



**GRAPHICAL AND MULTIMEDIA**

**SYSTEMS**

Robot Arm Project Report

Work Group #1

Adviser: ***Dr. Enric Martí Gòdia***

MASTER IN COMPUTER ENGINEERING

Group members:

**Md Ashraful Alam Milton**

**Negin Moradi**

**Marcelina Izworska**

**Pouya Ahmadmonfared**

**Kian Seif**

June 2015

**Outline**

**1. Introduction**

1.1 Background

1.2 Goal & Aim of Project

1.3 Project Selection

1.4 Robot Arm : Project Topic Details

**2 Robot Arm Design**

2.1 Research on Robot Arm

2.2 Inverse Kinematics

2.3 Initial Design

2.4 Final Design

**3. Development**

3.1 Platform & Environment of Development

3.2.1 Technologies: OpenGL

3.2.2 Technologies: QT

3.2.3 Eigen

3.3 Class Diagram

3.4 Application Structure

3.5 Robot Arm: Game Strategy

**4. Results**

4.1 Demo & Screen shots

4.2 Test

**5. Conclusion and Future works**

**6. Bibliography**

**1. Introduction**

**1.1 Background:**

Computer Graphics is a revolutionary technology that changes the aspect of how we interact with computer. From Animation movie to Flight simulator, computer graphics is a very practical and versatile technology. In our Graphical and Multimedia Application course we have been taught the computer graphics and its application by PBL(Project Based Learning ) methodology. In this course, we have been doing to do a project to follow project based learning paradigm.

**1.2 Goal & Aim of Project**

Computer graphics theories can be vast and extensive. Computer graphics is very hard to remind this theories in mind without practical project. Also, practical project gives a clear and concise view of about what theories have learnt. Our project”s main goals are same as above and also introduce ourselves in the latest graphics stack to implement the knowledge we have learnt.

**1.3 Project Selection**

In the beginning of the semester, we had some basic theory classes on computer graphics including mathematics, coordinate system, opengl. We started learning computer graphics before starting the project. With the help of Dr. Enric Martí Gòdia, we built a steady progress in computer graphics. He proposed us several project like Lights Simulation, Robot Arm, a Computer Game (3D Tetris), etc. All these projects were very interesting to us.

Our group started to discuss about these topics to find the best option that was appropriate for us and for our skills. After discussing, we found out that 3D Tetris and Robot Arm are good options but we should choose one of them. Some members had an interest in Robot Arm and other ones liked to choose Computer Game. For considering 3D Tetris we thought about creating 3D elements with shapes different then in usual one. Finally, we voted to choose one of them so the result was Robot Arm.

**1.4 Robot Arm : Project Topic Details**

We should design a model a Robot Arm and then implement it with appropriate software.

For instance, the robotics company “Happy Robot” hires you to implement an application that represents a robot arm with more than 2 joints with a clamp and hand. Keep in mind restrictions in the arm motion. We should represent movements of the robot with both strategies movement: In kinematics from key movements (keyframes) and inverse kinematics, giving the final position where we want to place the clamp.

At first, in result of our brainstorming, we created idea to use 3d model and move the using computer graphics techniques. Additionally add some other features and maybe create kind of game that two users can play each other in using robot arm to collect 3d objects.

This Robot Arm should have several segments which have their own constraint for movement. Showing the sample for previous students projects help us to have more idea about the robot arm.

For gaining these purposes we should find software for modeling and designing that, so we started to research through the internet and related books for finding some resources that can help us to make whole project.

**2 Robot Arm Design**

**2.1 Research on Robot Arm**

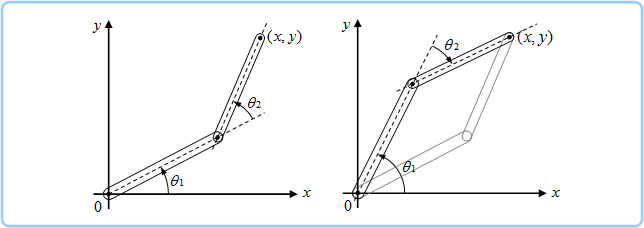
Absolutely, one of the best resource in each research is the internet, thus we tried to search through the internet to find some tutorials about useful software for modeling and also for implementing the model loading it.

