MOBILE ROBOT SIMULATOR WITH JMONKEYENGINE (JAVA BASE PROGRAMMING)

NG WENG KAI

BACHELOR OF MECHATRONICS ENGINEERING (HONS.)

UNIVERSITY MALAYSIA PAHANG

MOBILE ROBOT SIMULATOR WITH JMONKEYENGINE (JAVA BASE PROGRAMMING)

NG WENG KAI

Thesis submitted in partial fulfillment of the requirements
for the award of the degree of
Bachelor of Mechatronics Engineering (Hons.)

Faculty of Manufacturing Engineering
UNIVERSITY MALAYSIA PAHANG

JUNE 2015

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name	:	NG WENG KAI
Identification Card No	:	9105100759036
Title	:	ROBOT SIMULATOR WITH JMONKEY ENGINE (JAVA PROGRAMMING BASE)
Academic Session	:	2014/2015

I declare that this thesis is classified as:

☐	CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1972)
☐	RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☐	OPEN ACCESS	I agree that my thesis to be published as online open access (Full text)

I acknowledge that Universiti Malaysia Pahang reserve the right as follows:

1. The Thesis is the Property of University Malaysia Pahang.
2. The Library of University Malaysia Pahang has the right to make copies for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

(Author's Signature)	(Supervisor's Signature)
NG WENG KAI	DR. NG LIANG SHING
Date: 16 JUNE 2015	Date: 16 JUNE 2015

SUPERVISOR’S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequatein terms of scope and quality for the award of Bachelor of Mechatronic Engineering.

Signature	:
Name of Supervisor	: Dr. Ng Liang Shing
Position	: Senior Lecturer
Date	:

STUDENT’S DECLARATION

I hereby declare that the work in this thesis is my own except for quotations and summaries which have been duly acknowledged. The thesis has not been accepted for any degree and is not concurrently submitted for award of other degree.

Signature	:
Name	: Ng Weng Kai
ID Number	: FB 11053
Data	:

ACKNOWLEDGEMENT

I am thankful and would like to express my greatest gratitude to University Malaysia Pahang for giving me opportunity to being a student and fulfill my dreams here. I would also like to thanks to Faculty of Manufacturing Engineering and all the staff for the facility and support provided during the entire project. Not forget also to my supervisor Dr. Ng Liang Shing, I am truthfully appreciate and thankful for his guidance and words of encouragement throughout the project period. Without his support, the completion of this project will not be possible. Plenty special thanks to all my fellow friends for the support, advice and inspirations through this project. Lastly, I acknowledge my sincere indebtedness and gratitude to my parents for their love, dream, and sacrifice throughout my life.

ABSTRACT

Simulation has become an important part in design or development for an autonomous robot. Most simulation comes with accurate physic simulation, multi platforms, high quality rendering and open source code. There are several reasons indicated that computer simulation is essentially tool for robotics field, particularly for mobile autonomous robot. Firstly, robots are fragile and hard to test and develop due to their complexity. The ability to test and develop in simulation as in real world environment can reduce the risks and cost of hardware failure. Moreover, there will be costly in develop a robot if the risk of failure is high. Simulations allow us to test the robot in software while waiting for the delivery of components. Nowadays, service robots are becoming more and more visible in daily life. We can see that many countries have been started to undergo research on service robots. Even we search for services robot on YouTube, I can fairly certain that all the results can be summed into one little word, robot. Many researchers have been done on service robot by different organization. There are 2 types or service robot, remote controlled and autonomous. In ecological applications, service robots are used to collect waste and dangerous item in indoor and outdoor environment.

TRANSLATION OF ABSTRACT

Simulasi telah menjadi sebahagian penting dalam reka bentuk atau pembangunan robot autonomi untuk. Kebanyakan simulasi datang dengan simulasi fizik tepat, platform pelbagai, terjemahan berkualiti tinggi dan kod sumber terbuka. Terdapat beberapa sebab-sebab Yang Dinyatakan Itu alat simulasi komputer untuk robotik adalah dasarnya bidang, khususnya bagi robot autonomi mudah alih. Pertama, robot adalah rapuh dan sukar untuk menguji dan membangunkan kerana kerumitan mereka. Keupayaan untuk menguji dan membangunkan dalam simulasi seperti dalam persekitaran dunia sebenar boleh mengurangkan risiko dan kos kegagalan perkakasan. Lebih lebih, akan ada Mahal dalam membangunkan robot jika risiko kegagalan adalah tinggi. Simulasi membolehkan kita untuk menguji robot dalam perisian sementara menunggu penghantaran komponen. Pada masa kini, robot perkhidmatan menjadi lebih dan lebih jelas dalam kehidupan seharian. Kita boleh melihat thatmany negara havebeen mula menjalani penyelidikan mengenai robot perkhidmatan. Walaupun kami mencari perkhidmatan robot di YouTube, saya boleh agak tertentu Itu Semua keputusan boleh disimpulkan dalam satu perkataan kecil, robot. Ramai penyelidik havebeen dilakukan pada robot perkhidmatan oleh organisasi yang berbeza. Terdapat 2 jenis perkhidmatan atau robot, kawalan jauh dan autonomi. Dalam Aplikasi ekologi, robot Perkhidmatan digunakan untuk mengumpul bahan buangan dan berbahaya di ruang persekitaran dalaman dan luaran.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

2D	2 Dimensional
3D	3 Dimensional
SDK	Software Development Kit
ODE	Open Dynamics
ROS	Robot Operation System
SLAM	Simultaneous localization and mapping
LWJGL	Light Weight Java Game Library
OpenGL	Open Graphic Library
USARsim	Unified System for Automation and Robot Simulation
OpenCV	Open source Computer Vision

CHAPTER 1

1. INTRODUCTION

1.1.INTRODUCTION

This chapter is mainly focus on discussing about the background of study, problem statement, and objectives of the project and also the scope of work to be done.

1.2.BACKGROUND OF THE STUDY

Robotics is science of designing and building machine that can be programmed and communicated with human to perform more than one tasks. The word robot comes from a play written in 1920 by the Czech author Karel Capek. He was the inventor who invented humanlike machine designed to perform human task in many forms.

Nowadays, service robots are becoming more and more visible in daily life. We can see that many countries have been started to undergo research on service robots. Even we search for services robot on YouTube, I can fairly certain that all the results can be summed into one little word, robot. Many researchers have been done on service robot by different organization. There are 2 types or service robot, remote controlled and autonomous. In ecological applications, service robots are used to collect waste and dangerous item in indoor and outdoor environment.

The main purpose of mobile robotics is the implementation and design of autonomous systems. In an autonomous system, a mobile robot will execute taskswith increasing complexity. The complexity of a task refer to the input information from the sensor, manipulate signal, than develop a plan to execute the certain task to solve the arising problems. For an example, for a robot navigation problem, the task of the robot is going from X to Z. To execute this task, the robot needs to receive and collect the information about the current environment, knowledge about its own capability to execute and move the motion task by sensing the environment. A path planner module is needed to determine the optimum way taking into account all the obstacles. Finally, an execution controller is in charge of implementing the planned solution in the actual environment.

When we deal with more complex tasks, we need to have a platform to run our simulation and experiment so that our solution proposals could be verified before implementation in a real robot. That is why we need a platform (simulation tools) that able to let us test our theoretical methods by using the realistic virtual robots in different scenarios. In the simulator, we have to access to the different capabilities of the virtual robot both for actuation and sensing in a realistic manner so that the simulation results able to export and implement on the same task in real robot.

Therefore, the purpose of this project is to develop a robot simulator by using pure java programming software for mobile robot simulations. A virtual robot and some environments were developed in this project.

1.3. **PROBLEM STATEMENT**

Robot simulator was important for performing robot simulations to reduce the risks and cost of hardware failure in reality. In most cases, robot simulator able to applied in robot exploration that involve exploration of locations that are hazardous or inaccessible to human. Moreover, with the ability in robot simulator, experiments can be assembled faster in simulator than in reality.

1.4. **OBJECTIVES**

The objectives of this project are:

- To design and develop a virtual prototype for robot simulator.
- To design and develop difference kind of environments in robot simulator.

1.5. **SCOPE OF STUDY**

The scopes of this project are:

- JmonkeyEngine software
- Blender software
- Java Programming
- System interface

CHAPTER 2

2. LITERATURE REVIEW

2.1.INTRODUCTION

In this chapter, we will discuss about the theories and overview of various fields that involve in this project. Those theories will enable us to understand the later discussion on this project.

2.2.ROBOT SIMULATION PACKAGE

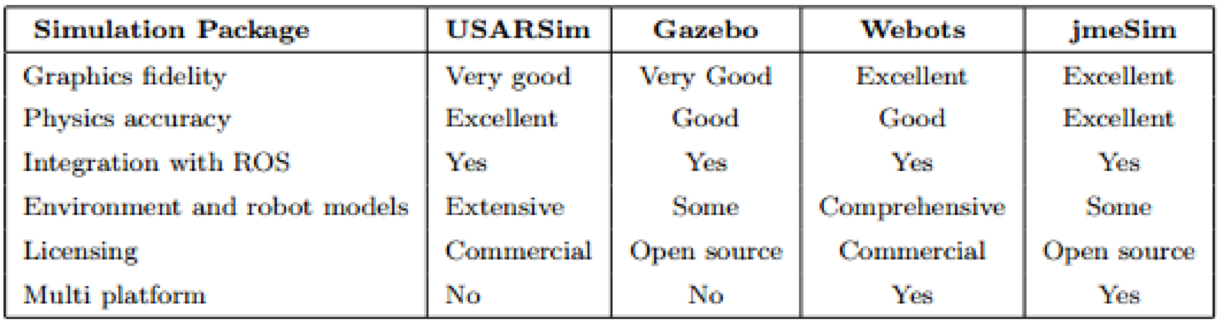
Simulation has become an important part in design or development for an autonomous robot. Most simulation comes with accurate physic simulation, multi platform, high quality rendering and open source code.

