May 2011

Development and Evaluation of a Simulation Tool for Robotics Education

Erick T. Read
Worcester Polytechnic Institute

Justin A. Gostanian
Worcester Polytechnic Institute

William Spencer Mulligan
Worcester Polytechnic Institute

DEVELOPMENT AND EVALUATION OF A SIMULATION TOOL FOR ROBOTICS EDUCATION

An Interactive Qualifying Project Submitted to the Faculty of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

By

_____

Justin Gostanian,
Computer Science Major

_____

William Mulligan,
Robotics Engineering Major

_____

Erick Read,
Robotics Engineering Major

May 14, 2011

_____

Taskin Padir, Project Advisor

TABLE OF CONTENTS

TABLE OF FIGURES

EXECUTIVE SUMMARY

This Interactive Qualifying Project (IQP) is aimed at constructing a virtual robotic simulator called Lua Visual and then testing its effectiveness as a learning tool. A major problem with most robotic learning platforms for robotics is that they are costly and require support staff to educate the students. The price of the software, hardware, and the support staff can create a cost that most schools cannot afford. Lua Visual is a virtual robotic simulator created within the scope of this IQP. The purpose of developing Lua Visual is to provide an educational tool that is be easy for students to use without any prior programming knowledge. Since Lua Visual is open source it requires no extra cost for parts or maintenance other than the computers that they are running on and the internet connection used to download it from the site.

With the objective of developing an easy to use education tool, the development of Lua Visual started. Construction of Lua Visual started by first deciding on what specifications the simulator should have and by researching existing platforms. To determining the goals for Lua Visual, other simulators and related products that aimed to accomplish a similar goal were researched. The research went over many different kinds of platforms that were both virtual and complete robotic kits. After reviewing related platforms we decided to produce our own simulator, using design concepts from the simulators we had researched. Concepts such as ease of use, programing language, and executable software were incorporated into Lua Visual to make it easier to use and appeal to a larger base demographic.

The first step in the construction of Lua Visual was to determine the programing language that we would base the software in. It was determined after looking through other

programing languages that Lua would be used for its simplicity and ability to be easily understood by students with no prior programming knowledge. Lua is a fast, lightweight, embeddable scripting language that is very easy to learn. Once we knew what language we wanted to use in the simulator, the construction of the simulator and the integration of the game engine was started. Work with the graphics engine, C++, and Lua were all required in order to get the simulator fully functional to allow user testing.

Once Lua Visual was developed, a time and place had to be determined to test the simulator on our target demographic. It was determined that working with the students participating at WPI's F.I.R.S.T. Regional Competition would be a great place to get volunteers to help determine the usefulness of our simulator. The usefulness of Lua Visual and the background of the students were determined by having the students fill out a survey after they completed their session. The survey was made to determine the skills of each participant, their background with other programing languages, the academic level of the student, and the participant's personal thoughts on their experience while using Lua Visual. At the competition twenty three high school students and two college level students participated in our Lua Visual challenge and completed the survey.

Once the results of the first test group had been collected it was determined that more data was needed to determine its usefulness and effectiveness. Since the first test group was made up of mostly high school students, it was decided that the second test should be run on college level students since they are also part of the target demographic. The help of ten robotic engineering students from WPI was recruited and each participated in the same challenges and survey as the first group. The participants that in the second test allowed for

further analysis of the results to compare high school level students versus college level students. This difference in grade level and programming experience allows a better understanding of the scale of usefulness that Lua Visual offers, and its appeal to more than just high school students with little programming background.

The data was compiled for both of the test groups. Using the information learned in the surveys, the academic grade and programming level of all the participants was determined. Using the surveys, the similarities and differences in how the students felt about Lua Visual as a learning tool was also determined. The results showed that although many of the participants did not have much of a background in programming they are still able to use Lua Visual and complete the tasks with minimal problems. It was determined that the students that have a stronger programming background enjoyed using Lua Visual and completed the challenges on at the same average speed that participants with little programming experience did.

Lua Visual fulfilled all of the goals that were set out to meet when the IQP was started. It is an easy to use, virtual robotics simulator that can be downloaded for free off of the website (www.luavisual.com). Lua Visual does not require students to have extensive knowledge about programming and is a useful tool for students to learn about robotics programming. Students interested in robotics programming can download the same challenges used in the testing and learn about problem solving, logic, creativity, and the integration of math and science in real world applications. Future development of Lua Visual can involve enhancing the physics of the simulation and providing more features to the user such as configurable sensors.

1. INTRODUCTION

This paper presents the details of the development and evaluation of a robot simulator designed within the scope of an Interactive Qualifying Project (IQP) by undergraduates at Worcester Polytechnic Institute. Interactive Qualifying Projects challenges undergraduate students to learn about the role of science and technology, and its impact on society.[1] Students are encouraged to use the IQP to expand their knowledge outside their major and to solve open ended, complex problems.[1] This project is typically completed in the students Junior year and is equivalent to three courses.

1.1. ROBOTICS

"I can envision a future in which robotic devices will become a nearly ubiquitous part of our day-to-day lives." stated Bill Gates in a December 2006 interview for Scientific American Magazine[2]. Robotics is a field of engineering that has yet to drastically alter our society, but many people like Bill Gates expect the impact of robots in our society to increase dramatically in the future.

In past years, figures have showed a constant decrease in the number of students entering engineering and science programs[3]. This alarming trend has fueled the need for creating math and science programs to engage students at an early age. Due to its hands-on

[1] WPI. <http://www.wpi.edu/academics/catalogs/ugrad/iqp.html>
[2] December 2006 interview for Scientific American Magazine
[3] Thibodeau, Patrick. "Obama backs U.S. return to math, science, tech".

and interdisciplinary nature, robotics has been able to seize student's interest as few other topics can[4].

Programs such as *For Inspiration and Recognition of Science and Technology (FIRST) Robotics Competition*[5] and *RoboCup*[6] have showed that robotics is able to attract tens of thousands of students on an international scale. These programs offer opportunities for students in middle school or high school to get hands on experience with robotics, but these competitions require a significant amount of resources from participating schools.

FIRST Robotics Competition is an international high school robotics competition where teams build up to 120 pound robots each year to complete a specific task. FIRST's mission is "to inspire young people to be science and technology leaders, by engaging them in exciting mentor-based programs that build science, engineering and technology skills, that inspire innovation, and that foster well-rounded life capabilities including self-confidence, communication, and leadership."[7] Figure 1 shows the impact FIRST has on students after high school.[8] As shown, students who participate in FIRST are four times more likely pursue a career in engineering.

---

[4] Carpin, Stefano. "USARSim: a robot simulator for research and education."
[5] "FIRST at a Glance." *USFIRST.org*. 9 Feb. 2011. Web. 10 Apr. 2011. <http://www.usfirst.org/>.
[6] RoboCup. Web. 1 May. 2011. <http://www.robocup.org/>
[7] "FIRST at a Glance." *USFIRST.org*. 9 Feb. 2011. Web. 10 Apr. 2011. <http://www.usfirst.org/>.
[8] "Impact." *USFIRST.org*. Web. 10 Apr. 2011. <http://www.usfirst.org/>.

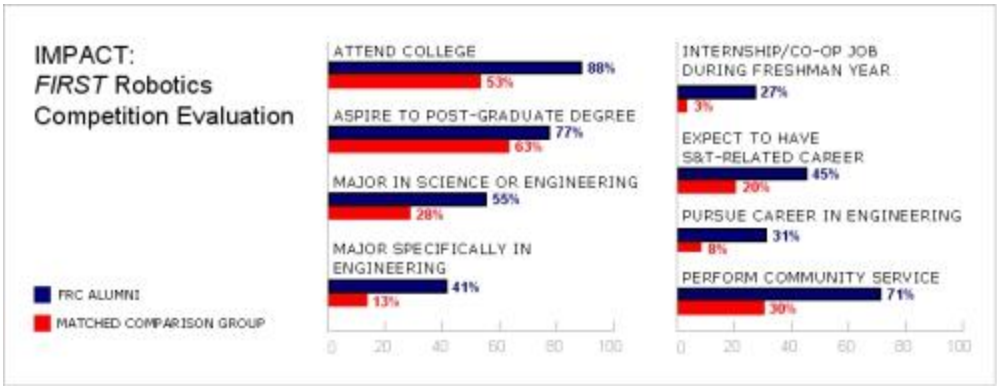


Figure 1: Impact of FIRST Robotics Competition on students after high school

RoboCup is another international competition aimed to develop autonomous soccer robots. RoboCup is aimed at higher level education by promoting robotics and AI research through a formidable challenge. RoboCup has several leagues, but most leagues involve using humanoid robots to play a game of soccer against other robots on a miniaturized soccer field. This challenge tests vision, locomotion, multi-robot communication, and artificial intelligence.[9]

1.2. THE PROBLEM

Robotic competitions and robotics education require a great deal of dedicated equipment, lab space, supporting personnel, and funding for robot components. Many robotic competitions require a machine shop for building and cutting materials for robots. Computers are required to program robots along with any necessary software for 3D modeling or programming. The amount of funding is dependent on the specific competition or project, but usually will require many thousands of dollars for materials and/or transportation. For example, the registration cost to compete in the FIRST robotics competition for 2011 is $5,000 USD[10]. In

[9] RoboCup. Web. 1 May. 2011. <http://www.robocup.org/>
[10] FIRST. "Registration Pricing 2011". Web May 14. <http://www.usfirst.org/roboticsprograms/frc/content.aspx?id=460>

addition to the cost of materials and software, robotics competitions also require teachers and support staff to be present to aid the students. In total, this places a large burden on any school who would like to participate in these robotics competitions or teach robotics in a classroom.

Due to the complexity and cost related to robotic competitions and education, this research group chose to develop and evaluate an alternative solution for educating students about robotics. Our approach would be to implement a robotic simulator and evaluate the ability for students to use and learn from the software. This decision was derived from the fact a simulator could be offered at no additional cost over the Internet, and could teach students many of the fundamental skills that the real life competitions offer.

1.3. LUA VISUAL

Our IQP group developed and evaluated the effectiveness of a robotic simulation as an educational tool for the learning of robotics and computer science. A robot simulator does not require schools to purchase any additional equipment beyond computers and would allow the instructor to provide a fun and educational method for teaching students valuable skills such as problem solving, logic, creativity, and the integration of math and science in real world applications.

As part of the IQP, we developed a simulator, called Lua Visual. Lua Visual is an executable program for Windows computers that allows the user to control a virtual robot. A virtual robot is a simulated representation of a robot that can navigate the environment created by the simulator with the logic that the students develop. The virtual robot shares many of the same principals as a real robot, like using sensors to perceive the

environment. This makes the simulator ideal to teach students the same skills that would traditionally require expensive hardware.

Lua Visual uses Lua to allow the user to program the behavior of the virtual robot. Lua is a scripting language which means that the program does not need to be compiling prior to execution.[11] This is ideal for inexperienced users who would have difficulty setting up the environment for languages like C++ and Java which require compilation. Lua has very little syntax which makes it easy for new users to quickly understand the program. Lua Visual could be considered an IDE or Integrated Development Environment.

1.4. OBJECTIVES

The objectives for this IQP are to develop and evaluate a robotics simulation for the purpose of education. The specific project objectives are:

1. Create a 3 dimensional robot simulator that uses Lua to program a virtual robot to navigate through a maze. Navigating a maze is a complex task that requires the robot to sense the environment and react accordingly. This provides a challenge for users of the simulator.
2. Create a website which contains tutorials and user instructions and the executable for the program.
3. Evaluate and investigate the benefits and disadvantages of the simulation tool by surveying high school and college students on the effectiveness and accessibility of the program.

Specifically, the robotics simulator called Lua Visual must meet the following objectives:

[11] PUC-Rio. "Lua." N.p., 6 9 2010. Web. 15 Dec 2010. <http://www.lua.org/home.html>

1. Allow users to create and edit Lua files inside the program. This is an important requirement since it allows the user to write code and run the simulation inside of a single application. The Lua scripts are used to control the virtual robots in the simulation.
2. This application should use an OpenGL engine which runs the simulation. OpenGL is a computer graphics API which allows rendering of 3D scenes. OpenGL has the advantage of being cross-platform, so future work could make the software work on Windows, Mac, and Linux.
3. The virtual robot should be a differential drive robot, which is a robot with two independently driven wheels, inside a predefined maze. A differential robot is the most common type of wheeled robot due to its agility and simplicity. The maze should have a start point and a goal for the robot to determine when the maze has been successfully navigated.
4. The virtual robot should know its current location in the maze using odometry and should have range-finding sensors on each side of the robot. Odometry involves taking data from moving sensors and estimating the position of the robot. This is the common technique for robots to estimate their position using internal sensors such as encoders.

1.5. RELATED WORK

While using simulators as a tool for robotics is not new, many of the existing simulators exist for the purpose of programming commercial robots or research at the college level. Our aim with Lua Visual is to create an application that can be used by non-programmers to learn about robotics. Studying the currently available software is an important part of determining what can be improved upon with Lua Visual. Some commonly used simulators are Player/Stage, Webots, USARSim, and OpenSim.

1.5.1. PLAYER/STAGE

One popular robotics simulator is Player/Stage. Stage simulates a population of mobile robots, sensors and objects in a two-dimensional bitmapped environment. Stage is designed to support research into multi-agent autonomous systems, so it provides computationally cheap models of lots of devices rather than attempting to emulate any device with great fidelity.[12] Stage is part of project to create free software for research into robotics and sensor systems. This project is called the "Player Project." Player is designed to be language and platform independent, though only languages that support TCP sockets can be used. TCP sockets is a network communication protocol which would allow any language or application to send commands to robots. Player is meant to be used in actual robots, but Stage is a simulator that implements Player's software to perform a simulation.[12] Despite this limitation of needing knowledge of TCP sockets, Stage has a lot of ideas that could be applied to Lua Visual. As can be seen in Figure 2, in Stage, the world created for the simulation is limited to a two-dimensional maze and the robots are simple polygons. These ideas have been used in the creation Lua Visual to make it a lot simpler. However, Lua Visual did not accurately attempt to simulate physics in the way Stage does. Position and velocity of the robot is all that is calculated in Lua Visual, while Stage attempts to give the most realistic simulation possible, as the physics of Stage are very accurate.[12]

[12] Hedges, Reed. "Stage." *Player Project*. Web. 30 Nov 2010. <http://playerstage.sourceforge.net/index.php?src=stage>.