In addition, Dr. Enric introduced us OpenGL as a powerful, professional-level system for 3D computer graphics programming. During the semester, Dr. Enric taught us more about OpenGL and its features. So, we’ve learnt to use OpenGL for 3D computer graphics programming. Also, we could find lots of tutorials about modeling software like 3D Max. As a brief definition Autodesk 3ds Max, formerly 3D Studio Max, is a professional 3D computer graphics program for making 3D animations, models, games and images. It is developed and produced by Autodesk Media and Entertainment. It has modeling capabilities, a flexible plugin architecture and can be used on the Microsoft Windows platform. It is frequently used by video game developers, many TV commercial studios and architectural visualization studios. It is also used for movie effects and movie pre-visualization.

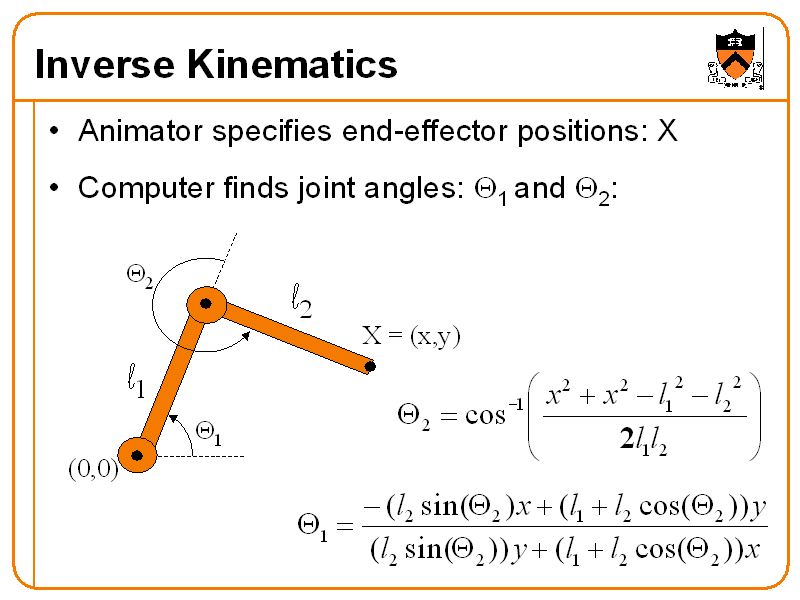
In addition, we could find some other software for instance Mudbox, Zbrush, Max, Modo.

**2.2 Inverse Kinematics**

Inverse kinematics refers to the use of the kinematics equations of a robot to determine the joint parameters that provide a desired position of the end-effector. Specification of the movement of a robot so that its end-effector achieves a desired task is known as motion planning. Inverse kinematics transforms the motion plan into joint actuator trajectories for the robot.



**Fig. Inverse kinematics**



The movement of a kinematic chain whether it is a robot or an animated character is modeled by the kinematics equations of the chain. These equations define the configuration of the chain in terms of its joint parameters. Forward kinematics uses the joint parameters to compute the configuration of the chain, and inverse kinematics reverses this calculation to determine the joint parameters that achieves a desired configuration.

For example, inverse kinematics formulas allow calculation of the joint parameters that position a robot arm to pick up a part. Similar formulas

determine the positions of the skeleton of an animated character that is to move in a particular way.

