() {
  // Make the rigid body not change rotation
    if (rigidbody)
    rigidbody.freezeRotation = true;
}

function Update() {}
```

A.2 ScoreCounter.js

```
#pragma strict

public static var CubeScore = 10;

function OnGUI() {
    GUI.Label(Rect(10,10,1000,20), "Cubes Remaining");
```

```
GUI.Label(Rect(10,30,100,20), CubeScore.ToString());
}
```

A.3 ScoreTime.js

```
#pragma strict

public static var TimeIsOn = true;
private var time = 0;

function OnGUI() {
    GUI.Label(Rect(10,50,1000,20), "Elapsed Time");
GUI.Label(Rect(10,70,100,20), time.ToString());
}

function Awake() {
TimerTick();
}

function TimerTick() {
    // while there are seconds left
    while(TimeIsOn)
    {
        // wait for 1 second
        yield WaitForSeconds(1);

        // increase the time
        time++;
    }
}
```

A.4 ScoreCountDown.js

```
#pragma strict

private var time = 180;

function OnGUI() {
    GUI.Label(Rect(10,50,1000,20), "Time Remaining");
GUI.Label(Rect(10,70,100,20), time.ToString());
}

function Awake() {
```

```
TimerTick();
}

function TimerTick() {
    // while there are seconds left
    while(time > 0)
    {
        // wait for 1 second
        yield WaitForSeconds(1);

        // increase the time
        time--;
    }
}
```

A.5 ScoreCountDown.js

```
#pragma strict

var forwardSpeed : float = 3;
var turnSpeed : float = 2;
var FrontLeftWheel : WheelCollider;
var FrontRightWheel : WheelCollider;

//Detect collision and perform appropriate actions.
function OnCollisionEnter(col : Collision) {
    if(col.gameObject.tag == "Cube"){
        ScoreCounter.CubeScore -= 1;
     Destroy(col.gameObject);
    } else if(col.gameObject.tag == "StartText"){
     ScoreTime.TimeIsOn = true;
     Destroy(col.gameObject);
    } else if(col.gameObject.tag == "FinishText"){
     ScoreTime.TimeIsOn = false;
     Destroy(col.gameObject);
    }
}

function Update() {
//The actual force applied to push the robot forward, i.e the speed with which the robo
var forwardMoveAmount = Input.GetAxis("Vertical")*forwardSpeed;

//The actual amount with which the robot turns.
```

```
var turnAmount = Input.GetAxis("Horizontal")*turnSpeed;
//Rotates the robot.
transform.Rotate(0,turnAmount,0);

//Pushes the robot forward with a force
rigidbody.AddRelativeForce(0,0,forwardMoveAmount);

FrontLeftWheel.steerAngle = 10 * Input.GetAxis("Horizontal");
FrontRightWheel.steerAngle = 10 * Input.GetAxis("Horizontal");
}
```

A.6 CameraSwitch.js

```
//Script for switching between cameras within your scene.

//You can place this script on an empty gameobject in your scene,
//just make sure to drag the appropriate camera into it's appropriate
//slot in the inspector.

public var fullFirstPersonCam : Camera;
public var fullThirdPersonCam : Camera;
public var orbitCam : Camera;

function Update() {

if (Input.GetKey(KeyCode.F1)) {
        orbitCam.rect = new Rect(0.5, -0.5, 1, 1);
        fullThirdPersonCam.depth = -5;
        fullFirstPersonCam.depth = -5;
    }

if (Input.GetKey(KeyCode.F2)) {
     orbitCam.rect = new Rect(0.7, -0.7, 1, 1);
     fullThirdPersonCam.depth = -5;
        fullFirstPersonCam.depth = 5;
    }

if (Input.GetKey(KeyCode.F3)) {
     orbitCam.rect = new Rect(0.7, -0.7, 1, 1);
     fullThirdPersonCam.depth = 5;
        fullFirstPersonCam.depth = -5;
    }
}
```

A.7 ControlScript.js

```
#pragma strict

var forwardSpeed : float = 3;
var turnSpeed : float = 2;
var FrontLeftWheel : WheelCollider;
var FrontRightWheel : WheelCollider;

//Detect collision and perform appropriate actions.
function OnCollisionEnter(col : Collision) {
    if(col.gameObject.tag == "Cube"){
        ScoreCounter.CubeScore -= 1;
     Destroy(col.gameObject);
    } else if(col.gameObject.tag == "StartText"){
     ScoreTime.TimeIsOn = true;
     Destroy(col.gameObject);
    } else if(col.gameObject.tag == "FinishText"){
     ScoreTime.TimeIsOn = false;
     Destroy(col.gameObject);
    }
}

function Update() {
//The actual force applied to push the robot forward, i.e the speed with which the robo
var forwardMoveAmount = Input.GetAxis("Vertical")*forwardSpeed;

//The actual amount with which the robot turns.
var turnAmount = Input.GetAxis("Horizontal")*turnSpeed;
//Rotates the robot.
transform.Rotate(0,turnAmount,0);

