|  |  |  |
| --- | --- | --- |
| BW | | |
| American University of Sharjah  Department of Computer Science and Engineering | | |
| Robotics Simulator for Artificial Intelligence Course | | |
| Senior Design Project Presented by: | | |
| Firas Sardast (ID: 47500) | | |
| Issa Haddad (ID: 47102) | | |
| Omar Al-Nabulsi (ID: 47476) | | |
| Supervised by: | |  |
| Dr. Michel Pasquier | |  |
| December, 2015 |  | |
|  | | |
|  | | |

Abstract

Artificial intelligence, as its name implies, is the intelligence exhibited by particular software or a given machine. In other terms, there exists a robot agent that can perceive the environment around it and take actions accordingly in order to maximize the chances of its success. Since the AI is an essential topic in the both computer science and engineering, there exists a large demand by the students on this course. Our project aims to facilitate the way students learn about the AI course. Doing hands on training using a robot simulator can complement the theory behind the course. All available robotics simulators in the market are either too complicated to use, too vague, or too expensive and contain many unnecessary features that AI students may not find useful. Throughout our project, we will be researching about the different simulators that are available in the market. We will be choosing the most optimum software, enhancing it in way to fit the needs of students enrolling in the AI course. Our finalized tool will have the capability to run AI simulations, debug AI code, and view demos about various AI agents. The aim of this project is to enable students to fully utilize our software given the lab session time constraint and to eventually enable as many people to contribute to the AI research in this university and beyond.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
| We dedicate this work to our parents & professors. |
|  |
|  |
|  |
|  |
|  |
|  |

Acknowledgements

We would like to express our gratitude to Dr. Michel Pasquier for guiding us throughout this project. Our thanks also go to Dr. Imran Zualkernan for his continuous support in our work.

Table of Contents

[I. Introduction 1](#_Toc437369020)

1.1 [Artificial Intelligence 1](#_Toc437369021)

1.2 [Importance of Robot Simulators 2](#_Toc437369021)

[II. Problem Statement and Design Objectives 3](#_Toc437369024)

2.1 [Problem Statement 3](#_Toc437369025)

2.2 [Design Objectives 3](#_Toc437369026)

[III. Literature Review 6](#_Toc437369027)

3.1 [Gazebo 6](#_Toc437369025)

3.1.1 specifications 7

3.1.2 Robots and objects 7

3.1.3 Sensors 8

3.2 [V-rep 9](#_Toc437369026)

3.2.1 specifications 9

3.2.2 Robots and Sensors 10

3.3 [Morse 11](#_Toc437369026)

3.4 [Opennero 14](#_Toc437369026)

3.4 [Robotics simulators evaluation 18](#_Toc437369026)

[IV. System/Sub-System Specifications 22](#_Toc437369028)

4.1 [Functional Requirements 22](#_Toc437369026)

4.2 [Non-functional Requirements 24](#_Toc437369026)

[V. Technical Approach and Design Alternatives 25](#_Toc437369035)

[VI. Testing 34](#_Toc437369037)

[VII. Project Global, Economic, Societal Impact 36](#_Toc437369038)

[Viii. Perliminary cost Estimated Components 37](#_Toc437369040)

I[X. Project Management 38](#_Toc437369041)

[X. Standards 39](#_Toc437369042)

List of Figures

[**Figure 1- A display of an Environment in Gazebo……………………………………8**](file:///D:\Logs\Downloads\COE490_SDP%20Report%20_Fall%202015_v1.5.doc#_Toc437368989)

**Figure 2- Testing NAO robot on V-rep……………………………………………….10**

**Figure 3- Display of environment and robots………………………………………...11**

**Figure 4- Another display of an environment and robot……………………………12**

**Figure 5- Components in a Morse environment……………………………………...13**

**Figure 6- Environment in OpenNero…………………………………………………16**

**Figure 7- Another environment in OpenNero………………………………………..17**

**Figure 8- Agent action flow chart……………………………………………………..18**

**Figure 9- V-rep’s architecture environment………………………………..……….. 25**

**Figure 10- Simple Reflex Agent……………………………………………..……….. 28**

**Figure 11- Goal-based Agent……………………………………………..………….. 28**

**Figure 12- Flow chart for the move forward LUA function………………………...31**

**Figure 13- First Demo Environment………………………..………………………...32**

**Figure 14- Second Demo Environment……………………..………………………...32**

Figure 15- grid used to adjust the 90 degree rotation………………………………...33

List of Tables

[**Table 1- Specification**](file:///D:\Logs\Downloads\COE490_SDP%20Report%20_Fall%202015_v1.5.doc#_Toc437368989) **comparison table ………………………………………… 18**

**Table 2- Robots and sensors comparison table………………………………….. 19**

**Table 3- Standards………………………………………………………………… 39**

# Introduction

## Artificial Intelligence (AI) needs to be taught by practice, and what better way is to teach AI through simulating robots and allowing them to move based on certain algorithms. Testing those algorithms on robots can help students visualize the movement of the robots, thus understand the purpose of every algorithm. Are the available Robotics simulator tools efficient enough to aid students in the understanding of AI?

## *1.1 Artificial Intelligence*

Artificial intelligence is a discipline that combines methods of behavior, thought processes and, reasoning to achieve both rationality and intelligence. In order to achieve the purpose behind AI, scientists use what they call an agent. Agent is simply anything that can gather information from its surroundings by using multiple sensors. To assess any AI agent there is a performance measure that determines the successfulness of a certain agent. As the nature of an agent changes, the performance measures’ criteria changes accordingly. Moreover, the agent should be able to learn and estimate some information regarding the environment it is placed in. Therefore, there are a lot of tested AI algorithms that help achieve the main goal of learning AI. The main goal of learning Artificial Intelligence is to be able to build an intelligent system that can rationally interact with it is surroundings. Some of the famous AI algorithms are tree-search, graph-search, reflex-vacuum-agent, and table-driven-agent [1].

## *1.2 Importance of Robot Simulators*

The best way to teach AI is to explain the movement of the agent in response to AI algorithm used. The usage of graphs or trees would not give enough vision for student to understand the use of such algorithm. Therefore, many researchers and professors use robot simulators to test AI algorithms because most of the robot simulators allow the users to view the movement of the agent they created. To enhance the learning process of students enrolling in an Artificial Intelligence course, CMP 433, professors should consider providing students with available free robot simulators. Students should have to apply AI algorithms and test them using robot simulators because it will give them insight into the tested algorithm and develop expertise in using it [2]. Moreover, students can view how the agent created interacts with the environment they chose and how it moves around and collects information. The AI algorithm will become more understandable to students after viewing a real life application of it. Furthermore, students can then test their understanding by using other AI algorithms and inspecting the results on the agents.

# Problem Statement and Design Objectives

## *2.1 Problem Statement*

The main challenge is to find a free open-source tool that can incorporate the library that covers the topics included in the Artificial Intelligence: A Modern Approach (AIMA). Another challenge is to make the tool user friendly so that students can directly write their codes and view the output without any complications. Most of the available robotics simulator tools are commercial and their source code is not available. The commercial tools’ license is usually expensive to obtain even for a single user. Moreover, receiving one license is not enough because the aim of this project is to provide the software for students enrolling in the artificial intelligence course. The availability of the source code is important in our case because we may need to modify and enhance the tool to make it easier for the students to use. Furthermore, nearly all of the tools available do not include ready libraries with AI algorithms. There are certain algorithms that are covered in the AIMA textbook that need to be available in the simulator’s library. The tool should maintain a user friendly GUI which will make it easy for students to use and access the several functions that the tool provides.

## *2.2 Design Objectives*

The main objective of our senior design project is to identify the best available robot simulation tool and to modify its source code. First we need to research the free available tools online. Second we need to compare the tools we researched and choose the best which appeals more to our set of requirements. Third we need to modify the source code accordingly to meet our set of requirements.

The research measures we are following focuses on:

* Mandatory requirements:
  + Free tool
  + Active and supported
  + Available source code
  + Available physics engine
  + Mobile Robots
  + Available sensors
  + The API interoperability with the AIMA library
* Optional Requirements:
  + Available for Windows
  + Various environments
  + Programming language and source code language in C++

Second step is to compare the different tools available. We are going to go through each one individually by matching their specifications to the set of requirements. Then we will create a comparison table which will include the best tools only. The comparison table’s main advantage is to aid us take a broad view of what every tool provides compared to the others. Then we should weigh every requirement according to it is importance because the process of choosing the best tool requires a tradeoff. We will assess every tool based on the importance of the requirement it incorporates. The step of choosing the right tool ends here.

Third, the missing requirements in the tool we have chosen needs to be implemented according to its importance. Therefore, we will go through the source code in order to enhance the tool and add the missing parts. One important step we need to make while enhancing the tool is to include the library provided by AIMA (Artificial Intelligence: A Modern Approach). Moreover, there is an extra step we will do to make the software available for students outside campus labs without worrying about the type of operating system used. We will try to integrate the enhanced software into an operating system image that will be installed on a virtual machine created by VMware. This would reduce the installation time and the installation complexity of the software for students.

# Literature Review

In this section of the report we will mention in details the specifications of the tools we researched and we will expand on the comparing process we went through to choose the best tool. Here are the various available robot simulators with their specifications

## *3.1 Gazebo*

Gazebo is a robotics simulation tool that was developed back in 2002 in the University of Southern California. Gazebo’s main purpose was to enable users to simulate robots that can undergo several circumstances in an outdoor environment [3]. A significant fact to mention is that Gazebo is supported by an agency of the U.S. Department of defense which is the Defense Advanced Research Projects Agency (DARPA). Moreover, DARPA is looking forward to introduce Gazebo a learning tool for students majoring in the robotics engineering fields [4]. Gazebo simulator is not a commercial product, therefore requires only an Apache 2.0 license for installation. Furthermore, it is an open-source tool and the source code can be downloaded from Gazebo’s main website. The source code includes the GUI header files, sensors, and the various tools used [4]. Therefore, we can modify the code or enhance it according to the requirements of our project.