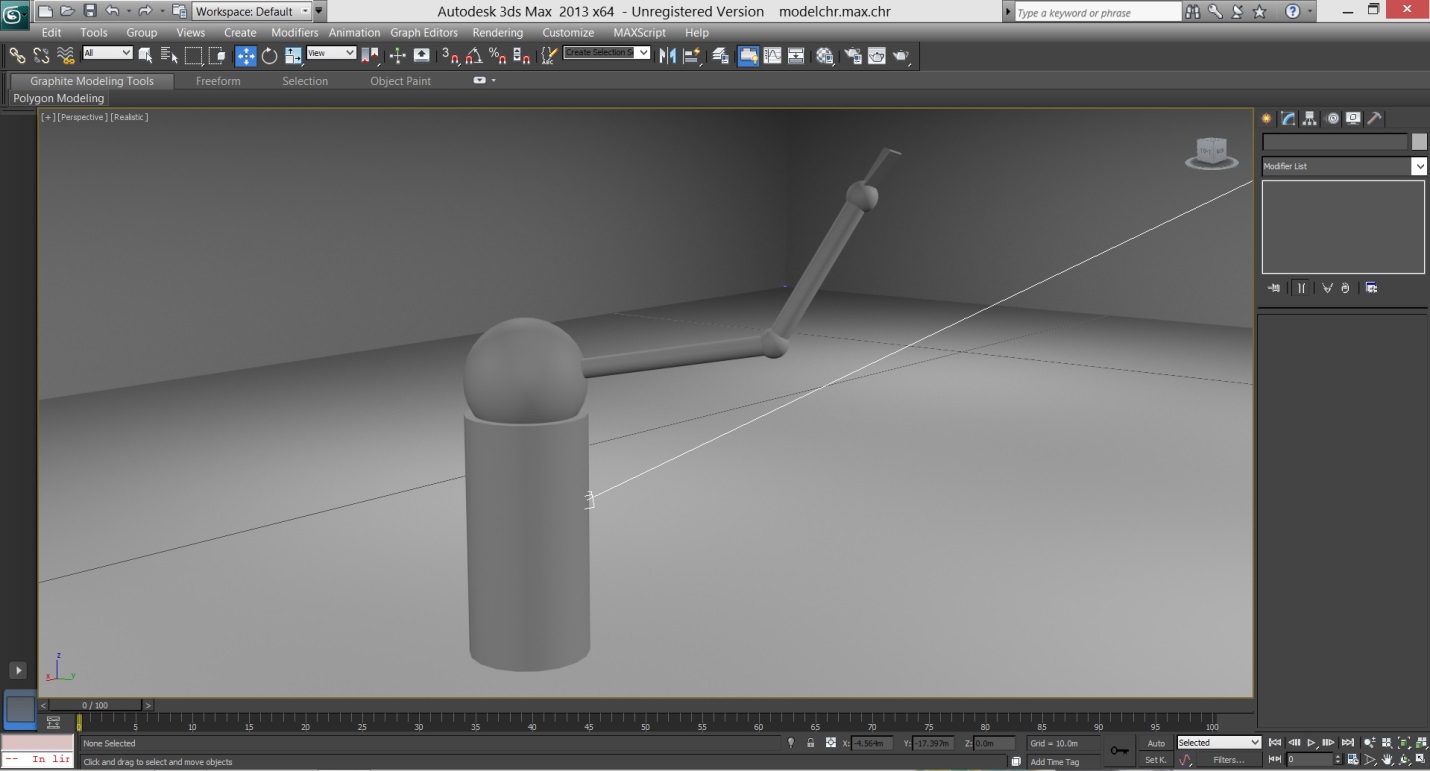
**2.3 Initial Design**

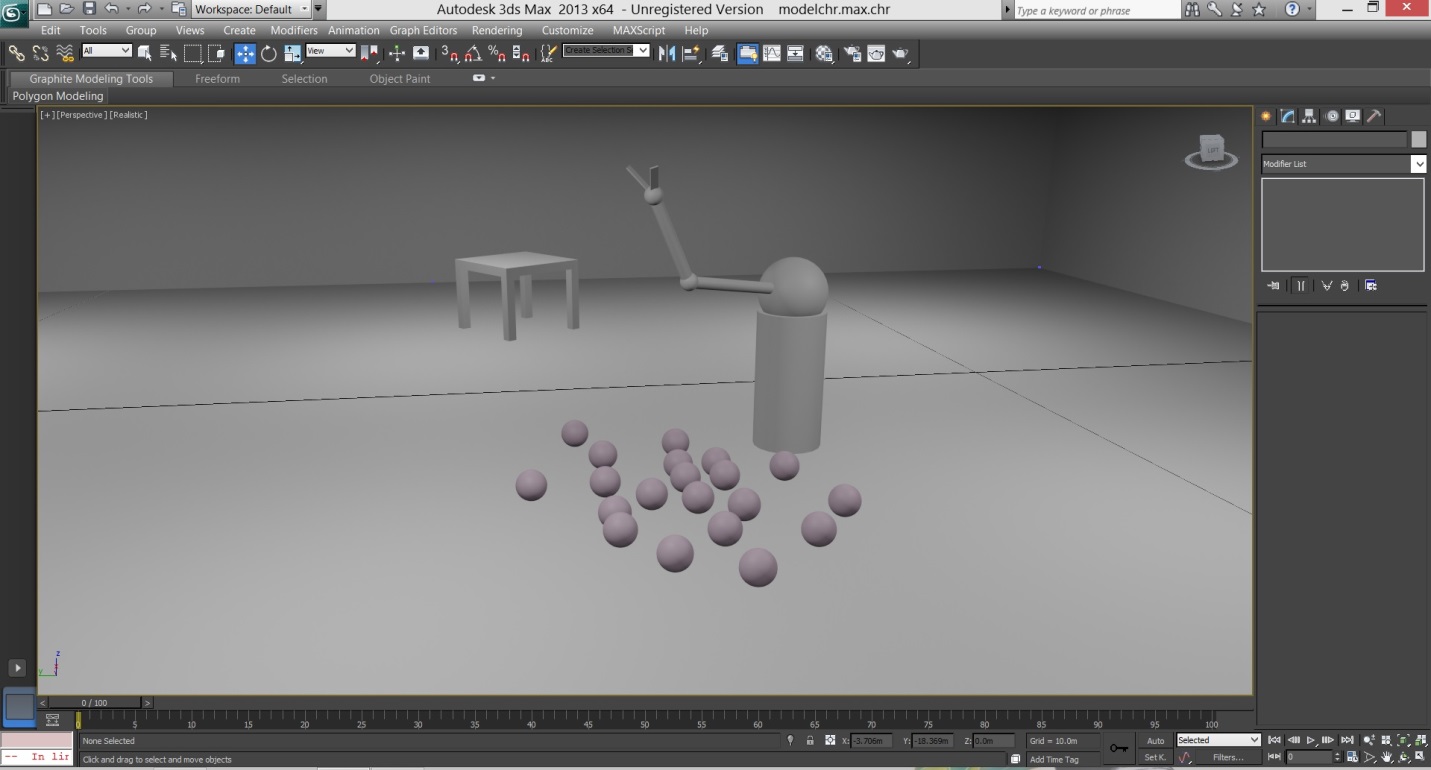
We tried to design an Arm like a Robot Arm which was our purpose for this part of our project.

**Functions:**

In continue we decided a robot arm with more than 2 joints with a clamp and hand also Keep in mind restrictions in the arm motion but we decided to model and represent the arm and hand as much as look like to real hand. The last but not final strategy is to move the object (which has specific position) from one position to another position we represented movements of the robot arm and also objects with progressive and inverse kinematics. It had a fixed base with three parts which each one has an individual movement. Also, we gave an idea to make this project as a kind of game.

Following pictures show our first model by 3D Max:





**2.4 Final Design**

****

due to our lack of knowledge and expertise, it was very hard for use to implement a complex model in our robot arm project. We decided to build the robot arm with n-segment of triangles. We used 6 segments

.