There are several reasons indicated that computer simulation is essentially tool for robotics field, particularly for mobile autonomous robot. Firstly, robots are fragile and hard to test and develop due to their complexity. The ability to test and develop in simulation as in real world environment can reduce the risks and cost of hardware failure. Moreover, there will be costly in develop a robot if the risk of failure is high. Simulations allow us to test the robot in software while waiting for the delivery of components.

In addition, with simulation, we got the ability to try our robot in difference kind of environment which not feasible to create in reality, such as fire and disaster scenarios. Finally, the most attraction for the software simulation was the ability to test run the robot as many time as we want. This process is needed for developing a complex algorithm and programming code.

There are plenty simulations package today. All of them got their advantages and disadvantages.

Table 2.1: Summary of the capabilities of some popular robotic simulation packages



Simulation Package	USARSim	Gazebo	Webots	jmeSim
Graphics fidelity	Very good	Very Good	Excellent	Excellent
Physics accuracy	Excellent	Good	Good	Excellent
Integration with ROS	Yes	Yes	Yes	Yes
Environment and robot models	Extensive	Some	Comprehensive	Some
Licensing	Commercial	Open source	Commercial	Open source
Multi platform	No	No	Yes	Yes

Source: Adam Haber (2013)

2.2.1. USARSim

USARSim is a commercially available game engine, originally based on the Unreal Game Engine2.0 high fidelity robot simulator. It was an open source software that we able to found many source via internet. Unreal 3.0 used hardware acceleration to produce state of the art graphics rendering, and the NVIDIA PhysX physics simulation engine. (Bastiaan, 2010).

The main disadvantages of USARSim are the fact that the Unreal game engine is commercially licensed. This means that user must purchase the game engine in order to use the simulation package inside. Without purchase the game engine, user can only access to the low level of the simulation package. The purchase fee is high and really a burden for a student.

2.2.2. Webots

Webots is a commercial robot simulation package that is the most developed and fully featured of those surveyed here (Michel, 1998). Webots is built on top of the Open Dynamics Engine (ODE) physics engine. It come with high rendering capabilities and more impressive. It can accept robot control code form difference type of programming. It include package of indoor and outdoor objects, and fully functional robot model with difference robotic platforms.

Webots is cross platform, so it can be installable on many operating systems like Mac, Linux and Windows. However, same as USARSim, webots is commercially licensed. Software licenses for a full version of Webots need around US$2300. Further, being commercially licensed means that any result that develop by Webot can’t be access by other user as long as they did not purchase the license. Therefore, the customizability of the software itself was limited.

2.3.GAME DEVELOPMENTENGINE

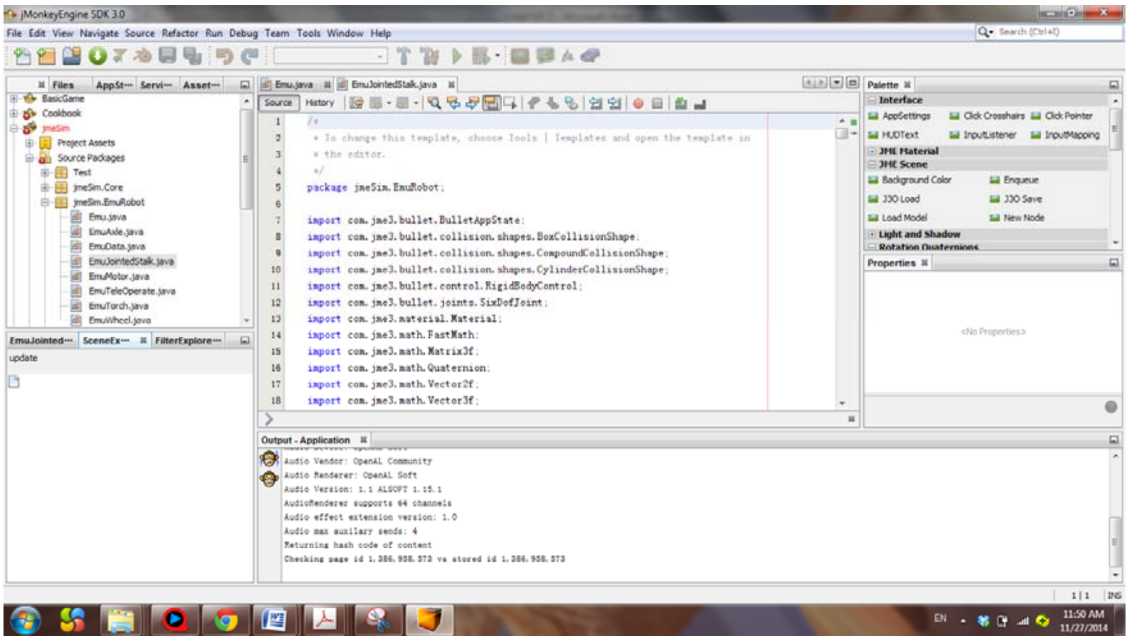
Game development engines can be classified base on 2 licensing format; commercial licensing and open source licensing engines. Some of popular commercial video game development engine are Unity, Havok and Unreal Engine. Many popular games were developed by these engines. As a request of their customer, they provide a wide support to use their environments. We able to use them in non commercial applications with low costs depend on the specific engines. Some of them offer the trial version for free for a certain period.

Another licensing format for game development engines is open source licensing. Some popular open source game development engines are Jmonkey, OGRE and Crystal Space. In the case of these engines, the professional support is poorer since it is open source. They probably support by a volunteer community and to learned lessons from other advanced users of the same environment.

The important issue arises of game development engine was the programming languages used for the development. They are many kind of programming languages used for this purpose, ie; C++, phyton and Javascript. Most of these engines able to run in different operating system like Windows, Mac and Linux. One of the most important features for game development engine was to be able of executing several processes concurrently. For this reason, most of the game development engines provide a simple mechanism to render model in the game scenario for programmed actions purpose regarding to different game interactions.

2.4.JMONKEY ENGINE SOFTWARE DEVELOPMENT KIT (SDK)

JMonkey Engine Software Development Kit (SDK) contains all the things I need to get started easily and efficiently in designing a virtual robot and environment. JMonkey Engine come with build in libraries, different type of plug-in, sample code, Javadoc, and plenty of virtual robot development utilities.



```
/*
 * To change this template, choose Tools | Templates and open the template in
 * the editor.
 */
package jmeSim.EmuRobot;

import com.jme3.bullet.BulletAppState;
import com.jme3.bullet.collision.shapes.BoxCollisionShape;
import com.jme3.bullet.collision.shapes.CompoundCollisionShape;
import com.jme3.bullet.collision.shapes.CylinderCollisionShape;
import com.jme3.bullet.control.RigidBodyControl;
import com.jme3.bullet.joints.SixDofJoint;
import com.jme3.material.Material;
import com.jme3.math.FastMath;
import com.jme3.math.Matrix3f;
import com.jme3.math.Quaternion;
import com.jme3.math.Vector2f;
import com.jme3.math.Vector3f;
```

```
Audio Vendor: OpenAL Community
Audio Renderer: OpenAL Soft
Audio Version: 1.1 ALSOFT 1.15.1
AudioRenderer supports 64 channels
Audio effect extension version: 1.0
Audio max auxilary sends: 4
Returning hash code of content
Checking page id 1,386,938,373 vs stored id 1,386,938,373
```

Figure 2.1: jMonkeyEngine(SDK) Platform

JMonkeyEngine (jME) is a game engine developed purposely for modern 3D development, as it applies shader technology extensively. JMonkey Engine is a purely java based and implement LWJGL as its default renderer. It fully supports OpenGL 2 through OpenGL4. Moreover, JMonkey Engine is an open source software and is free of cost for development a virtual robot and environment. It is used by some educational organization and commercial game studies as it readily integrated with an advance SDK.

Besides that, JMonkey Engine demonstrates the best ability to be integrated with libraries like ROS and OpenCV as all of them were written in java. It contains excellent graphics fidelity and physics accuracy. In addition, JMonkey Engine come with build in networking feature which enable the robot simulation run in multi computers and undergo multi robot simulation in more complex environment.

2.5.**MOBILE ROBOT SIMULATOR**

The mobile robot was the main element in the simulator. The virtual mobile robot is an autonomous entity with actuation capabilities and sensing. Those available in the real robot were represented by these capabilities. The robot simulation was executed independently with respect to other models in the scenario. Several mechanisms were needed for a robot to keep knowledge about itself and about the environment through its sensors.