## *3.1.1 Specifications*

Gazebo is available on Linux and the version available for Windows is still under testing. However, the MAC OS has been recently integrated but still under testing. Besides, the source code is available in C++ and the programming done by users to simulate the movement of robots is also in C++. It also incorporates a physics engine written in C++ and this engine is created to simulate kinematics related to the robot models. The physics engine takes into account collision detection, rotational functions, and many other useful features [5]. It also accesses several high performance physics engines such as ODE, Bullet, DART and Simbody. Gazebo also has a GUI interface that is user friendly. The environments can be either implemented in 2D or 3D depending on the purpose of the experiment conducted. The program provides a plain environment; however, the user can enhance it by creating his own environment depending on his needs. There is also a functionality that reviews the code itself and checks the code style. Another important feature is that Gazebo allows cloud simulation which is the ability to run simulations on online servers [3].

## *3.1.2 Robots and Objects*

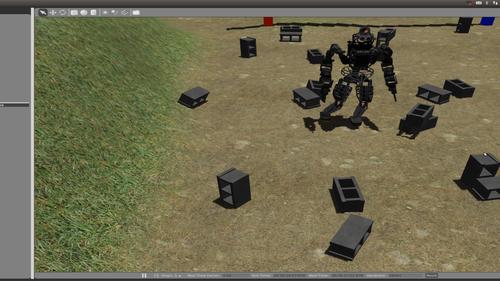
All objects in Gazebo have a well-defined mass, friction, velocity and many other attributes so that they will have in a realistic behavior. Such attributes will allow the simulated robot to push, pull, or carry these objects. Moreover, the robots available are dynamic structures made of rigid parts connected to each other. Those robots can undergo different forces to allow interaction with the environment they are placed in [5]. There are several robot models available in Gazebo such as PR2, Pioneer2 DX, iRobot Create, and TurtleBot.

Figure 1 A display of an Environment in Gazebo [5]

Likewise, users also have the freedom to build their own robot models using SDF and test them through Gazebo. SDF is a format developed alongside with Gazebo to describe aspects such as kinematics and surface properties of the designed robot. Gazebo also allows the specified model to act either as a ground mobile robot, a water robot, or an aerial robot. Also the robots can be as simple as a robotic hand/arm or as complex as a Humanoid or a human avatar [3].

## *3.1.3 Sensors*

Gazebo also uses a basic API to allow user to create robots and add to them sensors. Those sensors are basically abstract and need to be attached to the robot model in order to function. There are different sensors that are supported by the Gazebo simulator such as laser range finders, 2D/3D cameras, Kinect style sensors, contact sensors, and force-torque, collision sensors, and GPS. These various sensors being attached to robots can be used to test different AI techniques [3].

## *3.2 V-rep*

The Virtual Robot Experimentation Platform is a robot simulator with a unified development atmosphere and is based on distributed control architecture. The distributed control architecture functionality allows every object to be controlled individually which makes V-rep adaptable for multiple robotics applications [6].V-rep has aced both in academic and industrial fields because of it is ability to accomplish various tasks such as system verification, simulation of complex assembly chains in factory automation applications, algorithm optimization and robot task planner and controller [7].

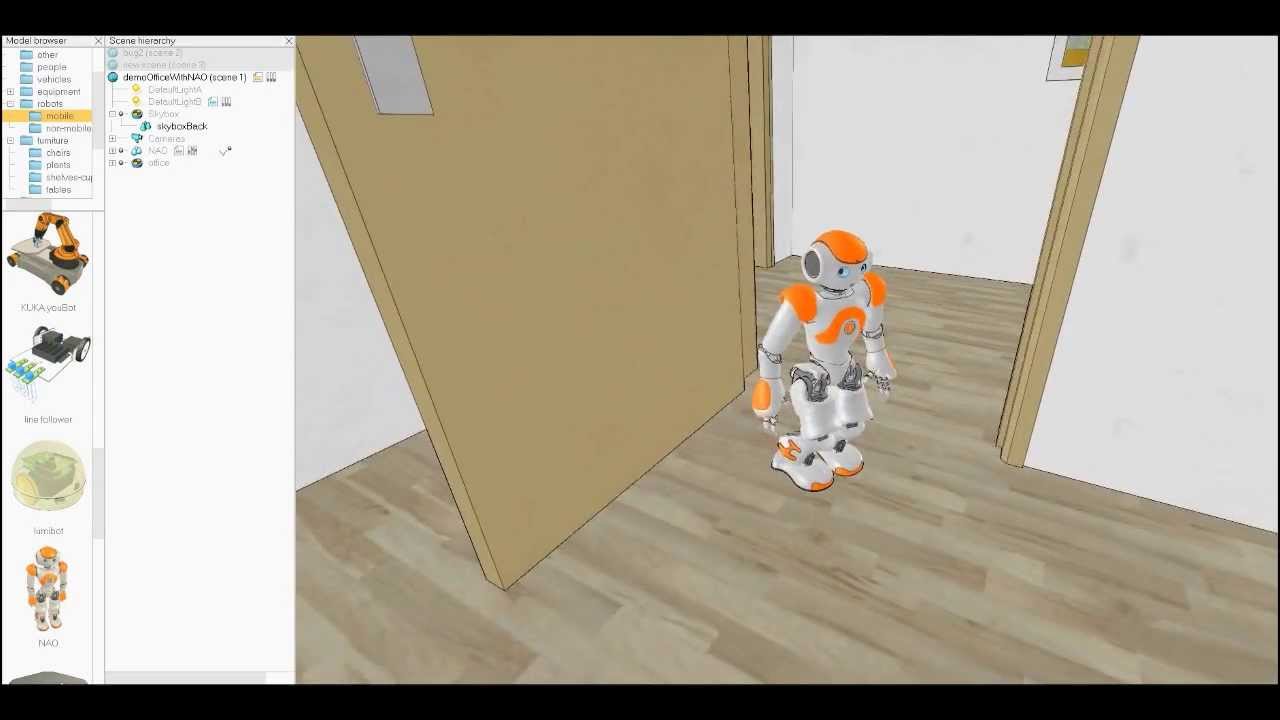
## *3.2.1 Specifications*

V-rep is available on multiple platforms such as Linux, Windows, and MacOS X. The source-code is available on their website for both commercial and educational use. V-rep allows the user to customize any aspect of the simulation process and the simulator itself because of the API they use. It also supports six programming approaches and every approach has its own language and functionalities. A significant fact to mention is that the user can use those six programming techniques simultaneously and symbiotically [7]. Away from the regular API V-rep also provides a remote API which allows the control of a simulation from within another application or even a remote hardware. Moreover, V-rep access four physics engines Bullet, ODE, Vortex dynamics, and Newton dynamics. The GUI interface is user friendly because most of the functions can be accessed from the toolbars on top of the window or from the menu bar. Simple simulations may require no or minimal amount of programming; therefore, this will allow student to focus on coding the AI techniques for simple scenarios [6]. One main purpose of V-rep is to ease user tasks and reduce complexity [7].

## *3.2.2 Robots and sensors*

V-rep provides various kind of robots such as mobile ground robots, aerial robots, robotic hands/arms, Humanoid, and human avatars. The manipulation of objects in V-rep is very simple they can be shifted or rotated by buttons. Moreover, the sensors it provides can be attached to other objects all combined to create a building simulation scene. Available sensors are proximity sensors, vision sensors, force sensors, and cameras. Every sensor has its own detailed specifications, for example the proximity sensor has six different models. With the help of the sensors V-rep simulator can provide measurements regarding distance calculations between objects; therefore, it can detect collisions between robots and other objects within their environment. The user is also allowed to control the speed of the simulation during the simulation itself [6].

Figure 2 Testing NAO robot on V-rep [6]



## *3.3 Morse*

Morse is an open robots simulator engine with contributors from more than 15 academic universities worldwide [8].This makes this particular simulator one of the most highly rated and highly efficient open source software on the net. Morse comes equipped with many useful standard sensors, highly professional actuators, and diverse list of a built in robotic bases. Much of its success comes from the fact that it is based on the Blender Game Engine as well as it uses state-of-the-art Bullet library for physics simulation [9]. These advanced features will be elaborated on further in the following paragraphs.

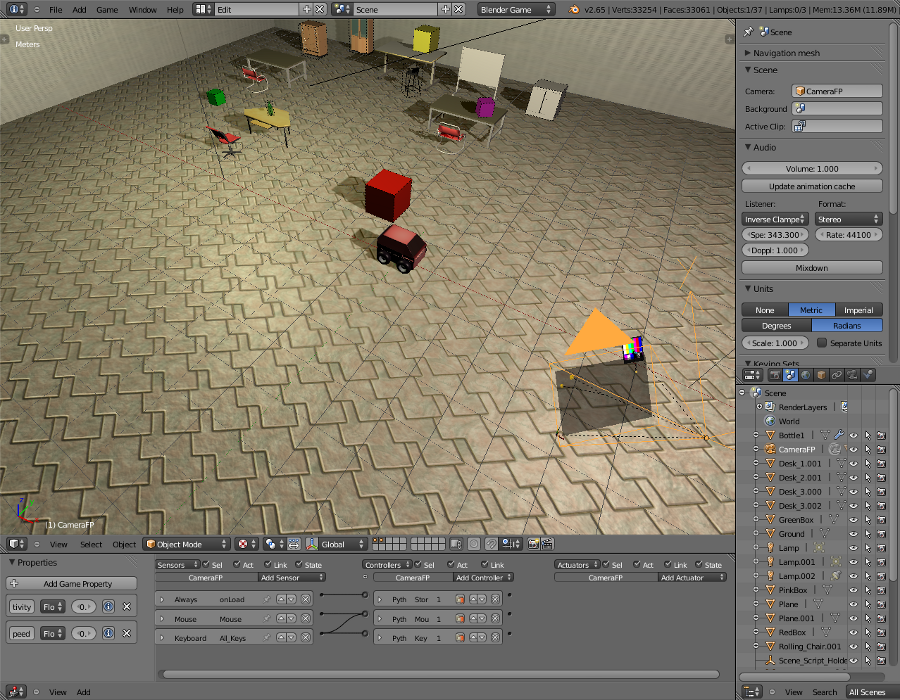
Blender Game Engine is a free three dimensional creation suit, which means that it can be used for multiple purposes such as 3D modeling, rigging, animation, simulation, rendering, compositing, motion tracking, video editing and game creation [10]. In Morse simulator, this can be used to create professional environments to run our robot simulation in. An example of an environment created using Blender Engine can be viewed below in the figure.