**3 Development**

**3.1 Platform & Environment of Development**

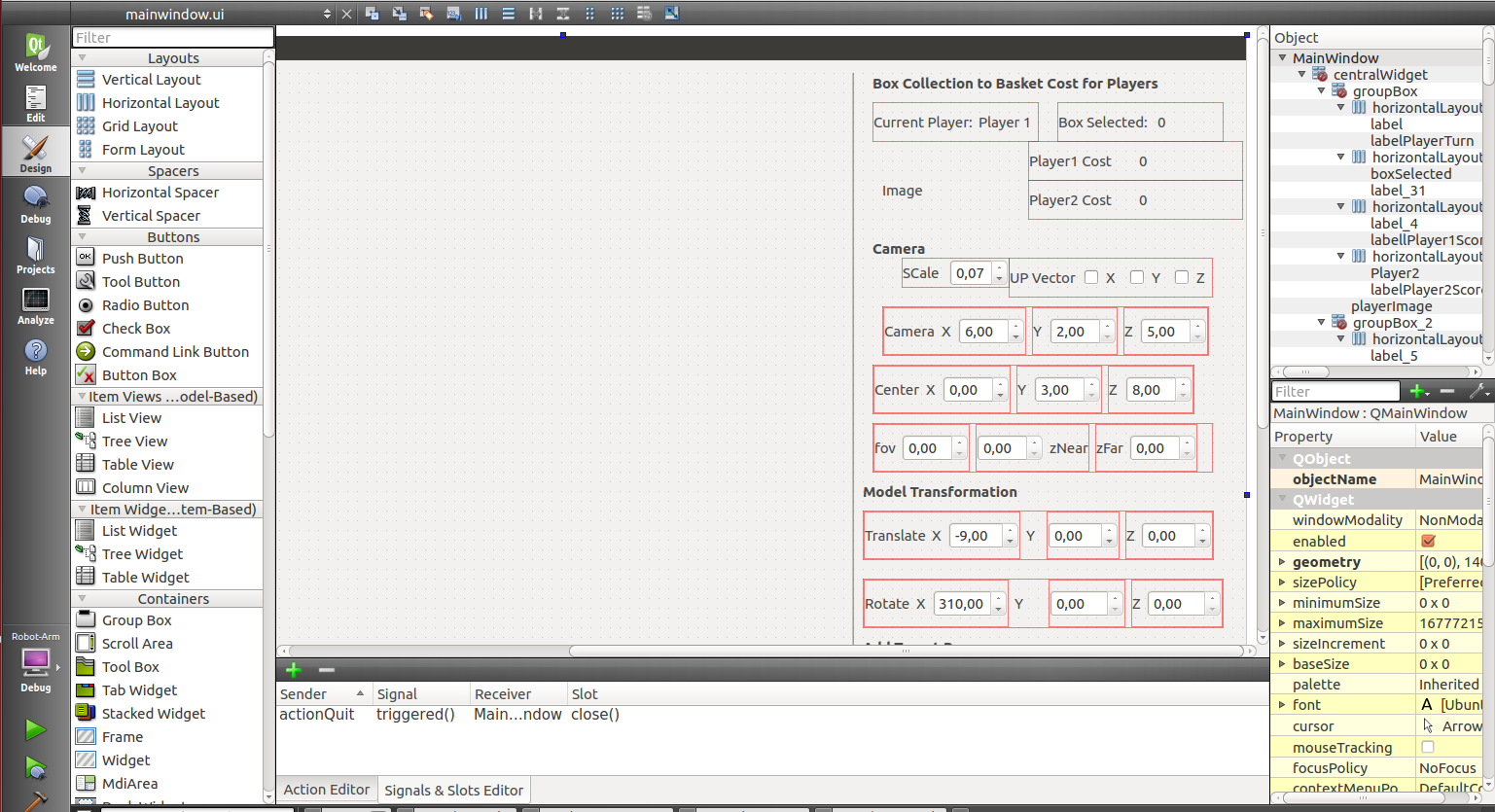
We worked on the QT Creator, so we implemented it in the environment of Linux but, we met some errors in this way, so we tried to remove them. We worked with OpenGL for implementing and 3D Max to design our basic model.