In the server-client architecture, the robot indicates the data server and the client indicate the programs that can access to the information generated by the robot, its temporal evolution, and its interaction with other objects in the scenario. Update models in the robot environment by using critical methods both in computational power and time was important in order to be able to simulate the virtual mobile robot in real time. We able to set up and keep update the time interval for each individual model in the development platform used for this system. This is critical to be able to simulate the motion primitives of the mobile robot in real time.

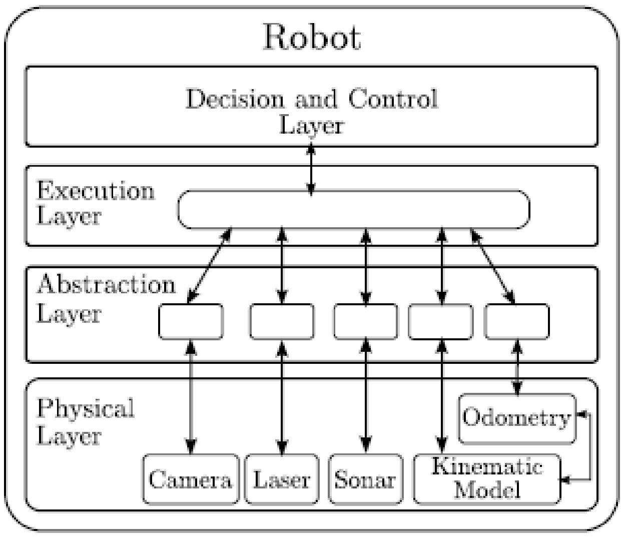


Figure 2.2: Robot Architecture

Source: Uriel H. Hernandez-Belmonte(2011)

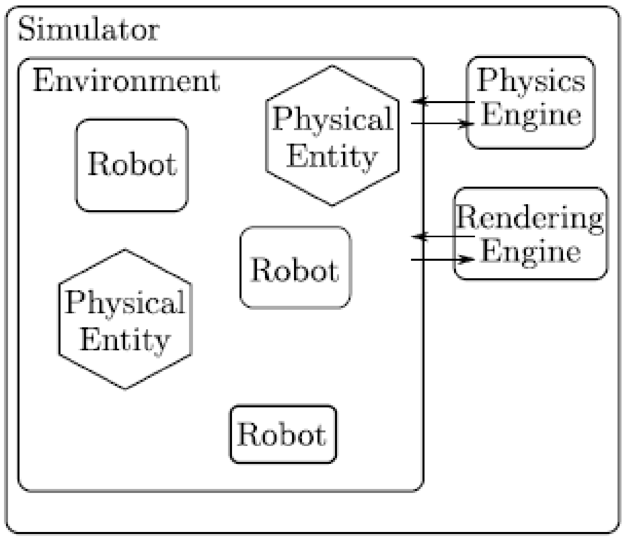


Figure 2.3: System architecture of the simulator

Source: Victor Ayala-Ramirez(2011)

CHAPTER 3

3. METHODOLOGY

3.1.INTRODUCTION

This chapter mainly discuss about the method used in this project. This project is fully software based. All the models and methods use in this project can be sum into one little world, programming. The details of the progress in those this project will be explained in this chapter.

3.2.PROCCESS FLOW

3.2.1. Overall

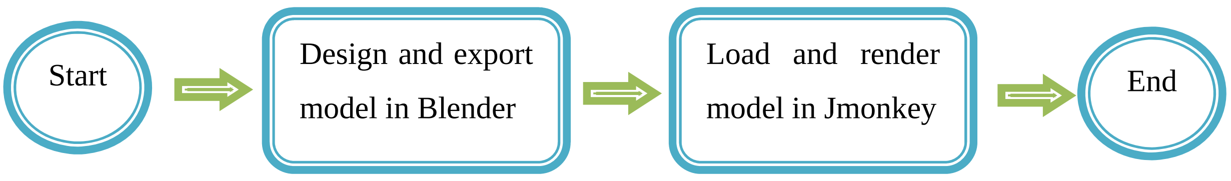
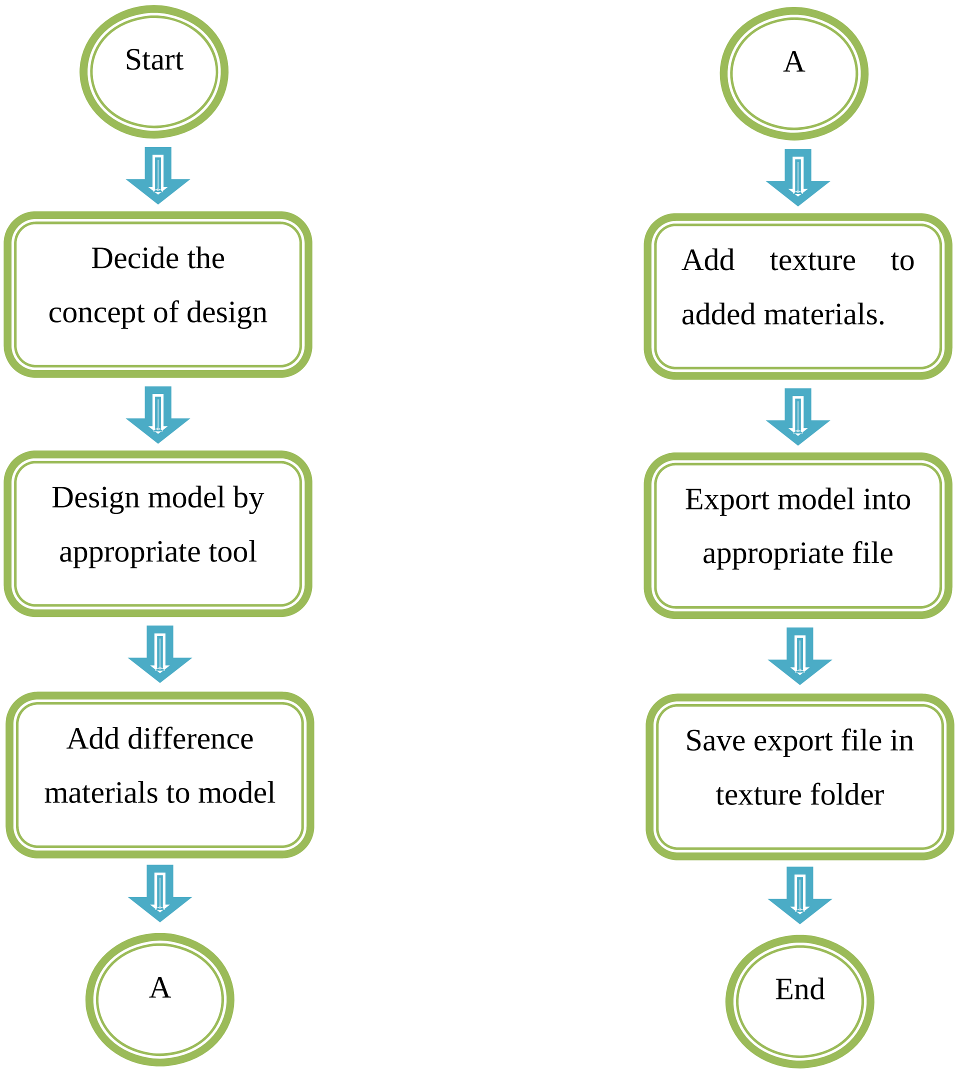