Figure 3 Display of environment and robots [9]

Bullet on the other hand is a collision detection and rigid body dynamics library, which means that using Bullet, one can create simulations with real time physics laws [11]. This peace of magnificent library was created initially by Erwin Coumans, who worked previously for Sony Computer Entertainment and who now works for Google. The library is very accurate that it is not only used in the gaming industry, it is also used in NASA Tensegrity Robotics Toolkit [12]. The figure below illustrates how this library can be used to study the physical simulation of many structures.

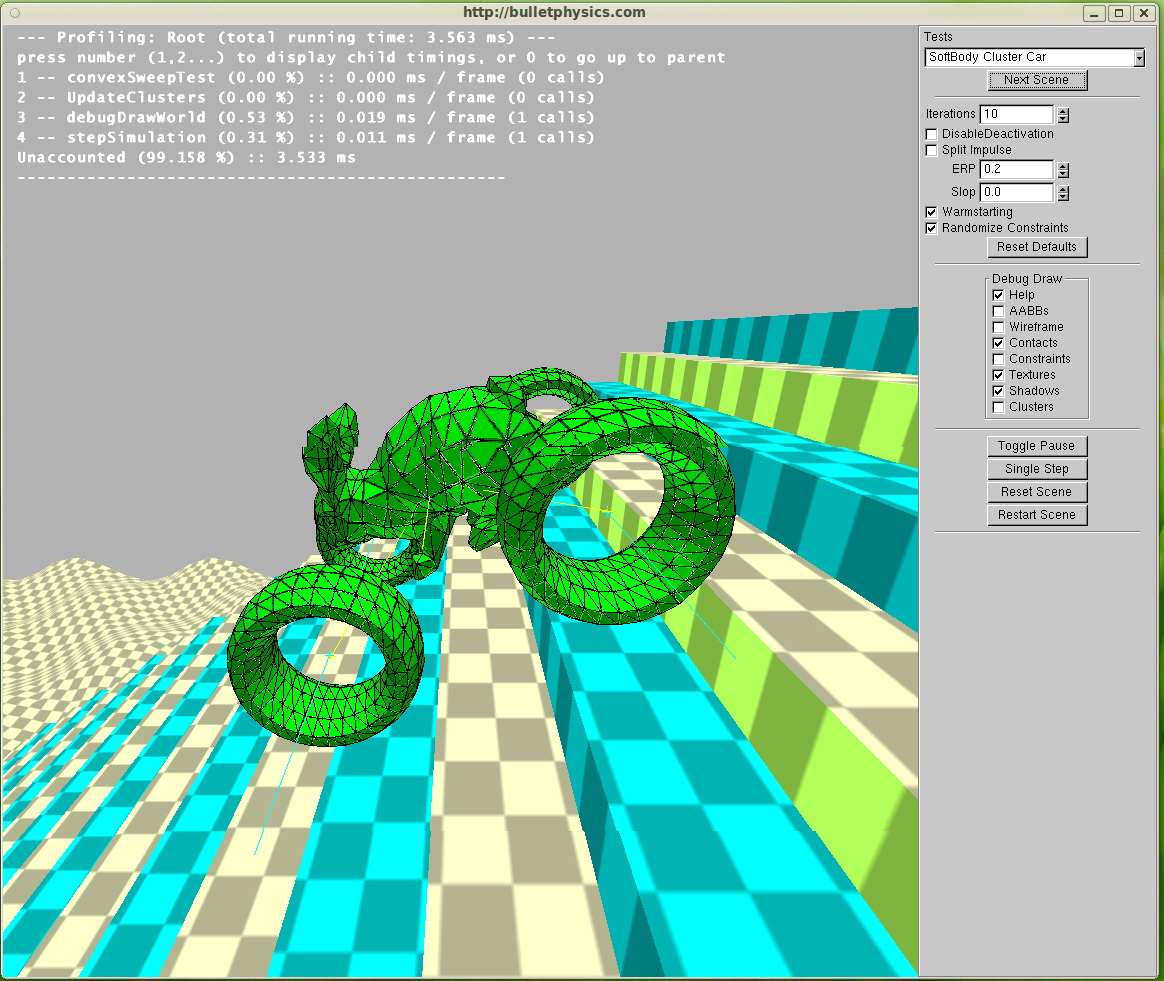


Figure 4 Another display of an environment and robot [19]

Simulations in Morse are completely handled with the command line. This means that the simulator offers very little or no graphical user interface. All simulations are to be executed using small python scripts. Given the many built in components that Morse offers to its clients, one important feature is that depending on your need, you can chose whether realism of motion is a necessity or whether supervision of your robot is the priority. You can manipulate the degree of effectiveness of you simulation simply by adding or removing components such as accurate camera sensors or waypoint controller [9].

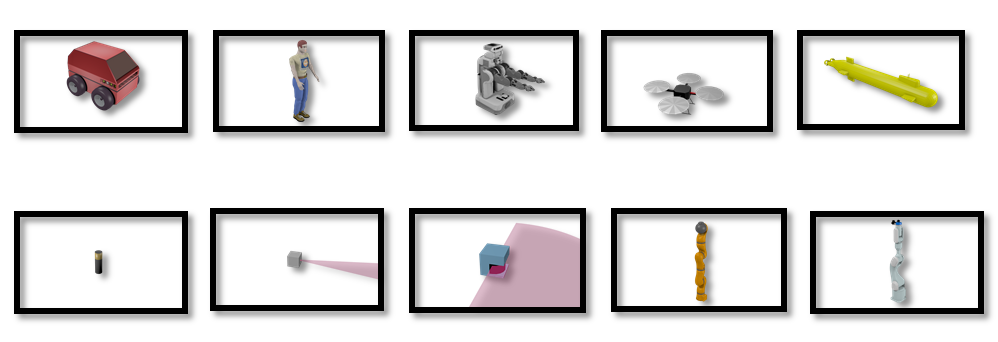
Morse offers a variety of built-in robotic bases. It offers land robots, flying robots, as well as underwater robots like the submarine. Morse also comes equipped with many useful sensors you can integrate such as the battery, depth camera, infrared, video camera, gps, and the accelerometer. Moreover, many actuators or in other terms a controlling mechanism are offered such as the teleport which teleports the robot to a specific coordinate position in the environment. Other actuators include the gripper which is capable of grabbing objects marked as Graspable. A waypoint actuator enables the robot to move towards any given coordinate in the environment while restricted to move only forward, turn around z-axis and going up or down [13]. The figure below shows how some of the components look inside the environment.

Figure 5 Components in a Morse environment [13]

Developers in Morse found out that it is a good idea to support multiple middleware in the simulator. Middleware is a tool or software that provides services that otherwise the operating system cannot offer. It is basically used to perform input and output communication between software components. Morse support only four open-source middlewares, these include ROS, YARP, Pocolibs and MOOS [9]. To better understand what a middleware is, take ROS for instance. One of the problems that ROS can solve in your simulation is done using robot geometry library. This library keeps track of where different sensors in the robot sit in relation to each other. Combining data from a camera sensor as well as data from a laser sensor, one must know where each sensor sits inside a common frame of reference; this is where ROS enters. Using robot description language, a set of tools offered by ROS can describe and model your robot so that it can be understood by all the ROS system [14].

Although Morse seems like the perfect software to use for our project, nevertheless it has some limitations that we need to consider before deciding which software tool we need to adopt. For example, Morse is supported only on Linux, which means that limited support is offered to Windows or MacOSX users. Morse does not guarantee that the user will be able to accurately simulate a robot arm for instance. Morse does not come equipped with advanced algorithms such as path planning and many others. Another limitation which was described earlier is the unavailability of graphical user interface, which brings down the simulator’s ease of use [9].

## *3.4 OpenNero*

Open Nero is an open source game platform that is used for educational and research purposes which is based on game platform Neuro-Evolving Robotic Operatives (NERO), as it is a game for teaching AI techniques by providing a visual game experience for students to demonstrate different AI techniques. Moreover, it was introduced due to NERO being very difficult to adapt new environments, game rules and learning algorithms. Furthermore, Open Nero is similar to other game engines as it provides common features such as 3D graphics, physics simulation and a graphical user interface. It also has AI techniques embedded in it, where they are demonstrated using different environments [15].

Additionally, Open Nero allows the user to create a 3D environment by randomly defining a 2D maze with a start and a goal state, and then a creation process that consists of 3 parts concludes the creation of the environment. First is the environment stage in which two tools are used to create the visual related layer. The importer tool is used to load 3D models and set the physical attributes, animations and sounds. The second tool which is the Builder tool is used to insert objects into the environment. Second is the task stage which requires the user to define the environment logic and sensors for detecting actions consumed by characters. The sensors request queries that are predefined in the simulator such as the camera, range sensors, object counters and filters. Lastly, the agent is defined by the user by defining the start, act and end methods of the agent in the script as the AI algorithms are already implemented in Open Nero [15].