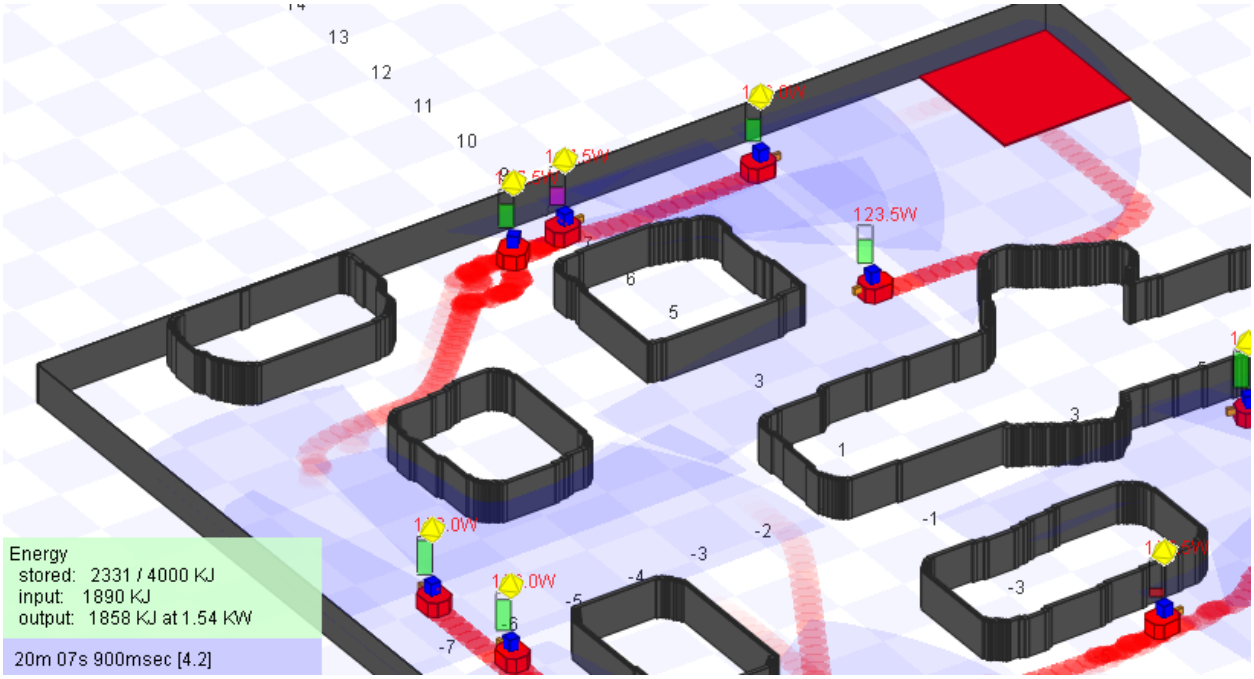


Figure 2: Multi-robot simulation inside of Stage[12]

1.5.2. WEBOTS

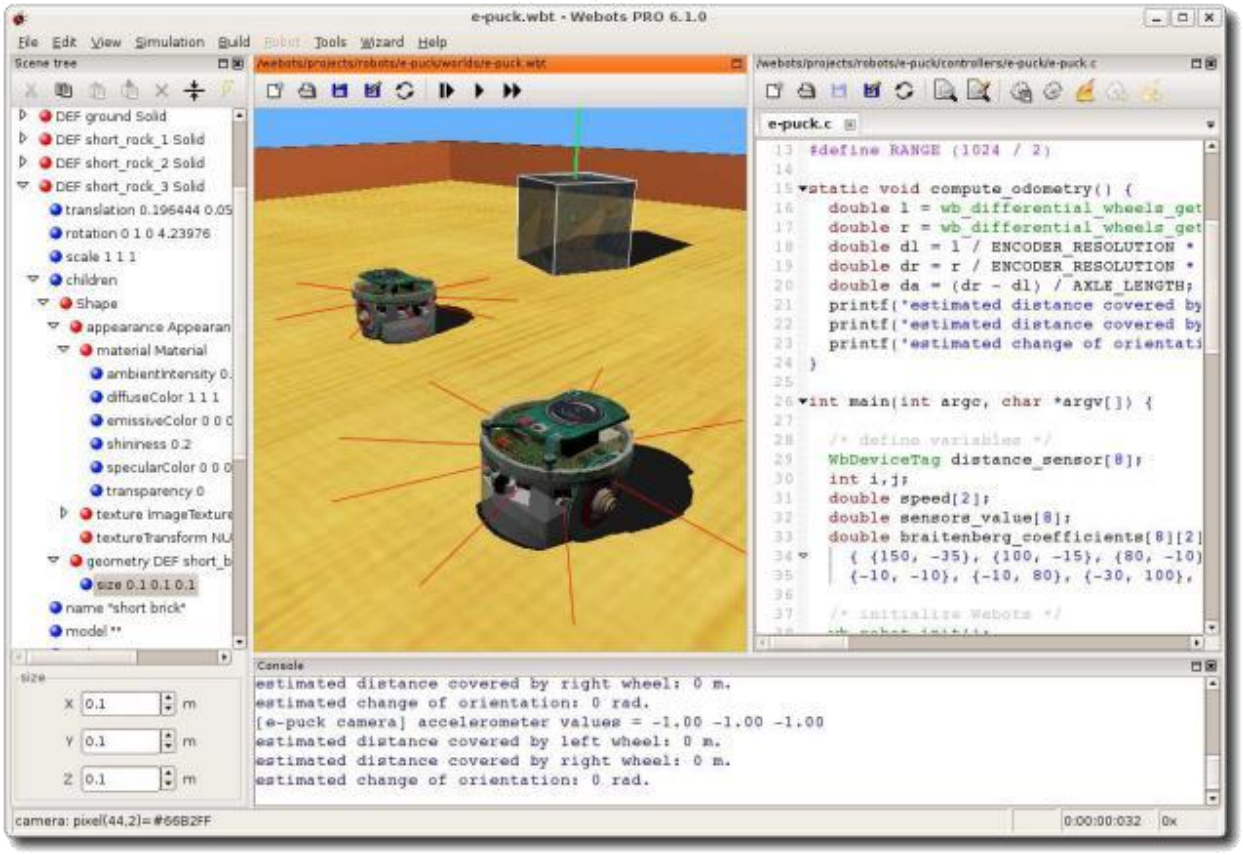


Figure 3: A virtual robot inside of Webots[13]

Webots, as seen in Figure 3, is a development environment used to model, program, and simulate mobile robots.[13] It allows the user to test robot behavior in a realistic world using simulated physics, and the controller programs can be transferred over to real robots. It supports programming in C, C++, Java, Python, URBI, and MATLAB. It is used as an educational tool – the website claims it is used in over 750 universities.[13] Webots is not quite the level of simplicity we are aiming for though, as it still relies heavily on the user's programming knowledge. Webots allows for very realistic simulations, but it is a complicated program. In order to make the simulator most accessible to those with little experience, Lua Visual has

[13] Webots. "Webots User Guide." Web. 10 Apr. 2011. <http://www.cyberbotics.com/guide.pdf>

simplified physics and requires less programming knowledge, while still being able to teach programming concepts. However, Webots is clearly a good example of a robotics simulator usable for education.

1.5.3. USARSIM

The USARSim[14] or Unified System for Automation and Robot Simulation is a general purpose research tool for high-fidelity simulations of robots and environments. It is built upon the popular Unreal Tournament game engine used in many modern video games. The simulation is primarily being used in the RoboCup rescue virtual robot competition as shown in Figure 4. USARSim was initially developed to support differential drive wheeled robots, but due to increased interest, USARSim now supports underwater vehicles, legged platforms, and humanoids.

USARSim offers many useful features to users to simulate a wide range of robots. Multiple sensors can be simulated on the robot with the ability to modify the location, position, and noise level of each sensor. Since the Unreal Tournament Engine is used, new environments can be created using the level creator. However, USARSim is partly limited due to its usage of the Unreal Tournament engine, which is not open source.

[14] USARSim <http://sourceforge.net/projects/usarsim/>

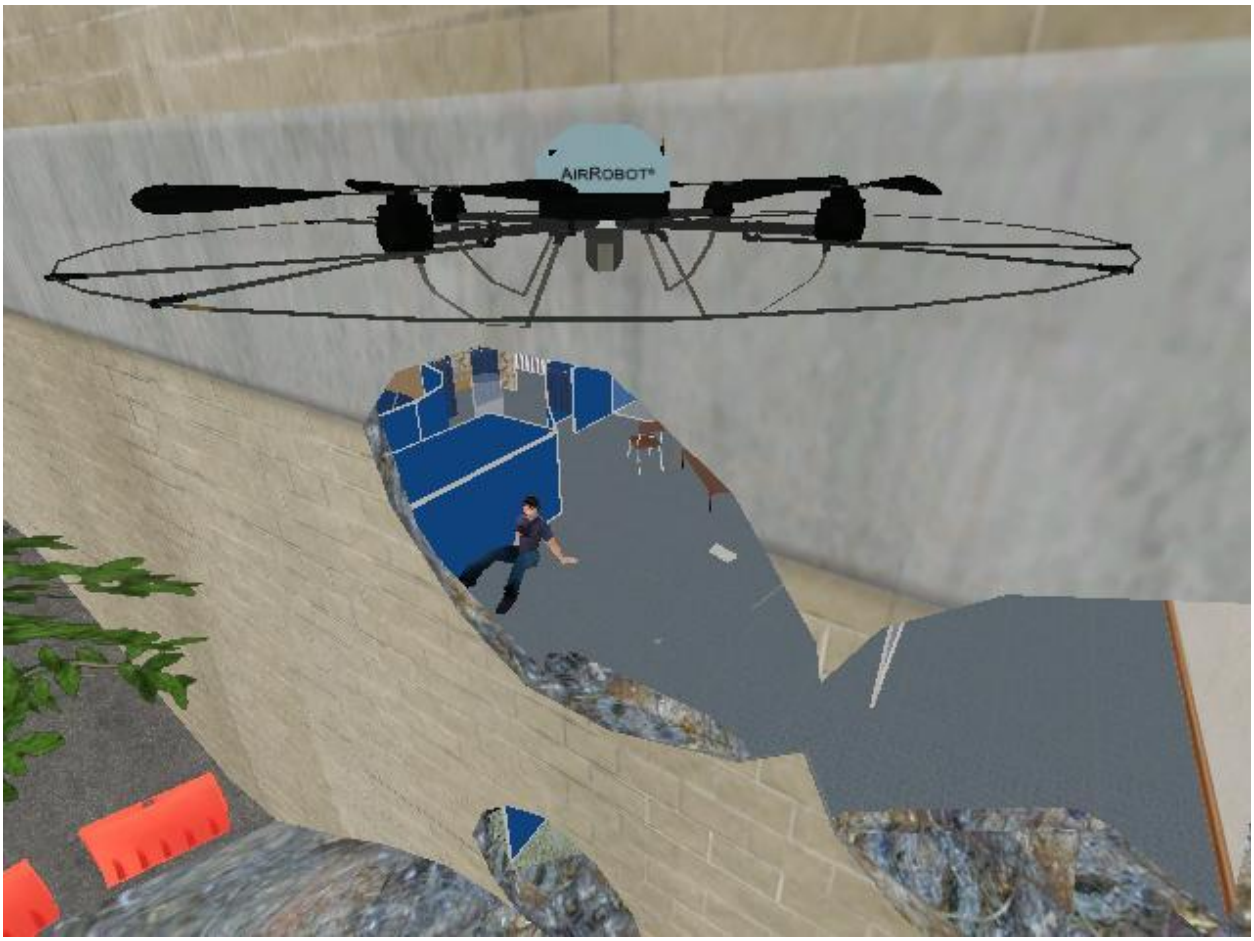


Figure 4: Aerial Robot looking inside a building using USARSim[14]

1.5.4. OPENSIM

OpenSim[15] is an open source robotics simulator, but it is primarily designed for a professional level (research and development). OpenSim's website explains its goal:

> *"The aim of this project is to create a tool for higher-fidelity real-time simulations of autonomous mobile robots. Real-time simulation also allows hardware-in-the-loop simulations. We're working toward a 3D simulator that uses OpenGL for real-time rendering of the robot environment as realistically as possible, including a physics engine to simulate dynamics in real-time."*

[15] OpenSim <http://opensimulator.sourceforge.net/>

OpenSim, seen in Figure 5, is an open source tool developed by an individual interested in robotics research, is complicated enough that it could never be used as an education tool for high school and lower level undergraduate students. Firstly, the robots are coded in Java, so anyone wishing to use it would have to know Java. Secondly, OpenSim gives a lot of power to the user, such as ability to control individual robot arms. These aspects are useful for those who know what they are doing, but for the purposes of this study. This is a good feature for experienced robotics developers, but complicated software would only confuse novice high school students. Though our simulator lacks the features of OpenSim (as in, OpenSim is capable of producing a more realistic simulation), the simplicity will make it more useful for teaching high school and freshmen-level college students robotics programming. There are not features in Lua Visual that would only make the experience frustrating for a person with limited knowledge in designing and programming robots. Rather than focusing on software that can be used for research and development, Lua Visual is dedicated to ease of use. OpenSim is merely a good reminder of how powerful a simulator can, but it is not something that can be hoped to accomplish in the scope of this IQP, and more importantly, it does not serve the purpose.