Start
Design and export model in Blender
Load and render model in Jmonkey
End

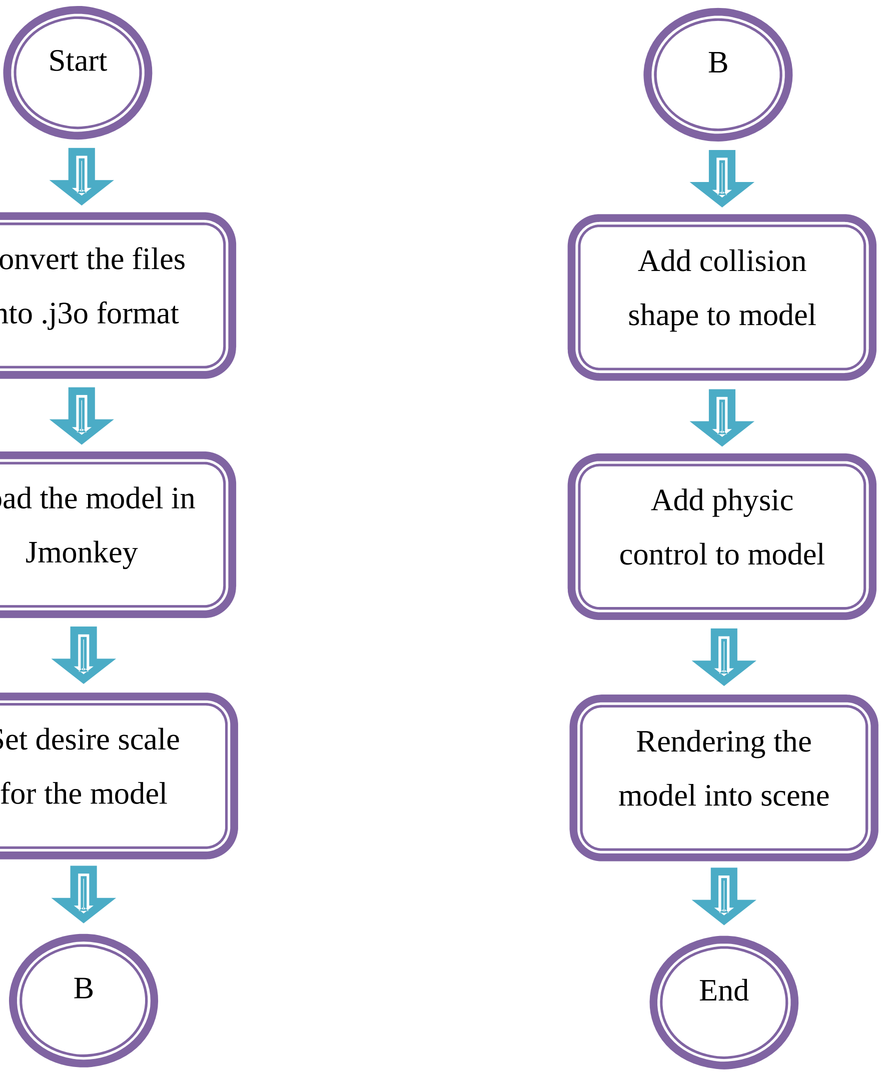
Figure 3.1: Overall project flow

3.2.2. Blender


Start
Decide the concept of design
Design model by appropriate tool
Add difference materials to model
A
A
Add texture to added materials.
Export model into appropriate file
Save export file in texture folder
End

3.2: Process flow in Blender

3.2.3. JmonkeyEngine


Start
Convert the files into .j3o format
Load the model in Jmonkey
Set desire scale for the model
B
B
Add collision shape to model
Add physic control to model
Rendering the model into scene
End

3.3: Process flow in JmonkeyEngine

3.3.**VIRTUAL ROBOT DESIGN**

Firstly, several design software such as webot, JmeSim, USARSim and Gazebo is surveyed base on it feature and capabilities. JmeSim is then to be chosen as the virtual robot development software in this project.

JMonkey Engine Software Development Kit (SDK) contains all the things I need to get started easily and efficiently in designing a virtual robot and environment. JMonkey Engine come with build in libraries, different type of plug-in, sample code, Javadoc, and plenty of virtual robot development utilities.

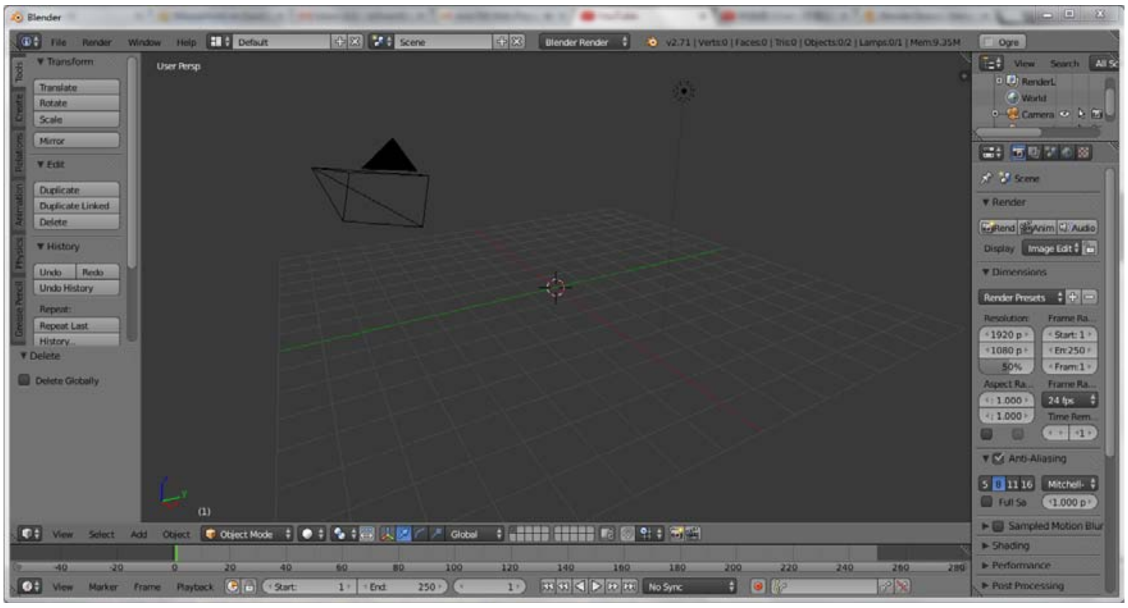
JmeSim is an robot simulator that build inside JmonkeyEngine. Same as other robot simulators, JmeSim is built base on the game engine, especially, the open source JMonkey Engine. Since the JmeSim and JMonkey Engine are purely java base, JmeSim is architecture neutral; it able to run on whichever operating system. JMonkey Engine has the ability which makes for fabulous graphical performance at real time frame rates on computer machine.

The physic simulation inside JmeSim is performed by jBullet where jBullet is a Java port of the Bullet Physics library. It makes work easier to test and analysis the physic simulation on virtual robot. Moreover, it provides fabulous stability, speed and accuracy which better than other commercially licensed physics engines.

In addition, JmeSim also offer a bundle of sensors which give data typically needed by an autonomous mobile robot. The JmeSim build in sensor are thermal camera, depth camera, sonar sensor, and laser range finder.

3.4.VIRTUAL ENVIRONMENT DESIGN

All the environments are design in Blender software. Blender is the free and open source 3D creation suite. It supports the entirety of the 3D pipeline—modeling, rigging, animation, simulation, rendering, compositing and motion tracking, even video editing and game creation. In this project, Blender software was used to design the model for the difference environments.



3.4: Blender software platform

Blender was used in this project because it able to export the file into Ogre3D format which are support by JmonkeyEngine. Blender exports the model into .mesh and .material files. Both files were converted into .j3o format inside JmonkeyEngine and load into the scene. JmonkeyEngine support the model file with format .j3o. Therefore, Blender was one of the design software that can export the design model into JmokeyEngine.

CHAPTER 4

4. RESULT AND DISCUSSION

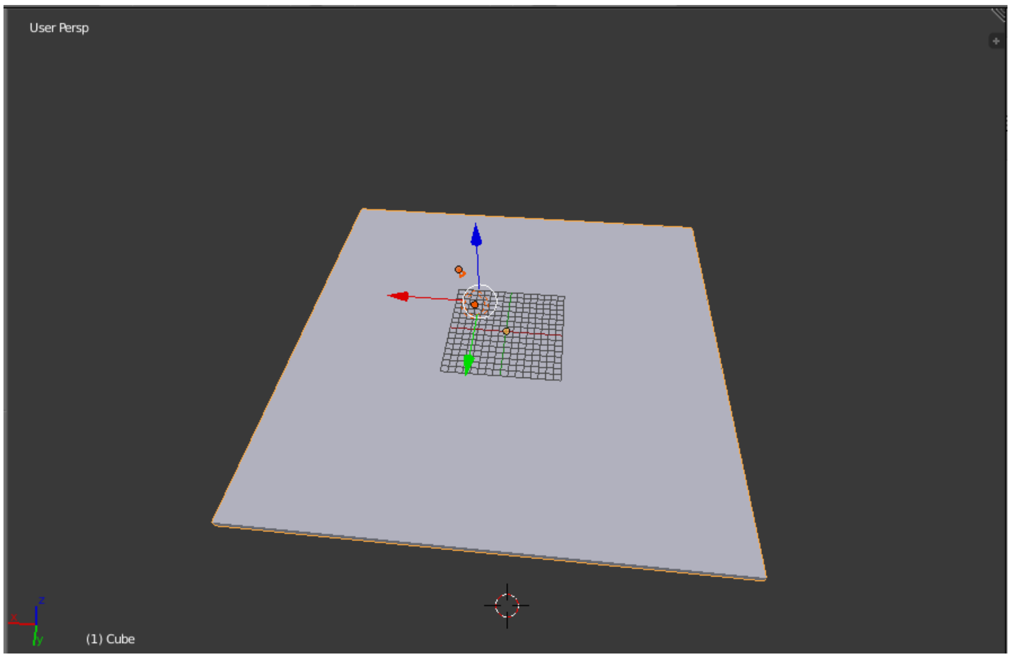
4.1. INTRODUCTION

This chapter mainly discuss about the outcome from the design, methodologies and software discussed in previous. The development of software will be discussed in detail in this chapter.

4.2. BLENDER SOFTWARE

Blender software was used to design the model and component in the indoor and outdoor environment in the robot simulation. The component that had been designed is floor, environments (indoor and outdoor), rubbish and tree.