An example of the environments in Open Nero is the maze mod which is a basically a maze environment. Open Nero provides features for such an environment such as colored markers and cubes for labeling the goal/ending point in the maze and the path taken by the agent to reach the goal starting from the beginning point using different AI techniques that are selected by the user. This helps in illustrating the logic of the technique and how it works better. Moreover, a red cube labels the goal point, a yellow marker to indicate the next location to be taken, a blue marker to show the path taken by the agent up until the current position, a green marker to label locations that the agent may return to and lastly a white marker to illustrate the overall correct path [16].

Furthermore, the agent goes through the maze, as shown in Figure 6, using AI search algorithms that are predefined in OpenNero. AI search techniques include [16]:

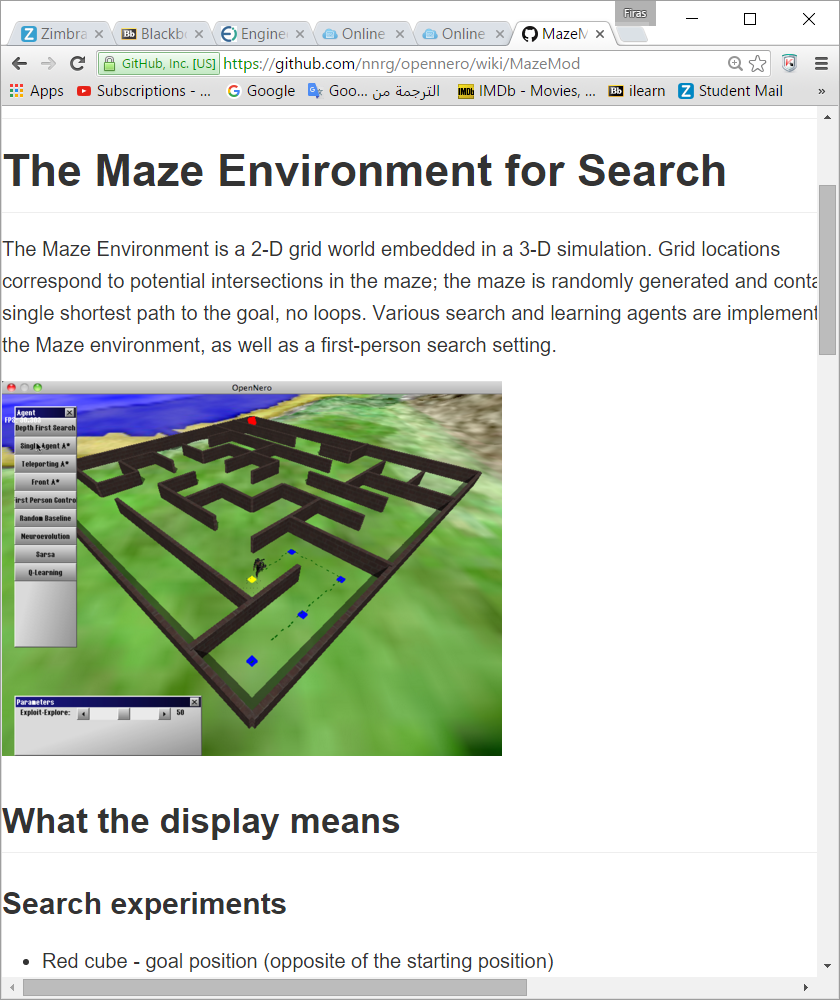
* Depth First Search
* Breadth First Search
* A\* Search with 3 types of visualizations:
* Single Agent A\* Search: the agent goes through the maze by making progress and back tracking to move on
* Teleporting A\* Search: the agent can teleport to the next open node instead of back tracking which is faster
* Front A\* Search: the agent can create multiple clones of itself when faced with different path
* First Person Control: it allows the user to navigate the maze using a first-person camera to try and solve the maze by himself.
* Q-Learning

Figure 6 Environment in OpenNero [16]

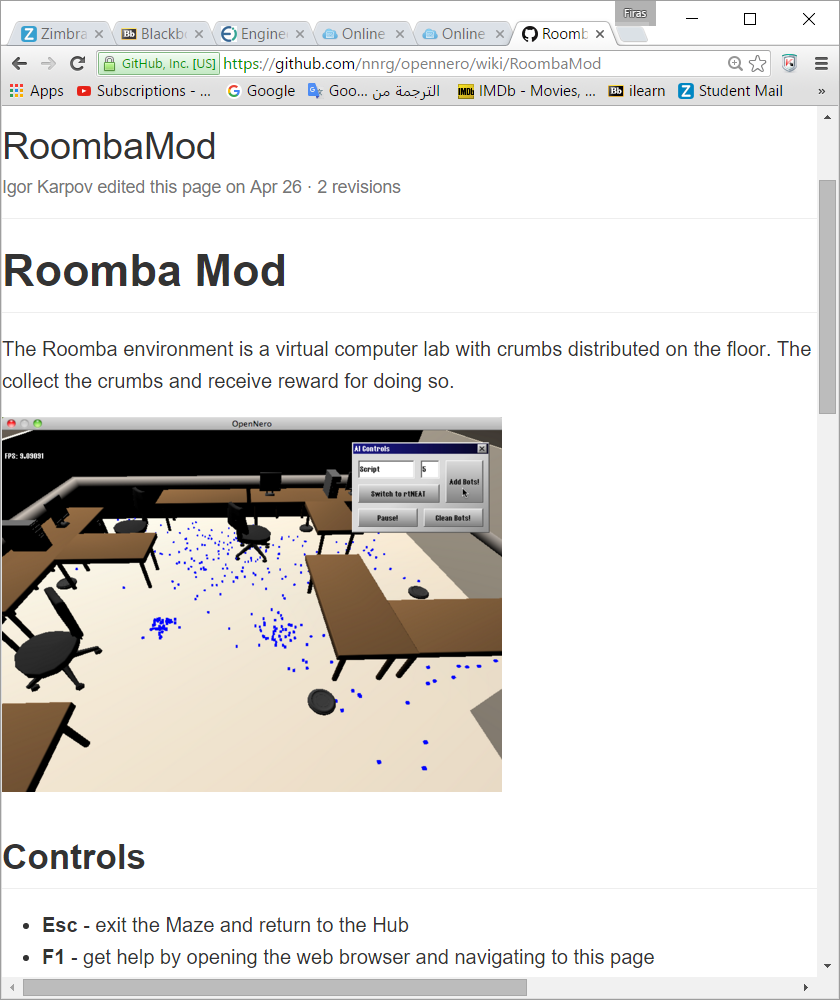
Moreover, there are other modes such as NERO mode where the user constructs a team to battle an opponent team. Roomba mode is an environment that reflects a computer lab with dirt particles on the floor, and the agent has to collect all of the dirt particles. This resembles a cleaner robot that cleans the floor from all kinds of dirt particles as shown in Figure 7 [16].

Figure 7 Another environment in OpenNero [16]

An overview of the overall system and how the agent and the environment interact is illustrated in Figure 8, where the agent does observe the environment, and reacts with appropriate actions accordingly. Whenever the agent performs a correct action it gets rewarded [16].

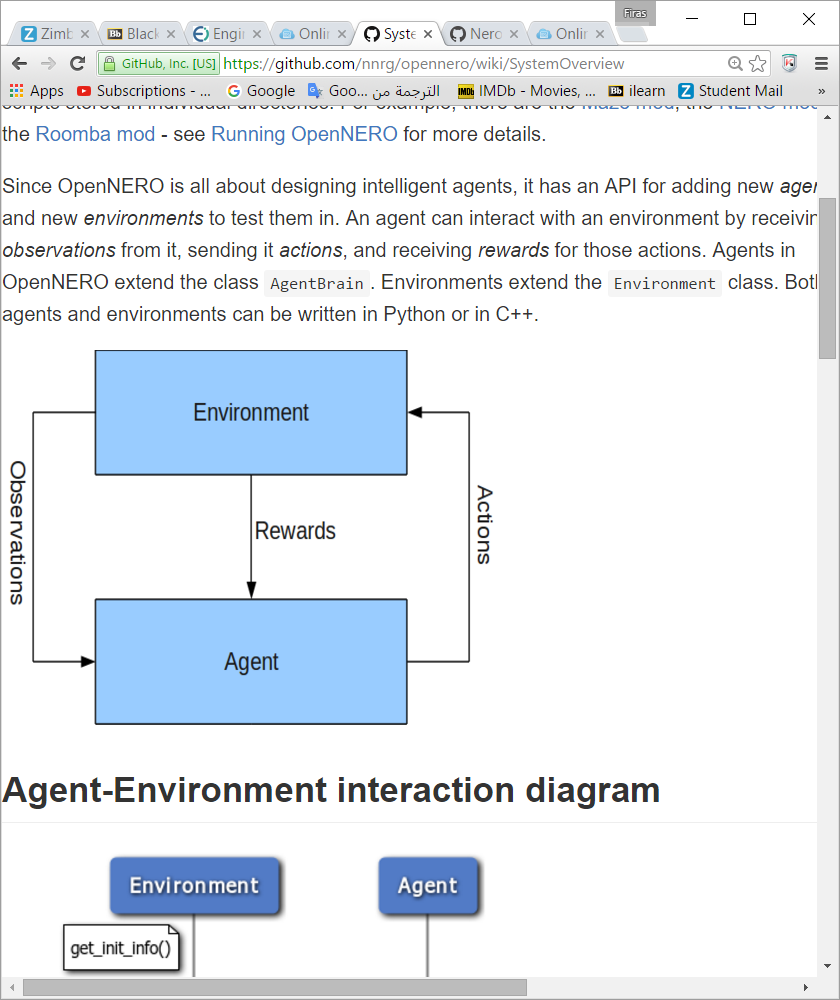


Figure 8 Agent action flow chart [16]