//Pushes the robot forward with a force
rigidbody.AddRelativeForce(0,0,forwardMoveAmount);

FrontLeftWheel.steerAngle = 10 * Input.GetAxis("Horizontal");
FrontRightWheel.steerAngle = 10 * Input.GetAxis("Horizontal");
}
```

A.8 WheelAlignment.js

```
 // Define the variables used in the script, the Corresponding collider is the wheel co
// the visible wheel, the slip prefab is the prefab instantiated when the wheels slide,
```

```
// value used to rotate the wheel around it's axel.
var CorrespondingCollider : WheelCollider;
var SlipPrefab : GameObject;
public var RotationValue : float = 0.0;

function Update () {

    // define a hit point for the raycast collision
var hit : RaycastHit;
// Find the collider's center point, you need to do this because the center of the coll
// the real position if the transform's off.
var ColliderCenterPoint : Vector3 = CorrespondingCollider.transform.TransformPoint( Cor

// now cast a ray out from the wheel collider's center the distance of the suspension,
// variable's data to find where the wheel hit, if it didn't, then se tthe wheel to be
if ( Physics.Raycast( ColliderCenterPoint, -CorrespondingCollider.transform.up, hit, Co
transform.position = hit.point + (CorrespondingCollider.transform.up * CorrespondingCol
}else{
transform.position = ColliderCenterPoint - (CorrespondingCollider.transform.up * Corres
}

// now set the wheel rotation to the rotation of the collider combined with a new rotat
// is the rotation around the axle, and the rotation from steering input.
transform.rotation = CorrespondingCollider.transform.rotation * Quaternion.Euler( Rotat
// increase the rotation value by the rotation speed (in degrees per second)
RotationValue += CorrespondingCollider.rpm * ( 360/60 ) * Time.deltaTime;

// define a wheelhit object, this stores all of the data from the wheel collider and wi
// the slip of the tire.
var CorrespondingGroundHit : WheelHit;
CorrespondingCollider.GetGroundHit( CorrespondingGroundHit );