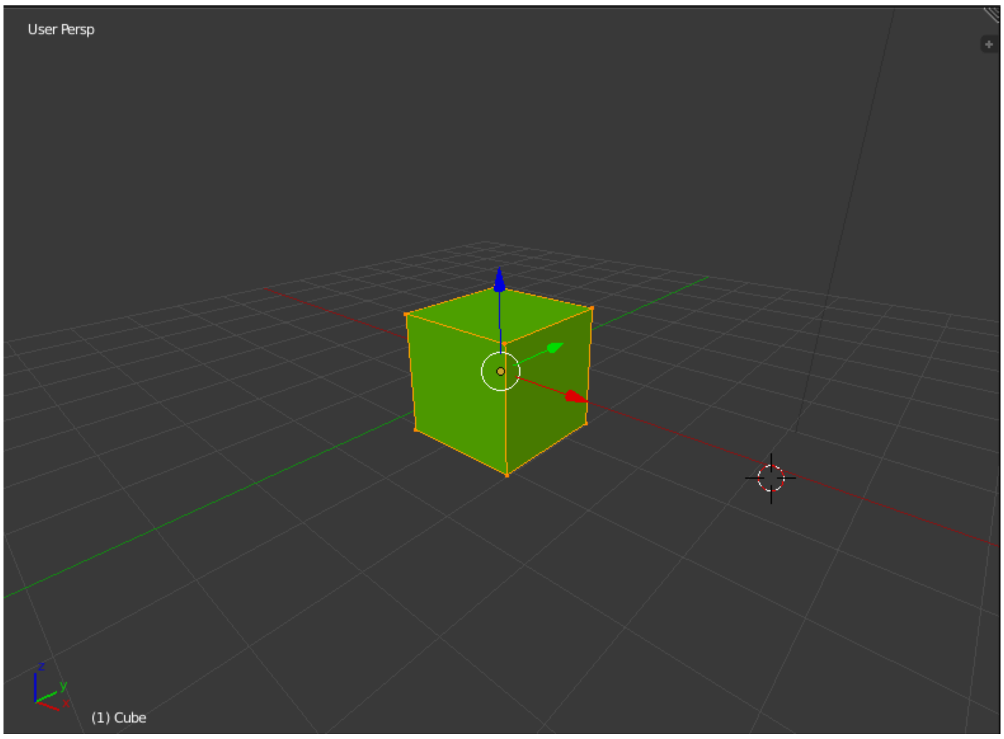
4.2.1. Floor



4.1: Floor in blender

In blender, a cube plane was extended and extruded to the parameter desired. Than the material and geometry was added to the cube mesh. After that, the cube plane was exported into 2 files; .material and .mesh. Those 2 files are than saved in the texture and was converted to .j3o file. The .j3o file is than saved in the model folder so that it can be loaded inside the robot simulation.

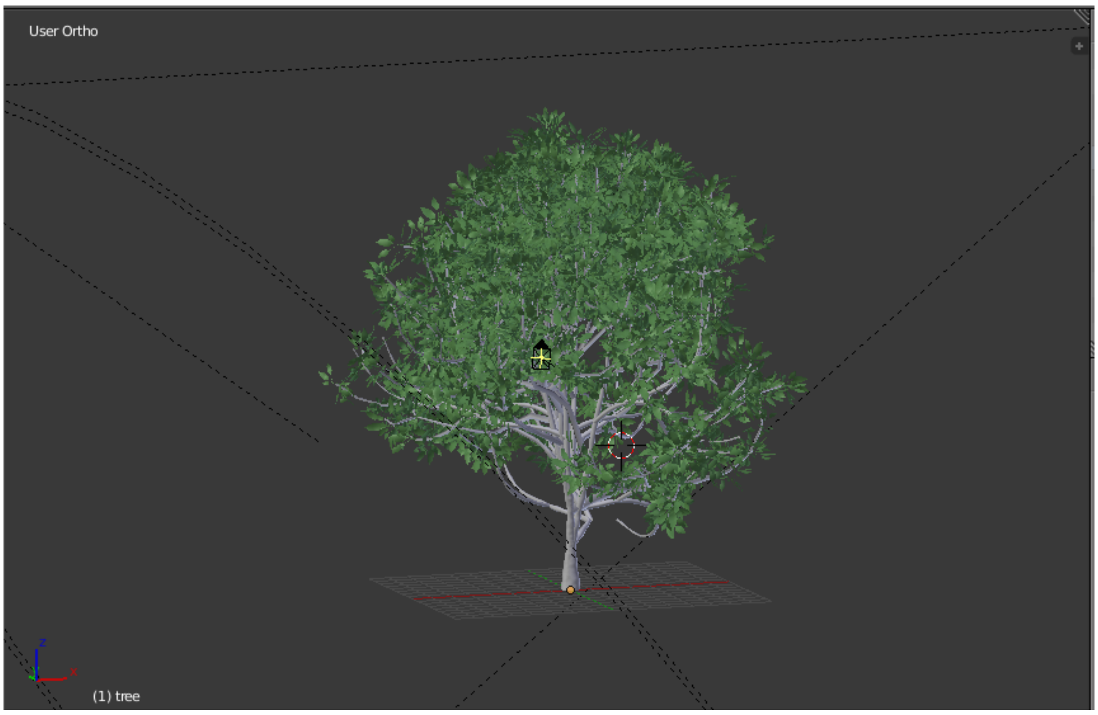
4.2.2. Virtual Obstacle



4.2: Virtual obstacle in Blender

A cube mesh was design in blender to indicate the rubbish inside the robot simulation. The cube mesh was chosen as the indicator for the rubbish because it won't roll if compare to circle when collide with the robot in robot simulation. Than the material and geometry was added to the cube mesh. After that, the cube mesh was exported into 2 files; .material and .mesh. Those 2 files are than saved in the texture and was converted to .j3o file. The .j3o file is than saved in the model folder so that it can be loaded inside the robot simulation.

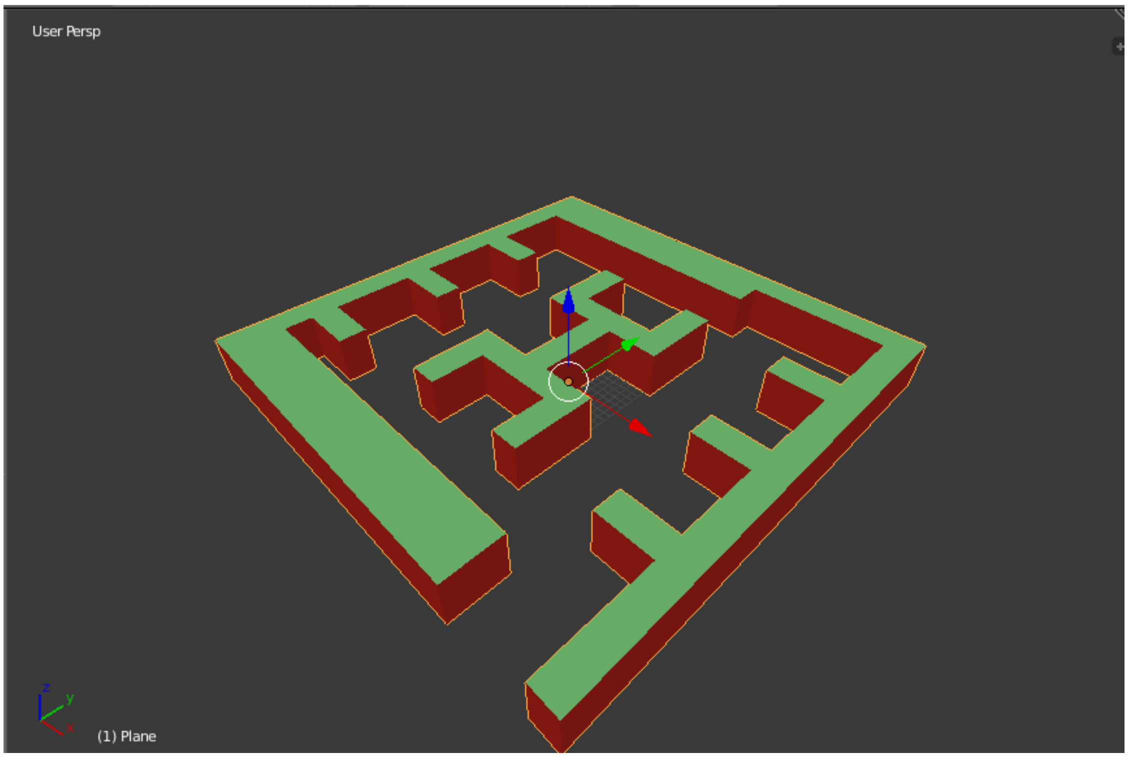
4.2.3. Virtual tree



4.3: Virtual tree in Blender

A cylinder mesh was used to design the tree model in blender while the circle plane was used to design the leave for the tree. Than the material and geometry was added to the tree model. After that, the tree model was exported into 2 files; .material and .mesh. Those 2 files are than saved in the texture and was converted to .j3o file. The .j3o file is than saved in the model folder so that it can be loaded inside the robot simulation.