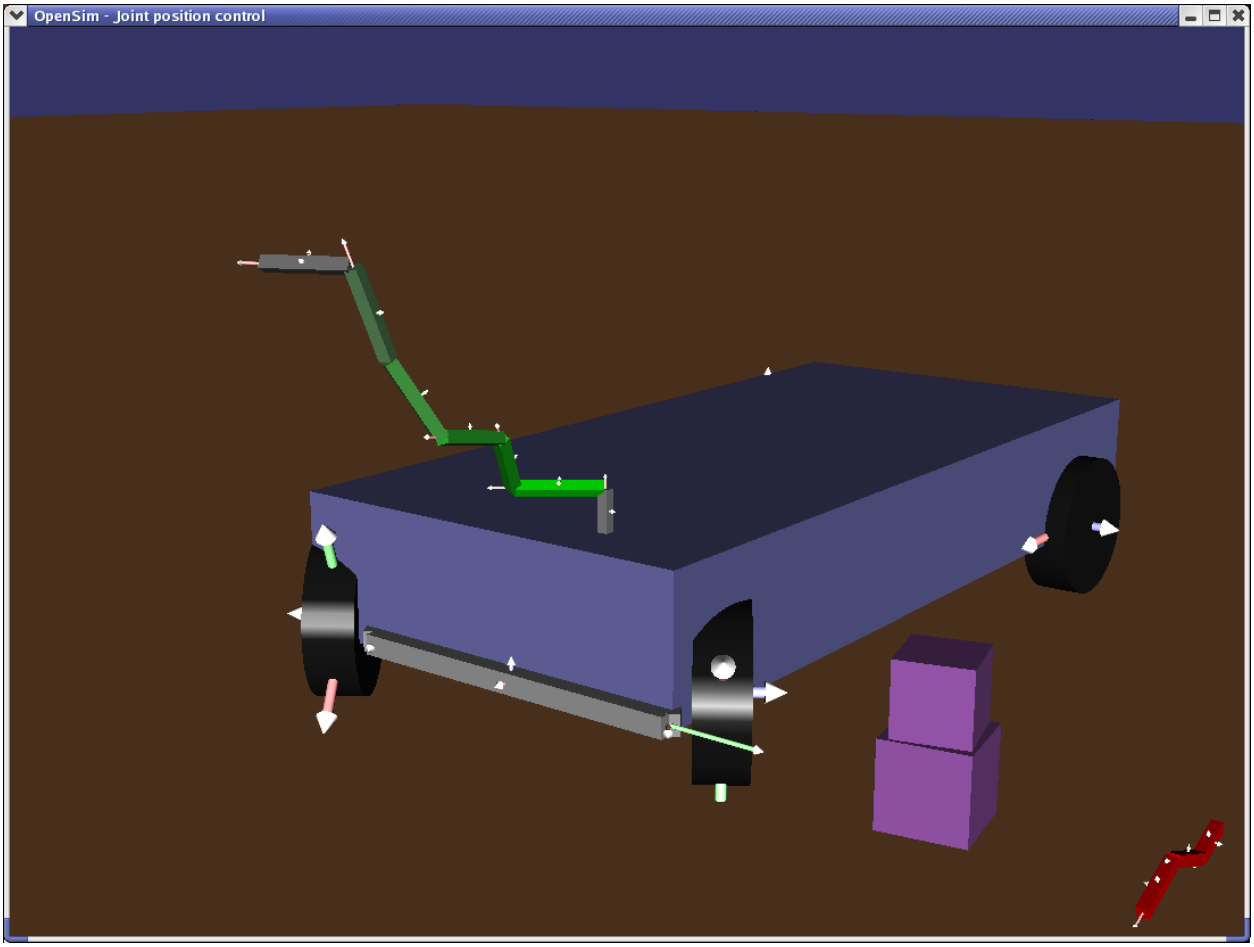


Figure 5: Wheeled robot inside of OpenSim[15]

1.5.5. SIMULATION COMPARISON

Figure 6 compares the features of a few currently available robotics simulation tools. The operating system column offers a look at which major operating systems can run the simulator and offers a look at how many computers can use it. The language column shows which programming languages can be used to program the logic of the simulated robots. The engine column shows whether the simulator uses a graphics engine for the 3D rendering. Finally, the supported robot platforms and sensors column provides a look at what kind of robots can be simulated.

SIMULATOR	OPERATING SYSTEM	LANGUAGE	ENGINE	SUPPORTED ROBOT PLATFORMS	SENSORS
PLAYER/STAGE	WINDOWS, MAC, LINUX	C++, JAVA, PYTHON	N.A.	WHEELED	RANGE, CAMERA, ENCODER, TOUCH
WEBOTS	WINDOWS, MAC, LINUX	C, C++, JAVA, PYTHON, URBI, AND MATLAB	OPENGL & OPEN DYNAMICS ENGINE	WHEELED, LEGGED, FLYING	RANGE, CAMERA, ENCODER, TOUCH
USARSIM	WINDOWS	TCP SOCKETS	UNREAL TOURNAMENT	WHEELED, UNDERWATER, LEGGED, HUMANIOD	RANGE, CAMERA, ENCODER, TOUCH
OPENSIM	WINDOWS, MAC, LINUX	N.A.	OPENGL	WHEELED	N.A.

Table 6 - Table of comparisons between various robotic simulators

1.5.6. NON SIMULATION ROBOTIC TOOLS

Tools for robotics education do exist, but unlike simulators, they do not focus on programming of robots. Lua Visual will not necessarily be a replacement to educational tools such as Vex[16] or Lego Mindstorm[17]. The difference between these platforms and Lua Visual is which focus of robotics are taught to the students. Lua Visual focuses on the programming concepts of robotics, such as interpreting sensor values. Vex and Mindstorm has a strong focus on the mechanical side and construction of robots, but a simulator removes the construction side of robotics entirely, allowing more focus on coding. In the Vex curriculum, EasyC is used, which makes programming simply dragging and dropping pre-made functions.[17] Mindstorm also uses a drag and drop environment but does allow Java programming as well.[17] The

[16] "Vex Educational Robotics Cirriculum." vexrobotics.com. <http://content.vexrobotics.com/docs/VEXCC-E-0908LR.pdf>

[17] http://www.ni.com/academic/mindstorms/works.htm

robotics kits emphasis is on building the robot, and the programming aspect is simplified to a drag-and-drop GUI using pre-made functions. There is no construction needed for the robot in the simulator, instead the aim is to teach students how to program a robot to perform tasks that are commonly assigned in both Vex and Mindstorm curricula, such as navigating a maze.[17] Both building and programming are doubtlessly important concepts for robotics education, but a benefit of Lua Visual over other educational tools will be the cost.

Both Vex and Mindstorm are costly, with Vex costing at least $700 for a kit and Mindstorm costing at least $200. Lua Visual will be freely distributed and free to use for everyone. Cost is without a doubt an issue for high schools across the country, and a free simulator would make it less expensive to teach robotics concepts. With an advanced enough simulator, that is still easy to use, schools that otherwise would not be able to provide a robotics education, could possibly have a course dedicated to the programming of robots.

1.5.7. PREVIOUS RESEARCH ON SIMULATORS IN EDUCATION

Simulations are used to create experiential environments within which learning can be done and analyzed.[18] Simulations were historically played as "management games" that were used to usually demonstrated tactics and strategies in warfare. It has been shown that using management games as education has been found to be generally effective. This research would suggest that a robotics simulator used for robotics education would also be effective.

Robotics has become an important part of university curriculum and research. A robotics course usually gives the practical exercises for effective learning which traditionally

[18] Bernard Keys and Joseph Wolfe, "The Role of Management Games and Simulations in Education and Research."Journal of Management (JofM), Vol. 16, No.2, 307 - 336.

requires equipment including robots and their controllers. However, it is not easily available to student in that it is either expensive or too large to fit in the classroom. A way around this is a solution called “e-learning”, which can be defined as the use of internet technologies to deliver a broad array of solutions that enhance knowledge and performance.[19] The website for Lua Visual employs some of the concepts presented in e-learning.

There are many e-learning advantages over traditional methods. The most obvious is the low cost. A good range of material can be accessed without the high cost of books and other such learning material. Another advantage is the simultaneous access to information. An application that allows users to communicate, such as a forum, would enable them to share knowledge. Also, while textbooks and other materials may go out of date, a website can be updated easily. All of these aspects could be used to improve robotics education.[19]

1.6. REPORT ORGANIZATION

The report is organized as follows:

- Chapter 2 covers the development of Lua Visual and the creation of the survey for high school and college students.
- Chapter 3 presents the results of the survey and analysis the data.
- Chapter 4 offers a conclusion to the project.

[19] Hamid, N.A.; Haron, H.; Jambak, M.I.; Sukimin, Z.; , "An Overview of Robotic Simulation E-learning," *Modelling & Simulation, 2009. AMS '09. Third Asia International Conference on* , vol., no., pp.566-571, 25-29 May 2009 doi: 10.1109/AMS.2009.59

2. METHODOLOGY

This section overviews the steps taken during the Interactive Qualifying Project to develop and evaluate Lua Visual as a robotics simulation for education. Specifically, this section goes into depth the decision process behind many of Lua Visual's features including:

1. Programming language
2. 3D Game Engine
3. GUI Library
4. Dynamic versus Static Libraries
5. Obstacle Detection

This section also provides a look at the decision process behind the survey of Lua Visual including rationale behind many of the questions presented in the survey.

2.1. LUA VISUAL OBJECTIVES

One of the main focuses of this Interactive Qualifying Project is to create and release a robot simulation which allows the user to view a virtual representation of a robot moving in a virtual environment. The objectives of creating Lua Visual is as follows:

- Allow students to program robots and further their education and interest in robotics engineering and computer science.
- Allow the user to program the movement behavior of a virtual robot through a maze towards a goal.
- In order to be an effective learning tool, the program must be simple to use and not require extensive previous knowledge in the areas of computer science or robotics.

One of the important design decisions for the robot simulation program is which language to use for the robot's logic. Possible languages included C++, Java, or Python which are the most widely used for this type of application. After some research, a language called Lua was

chosen due to its simplicity and efficiency.[20] Lua is a scripting language which means the code does not need to be compiled in order to execute.[21] This makes programming for the user much faster and easier since compiling code can be a frustrating experience if the computer environment is not setup correctly. Lua has automatic memory management which releases the burden of type casting variables and allows for a smaller learning curve for new users. The robot simulation will act as a Lua Integrated Development Environment which will allow the users to program Lua directly inside the simulation and execute their programs. The code sample below shows the simple syntax of Lua. It tells the robot to drive forward until the range finder sensor is less than five.

```
-- This is a comment in Lua
-- Below is a 'if' statement in Lua
if (getFrontRange() > 5) then
     DriveForward()
end
```

Code sample in Lua for driving the robot forward until a wall is detected.

In order to visualize the robot moving through the maze, the robot simulation generates a 3 dimensional world for the user which displays their robot navigating inside the maze. This 3 dimensional world is programmed in OpenGL which is an open standard for computer graphics on Windows, Mac OS, and Linux.

2.2. LUA VISUAL DEVELOPMENT

The development of Lua Visual required research into existing frameworks and different ways of creating GUI applications for users. Some of the key decisions were which game engine

[20] PUC-Rio. "Lua." N.p., 6 9 2010. Web. 15 Dec 2010. <http://www.lua.org/home.html>

and GUI framework to use for Lua Visual. These decisions would have a huge effect when trying to create the final product, so the final goals of the application had to be considered.

2.2.1. GAME ENGINE

Various game engines were looked at before the start of the development of Lua Visual. Figure 7 shows the game engines that were looked at during this research phase. The primary feature that is important was that the game engine was open source, so that it could distributed for free inside of Lua Visual. Other important aspects about which game engine was going to be used is the language it is programmed in. This varied from C++, Java or Python as shown in Figure 6. The benefit of choosing a game engine written in C++ is that Windows applications are natively programmed in C++. This makes combining the application with the simulation much easier. The primary decision to not using any of the commercial or free game engines was based on the fact that it would have been extremely difficult to embed the game engine inside of Lua Visual. Lua Visual needed to have an extensive GUI (Graphical User Interface) that most game engines could not support.

Game Engine	Type	Open Source	Language	Platforms	Commercial?	Website
Irrlicht	Free	Yes	C++	Windows, Linux, OSX	Yes (zlib)	irrlicht.sourceforge.net
Crystal Space	Free	Yes	C++	Windows, Linux, OSX	Yes (LGPL)	www.crystalspace3d.org
jMonkeyEngine	Free	Yes	Java	Windows, Linux, OSX	Yes (BSD)	www.jmonkeyengine.com
Panda3D	Free	Yes	Python/C++	Windows, Linux, OSX	Yes (BSD)	www.panda3d.org
Unity	Free/Comercial	No		Windows, Linux, OSX	Yes	unity3d.com
OpenSceneGraph	Free	Yes	C++	Windows, Linux, OSX	Yes (LGPL)	www.openscenegraph.org
Blender	Free	Yes	C++	Windows, Linux, OSX	GPL	www.blender.org

Table 7 - Table of existing 3D game engines

2.2.2. GUI LIBRARY

Since Lua Visual offers users the ability to code directly inside the application, this requires a fairly complex graphical user interface similar to existing Integrated Development Environments (IDE) like Eclipse. Features such as creating, editing, and saving files would have to be implemented. In order to do this, a cross-platform framework would have to be chosen. The two main frameworks are Qt[21] and wxWidgets[22]. Both of these frameworks offer the ability to create user interfaces across multiple platforms such as Windows and Mac OS X. Qt was chosen as the framework that would be used in Lua Visual due to its ease of setup and installation. Qt offers many great features including a module for OpenGL, which allowed the simulation to be placed directly inside the GUI. Figure 8 shows a screenshot of the graphical user interface for Lua Visual. The primary features include a menu bar, a workspace window, a text editor for editing Lua code, and a console for seeing errors in the code.