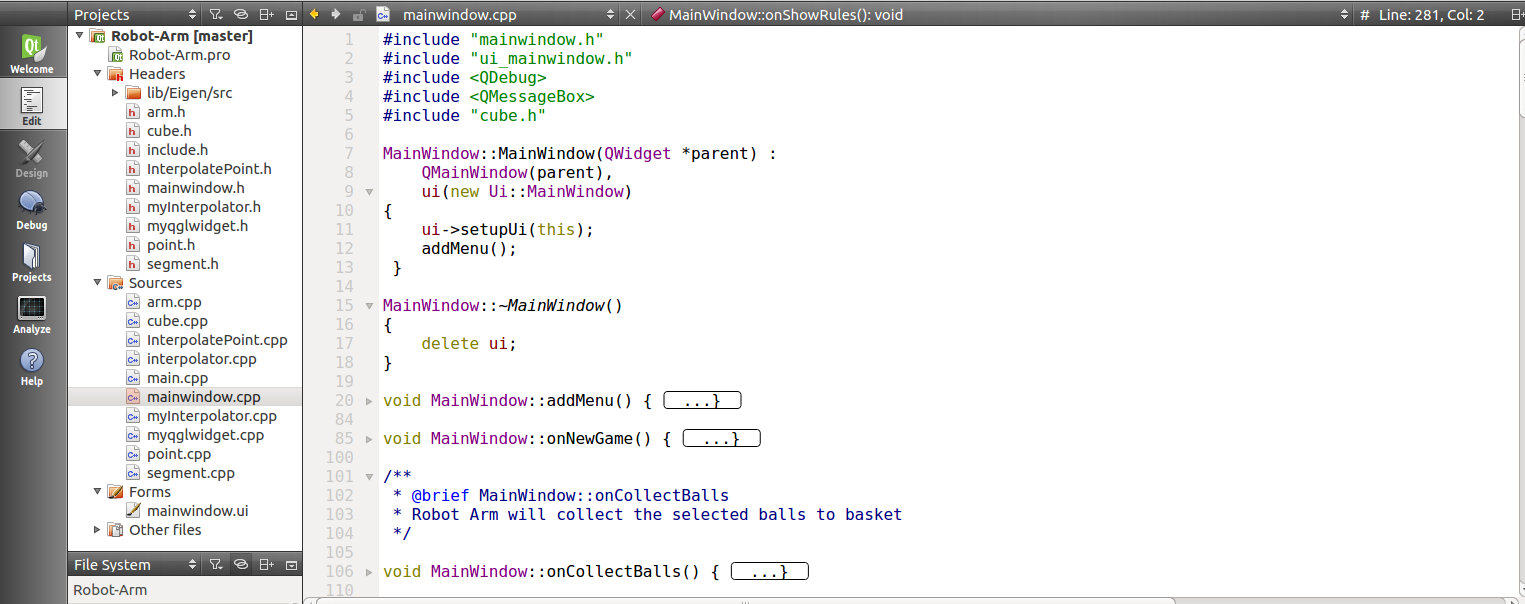
Consequently, we started to work with the “CGEnv.” and we knew that it is compatible with the “OpenGL v.1.1”!

On the other hand, we created a project named “Robotarm” in the QT creator by C++. We tried to add “.obj” file inside it.

In contrast, it is necessary to make a platform for implementation, therefore we set up the OpenGL project to load and move the objects of model intothe OpenGLand we built basis for importing the model into the project easily.

we used ubuntu operating system. As we describe a little in Introduction finally we use QT and C++ for





**3.2.1 Technologies: OpenGL**

OpenGL has its origins in the earlier GL (“Graphics Library”) system which was invented by Silicon Graphics Inc. as the means for programming their high-performance specialized graphics workstations. As time went on, people became interested in porting GL to other kinds of machine, and in 1992 a variation of GL – called OpenGL – was announced. Unlike GL, OpenGL was specifically designed to be platform independent, so it would work across a whole range of computer hardware – not just Silicon Graphics machines. The combination of OpenGL’s power and portability led to its rapid acceptance as a standard for computer graphics programming.