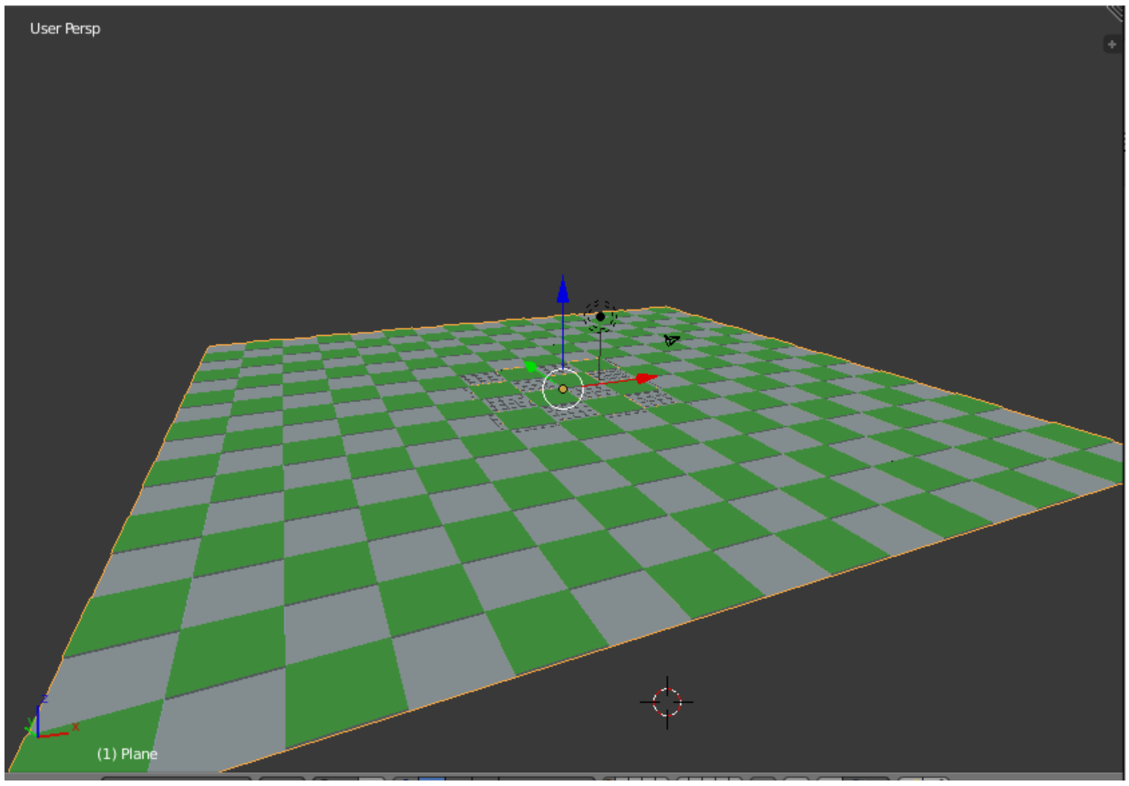
4.2.4. Virtual Environment (indoor)



4.4: Indoor environment in Blender

A cube plane was scaled to desired size and subdivided into scale 15. The shape of the environment is than design by selecting the small cubes accordingly and extrudes it by 5mm. The material and geometry was added to the environment model. After that, the environment model was exported into 2 files; .material and .mesh. Those 2 files are than saved in the texture and was converted to .j3o file. The .j3o file is than saved in the model folder so that it can be loaded inside the robot simulation.

4.2.5. Virtual Environment (outdoor)



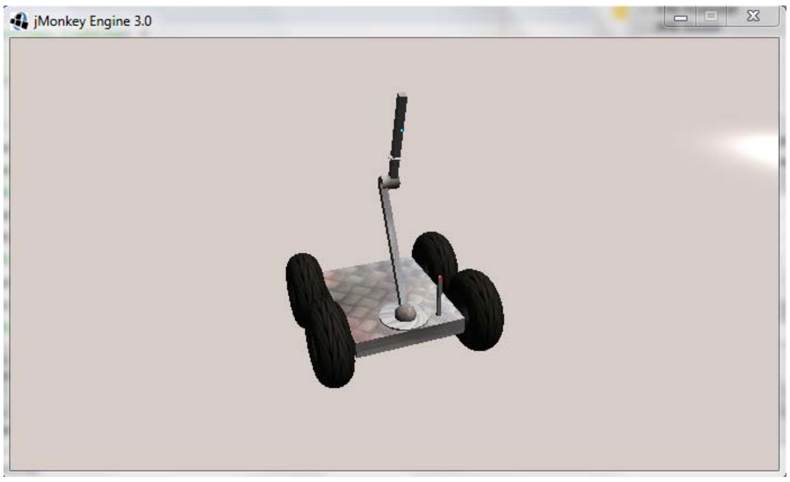
4.5: Outdoor environment in Blender

A cube plane is scaled and subdivided into scale 10. A step field is than designed by selecting the small cubes accordingly and extrude it to 0.5mm height. The uneven ground for outdoor environment was indicated by the step filed inside the robot simulation. The material and geometry was added to the environment model. After that, the environment model was exported into 2 files; .material and .mesh. Those 2 files are than saved in the texture and was converted to .j3o file. The .j3o file is than saved in the model folder so that it can be loaded inside the robot simulation.

4.3.**JmonkeyEngine**

JmonkeyEngineSDK was the main software used to develop the robot simulation. This is because the virtual robot and environment scene was developed in this software. All the model design in Blender was exported and loaded inside this software. In this software, all models was added and loaded to build a complete robot simulation. All the programming and 3D rendering happen here.

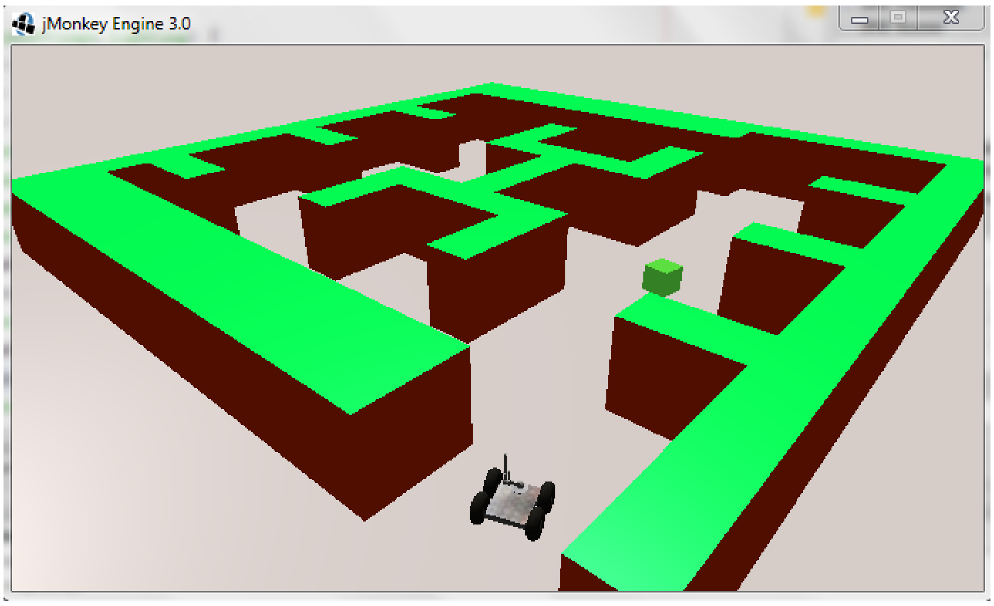
4.3.1. Virtual Robot



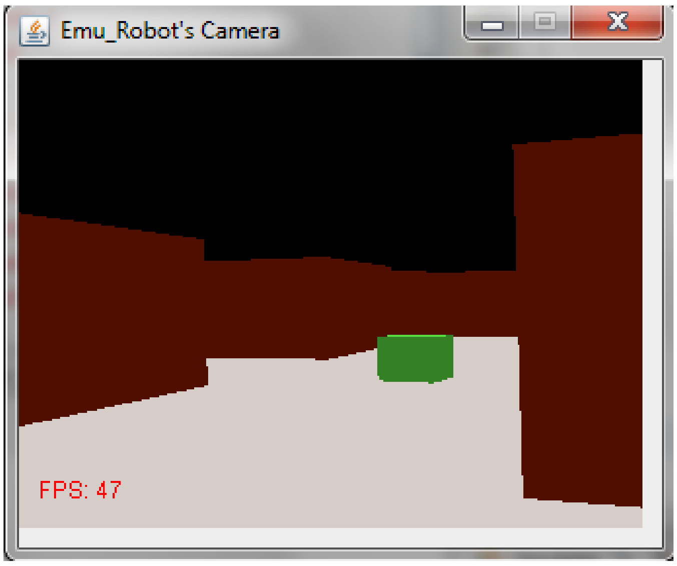
4.6: Virtual Robot in JmonkeyEngine

The design process for the virtual robot is all base on the Java programming in Jmonkey. There are 3 core object of virtual robot; chassis, wheel, and neck. Each of the core object is programmed part by part than at the end attached together to build the virtual robot. There was a camera sensor attached at the neck of the virtual robot and the scene was display at the small window frame during the simulation. Moreover, a torch light programming was attached on the chassis of the virtual robot. With the torch light, the virtual robot was able to run the simulation under dark environment.