[21] Nokia. Qt. <http://qt.nokia.com/products/>
[22] wxWidgets. <http://www.wxwidgets.org/>

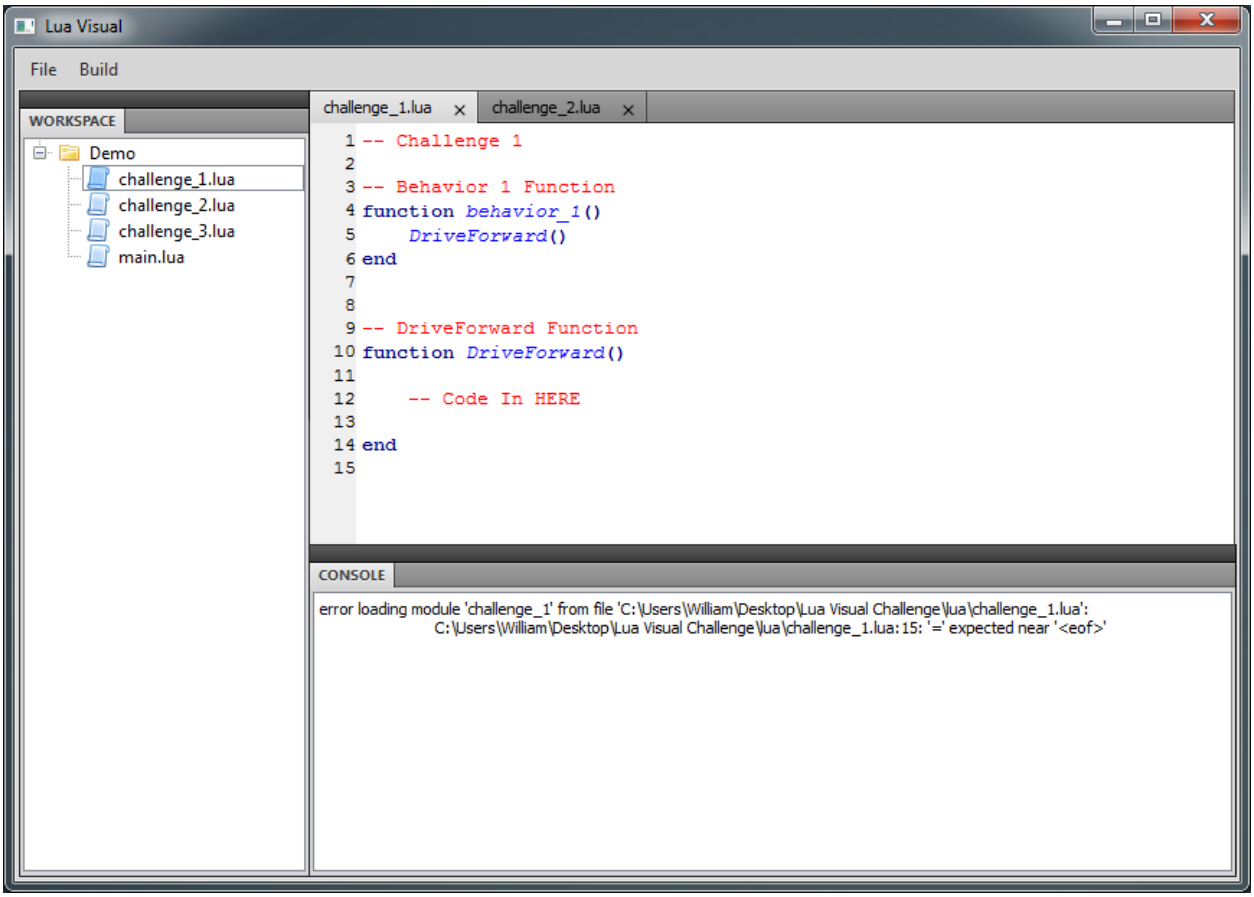


Figure 8 - Screenshot of the GUI programmed using Qt

2.2.3. DYNAMIC LINKED LIBRARIES VERSUS STATIC LINKED

During the development of the C++ executable program, one of the design decisions that needed to be made was whether the program would utilize DLL (Dynamic Linked Libraries) or static libraries. This is an important decision for any Windows application since this directly affects future development and support of the software. The final decision was to use DLL for Lua Visual due to the advantages provided.

Dynamic linked libraries offer the ability to split the executable into separate files ending in the '.dll' extension. This offers the advantage that future patches to the software would only

need to update the changed dynamic linked libraries versus the entire executable. It also offers the advantage of allowing multiple applications to access the same library, if this was a feature that was needed. The result is a smaller executable file but multiple dynamic linked libraries that are loaded during the startup of the program.

2.2.4. OBSTACLE DETECTION

A major hurtle that during the development of Lua Visual was not only the construction of objects but how to have the simulator register the collision of those objects. Objects are created by drawing boxes in the graphics engine between predetermined points. By handing in vectors of points we were able to construct mazes for the students to test their programming skills in.

After the ability to create mazes in Lua Visual was implemented, there needed to be a way to determine the distance between those objects. This is done by taking center points on the faces of the robot and then using a distance formula to determine the distance between that point and all other planes intersecting a line perpendicular to that point.

$$distance = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Then the smallest value of all the intersecting points was taken and this is the distance given from the robot to the nearest object.

As shown in Figure 9, the virtual robot is attempting to navigate the maze. It starts at the red square and must explore the maze until it reaches the green square. In order to do this, it will need to use its sensors to prevent it from running into walls.

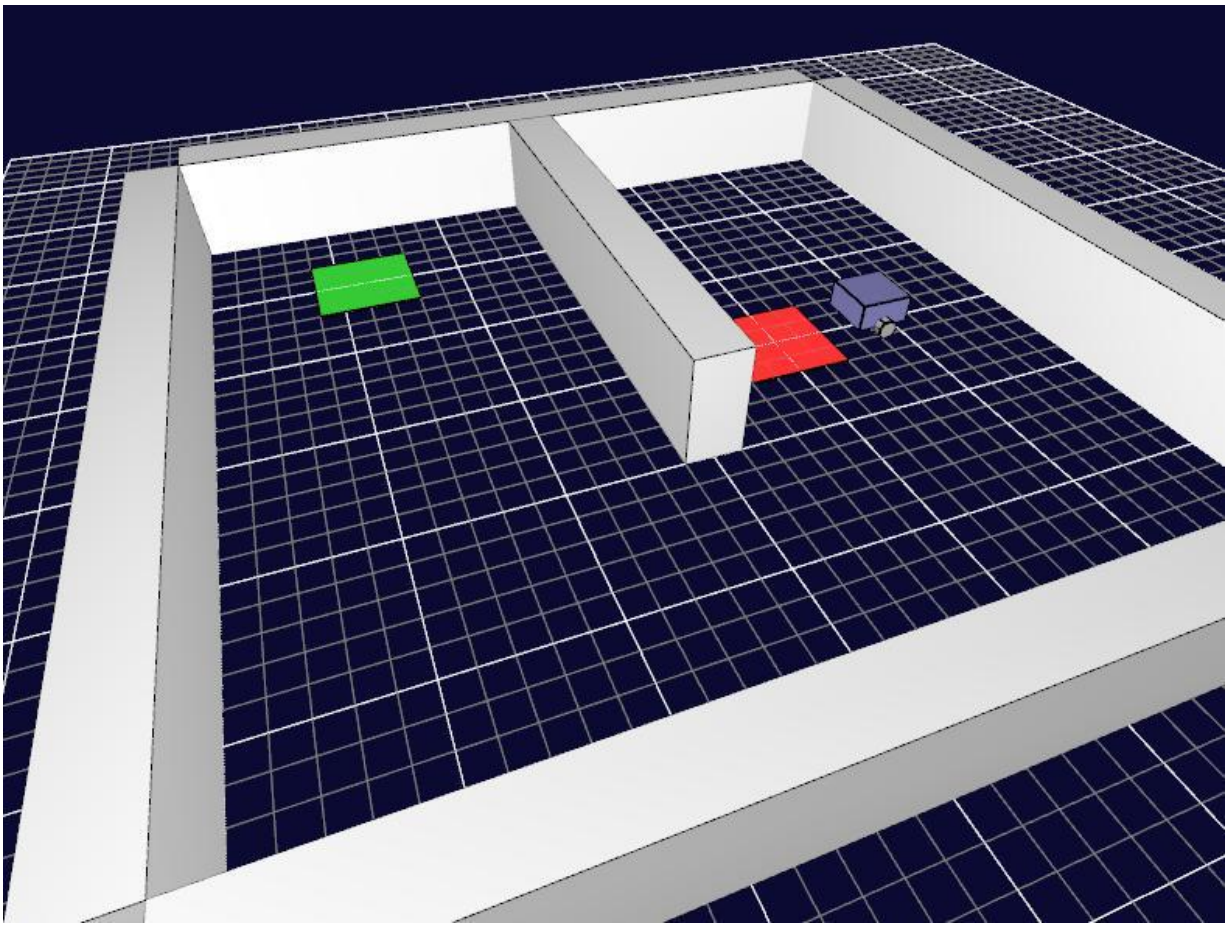


Figure 9: Virtual Robot inside of Lua Visual.

2.3. WEBSITE

In order to distribute the program and allow people around the world to use Lua Visual, a website was created for our program at 'www.LuaVisual.com'. The website provides the needed documentation for users to learn how to use our program, shown in Figure 10, and also allow users to download a copy of the program to their computer. The primary usage of the website will be to allow students and teachers to discover and download a copy of Lua Visual. It will serve a method of distributing Lua Visual and providing the needed documentation. A complete copy of the website can be seen in Appendix A.

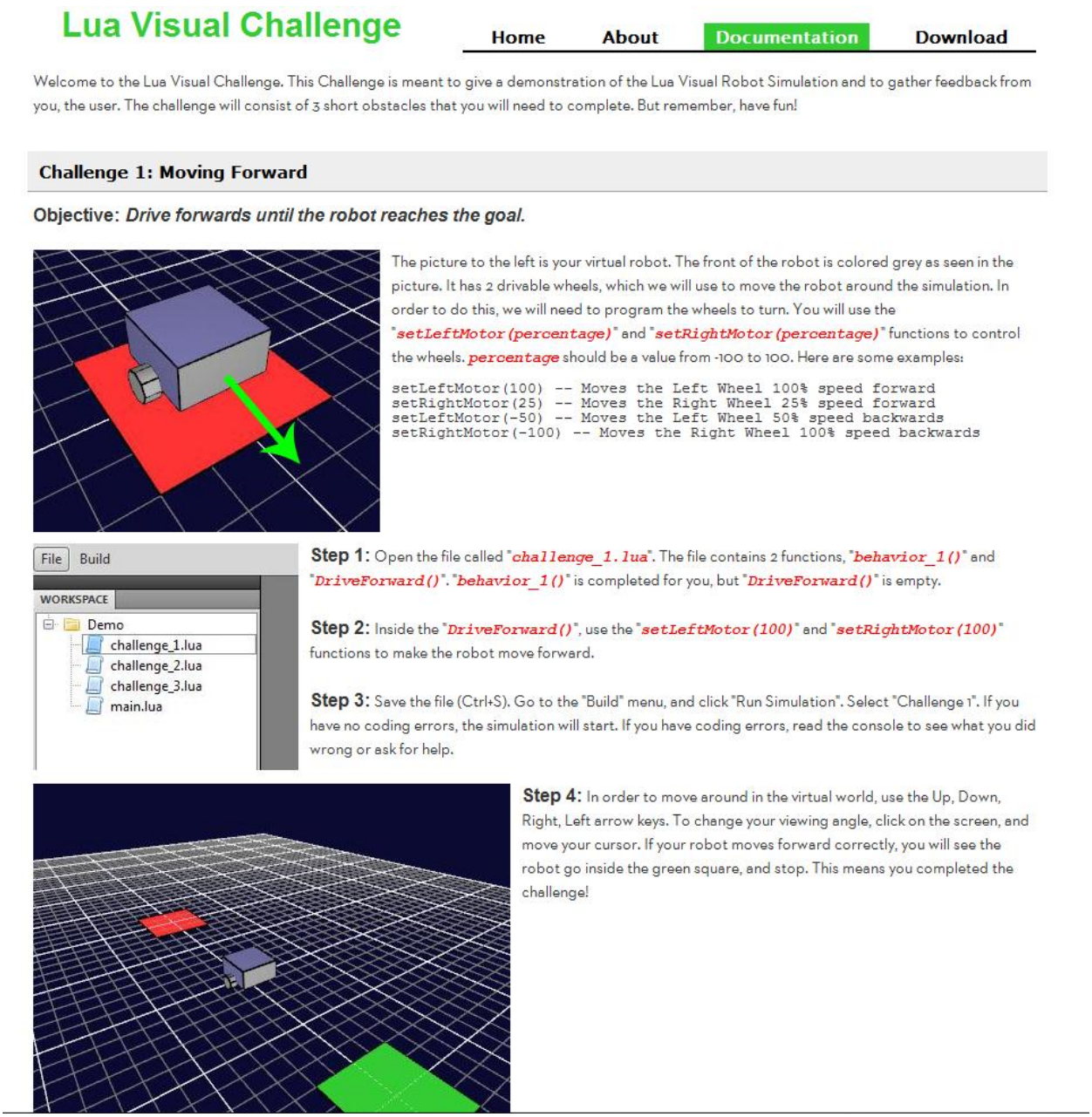


Figure 10 - Screenshot of the documentation page of Lua Visual website

2.4. SURVEY

To determine whether the developed robotic simulator called Lua Visual would be an effective educational tool, user testing was needed. To accomplish this high school and college students were asked to use the simulator and provide feedback on their experience using the

simulator. Tutorials and a library of reconstructed code were developed beforehand to aid those who have little programming knowledge. A survey was created to determine the effectiveness of Lua Visual on those who used it. The survey included the following questions:

1. What grade are you in?
2. What school or college do you attend?
3. On a scale from 1 to 5, rate your C/C++ programming knowledge.
4. On a scale from 1 to 5, rate your Java programming knowledge.
5. How many programming classes have you taken?
6. Have you taken a robotics class?
7. If any, what other programming languages besides C, C++, or Java do you know?
8. On a scale of 1 to 5, how easy was the software to use?
9. Were you able to complete the objective?
10. What did you find difficult about the software?
11. How useful do you think this software would be for learning robotics programming?
12. What did you like about the software?
13. What did you dislike about this software?

After we determined what questions we wanted to ask the students that participated in our survey we had to get it approved by WPI's Institutional Review Board (IRB), shown in Appendix D. Our questions were intended to gather background information on the participants and then learn their opinion of the software. The first six questions asks the students their history with programming and robotics, giving us the background information we need, and the last six questions gives us the students' reaction to Lua Visual.

2.4.1. SURVEY DESIGN

The “What grade are you in?” question was asked because it gives a useful piece of background information to make comparisons against. Most likely, the greater the grade level, the more experience a student would have with programming and robotics. How the students react to Lua Visual might be influenced by how experienced they are. It also made our results easier to organize into two different sets - high school student and college students.

The questions asking how much programming experience the students think they have has the similar purpose to the above question. It allows us to compare how experienced the student is against how they reacted to the software and see how programming knowledge influences their opinion. The survey specifically asks the students about their knowledge of JAVA and C because those are the most popular programming languages, so those were likely the languages the students knew the best. To see if they had experience with other languages, the survey asked, “If any, what other programming languages besides C, C++, or Java do you know?”

Likewise, the two questions, “How many programming classes have you taken?” and, “Have you taken a robotics class?” would provide details into how much background the participants had in programming.

The remaining questions were for collecting the student’s opinion of Lua Visual after their 10-20 minute usage of the simulator. Lua Visual was intended to be usable by even people with no programming experience, so the question, “On a scale of 1 to 5, how easy was the software to use?” was intended to determine if students that lacked programming experience could still figure out how to use the program. With this information, it would be

possible to compare programming experience against ease of use. To give the participants an outlet to explain any difficulties they had, the "What did you find difficult about the software?" was provided.

Lua Visual is intended as an educational tool. To see if students thought the program could be used as such, we asked, "How useful do you think this software would be for learning robotics programming [on a scale of 1 to 5]?" Asking this would make it possible to determine if students could seem themselves using this software in a classroom.

Finally, the last two questions asked for anything else the students either liked or disliked. This would allow us to determine what aspects students responded best to, and what aspects they did not respond well to. These open ended questions were intended to confirm our ideas about students finding simulators interesting, but also give an outlet for any unanticipated criticism. Any criticism could be used for determining what future development of the software was need.