## *3.5 Robotics Simulators Evaluation*

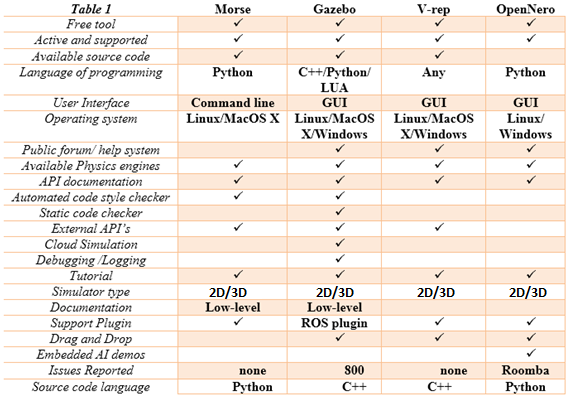
****

Table 1 Specification comparison table

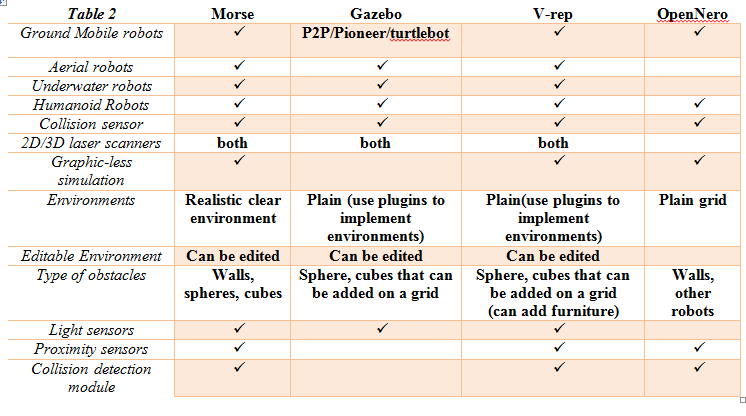


Table 2 Robot and sensor comparison table

After researching about the different simulators we could possibly use for our project, we created a table to help us better view the different features offered by each of the simulators. Our next objective will be to compare those listed features and to determine the most suitable one, in order to simplify the work done by us as designers and for the student who will be using the finalized software.

One of the most important criteria to measure is the operating system in which the software will be accessed on. Windows and MacOS are relatively easier to work on than Linux, this is mainly because Linux relies heavily on the command line which many student may find as not user-friendly. Morse can run only on Linux which makes it less likely to be chosen by us as the most efficient simulator; on the other hand, Gazebo, V-rep, and OpenNero can be downloaded on any operating system.

Another important criterion to note is the fact that students have to be comfortable with the programming language they will be using. Gazebo, V-rep, and OpenNero all provide remote Application Programming Interfaces that run Java, Python, or C++. In the case of OpenNero it only allows Python, and in an early development stage we were not sure of our capability with Python; therefore, we dropped OpenNero from the list.

Being limited down to Gazebo and V-rep, it was a hard decision to choose between them by just looking at their specifications without actually trying them out. However, we first took a look on their specifications just to note down what to test after downloading each of the simulators. Moreover, looking at the specifications table did not provide us with a clear distinction between both simulators because there were minor insignificant differences between the two. For example, Gazebo provides cloud simulation which is an interesting feature; but unfortunately it is not a main requirement in the simulator we want to develop. However, taking a look at the issues reported by users on both simulators, we clearly saw that V-rep didn't receive any issue reports compared to the 800 issues reported in Gazebo.

Moving on to the Robot and sensor comparison table we took a look at the specifications they both provide when building an environment and choosing the right robot model to test. There are two features that we took into account which made V-rep superior to Gazebo which are the availability of proximity sensors and collision detection modules. We cannot judge which the best program is through the features only, we needed to install both and give them a run because Gazebo may have some alternative features to the ones it is missing.

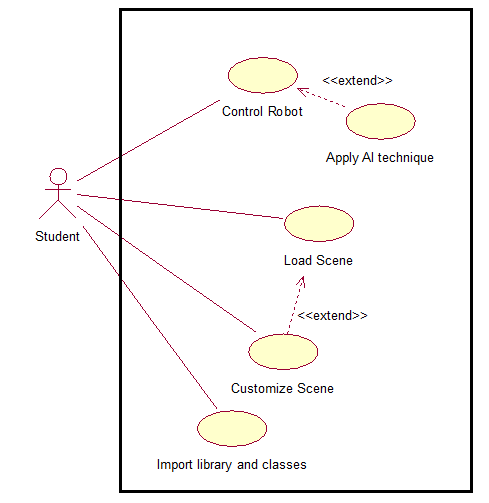
We tried to install Gazebo on different PC's in the university and at home; however, it did not work and their website went down for a couple of weeks. We could not waste time trying to install it; therefore, we went on V-rep's website and downloaded their educational version. That is how we got to reach that V-rep is the most efficient simulator among the ones we researched about. Our work did not end here; on the contrary it was just the first step toward completing our senior design project. We opened multiple forums, tutorials, and websites in order to have a thorough look at the functionalities of V-rep so we can have a better idea of how to adjust it by using the features that are suitable to our framework.

# System/Sub-System Specifications

***4.1 Functional Requirements***:

1. Load and customize scene in v-rep
   1. The user should be able to load a scene in v-rep with a predefined environment and agent.
   2. User should be able to customize the environment by adding and removing objects
2. Import v-rep library and classes
   1. Student should import v-rep library and include classes into their project
3. Student should be able to control the robot from python
   1. The user should be able to use predefined functions that control the agent from python
      1. The move forward function: this function moves the agent one step forward
      2. The turn left function: this function turns the agent 90 degrees to the left
      3. The turn right function: this function turns the agent 90 degrees to the right
      4. The get proximity sensor function: returns a reading that indicates if there is a wall ahead of the agent
   2. The user should be able to use these functions to write a script that directs the agent to reach the destination
4. Student should be able to customize the agent according to their own preference

**Use case representing the V-rep software with the students’ interactions:**



## *4.2 Non-functional requirements:*

1. Ease of use
2. Availability
3. Compatibility
4. Integrity
5. Attractiveness
6. Durability

# Technical Approach and Design Alternatives

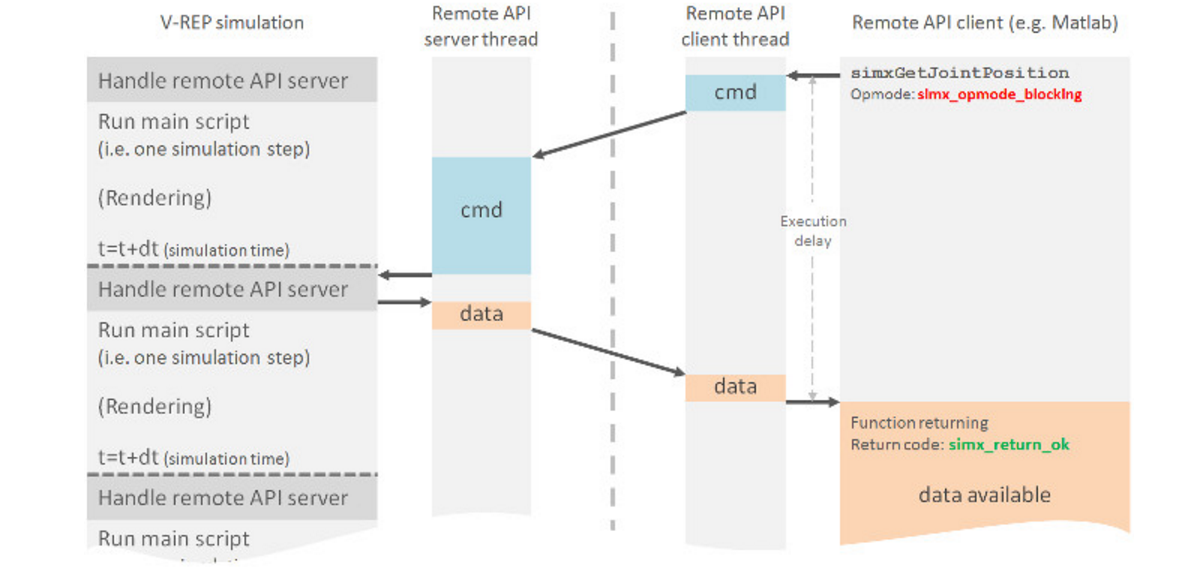
V-rep offers a remote API that allows an external application to control the simulation. The remote API contains approximately one hundred specific functions and one generic function that can be called from a python script. Moreover, the remote API interacts with v-rep via a socket communication in a way that reduces lag and network load [17]. Furthermore, we integrated v-rep with Spyder which is an IDE for python by including the v-rep library and classes in the same python project. Moreover, we used 2-tier architecture as v-rep facilitates the use of this architecture, and the following graph illustrates our architecture.

Figure 9 V-rep’s architecture [27]

V-rep robotics simulator allows its users to create a remote Application Programming Interface (API), so they would not be restricted to writing their codes in its main language LUA. At first, we connected V-rep to Eclipse which provides a Java integrated development environment (IDE) because students in the American University of Sharjah are more familiar with Java language than with Python. However, the remote API functions provided from V-rep to Java users are complicated due to the amount of classes that need to be defined. Therefore, we switched to python even though many students are not familiar with such a language. The easy syntax that python provides makes it an easy language to learn, thus not being familiar with it is an insignificant issue. Using python we were able to use the remote API functions provided by V-rep in a simpler and faster manner.

Moreover, python has a certain advantage that we seek for that Java fails to provide. The purpose of our project is to provide simplicity to the users when developing their codes. Python codes take less time to develop, and are generally shorter by 3 to 5 times than their equivalent Java codes. Python doesn't require its users to waste time declaring variables and defining classes; however, it allows the user to run the code directly [18].