4.3.2. Indoor Environment (Bright)

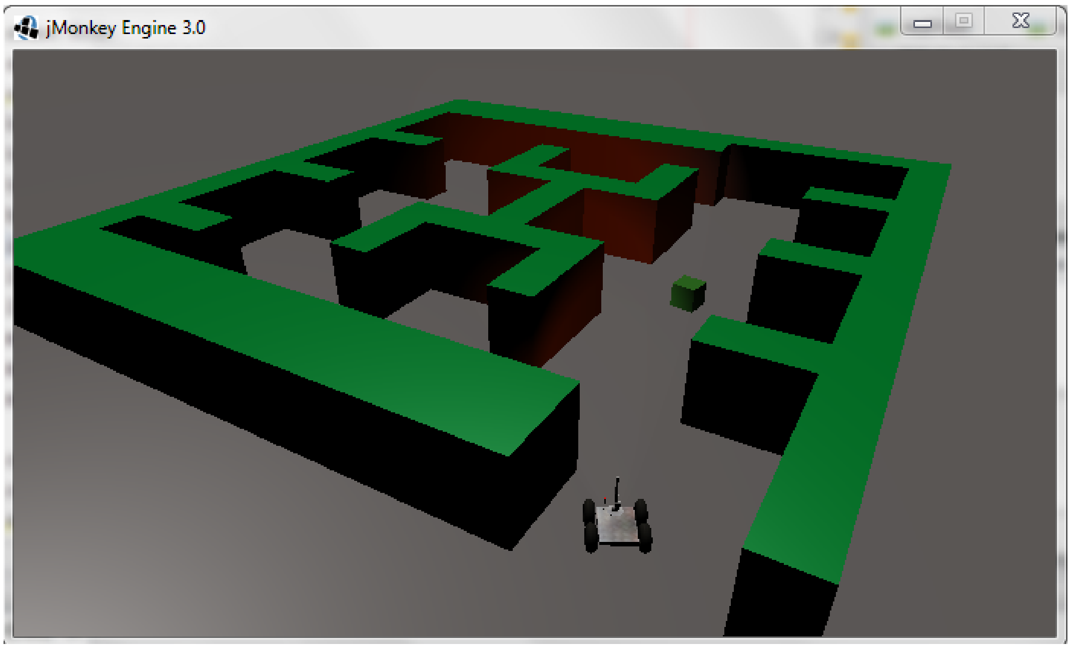


4.7: Indoor Environment (Bright) in JmonkeyEngine

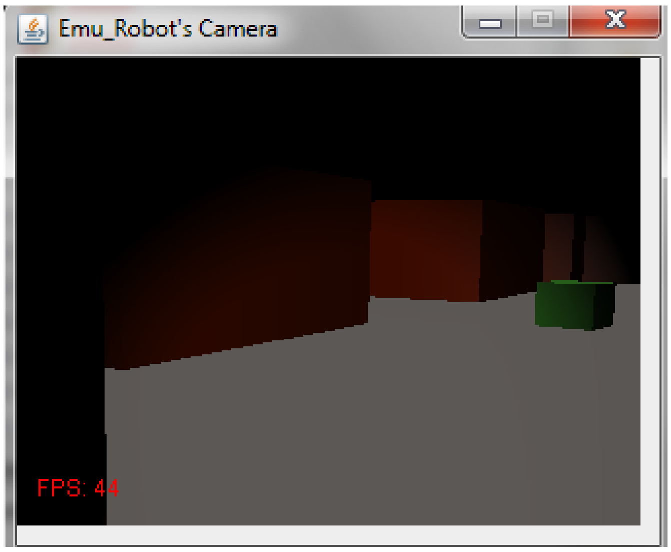


4.8: Camera display (Indoor Bright)

4.3.3. Indoor Environment (Dark)

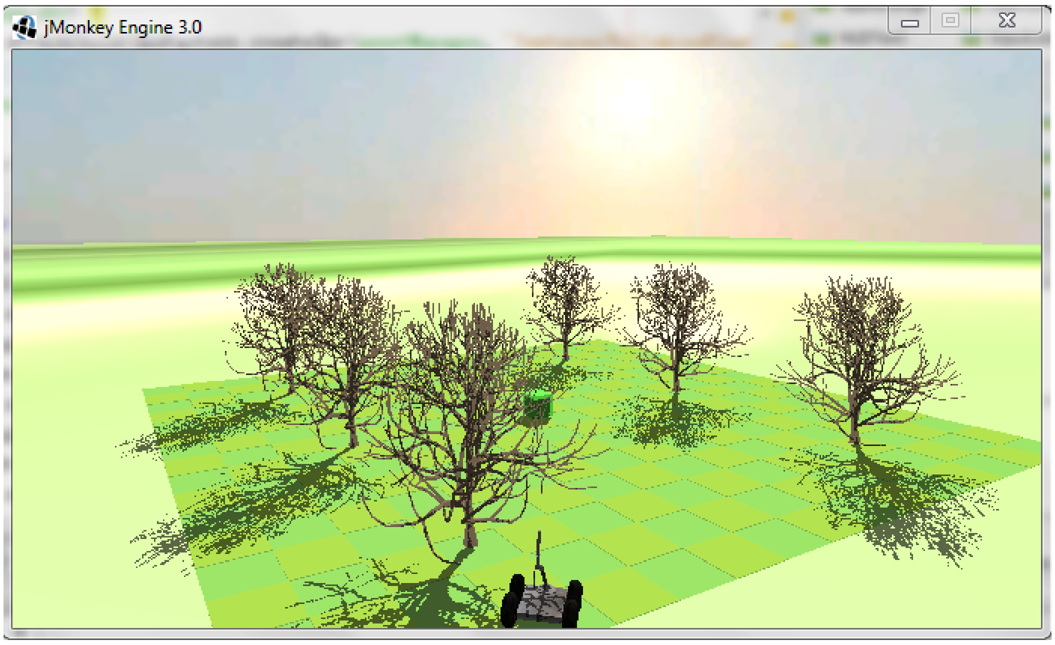


4.9: Indoor Environment (Dark) in JmonkeyEngine

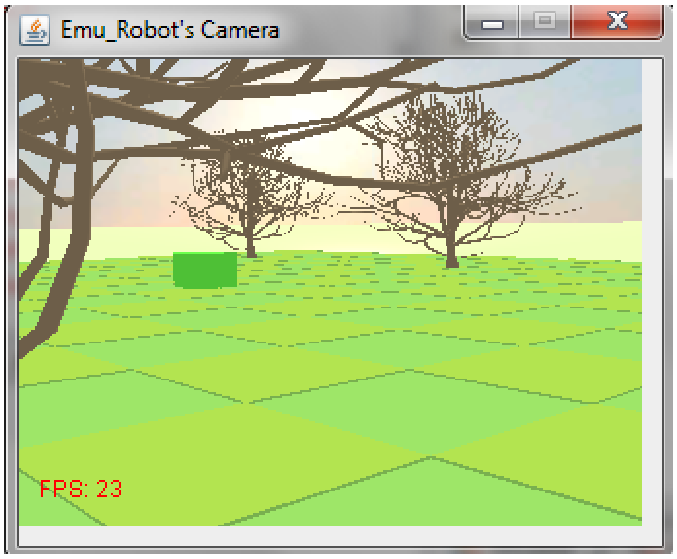


4.10: Camera display (Dark)

4.3.4. Outdoor Environment



4.11: Outdoor Environment in JmonkeyEngine.



4.12: Camera display (Outdoor)

4.4.DISCUSSION

In the indoor environment (Bright), the model that design in blender (environment, rubbish and floor) had been loaded. All models were added with the physic control. With the physic control, all the models contain their own mass and collision shape so that they able to collide with each other. The position for the virtual robot was set at the origin inside the world scene. The virtual robot can be navigated by reading the input from the keyboard. The ambient color was set to white color so that it looks bright for the environment. The brightness was scaled to 0.5 so that the environment won't look too bright.

For the indoor environment (Dark), the model loaded was same as the bright environment. The only difference was the ambient light for the scene. In the dark environment, the ambient light was set to black color and the brightness was set to 1 so that the whole scene becomes darker. The torch light specification was applied in the dark environment so that the camera sensor on virtual robot able to display the scene on the small window frame. For each time the robot simulation was run, the small window frame will keep update and display the scene that receive from the camera sensor on the virtual robot.

The difference model was loaded in outdoor environment. For the outdoor environment, the extra model of the step field and tree was loaded into the simulation. The step fields indicate that the uneven ground surface for outdoor environment. Six model trees were loaded in parallel. The sunlight was programmed in the outdoor environment. With the sun light, the shadow for all models was implemented. As usual, a small window frame will display and update the image from camera sensor on the virtual robot.

CHAPTER 5

5. CONCLUSION AND RECOMMENDATION

5.1.**INTRODUCTION**

This chapter was mainly discussed about the conclusion regarding for the overall project and also some recommendations for future improvement of this project.

5.2.**CONCLUSION**

In the conclusion, this project was successfully done as the objective of this project was achieved. The robot simulation is successfully run in JmonkeyEngine. Different environment was programmed with different scene inside simulation. The robot simulation was able to access the real time video from external webcam for navigation and object detection.

Moreover, the robot simulation was able to interface with arduino. Therefore, the virtual robot was able to synchronies with the real robot.

5.3.**RECOMMENDATION**

Some of the recommendation has been made according to this overall project:

- Other types of microcontroller can be used to replace Arduino such as raspberry pi, which is a user friendly with java programming to compare its result and effectiveness.

- Other types of software can be used to replace JmonkeyEngine to compare its result and effectiveness.

- The scope of project can be enlarged by developing the robot simulation in networking. By developing the networking, multi virtual robot from different system able to run the simulation in the same environment and scene.

- SLAM technique can be developed in the robot simulation so that a 2d vector map will produce at the end of the robot simulation.

- A gripper can be programmed and attach on the virtual robot so that the virtual robot able to pick the rubbish inside the robot simulation.