2.4.2. TARGET DEMOGRAPHICS

This IQP focused on two different control groups for user testing and completing the survey. The first group is entry level college students. The software was distributed and used then feedback will be received from the group discussing what they feel worked or didn't work regarding the functionality of the software. Also surveys analyzing the understanding of programming will be collected from the group over time to see what kind of improvements have been made.

The second group that this IQP focused on is high school level students that may have little to no prior understanding of programming and its essentials. Because of this fact we

needed to work with and ask questions to these students on how to improve the software to fit their needs. These questions are be geared towards finding out what level of understanding if any the test group has.

2.4.3. SURVEY PROCEDURE

In order to get a good understanding of the usefulness of our project we needed a way to gather information on its effectiveness on the target audience. There were a specific set of questions that we really wanted to get answered in order to discover whether Lua Visual meet its goals of being easy to use and a good educational tool.

The testing for our IQP took place on the WPI campus. We were given permission to conduct our testing in Harrington auditorium on March 10th during the FIRST robotics competition. This location was picked because we believed that it would have the best chances of hitting the demographic that we created the program for. This competition hosted dozens of high school teams from around New England which gave us a wide range of students to gather feedback from.

The users were asked to read the documentation, seen in Appendix A, and to follow the instructions provided to use Lua Visual. The participant would on average take 10 to 20 minutes to complete the instructions and would then be asked to complete the survey. If the participant had trouble understand a particular step, one of the test administrators would offer assistance.

2.5. TIMELINE

Construction and testing of the Software is scheduled to go as follows.

October 26 – December 16 : Create a project proposal for Lua Visual and define long term goals.

January 13th – February 7th : Start construction of Lua visual and define problem areas that will need more time to be specifically tackled.

January 31st – February 14th : Worked on the modeling of the robot and being able to control the models movements around the virtual world.

February10th - February 17th: Constructed the range finding ability so that we could determine the distance between objects on the virtual world.

February 15th - February 21st : Created the different environments for each of the challenges and tested them to make sure that all of the components were working together correctly.

February 22nd - March 14th : Open LuaVisual.com to the public and make all pages functional on the site.

February 28th - March 7th : Created survey for use in determining the previous knowledge and the usefulness of Lua Visual as a learning tool.

March 10th : Tested Lua Visual at WPI F.I.R.S.T. Robotics competition. After each student was done with testing the software we had them fill our out survey.

April 14th : Ran a second test on Lua Visual with Robotics students.

March 29th – May 2nd : Compile all research and data together and start work on the final revision of the project paper.

3. RESULTS

3.1. THE LUA VISUAL CHALLENGE

We had the opportunity to test how students reacted to Lua Visual at the 2011 FRC WPI Regional, and later tested it on WPI students. Students were given instructions through the Lua Visual website, with extra documentation explaining basic programming concepts they would need to follow the instructions. The objectives of the challenge were to program the robot to:

1. Drive forwards until the robot reaches the goal.

2. Drive forwards until near the wall. Then drive backwards until the robot reaches the goal.

3. Drive forwards until near the wall. Then turn the robot 90 degrees to the left, and repeat until the robot reaches the goal.

These objectives were picked because they are very straightforward, allowing the students to complete the challenge within 30 minutes. More detail into these objectives can be seen in Appendix A.

After completing the programming challenges given, the participants were given a survey that asked questions on their programming experience and their opinions on the software they had just used.

3.2. FRC WPI REGIONAL COMPETITION SURVEY

Figure 11: High school students participating in the survey

For the survey at the FIRST Robotics Competition WPI Regional, there were a total of 25 students who participated in the survey. The participates were a fairly even distribution of different grade levels of high school students, with 6 Freshman, 6 Sophomores, 4 Juniors, and 7 Seniors, and also with 2 college level students as shown in Figure 12.

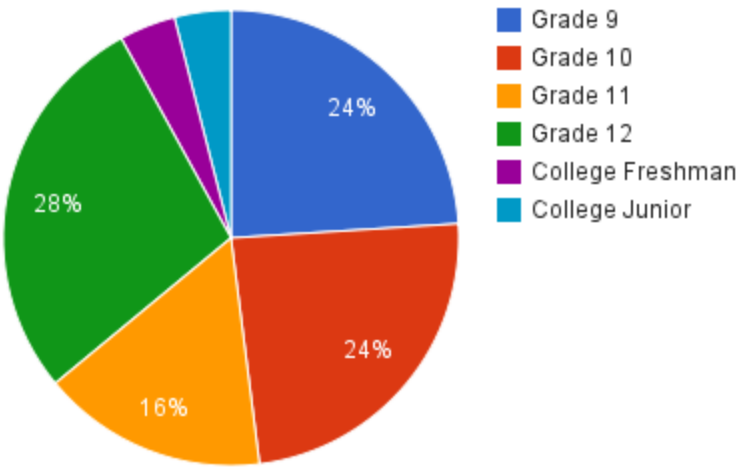


Figure 12: Percentage of Students in each grade

3.2.1. STUDENT BACKGROUND

Even though our test group was so evenly distributed, it appears majority of participants had little knowledge of programming, or at the very least, were not very confident in their programming skills. On the question asking them what their experience in programming (Java) was on a scale from 1(no experience) to 5 (expert), the average of each grade level was 1.83 at the 9th grade level, 2 at the 10th grade level, 1.75 at the 11th grade level, and 1.57 at the 12th grade level. There were no students that identified themselves as an expert (5), but there were 3 students that identified themselves as a 4 on the scale in C programming, and 2 different students that said they were a 4 in Java. Figure 13 presents the result from the Java programming experience question as the majority of students answered 1 to the C/C++

question, meaning that this result is a better representation of their programming experience. Another piece of information that we gathered was that, despite this being a robotics competition, only 10 of the 25 students had ever taken a robotics course. Most of the participating students were simply part of a club or similar group, which is not particularly surprising as FRC is an extracurricular activity.

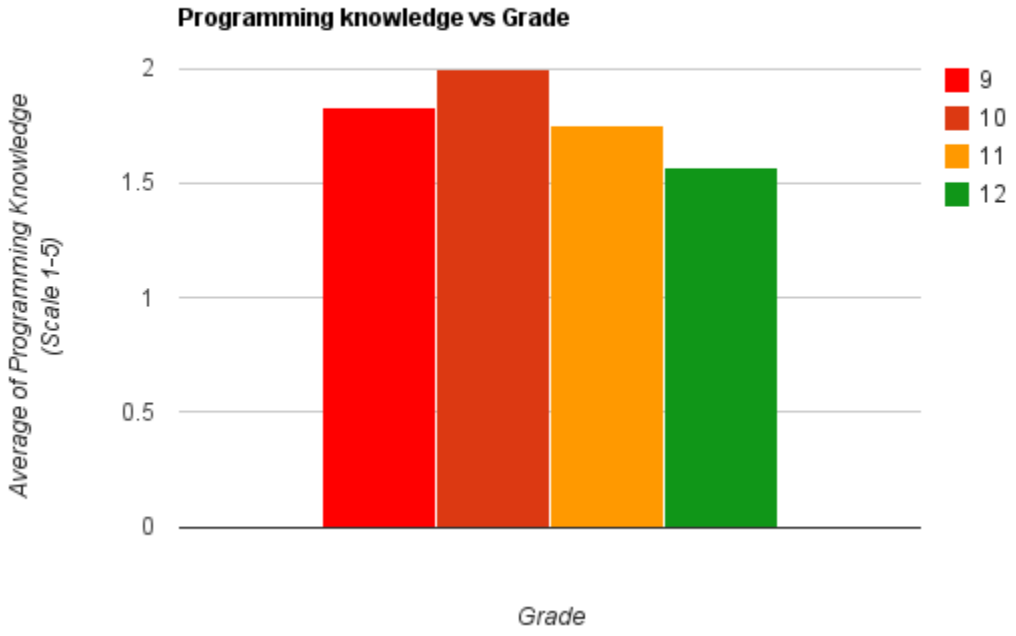


Figure 13: Students across all grade levels rate their programming knowledge fairly low

It was clear that this group of students, despite working with robots, had very little knowledge of how to program. They adequately represented the intended audience of Lua Visual - high school students interested in robotics but with little programming experience.

3.2.2. STUDENT OPINION OF LUA VISUAL

The next question now was how effective the software was for teaching. Though difficult to determine this by having students only use Lua Visual for about 20 minutes each, the survey asked them their general opinions based on the short time they had. The group was also polled both on how easy to use and how useful they believed the program was, once again using a scale of 1(not easy or not useful) to 5 (very easy or very useful). The results show that regardless of programming experience, most students thought that the program was both easy to use and useful, as the average of both of these polls was around a 4 at all programming levels. Number of programming classes also did not notably influence how easy students found the software to use. Also, out of the 25 students that participated in the challenge, 23 were able to complete the challenge. The two who were unable to finish only quit because of lack of time.

For the question, "What did you find difficult about the software," 14 of the 25 students responded that there was nothing that they had trouble with. Examples of difficulties were confusion with what was actually required to complete the programming requirements for the challenge. There were also criticism of not being able to easily move the camera angle around and zoom in and out when the simulation was running.

Lastly, when asked what aspects of the tool students liked and disliked, most specifically noted that it was easy to use and that they liked the graphical interface. Still others brought up topics such as the models that were used in the simulation. The challenges being made to easy to just copy and paste and finish. But for all of the problems people saw with Lua Visual we

received a lot more praise when we asked the participants what they liked about our software. Most said that it was easy to use and the virtual simulation makes use a lot more interesting. others said that they enjoyed the simplicity that came with using Lua since they had little previous programming experience. The results pointed to Lua Visual as being a positive and effective tool for learning. With the majority of the participants we had test at the FIRST competition were in our target demographic and reported having a positive experience we determined that Lua visual was a possible and successful tool. The specific comments from the students can be seen in appendix C.

Figure 14 shows that regardless of how students rated their own programming experience, on average, they felt the simulator was easy to use. Figure 15 shows that the number of programming classes a student had taken did not affect ease of use. Likewise, Figure 16 shows no significant difference between those who had taken a robotics course and those who did not. Figures 17 and 18 present that programming experience and number of programming courses taken had no influence on how useful they thought the program could be.

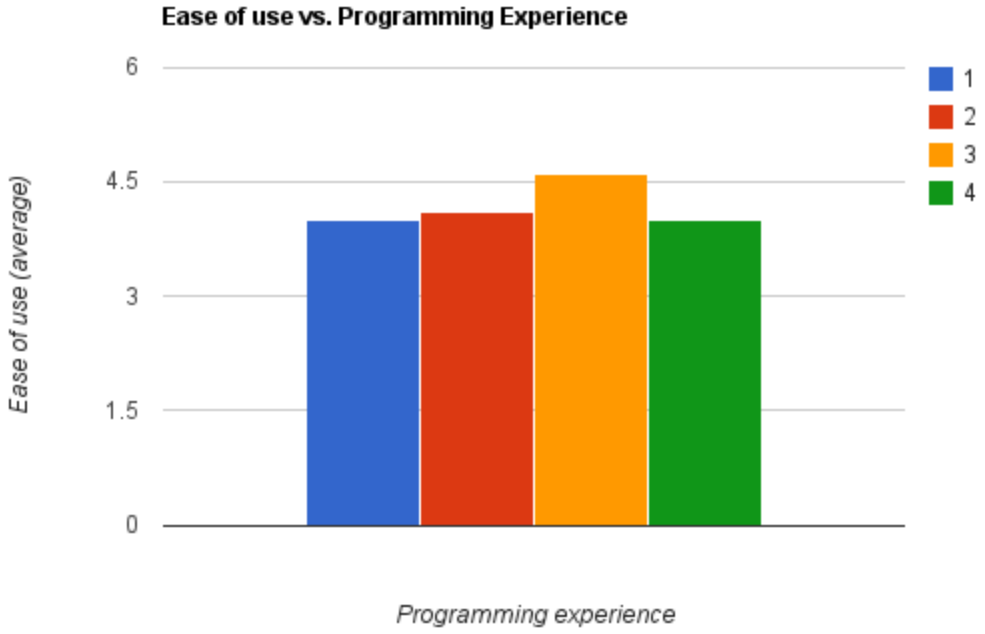


Figure 14: Regardless of programming experience, all participants said the program was easy to use

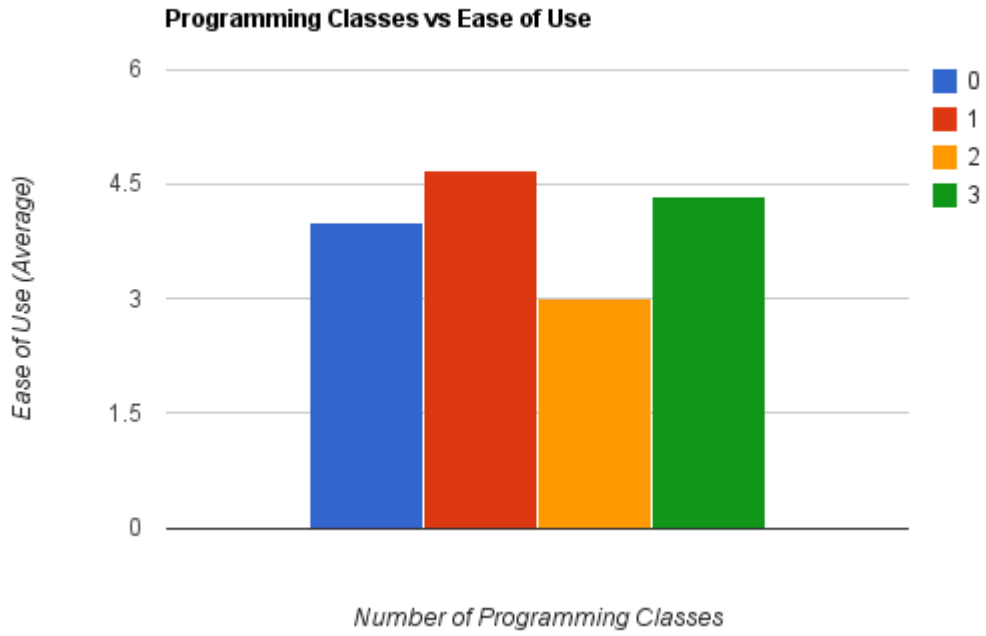


Figure 15: Even those who took few programming classes found the software easy to use