An agent or a robot can be classified into multiple program types. The following is a list of the different types that can be used to describe the level of intelligence of any particular agent. Particularly, the further the type of the agent in the list, the more sophisticated the agent is.

1. Simple reflex agent
2. Model-based reflex agent
3. Goal-based agent
4. Utility-based agent
5. Learning agent

This list can be generalized into two types of agents only, either a simple reflex agent or a goal based agent. The rest can be thought of as just an enhancement to either of the two types. These two are specifically the types that we’ll be implementing in our simulator. On the one hand, a simple reflex agent is simply an agent that is given a set of rules on how to behave when it receives any kind of data from its sensors. This type agent can be executed only using multiple conditional statements. On the other hand, a goal based agent is an agent that needs to be fed with information on the overall goal objective. This agent combines percepts, knowledge, and information to decide on a specific action sequence plan that would achieve the goal. It cannot be executed using simple conditioning statements; rather it requires a specific library that can handle the logic. The library we are going to use is called the AIMA library. It has all the necessary features required for the goal based agent to function.

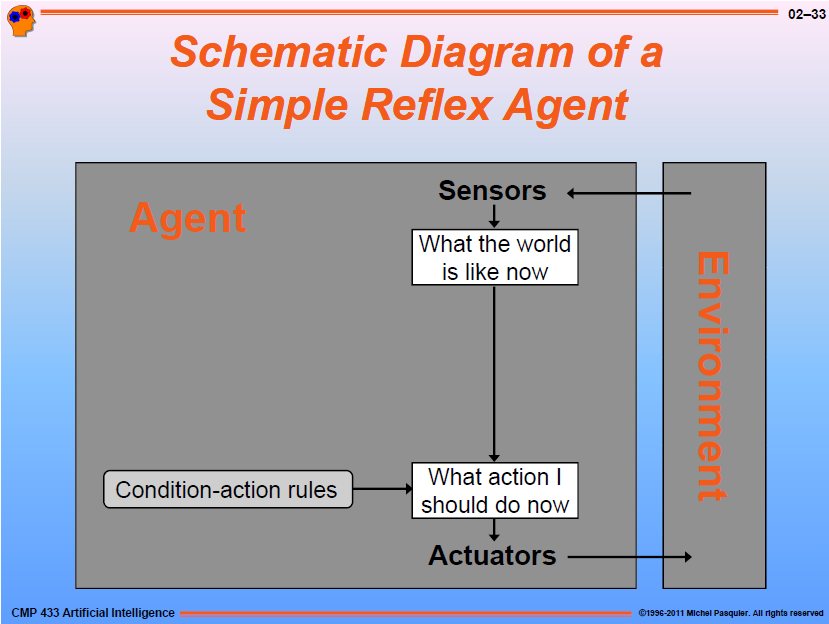


Figure 10 Simple Reflex Agent [1]

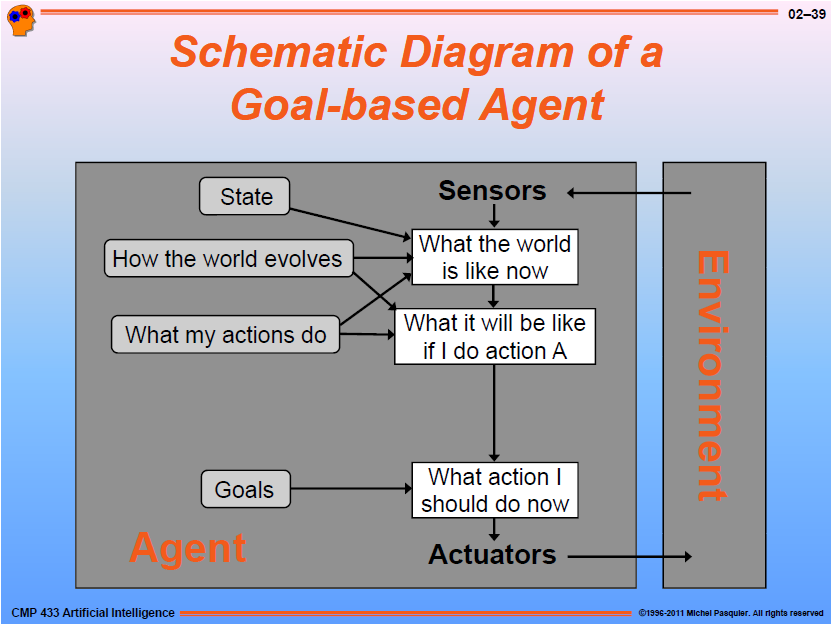


Figure 11 Goal-based Agent [1]

We’ll be creating two demos for our senior project, one for each type of agent we are implementing. An illustration of the first environment can be viewed below. The robot will be placed in any of the squares, facing any given direction. The robot should then find its way to the goal block which is marked by an “x” sign.

To begin creating any demo, three steps should be completed before any student can test his or her algorithm on the simulator.

1. Build Environment
2. Build Agent
3. Write Functions

First, we need to build the environment necessary for the demo. Second, we need to build our robot, and equip it with the required sensors and actuators to freely move around and detect objects such as walls. After creating our environment and agent, we should finally write functions that students will call in order to necessary manipulate the movement of the agent.

In our first demo, we created an environment similar to that shown in **Figure 13**. The agent can be placed anywhere and facing any direction in the environment. The goal state in our first demo is marked by an ‘X’ sign. The robot has to find its way using well defined algorithms and functions to reach the goal state. Our agent will be equipped with four proximity sensors, one on each side of the agent. The proximity sensor will be used to detect different objects and their distance around it. Our agent must be able to rotate exactly 90 degrees around its own axis without moving beyond the block it stands on. Furthermore, the agent should be able to move forward to any adjacent block. A solution to the specifications described was to place two joints, one on the right and one on the left of the robot, which will act like wheels. If the velocity of both joint is set to a positive constant value, the agent will be able to move forward. If one joint has a negative constant value and the other has the same constant value but negative, the agent should be able to rotate either clockwise or anticlockwise depending on which joint is positive and which is negative. Programming scripts in V-rep is written in LUA. We will have to define functions that control the movement of the agent in these scripts before we can call them from a separate remote API.

In our second demo, we created an environment similar to that shown in **Figure 14**. We are using the same agent as that in the first demo. Our functions in the simulator script are also the same as the first demo. The difference between this and the previous demo is that we are using the AIMA library here. Inside the remote API, we called functions specific to the AIMA library. We wrote different algorithms that are suitable for the goal-based agent

Moving the agent forward from one block to any adjacent one results in the position of the agent to increase or decrease by 0.5 and either on the x-direction or on the y-direction. Meanwhile rotating the agent 90 degrees clockwise or anticlockwise results in the orientation of the agent to increase or decrease by 90 degrees. Refer to the figure below. The appendix contains a simple “Move forward” function that we created within the scripts of our simulator. Refer to figure below for the activity diagram of the move forward function.

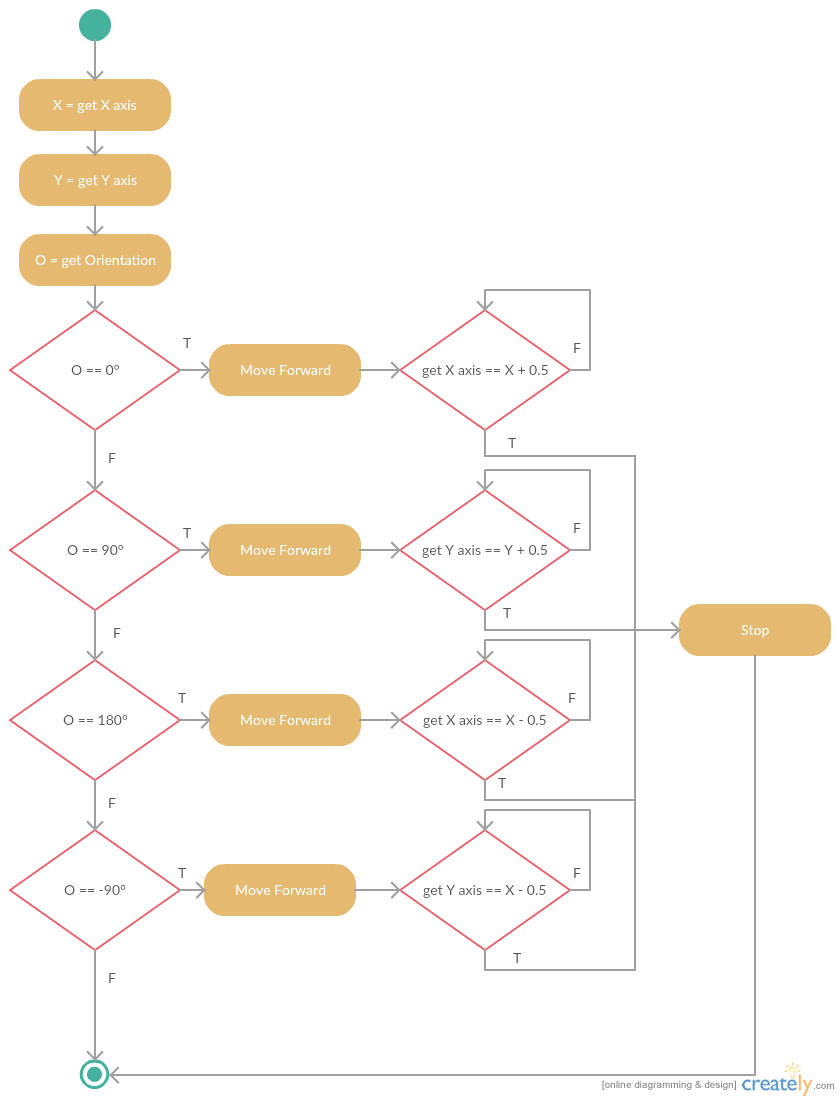


Figure 12 Flow chart for the move forward LUA function

# 

Figure 14 Second Demo Environment

Figure 12 First Demo Environment

Whilst working on our first demo, which is the simple reflex agent, we stumbled against many glitches that were related to our logic, simulator, and agent of choice. Three of our main difficulties are listed below in bullet points.