- Another alternative method in opencv can be replaced with the object detection by background subtraction method that implement in this project to detect rubbish and obstacle.

REFERENCES

- Uriel H. Hernandez-Belmonte, Victor Ayala-Ramirez and Raul E (2012). *A Mobile Robot Simulator Using a Game Development Engine.*

- Adam Haber, Matthew McGill, Claude Sammut (2012). *jmeSim: An Open Source, Multi Platform Robotics Simulator*.

- CatalinBuiu (2008) Design *and Development of a Waste Cleanup Service Robot.*

- B. Palaniappan, V. Ramya, T. Akilan (2013). *Embedded System for Robotic Arm Movement Control using Web Server and ZigBee Communication.*

- Pom Yuan Lam (BEng) (2010). *Graphical Embedded System Design Empowers Life Saving Spider Robots.*

- C˘at˘alinBuiu (2009). *Integrated System for Stereoscopic Cognitive Vision, Localization, Mapping, and Communication with a Mobile Service Robot.*

- Morgan Quigley_BrianGerkeyy, Ken Conley(2010). *ROS: an open-source Robot Operating System.*

- K.M.M. Rao*,Deputy Director,NRSA,Hyderabad-500 037 (2000). *OVERVIEW OF IMAGE PROCESSING.*

- SungHwanAhn· Jinwoo Choi · Nakju Lett Doh·Wan Kyun Chung (2008). *A practical approach for EKF-SLAM in an indoor environment:fusing ultrasonic sensors and stereo camera.*

- E. Prassler, E. Stroulia, M. Strobe1(2011). *Office Waste Cleanup: An Application For Service Robots.*

- Olivier Michel (2008). *WebotsTM: Professional Mobile Robot Simulation*
- Louis Hugues, Nicolas Bredeche (2007). *Simbad: An Autonomous Robot Simulation Package for Education and Research*
- Tim Laue, Kai Spiess, Thomas Röfer (2008). *SimRobot – A General Physical Robot Simulator and Its Application in RoboCup*
- Andreas Koestler, Thomas Bräunl (2009). *Mobile Robot Simulation with Realistic Error Models*
- Stephen J. Hartley, Drexel Univ., Philadelphia, PA (2005). *Concurrent programming: the Java programming language*
- Roland Hess (Book). *The Essential Blender: Guide to 3D Creation with the Open Source Suite Blender*
- Addison Wesley Professional (Book). *THE Java™ Programming Language, Fourth Edition*

APPENDICES A1

INDOOR ENVIRONMENT CODING

```
Spatial Maze = assetManager.loadModel("Models/lab_1.mesh.j3o");
Maze.setLocalScale(15f);
CollisionShapesceneShape =
CollisionShapeFactory.createMeshShape((Node) Maze);
mazeControl = new RigidBodyControl(sceneShape,0);
Maze.addControl(mazeControl);
mazeControl.setPhysicsLocation(n);
rootNode.attachChild(Maze);
bulletAppState.getPhysicsSpace().add(mazeControl);

Spatial cube = assetManager.loadModel("Models/Cube.mesh.j3o");
cube.setLocalScale(0.5f);
cubeControl = new RigidBodyControl(1);
cube.addControl(cubeControl);
cubeControl.setPhysicsLocation(new Vector3f(1f,1f,20f));
rootNode.attachChild(cube);
bulletAppState.getPhysicsSpace().add(cubeControl);

Spatial Base = assetManager.loadModel("Models/base.mesh.j3o");
Base.setLocalScale(10f);
floorControl = new RigidBodyControl(0);
Base.addControl(floorControl);
floorControl.setPhysicsLocation(new Vector3f(10f,25f,0f));
rootNode.attachChild(Base);
bulletAppState.getPhysicsSpace().add(floorControl);
```

APPENDICES A2

OUTDOOR ENVIRONMENT CODING

```
l = new DirectionalLight();
l.setDirection(new Vector3f(-1, -1, 1));
rootNode.addLight(l);
AmbientLight al = new AmbientLight();
al.setColor(ColorRGBA.White.mult(1.5f));
rootNode.addLight(al);
Spatial sky =
SkyFactory.createSky(assetManager, "Scenes/FullskiesSunset0068.dds", false);
sky.setLocalScale(350);
rootNode.attachChild(sky);

Spatial Maze = assetManager.loadModel("Models/Step1.mesh.j3o");
Maze.setShadowMode(ShadowMode.Receive);
Maze.setLocalScale(15f);
mazeControl = new RigidBodyControl(0);
Maze.addControl(mazeControl);
mazeControl.setPhysicsLocation(n);
rootNode.attachChild(Maze);
bulletAppState.getPhysicsSpace().add(mazeControl);
```

APPENDICES A3

SETUP TREE CODING

```
spatial = assetManager.loadModel("Models/tree.mesh.j3o");
spatial.setLocalScale(0.3f);
control = new RigidBodyControl(0);
spatial.addControl(control);
control.setPhysicsLocation(new Vector3f(loc));
spatial.setShadowMode(ShadowMode.CastAndReceive);
rootNode.attachChild(spatial);
bulletAppState.getPhysicsSpace().add(control);
```

APPENDICES A4

SETUP SHADOW CODING

```
dlsr =
newDirectionalLightShadowRenderer(assetManager, SHADOWMAP_SIZE, 3);
dlsr.setLight(l);
dlsr.setLambda(0.55f);
dlsr.setShadowIntensity(0.6);
dlsr.setEdgeFilteringMode(EdgeFilteringMode.Nearest);
viewPort.addProcessor(dlsr);

dlsf = new DirectionalLightShadowFilter(assetManager, SHADOWMAP_SIZE, 3);
dlsf.setLight(l);
dlsf.setLambda(0.55f);
dlsf.setShadowIntensity(0.6f);
dlsf.setEdgeFilteringMode(EdgeFilteringMode.Nearest);
dlsf.setEnabled(false);

FilterPostProcessorfpp = new FilterPostProcessor(assetManager);
fpp.addFilter(dlsf);

viewPort.addProcessor(fpp);
```

APPENDICES A5

CAMERE SENSOR CODING

```
public class CameraSensor extends SensorDevice {

protected Vector3f prev_loc;
   protectedint c = 0;

publicCameraSensor(Robot rob) {
super(rob);
name = (robot.getName() + "Cam");
createViewPort();
createDisplayFrame(robot.getName() + "'s Camera", 25, dim.height / 2 - ( 25 + 3 *
RobotCam.getHeight() / 2));
setupOffscreenView();
}

publicCameraSensor(Robot rob, Vector3f offset) {

super(rob, offset);
name = (robot.getName() + "Cam");
createViewPort();
createDisplayFrame(robot.getName() + "'s Camera", 25, dim.height / 2 - ( 25 + 3 *
RobotCam.getHeight() / 2));
setupOffscreenView();
   }
```

```
publicCameraSensor(Robot rob, Vector3f actuatorLocation, Vector3f offset)
   {
super(rob, actuatorLocation, offset);
name = (robot.getName() + "Cam");
createViewPort();
createDisplayFrame(robot.getName() + "'s Camera", 25, dim.height / 2 - ( 25 + 3 *
RobotCam.getHeight() / 2));
setupOffscreenView();
   }
```

APPENDICES B1

FINAL YEAR PROJECT 1 GANTT CHART

No	Task	Status	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Discuss topic with supervisor	P	🟦													
		A	🟩													
2	Brainstroming	P		🟦	🟦											
		A		🟩	🟩											
3	Seaching for information	P			🟦	🟦	🟦									
		A			🟩	🟩	🟩									
4	Survey for component	P						🟦	🟦							
		A						🟩	🟩							
5	Complete chapter 1	P								🟦						
		A								🟩						
6	Discuss information	P									🟦	🟦				
		A									🟩	🟩				
7	listing materials	P											🟦			
		A											🟩			
8	Complete Chapter 2	P												🟦		
		A												🟩		
9	Presenting fyp1	P													🟦	🟦
		A													🟩	🟩

P	Plan	on plan	🟦
A	Action	done	🟩
		pending	🟥

APPENDICES B2

FINAL YEAR PROJECT 2 GANTT CHART

			Week													
No	Task	Status	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Study Jmonkey	P	🟦	🟦	🟦	🟦	🟦	🟦	🟦							
		A	🟩	🟩	🟩	🟩	🟩	🟩	🟩							
2	Study blender	P			🟦	🟦	🟦	🟦	🟦	🟦	🟦					
		A			🟩	🟩	🟩	🟩	🟩	🟩	🟩					
3	Design model in blender	P							🟦	🟦	🟦	🟦				
		A							🟩	🟩	🟩	🟩				
4	Design scene in Jmonkey	P									🟦	🟦	🟦			
		A									🟩	🟩	🟩			
5	Complete chapter 4	P							🟦							
		A							🟩							
6	Design difference environment	P					🟦	🟦	🟦	🟦	🟦	🟦				
		A					🟩	🟩	🟩	🟩	🟩	🟩				
7	Complete chapter 5	P											🟦			
		A											🟩			
8	Final checking	P													🟦	
		A													🟩	
9	Presenting fyp2	P														🟦
		A														🟩

P	Plan	on plan	🟦
A	Action	done	🟩
		pending	🟥