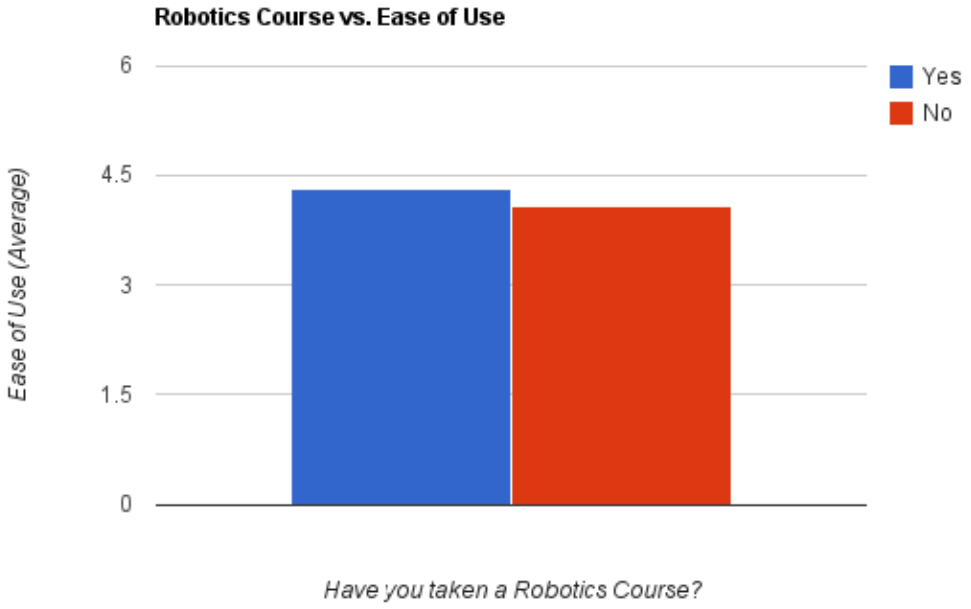


Figure 16: Taking or not taking a robotics course did not have a significant effect on accessibility

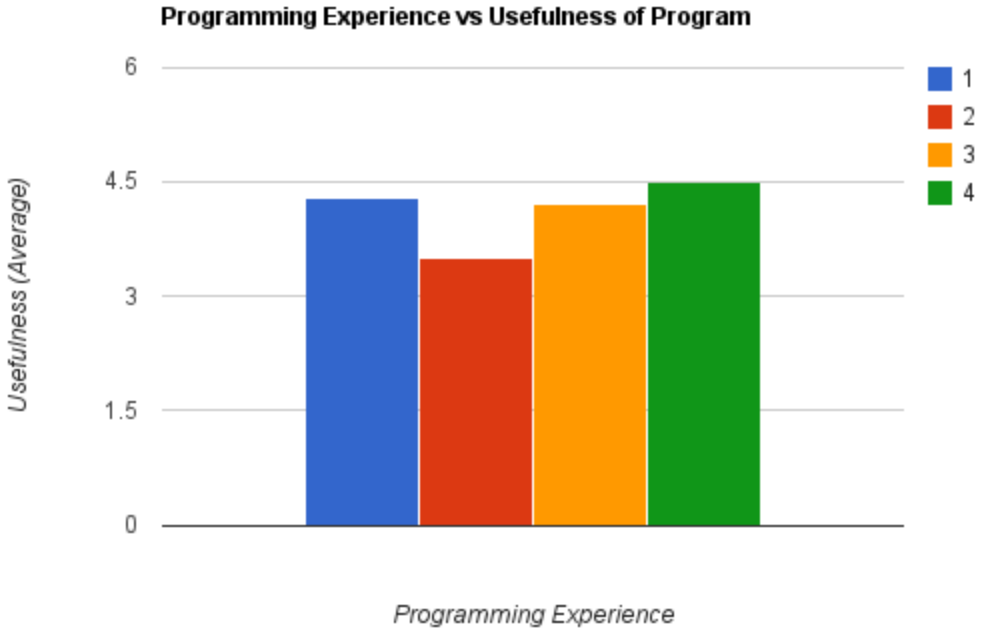


Figure 17: How much programming experience a student had did not heavily influence usefulness

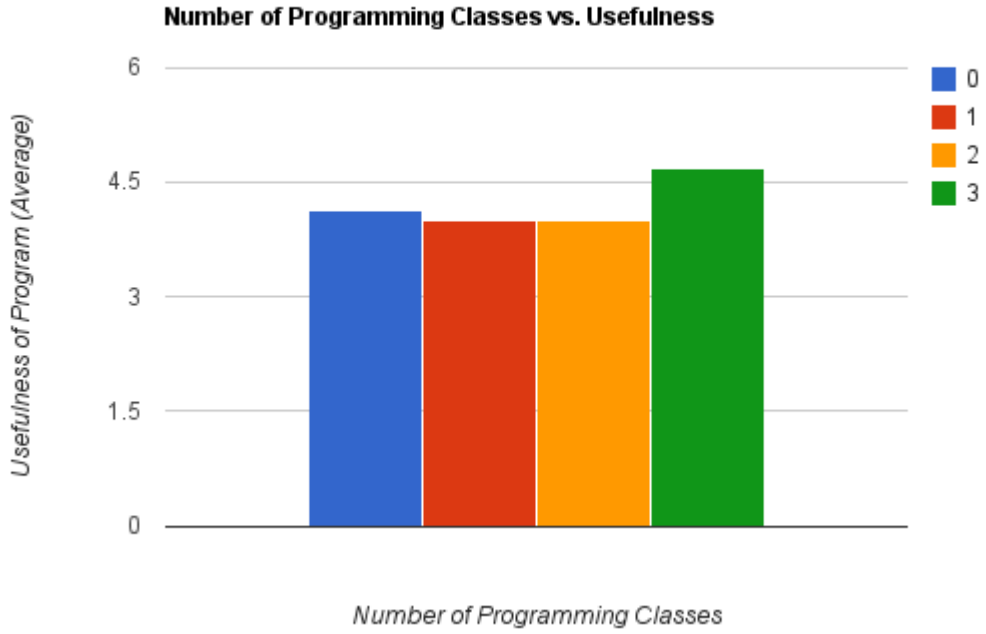


Figure 18: Number of Programming Classes did not greatly affect perception of usefulness

Overall, the participating students at the FIRST competition that helped in the testing of our software said that, regardless of their experience with programming, they all found the software easy to use. Also, the majority of the participants said that the user interface for the software looked nice and that the visual simulation was a useful feature.

3.3. WPI STUDENTS TEST

On April 14, we tested Lua Visual on RBE majors at WPI. A total of 10 students participated in this study. Our participants were volunteers from the robotics lab. Combined with the 2 students surveyed at the FIRST competition, we analyzed the responses of all 12 students.

7 of the 12 students were freshmen, the intended class level of the survey, but 1 sophomore, 3 juniors, and 1 senior were also included as shown in Figure 19.

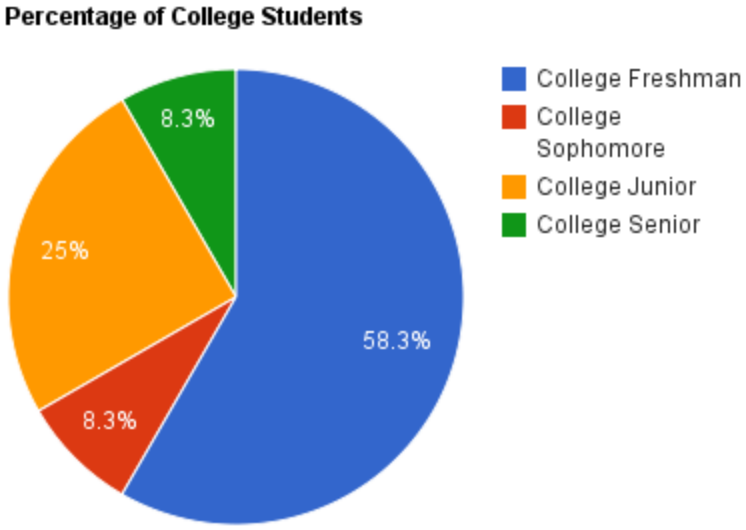


Figure 19: The distribution of college level participants

3.3.1. WPI STUDENT BACKGROUND

College students, as predicted, ranked themselves much higher in terms of programming experience. Across all grade levels, the average of what students ranked themselves in C proficiency was exactly a 3, on a scale of 1 to 5. The average of just the freshman was 2.57. The ranks for JAVA were slightly lower - a 1.92 average for all grade levels

and 1.43 for freshman as seen in Figure 20.

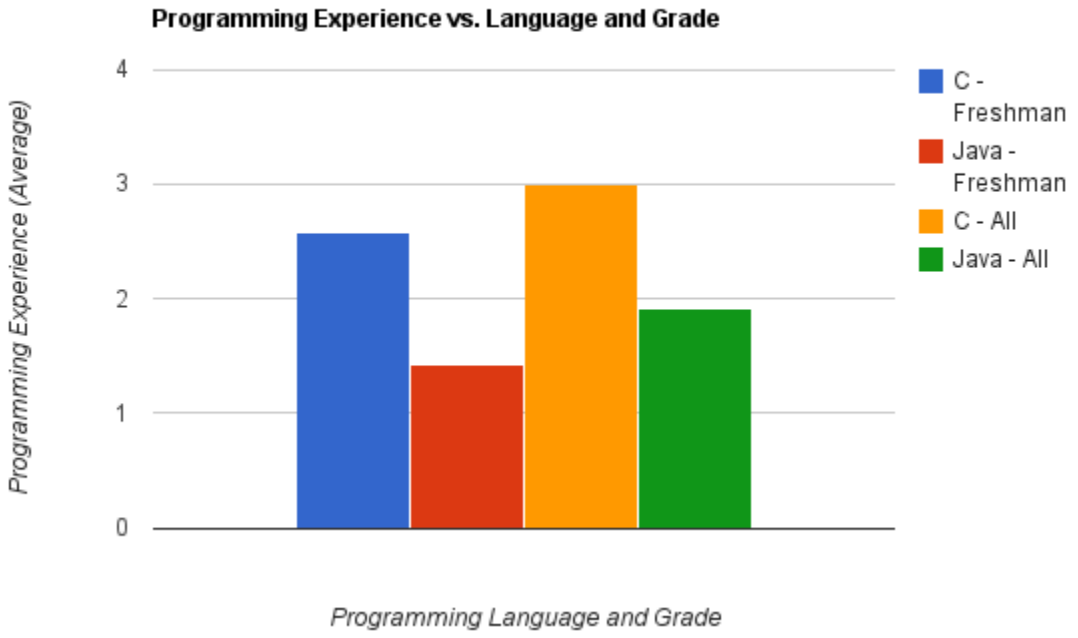


Figure 20: Freshman had significantly less experience than all participants combined

3.3.2. WPI STUDENT OPINIONS OF LUA VISUAL

Despite the increased experience in programming, most students still agreed that the program was both easy to use and potentially useful. Ease of use averaged at a 4.67 (freshman averaged slight lower at 4.57) and usefulness at 4.25 (slightly higher for freshman at 4.43) as seen in Figure 21 and 22. For specific aspects, most students noted that they liked the Lua's syntax and the simplicity of the program. The only complaints tied directly to the program were some of the more experienced participants noted how limited the Lua Visual was in terms of what kinds of robots could be built, and the lack of any physics engine. Every other

complaint was directed at minor issues, such as camera control.

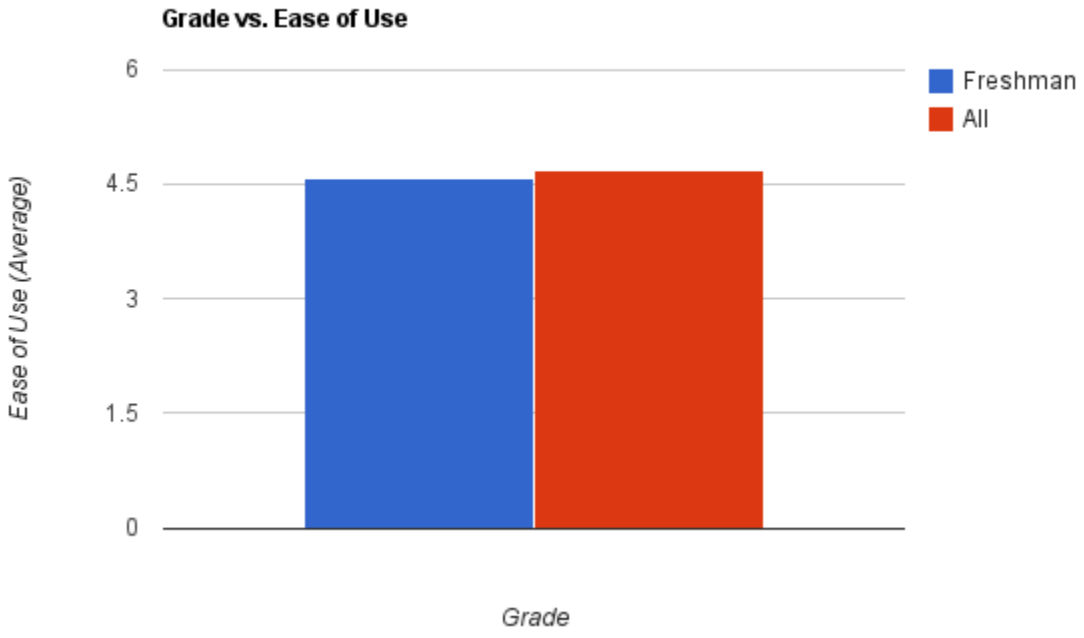


Figure 21: Both groups in the WPI test thought the program was easy to use

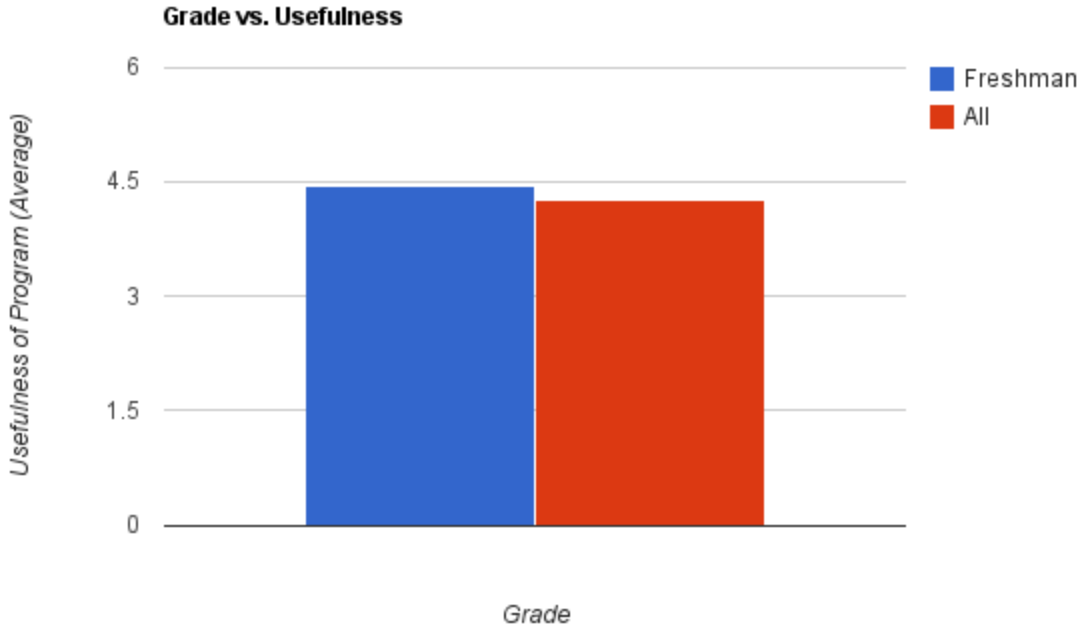


Figure 22: Students in the WPI test thought Lua Visual could be useful for teaching programming

During this second testing session we also asked the participants what they thought of our software and if they had any suggestions on how to improve it. Having some of the students that participated be juniors and seniors in college we received a lot heavier criticism and suggestions for our software. There were only a few difficulties that were stated in the WPI test group. These difficulties included the starting position and movement of the camera when running the simulation. Other areas people had problems were syntax when moving from other programing languages to Lua Visual and getting confused while reading through the readme.