// if the slip of the tire is greater than 2.0, and the slip prefab exists, create an i
// a zero rotation.
if ( Mathf.Abs( CorrespondingGroundHit.sidewaysSlip ) > 1.5 ) {
if ( SlipPrefab ) {
Instantiate( SlipPrefab, CorrespondingGroundHit.point, Quaternion.identity );
}
}
}
```

Appendix B

User testing: Interface 1

The following appendix contains the data of one of the interface in the simulation that proved the least successful. In this case, these diagrams will correspond to the simulations utilizing the first interface.

B.1 Experience with simulations

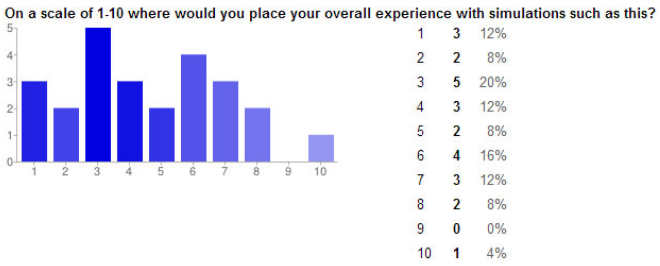

Figure B.1. Interface 1: Experience with simulations.

B.2 Time trial time

Times: 65, 67, 64, 69, 80, 90, 68, 75, 84, 78, 74, 68, 73, 73, 65, 69, 66, 70, 70, 63, 72, 84, 64, 66 and 60 seconds. **Mean:** 71.08 seconds.

B.3 Time trial difficulty

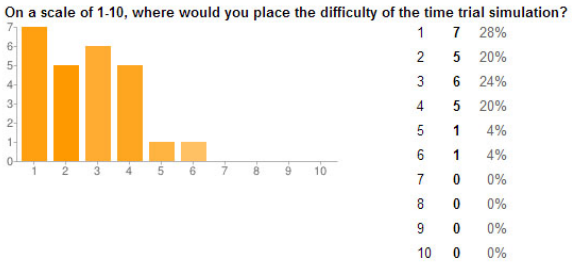

Figure B.2. Interface 1: Time trial difficulty.

B.4 Remaining cubes

Cubes remaining: 6, 7, 5, 4, 8, 4, 4, 5, 7, 5, 5, 6, 5, 6, 6, 5, 7, 8, 4, 4, 7, 5, 4, 2 and 5 cubes. **Mean:** 5.36 cubes.

B.5 Find cubes difficulty

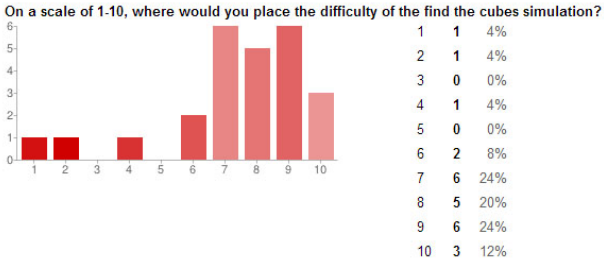

Figure B.3. Interface 1: Find cubes difficulty.

B.6 Maneuverability

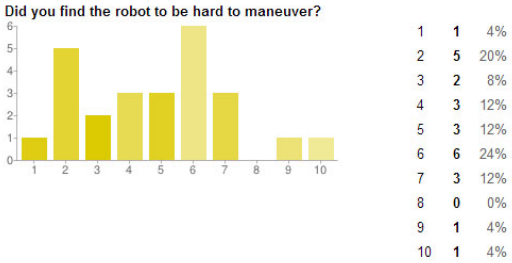

Figure B.4. Interface 1: Maneuverability.

B.7 Interface ease of use

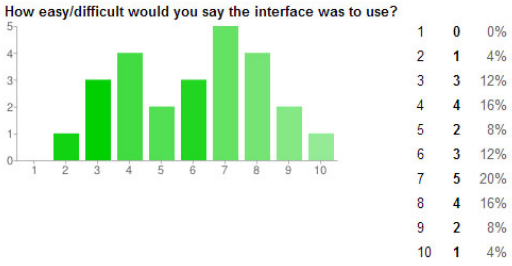

Figure B.5. Interface 1: Interface ease of use.

Appendix C

User testing: Interface 2

The following appendix contains the data for the interface of the simulation that did not prove successful, but still more successful than the first simulation, in this case, these diagrams correspond to the interface utilized by the second simulation.

C.1 Experience with simulations

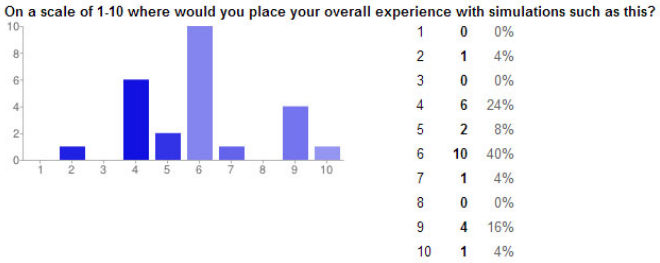

Figure C.1. Interface 2: Experience with simulations.

C.2 Time trial time

Times: 72, 70, 64, 64, 70, 64, 59, 62, 60, 62, 70, 69, 64, 65, 62, 63, 61, 61, 65, 66, 59, 61, 66, 71 and 63 seconds. **Mean:** 63.32 seconds.

C.3 Time trial difficulty

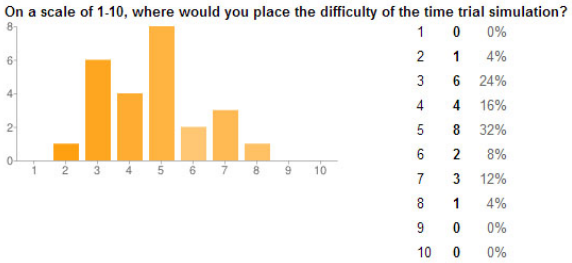

Figure C.2. Interface 2: Time trial difficulty.

C.4 Remaining cubes

Cubes remaining: 5, 6, 2, 5, 6, 3, 4, 3, 6, 4, 5, 7, 4, 8, 4, 2, 3, 6, 3, 1, 6, 5, 5, 1, and 1 cubes. **Mean:** 4.2 cubes.

C.5 Find cubes difficulty

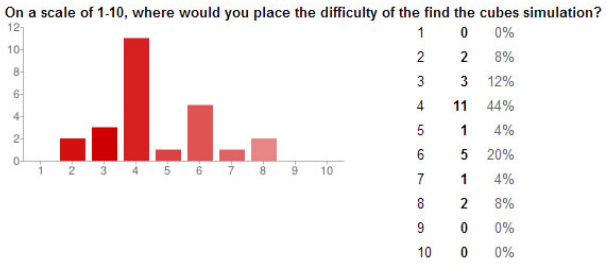

Figure C.3. Interface 2: Find cubes difficulty.

C.6 Maneuverability

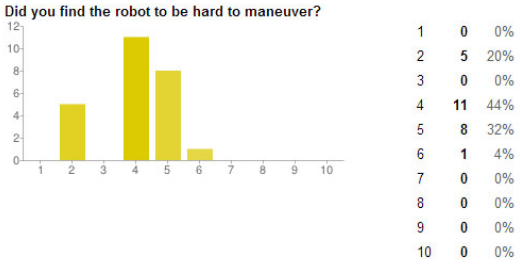

Figure C.4. Interface 2: Maneuverability.

C.7 Interface ease of use

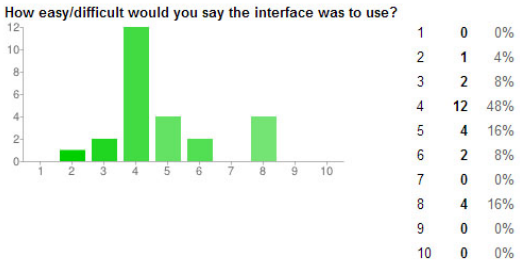

Figure C.5. Interface 2: Interface ease of use.