* In our first demo, we used an agent that has only two wheels, one on each side. When setting both wheels velocity to a positive constant speed, the agent moves forward. When we set one wheel velocity as positive and the other as negative, the agent rotates to the side, and so on. Our agent should be able to rotate clockwise or anticlockwise without getting outside of the boundary of the block where it is sitting in. But due to the asymmetrical shape of robot, the robot gets out of position on the long run. This bug didn’t occur initially as the robot worked fine in the short run.
* Our agent should be able rotate exactly 90 degrees, otherwise the agent will not behave as expected to reach the goal. We found a bug in the simulator itself, where one the predefined functions in the simulator responsible for setting the orientation of the agent is faulty. We had to find another way to quickly resolve this matter. We made the robot slow down right before it reaches 90 degrees just as normal person would slow down to stop for a red light. That way, we managed to precisely rotate the agent as expected.
* We should also link our simulator to an external IDE. Moreover, we faced multiple issues when integrating the IDE with V-rep. Moreover, we started with integrating a java IDE (Netbeans) with v-rep. However, the documentation was not straight forward, and it took a lot of research and trial and error to figure out. Moreover, we were simultaneously trying to integrate python with vrep as an alternative if the integration with java didn’t work out. Furthermore, fortunately we were able to make both of them work.
* Another issue was the functions that were provided by the V-rep library in both python and java. Although V-rep provided a partially decent documentation for its functions, it didn’t illustrate which functions in java are responsible for controlling the robot and invoking the LUA script in V-rep. Hence, we spent a lot of time researching and trying multiple approaches to try to control the robot.

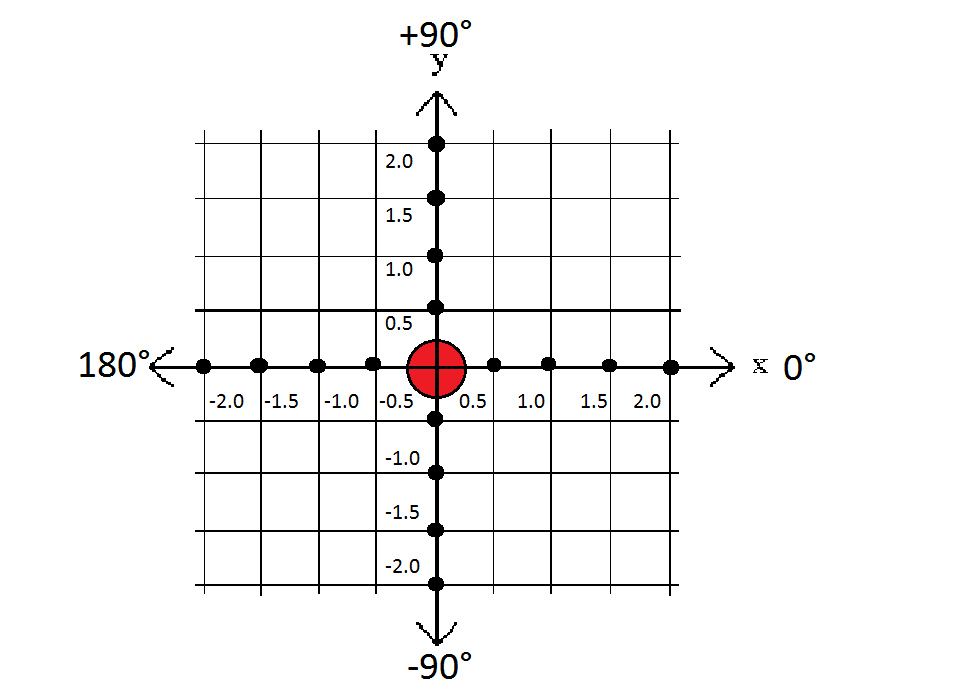


Figure 15 grid used to adjust the 90 degree rotation

# Testing

To evaluate our software we will test the algorithms and create demos corresponding to the algorithms taught in the AI course. We will also test the software to assure it fulfils it is purpose. We need to check if the software will be good enough to act as an educational tool for the Artificial intelligence course.

**Test process:**

The following are two important testing strategies that we adopted for our project:

* + Black-Box testing
  + White-Box testing

In our first strategy which is the black-box testing, we checked the reliability of the python remote API. We established connection set up between our client side which is the API and our simulator which acts like the server. We tested simple functions like “Move Forward” and “Stop”. We made sure that our simulator grasped those functions and applied them correctly to the agent. Our second strategy which is the white-box testing, we dove deep into our code. We created multiple test cases based on selective path testing. We made sure that each test case was carried appropriately and without any malfunctions. The following is a list of the different test cases we applied:

* + Turn clockwise 90 degrees when facing down
  + Turn clockwise 90 degrees when facing up
  + Turn clockwise 90 degrees when facing right
  + Turn clockwise 90 degrees when facing left
  + Turn anti-clockwise 90 degrees when facing down
  + Turn anti-clockwise 90 degrees when facing up
  + Turn anti-clockwise 90 degrees when facing right
  + Turn anti-clockwise 90 degrees when facing left
  + Turn clockwise 90 degrees twice in a row
  + Turn anti-clockwise 90 degrees twice in a row
  + Move forward one block
  + Move forward one block then turn clockwise 90 degrees
  + Move forward one block then turn anti-clockwise 90 degrees
  + Move forward one block then stop

By doing both of the testing strategies, we ensured the full delivery of the requirement functionality proposed by our project. We also managed to drastically minimize and prevent future errors.

# Project Global, Economic, Societal Impact

**Educational Impact:**

We want to develop this software and make it available for students enrolling in the AI course. The software will help students test their understanding of the theoretical parts of the course by trying to visualize how certain algorithms run. Therefore, students will have the opportunity to create various environments and agents to test the algorithms they studied in the course. Being able to view the movement of robots in response to certain algorithms, the students will thoroughly understand the content and structure of the algorithm.

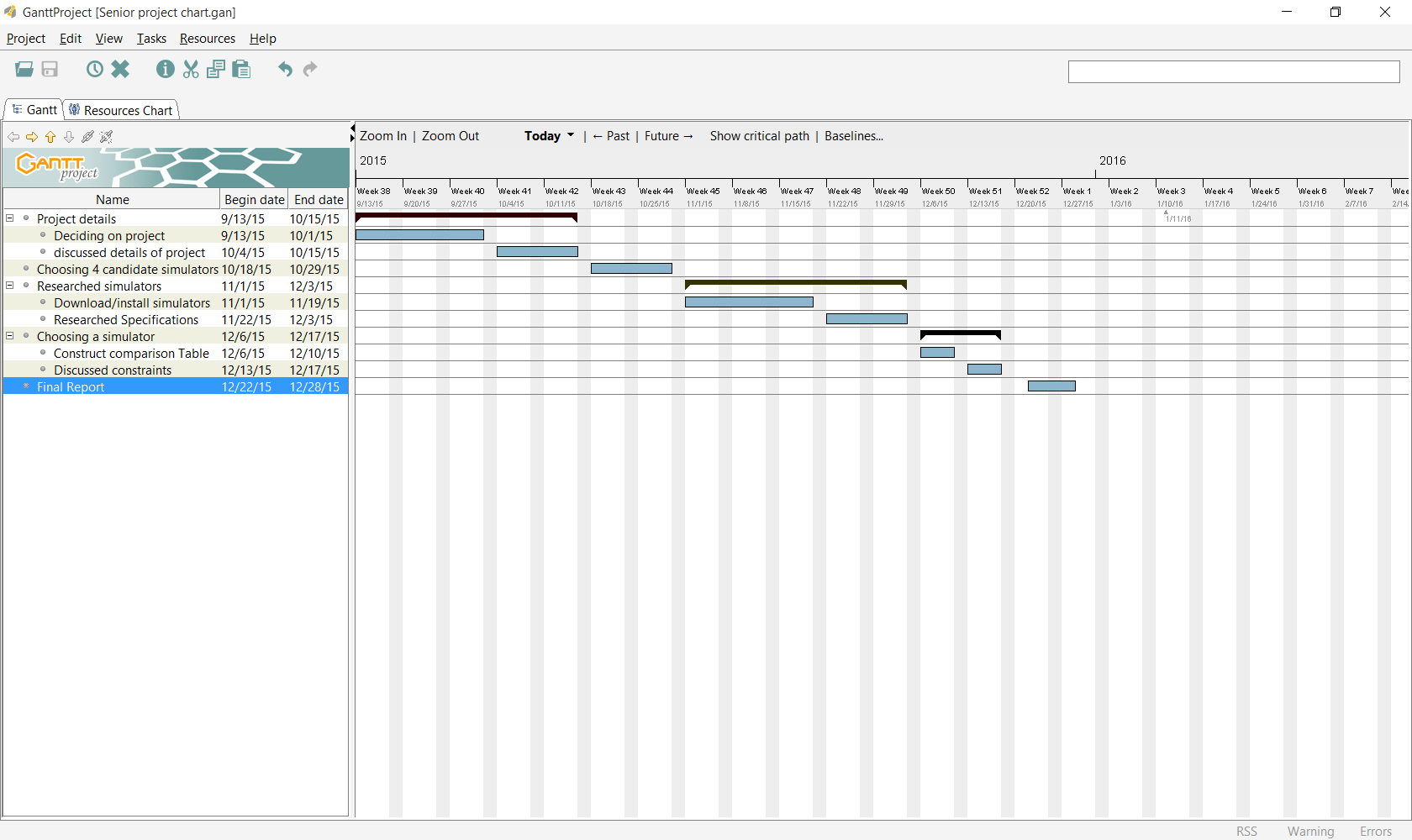
**Societal Impact:**

The software we are using allows the users to simulate real life environments and create agents that will interact with it depending on the AI code the user specifies. That is resource efficient because instead of building agents and testing them in real life and perhaps damaging them in certain cases, the user would simulate his code and verify if it will work efficiently then go on to build the agent desired.