The participants of this test were also asked if they had any likes or dislikes about Lua Visual. Dislikes included challenges not presenting enough of a challenge for some students. Others brought up the topics of the colors chosen for comments as the bright red kept getting confused for error messages. It was also stated that the simulator felt like it was to limited in

what it was capable of letting users simulate in its current form. The list of likes amongst the WPI testing group was quite broad and touched on all parts of the simulator and the website. Participants said that the syntax was very easy to use and that running the simulations were also easy. Some students stated that they liked the instant feedback that they received from the simulation over using an actual robot in which it would take longer to see if the program is written and executed correctly. Although not perfectly polished, Lua Visual was welcomed by both high school and university students. At the very least, the concept of a simulator is something that students reacted positively to.

3.4. SIMILARITIES AND DIFFERENCES BETWEEN TEST GROUPS

As predicted, college students had slightly more programming experience than the high school students, even with most of the college students being freshman. The JAVA programming experience of high school students averaged at 1.8, while the college students averaged at 1.92. The difference in C/C++ programming experience was much greater, with high school students only averaging at 1.76 compared to the college student's 3. Similarly, college students took far more programming classes, with high school students only taking 0.81 programming classes on average and college students at 1.83.

Despite difference in skill, both groups thought the program was very easy to use - high school participants averaged at 4.13 and college students averaged 4.67. This is also reflected in that all but 2 of the high school students could complete the challenge and every single college student could do so as well. In addition to being easy to use, both groups of students

agreed that Lua Visual could possibly be useful for learning robotics programming (4.25 average for college students and 4.09 for high school students).

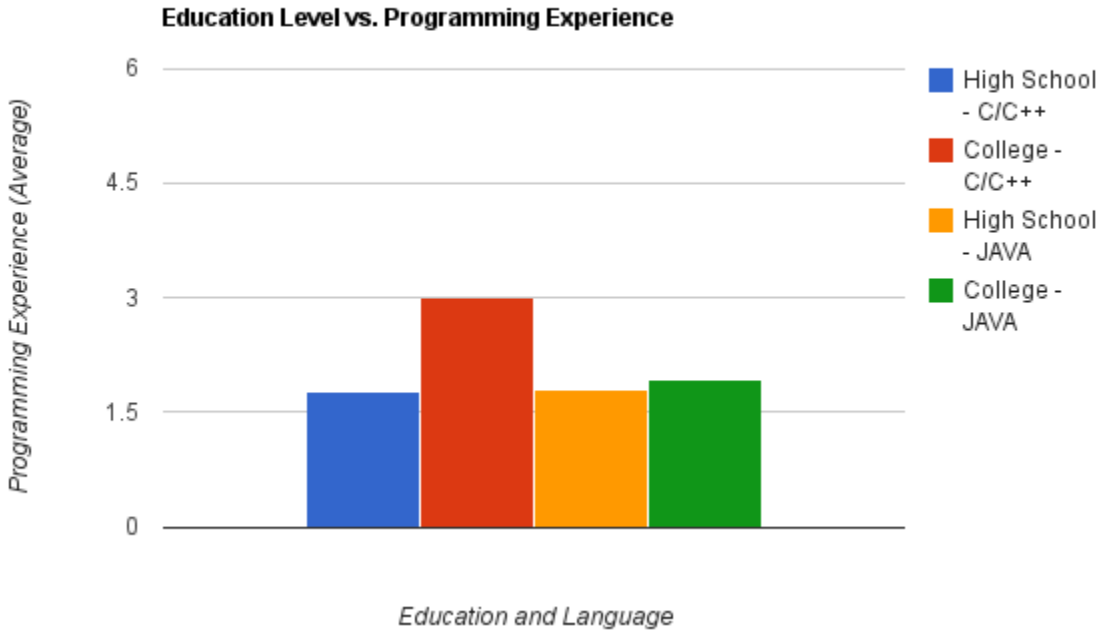


Figure 23: College students were much more confident in their programming skills

Figure 23 is a representation of the data we collected showing the difference between general programming experience and the current education that the participant was undergoing. In general most students, regardless of education, had little experience with JAVA but many college level students that volunteered had some kind of experience with C/C++.

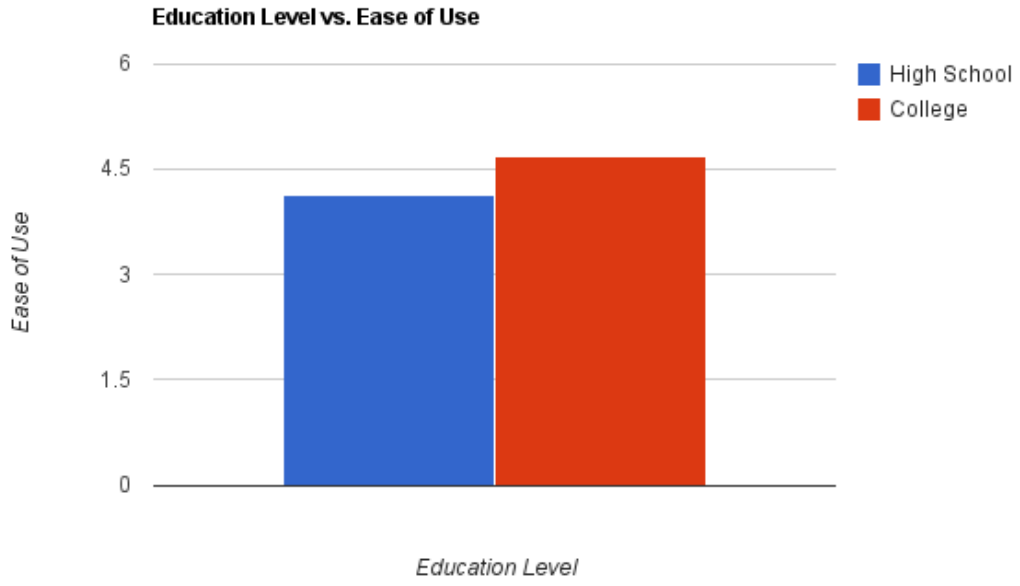


Figure 24: Both test groups agreed the program was easy to use

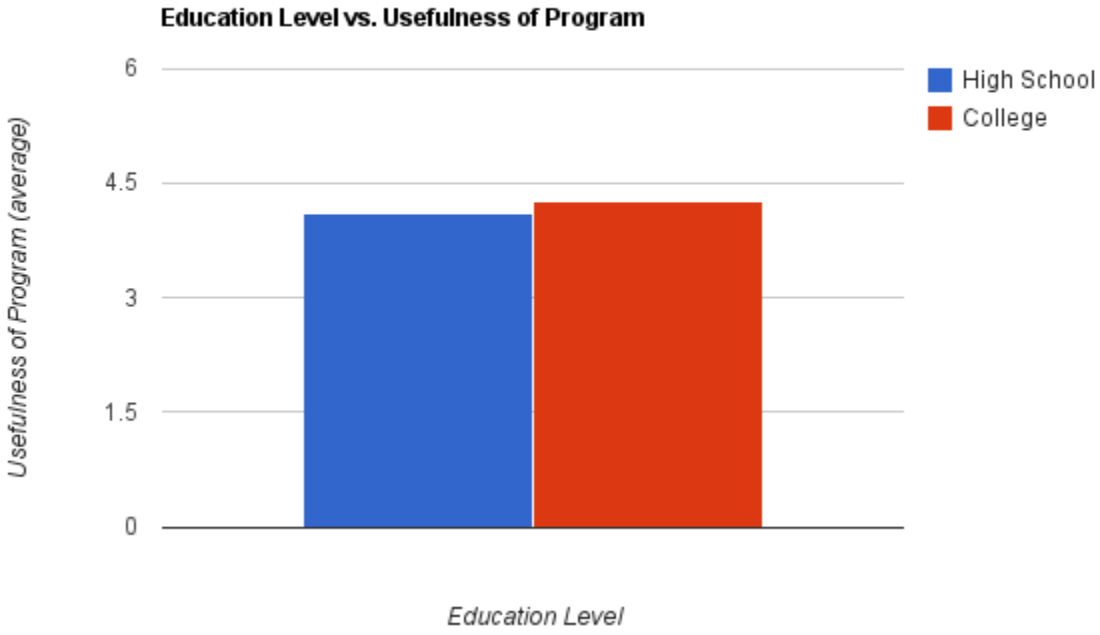


Figure 25: Students in both groups thought the program was useful

4. CONCLUSION

Our research provided us with the information that other robotic simulators do exist. Although none of the simulators that we were able to find were actually aimed towards the education of students, most were created with the purpose of testing equipment before money was spent or materials were chosen. Lua Visual seeks to provide a platform that is more of an educational tool than a simulation for a business environment. With similar assignments as Vex and Mindstorm, Lua Visual emphasizes more on the user's ability to understand programming structure and concepts. Specifically Lua Visual works with support staff to promote better understanding of good programing methods to complete a predetermined task.

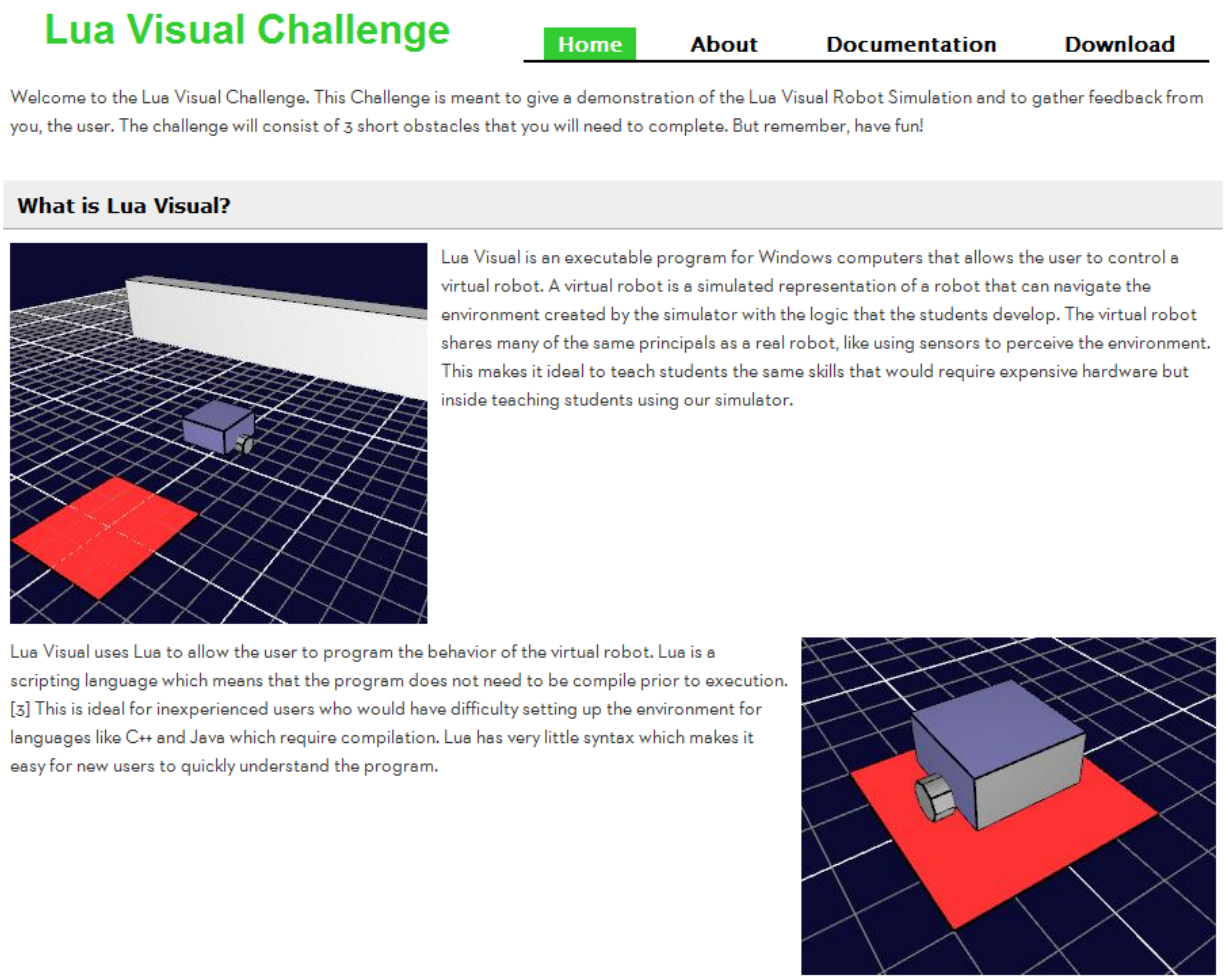
Lua Visual can be accessed using our website, which provides the needed documentation for users to learn about the program and also allows users to download a copy of the simulator to their computer. With this copy users will have access to all of the challenges and resources that the students that participated in our surveys would have had. The ReadMe file that was used to instruct users on how to use the software and complete the challenges can also be found on our website.

Gathering feedback was an important part in this IQP. Since Lua Visual is a brand new piece of software it is important to determine its usefulness and appeal to those who will be using it. The survey was conducted on a wide range of students with different backgrounds in programming. The survey revealed that despite having little or no experience with programming that the users participating in the survey, were still able to easily understand and control the virtual robots. Almost all of the participants were able to complete the challenge given and many of the students were interested in learning more about the simulator.

Using Lua Visual as the testing instrument, this IQP aimed to test the effectiveness of simulators and see if there was a possible positive effect on robotic education. Lua Visual was designed and tested to teach students who had limited prior programming experience on how to program robots. Through student feedback and research, our IQP determined that programs like Lua Visual can benefit education and spark interest in the field of robotics.

5. APPENDIX A: WEBSITE PAGES

Home Page

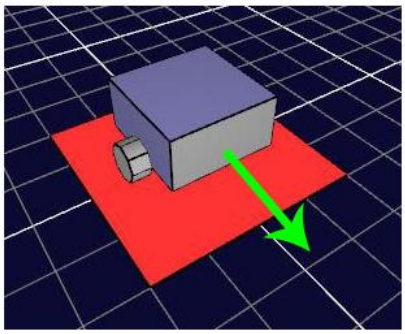


Lua Visual Challenge

Welcome to the Lua Visual Challenge. This Challenge is meant to give a demonstration of the Lua Visual Robot Simulation and to gather feedback from you, the user. The challenge will consist of 3 short obstacles that you will need to complete. But remember, have fun!

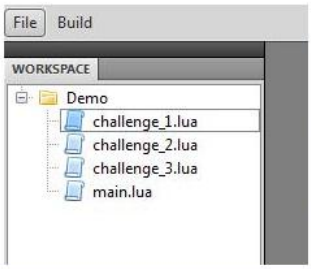
Challenge 1: Moving Forward

Objective: ***Drive forwards until the robot reaches the goal.***



The picture to the left is your virtual robot. The front of the robot is colored grey as seen in the picture. It has 2 drivable wheels, which we will use to move the robot around the simulation. In order to do this, we will need to program the wheels to turn. You will use the "`setLeftMotor(percentage)`" and "`setRightMotor(percentage)`" functions to control the wheels. `percentage` should be a value from -100 to 100. Here are some examples:

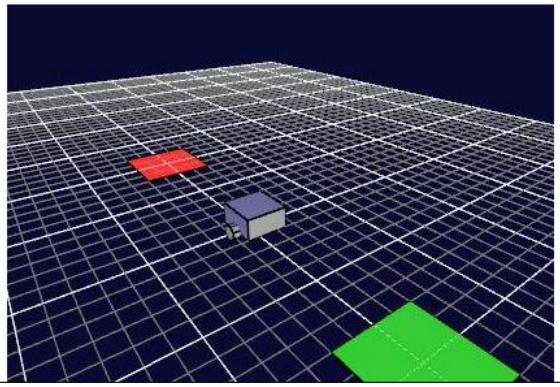
```
setLeftMotor(100)  -- Moves the Left Wheel 100% speed forward
setRightMotor(25)  -- Moves the Right Wheel 25% speed forward
setLeftMotor(-50)  -- Moves the Left Wheel 50% speed backwards
setRightMotor(-100)  -- Moves the Right Wheel 100% speed backwards
```



Step 1: Open the file called "`challenge_1.lua`". The file contains 2 functions, "`behavior_1()`" and "`DriveForward()`". "`behavior_1()`" is completed for you, but "`DriveForward()`" is empty.

Step 2: Inside the "`DriveForward()`", use the "`setLeftMotor(100)`" and "`setRightMotor(100)`" functions to make the robot move forward.

Step 3: Save the file (Ctrl+S). Go to the "Build" menu, and click "Run Simulation". Select "Challenge 1". If you have no coding errors, the simulation will start. If you have coding errors, read the console to see what you did wrong or ask for help.



Step 4: In order to move around in the virtual world, use the Up, Down, Right, Left arrow keys. To change your viewing angle, click on the screen, and move your cursor. If your robot moves forward correctly, you will see the robot go inside the green square, and stop. This means you completed the challenge!

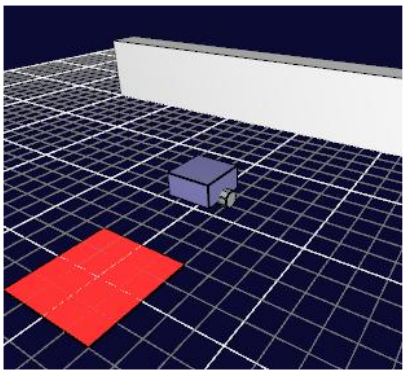
Documentation Page 2

Reality Check

Did you notice anything strange about the robot's movement? Newton's first law of motion states that a object in motion wants to stay in motion. This is the basic concept of Inertia. Our simulator ignores this law, since our virtual robot is able to instantly change speed. If this was a real robot, you would see the robot slowly increase in speed before it reached its max speed. Remember this when programming real robots!

Challenge 2: Detect Walls

Objective: ***Drive forwards until near the wall. Then drive backwards until the robot reaches the goal.***



Now its time to sense our environment. This virtual robot has a range finding sensor on the front which will allow you to program the robot to avoid crashing into walls. To use the range finding sensor, use the "*getFrontRange()*" function. This will return the distance to the closest wall directly in front of your robot. Here is a example:

```
-- Drive Forward until a wall is less than 5 units away.

if (getFrontRange() > 5) then
        DriveForward()
end
```

Step 1: Open the file called "*challenge_2.lua*". The current code will tell the robot to drive up to a wall and stop when within 5 units. But we want to robot to move backwards after it detects the wall.

Step 2: Inside "*behavior_1()*", call the "*ChangeBehavior(2)*" function inside the "*else*" statement. This will switch the behavior to the "*behavior_2()*" function when the robot gets near the wall.

Step 3: Save the file and run the simulation.

Reality Check

In reality, sensors are not prefect. All sensors have a margin of error, and most sensors can return completely wrong values. In order to use sensors on real robots, we must filter that sensor data and remove any data that might be incorrect. Thankfully, in this virtual world, the sensor values are always accurate.

Documentation Page 3

Challenge 3: Turning

Objective: ***Drive forwards until near the wall. Then turn the robot 90 degrees to the left, and repeat until the robot reaches the goal.***

So far we have only moved the robot forward and backwards. Now its time to turn the robot. There are a few functions you must use to complete a turn. "`setDesiredAngle(angle)`" should be used to set the amount you want to turn. "`angle`" should be a positive number for turning left or negative number for turning right. For example:

```
setDesiredAngle(90) -- Turns the robot 90 degrees left
setDesiredAngle(180) -- Turns the robot 180 degrees left
setDesiredAngle(-360) -- Turns the robot 360 degrees right
```

Step 1: Open the file called "`challenge_3.lua`".

Step 2: Inside "`behavior_1()`", modify the "`setDesiredAngle(0)`" to turn the robot 90 degrees to the left.

Step 3: Inside "`behavior_2()`", add "`TurnLeft()`" inside the "`if`" statement. This will cause the robot to turn left until the angle you set is reached.

Step 4: Save the file and run the simulation. If done properly, the robot will keep driving, and not crash into any walls.

Reality Check

How do you think you would program the `getAngle()` *function?* On a real robot, it is difficult to determine the current angle of your robot without additional sensors. Sensors such as wheel encoders can be very useful for calculating the angle your robot has turned. More advanced sensors such as gyroscopes or acceleratometers can also be used.

6. APPENDIX B: SURVEY PICTURES


FIRST
WELCOME
WPI
Team Work.
Worcester Polytechnic Institute


WPI

7. APPENDIX C: SURVEY DATA

What grade are you in?	What school or college do you attend?	On a scale from 1 to 5, rate your C/C++ programming knowledge.	On a scale from 1 to 5, rate your Java programming knowledge.
9th Grade	ASL	1 - No experience.	2
9th Grade	The American School in London	1 - No experience.	3
10th Grade	Newton South High School	1 - No experience.	4
12th Grade	UHSSE	2	2
10th Grade	Trinity high school	1 - No experience.	1 - No experience.
12th Grade	Ballston Spa High School	1 - No experience.	1 - No experience.
9th Grade	Ballston Spa High School	5 - Expert.	3
9th Grade	Ballston Spa High School	1 - No experience.	1 - No experience.
12th Grade	Newton South High School	4	3
11th Grade	Newton South High School	1 - No experience.	1 - No experience.
11th Grade	Newton North High School	4	1 - No experience.
College Freshman	WPI	3	1 - No experience.
12th Grade	South High	1 - No experience.	2
10th Grade	High Shcool of Science and Technology	1 - No experience.	1 - No experience.
12th Grade	Metropolitan Learning center	1 - No experience.	1 - No experience.
12th Grade	Ballston Spa High School	1 - No experience.	1 - No experience.
10th Grade	Agawam High School	1 - No experience.	1 - No experience.
11th Grade	algonquin	3	4
11th Grade	Hauppauge High School	1 - No experience.	1 - No experience.
10th Grade	Hauppauge High School	2	2
9th Grade	ShrewsburyHigh School	1 - No experience.	1 - No experience.
9th Grade	American School in London	1 - No experience.	1 - No experience.
12th Grade	Dhorety high	1 - No experience.	1 - No experience.
College Junior	WPI	4	3
10th Grade	American School in London	1 - No experience.	3
College Junior	WPI	4	3
College Junior	WPI	4	4
College Freshman	WPI	2	3
College Freshman	WPI	2	1 - No experience.
College Freshman	WPI	3	1 - No experience.
College Freshman	WPI	2	1 - No experience.
College Freshman	WPI	4	2
College Senior	WPI	2	1 - No experience.
College Sophmore	WPI	4	2
College Freshman	WPI	2	1 - No experience.

participant number	How many programming classes have you taken?	Have you taken a robotics class?	If any, what other programming languages besides C, C++, or Java do you know?
1	0	No.	None
2	1	No.	N/A
3	3	Yes.	Python
4	1	Yes.	LabVIEW, Python, BASIC
5	0	No.	none
6	0	No.	None
7	1	No.	PHP,Javascript,LabView,Lua,X86ASM,lolcode
8	0	No.	None
9	1	Yes.	Python, Objective C
10	None	No.	none
11	0	Yes.	none
12	0	No.	MATLAB, MAPLE
13	2	No.	English
14	none	No.	none
15	0	Yes.	none
16	0	Yes.	Basic labview, and BOE bots (I forget the name of the language)
17	0	No.	python, html
18	3	No.	python
19	none	No.	none
20	0	Yes.	None
21	none	No.	HTML
22	0	No.	None
23	1	Yes.	vb
24	3	Yes.	Scheme
25	1	Yes.	Objecitve C
26	2	Yes.	C#. Python, Ruby
27	7	Yes.	Scheme
28	2	Yes.	Racket / Scheme
29	1	Yes.	None
30	1	Yes.	Racket / Scheme
31	0	Yes.	none
32	none	Yes.	LabVIEW, GML, easyC
33	0	Yes.	None
34	6	Yes.	Scheme
35	0	Yes.	none

participant number	On a scale of 1 to 5, how easy was the software to use?	Were you able to complete the objective?	What did you find difficult about the software?	How useful do you think this software would be for learning robotics programming ?	What did you like about the software?	What did you dislike about this software?
1	4	Yes!	I was a little confused, but the software was easy to use.	4	It was easy to use, and easy to see what happened to the Robt	It was a little bit slow
2	4	Yes!	N/A	3	The code was simple and easy to learn.	The process of running the code was a bit tedious.
3	4	Yes!	5 - Very useful.		The commands are simple to comprehend. THe graphics interface allow people to practise with a virtual world	nothing :D
4	5 - Very easy.	Yes!	Nothing	3	The graphical representation makes it easy for people to quickly change code and see their results, making it simple and easy for people to experiment and learn.	Python is pretty simple compared to the languages that are actually used for robotics programming. I don't believe it would translate well over, so if somehow this could be done in a C-based language, or a graphical programming similar to LabVIEW, then it would work.
5	4	Yes!	nothing but a little explaining helped a lot	4	simple	nothing
6	4	Yes!	5 - Very useful.		The instructions were easy to follow.	

7	5 - Very easy.	Yes!	nothing	4	UI,graphics	Lua
8	4	Yes!	Not Much	5 - Very useful.		User Friendlyness
9	4	Yes!	Mild crashing problems	5 - Very useful.	the interface was simple but pleasing, and the graphics was the same.	simple easy to follow, firmiliar syntax.
10	3		No...?	2		
11	4	Yes!	5 - Very useful.		Simplicity	Can't Handle a certain speed, but it's all good
12	5 - Very easy.	Yes!	Talking to eric there were so many studentsw that made it difficult to ask for help.	4	It represents stuff very well! I liked seeing the robots move.	I didnt like the grid i didnt see the point of it.
13	3	Yes!	Reading	4	The moving robot	Nothing
14	3		Yes!	3	yes	
15	2	No...?	I could not see where i needed to fill in "100"	4	It is virtual, so you can see what exactly you have acomplished.	n/a
16	5 - Very easy.	Yes!	Understanding the programming logic with limited experience. However, the software was very straight forward, and I feel it would be a perfect tool for teaching other novice programmers.	5 - Very useful.	The interface was very simple and easy to understand.	Though there are areas in which it could be improved, such as graphics, I would not say there was anything I disliked about the software.
17	5 - Very easy.	Yes!	nothing	5 - Very useful.	easy, in basic engligh, not too many symbols used	nothing
18	4	Yes!	syntax lack of semicolons	4	stuff	lack of semicolons
19	5 - Very easy.	Yes!	Nothing	4	Directions	nothing at all

20	4	Yes!	I didn't realize I had to change the challenge selected under the Build menu	3	Interaction/navigation in the virtual world	Too easy to copy and paste without much thought

8. APPENDIX D: IRB APPROVAL LETTER

100 Institute Road
Worcester, MA 01609-2280, USA
508-831-5000, Fax: 508-831-6090
www.wpi.edu

Worcester Polytechnic Institute IRB #1
IRB 00007374

1 March 2011
File: 11-023

Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609

Re: IRB Expedited Review Approval: #11-023 "Robotics Simulation Education"

Dear Prof. Padir,

The WPI Institutional Review Committee (IRB) approves the above-referenced research activity, having conducted an expedited review according to the Code of Federal Regulations 46.

The period covered by this approval is 1 March 2011 to 28 February 2012, unless terminated sooner (in writing) by yourself or the WPI IRB. Amendments or changes to the research that might alter this specific approval must be submitted to the WPI IRB for review and may require a full IRB application in order for the research to continue.

Please contact the undersigned if you have any questions about the terms of this approval.

Sincerely,

Kent Rissmiller
WPI IRB Chair