# Preliminary Cost Estimates and Components

The only programs that are going to be used are V-rep and Spyder. Both are free open source tools.

# Project Management



# Standards

Here are certain IEEE standards that our software will be following:

|  |  |  |
| --- | --- | --- |
| ID | Area | Description |
| ISO/IEC 90003 | Software and Systems Engineering-- Guidelines for the Application of ISO 9001:2000 to Computer Software | This standard provides guidance for organizations in the application of ISO 9001:2000 to the acquisition, supply, development, operation and maintenance of computer software. |
| ISO/IEC 14764:2006 | Information Technology-- Software Maintenance | This standard elaborates on the maintenance process provided in ISO/IEC 12207. It provides guidance in implementing the requirements of that process. |
| SO/IEC 15026:1998 | Information Technology-- System and Software Integrity Levels | This International Standard introduces the concepts of software integrity levels and software integrity requirements. It defines the concepts associated with integrity levels, defines the processes for determining integrity levels and software integrity requirements, and places requirements on each process. |
| ISO/IEC 15939:2002 | Software Engineering-- Software Measurement Process | This standard provides a life cycle process for software measurement. The process is suitable for use with IEEE/EIA 12207. |

**Table 3 Standards [25]**

# Glossary

|  |  |
| --- | --- |
| AI | Artificial Intelligence |
| AIMA  DARPA | Artificial Intelligence: A Modern Approach  Defense Advanced Research Projects Agency |
| GUI | Graphical User Interface |
|  |  |
|  |  |

# Bibliography

|  |  |
| --- | --- |
| [1] | S. J. Russell and P. Norvig, Artificial Intelligence A Modern Approach, New Jersey: Prentice-Hall, 1995. |
| [2] | "Nuffield Foundation," Nuffield Foundation, 2015. [Online]. Available: http://www.nuffieldfoundation.org/. [Accessed 23 December 2015]. |
| [3] | "Gazebo," Open Source Robotics Foundation, 2014. [Online]. Available: http://gazebosim.org/. [Accessed 25 November 2015]. |
| [4] | M. Chaoui, T. DeSilva, L. Johnston and C. Wall-Epstien, "Evaluating Simulators in FRC," Worcester Polytechnic Institute, Worcester, 2014. |
| [5] | N. Koenig and A. Howard, "Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator," University of Southern California, Los Angeles. |
| [6] | "v-rep virtual robot experemintation platform," Coppelia Robotics, 13 October 2010. [Online]. Available: http://www.coppeliarobotics.com/index.html. [Accessed 11 December 2015]. |
| [7] | E. Rohmer, S. P. Singh and M. Freese, "V-REP: a Versatile and Scalable Robot Simulation Framework," in *International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, 2013. |
| [8] | "GitHub," GitHub, Inc, 2015. [Online]. Available: https://github.com/morse-simulator/morse. [Accessed 14 December 2015]. |
| [9] | "openrobots," LAAS-CNRS/ONERA , 2015. [Online]. Available: http://www.openrobots.org/morse/doc/stable/morse.html. [Accessed 12 December 2015]. |
| [10] | "blender," Blender Institute BV, 2015. [Online]. Available: http://www.blender.org. [Accessed 22 December 2015]. |
| [11] | "Bullet Collision Detection & Physics Library," doxygen , 12 August 2015. [Online]. Available: http://www.bulletphysics.org/Bullet/BulletFull/. [Accessed 21 December 2015]. |
| [12] | "Bullet Physics Library," Real-Time Physics Simulation, 2015. [Online]. Available: http://bulletphysics.org/wordpress/. [Accessed 24 December 2015 ]. |
| [13] | "openrobots," LAAS-CNRS/ONERA , 2015. [Online]. Available: http://www.openrobots.org/morse/doc/stable/components\_library.html. [Accessed 12 December 2015]. |
| [14] | "ROS," Open Source Robotics Foundation, 2015. [Online]. Available: http://www.ros.org/core-components/. [Accessed 22 November 2015]. |
| [15] | I. V. Karpov, J. Sheblak and R. Miikkilainen, "OpenNERO: a Game Platform for AI Research and Education," University of Texas, Austin , 2008. |
| [16] | "GitHub," GitHub, Inc, 2015. [Online]. Available: https://github.com/nnrg/opennero/wiki. [Accessed 12 December 2015]. |
| [17] | "Remote API," coppeliarobotics, [Online]. Available: http://www.coppeliarobotics.com/helpFiles/en/remoteApiOverview.htm. [Accessed 12 March 2016]. |
| [18] | "Comparing Python to Other Languages," Python, [Online]. Available: https://www.python.org/doc/essays/comparisons/. [Accessed 24 April 2016]. |
| [19] | "BULLET PHYSICS LIBRARY," 80 level, 2015. [Online]. Available: http://80.lv/vendors/bullet-physics-library/. [Accessed 17 November 2015]. |
| [20] | L. Nogueira, "Comparative Analysis Between Gazebo and V-REP Robotic Simulators," Universidade de Campinas, 2009. |
| [21] | "SDFormat," Open Source Robotics Foundation, 2014. [Online]. Available: http://sdformat.org/. [Accessed 26 December 2015]. |
| [22] | "The main client application," coppeliarobotics, 2015. [Online]. Available: http://www.coppeliarobotics.com/helpFiles/en/mainClientApplication.htm. [Accessed 26 December 2015]. |
| [23] | "Plugins," coppeliarobotics, 2015. [Online]. Available: http://www.coppeliarobotics.com/helpFiles/en/plugins.htm. [Accessed 26 December 2015]. |
| [24] | "SOURCEFORGE," Slashdot Media, 2015. [Online]. Available: http://sourceforge.net/p/ikvm/wiki/User%27s\_Guide/. [Accessed 26 December 2015]. |
| [25] | "SOURCEFORGE," Slashdot Media, 2015. [Online]. Available: http://sourceforge.net/p/ikvm/wiki/Convert\_a\_jar\_file\_to\_a\_dll\_and\_use\_it\_as\_library/. [Accessed 26 December 2015]. |
| [26] | "Comprehensive List of Important IEEE Software Engineering Standards," PennState Harrisburg, 2015. [Online]. Available: https://turing.cs.hbg.psu.edu/cmpsc487/IEEEStds\_List.htm. [Accessed 2 January 2016]. |
| [27] | "Remote API modus operandi," coppeliarobotics, [Online]. Available: http://www.coppeliarobotics.com/helpFiles/en/remoteApiModusOperandi.htm. [Accessed 12 March 2016]. |

# Appendix A

**LUA CODE:**

**moveForward=function()**

**-- move forward**

**IOcheck = simGetObjectOrientation(robot,-1)**

**-- 0=down, 1=right, 2=up, 3=left**

**if(IOcheck[3]<-1.4 and IOcheck[3]>-1.7) then**

**state = 3**

**elseif (IOcheck[3]<0.3 and IOcheck[3]>-0.3) then**

**state = 0**

**elseif (IOcheck[3]<1.7 and IOcheck[3]>1.4) then**

**state = 1**

**else**

**state = 2**

**end**

**simSetJointTargetVelocity(motorLeft,1)**

**simSetJointTargetVelocity(motorRight,1)**

**-- get new position**

**IPnew = simGetObjectPosition(robot,-1)**

**-- check when to stop**

**if(state==0) then**

**if(IPnew[2]<=oldy-0.45) then**

**simSetJointTargetVelocity(motorLeft,0.1)**

**simSetJointTargetVelocity(motorRight,0.1)**

**if (IPnew[2]<=oldy-0.5) then**

**simSetJointTargetVelocity(motorLeft,0)**

**simSetJointTargetVelocity(motorRight,0)**

**-- update old position to be able to make it moveforward again, (if below code is uncommented, robot will keep moving)**

**-- oldy = oldy - 0.5**

**end**

**end**

**end**

**if(state==1) then**

**if(IPnew[1]>=oldx+0.45) then**

**simSetJointTargetVelocity(motorLeft,0.1)**

**simSetJointTargetVelocity(motorRight,0.1)**

**if (IPnew[1]>=oldx+0.5) then**

**simSetJointTargetVelocity(motorLeft,0)**

**simSetJointTargetVelocity(motorRight,0)**

**-- update old position to be able to make it moveforward again, (if below code is uncommented, robot will keep moving)**

**-- oldx = oldx + 0.5**

**end**

**end**

**end**

**if(state==2) then**

**if (IPnew[2]>=oldy+0.45) then**

**simSetJointTargetVelocity(motorLeft,0.1)**

**simSetJointTargetVelocity(motorRight,0.1)**

**if (IPnew[2]>=oldy+0.5) then**

**simSetJointTargetVelocity(motorLeft,0)**

**simSetJointTargetVelocity(motorRight,0)**

**-- update old position to be able to make it moveforward again, (if below code is uncommented, robot will keep moving)**

**-- oldy = oldy + 0.5**

**end**

**end**

**end**

**if(state==3) then**

**if (IPnew[1]<=oldx-0.45) then**

**simSetJointTargetVelocity(motorLeft,0.1)**

**simSetJointTargetVelocity(motorRight,0.1)**

**if (IPnew[1]<=oldx-0.5) then**

**simSetJointTargetVelocity(motorLeft,0)**

**simSetJointTargetVelocity(motorRight,0)**

**-- update old position to be able to make it moveforward again, (if below code is uncommented, robot will keep moving)**

**-- oldx = oldx - 0.5**

**end**

**end**

**end**

**end**