Firstly, we couldn’t implement and load our model, unfortunately. So, we got some helps from Dr.Enric, therefore he gave us a prepared sample to gain this project.

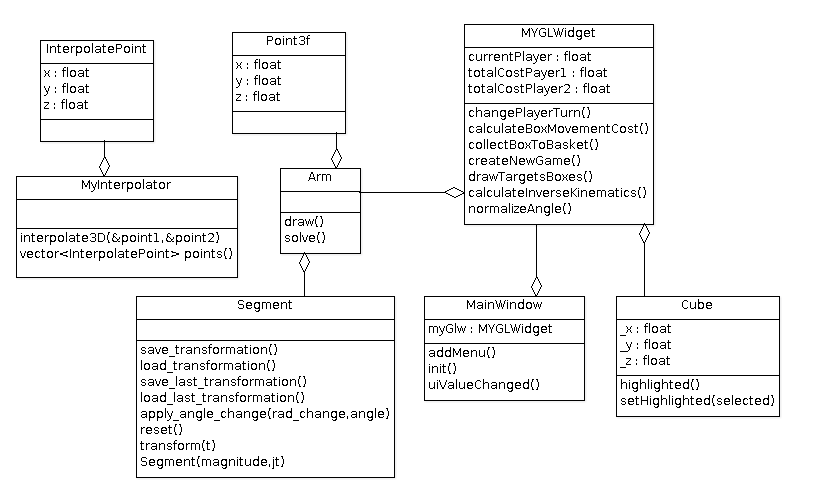
**3.2.2 Technologies: QT**

Qt is a cross-platform application framework that is widely used for developing application software that can be run on various software and hardware platforms with little or no change in the underlying codebase, while having the power and speed of native applications. Qt is currently being developed both by the Qt company, a subsidiary of Digia, and the Qt project under open-source governence, involving individual developers and firms working to advance Qt. Qt is used mainly for developing application software with graphicla user interface (GUIs)

**3.2.3 Eigen**

*Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms. It is widely used for high performance, fast computation in computer graphics and computer vision.*

**3.3 Class Diagram**



**3.4 Application Structure**

**User Interface**: The application has one window named MainWindow. It was created by QtCreator. It has drag and drop facility and easy to design. Every 30 miliseconds this window is refreshed and redrawn.

**OpenGL Widget**: to facilitate opengl calls in ui window, we extended the QGLWidget provided by default in Qt framework. The extended class is called MYGLWidget.

**User Controls**: On the right part of MYGLWidget there are many controls for user like Camera, Scale, Rotate, Translate

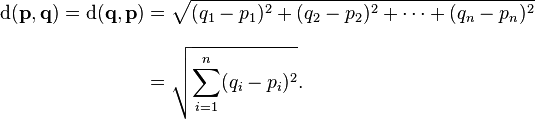
**Robot Arm**: Robot Arm is constructed by 6 segment of Segment class that forms the Arm class.

**Selection of Boxes**: User can select only 5 boxes. Rest of are discarded from the queue while selecting new box.

**3.5 Robot Arm: Game Strategy**

We implemented the Robot Arm project as a strategy game. There will be two players in the game. Every player has to collect boxes. Each time a player can select maximum 5 boxes by mouse click. The boxes should be near to arm. More the box is near to arm, less the cost it takes to be collected.The cost to collect a box to basket is calculated by comparing 3D euclidean distance from the bottom segment of robot arm.

**Calculate Cost of Moving Box to Basket**



we used 3D **Euclidean distance** between two target box and reference or pivot point of robot arm to calculate the distance which will be used as cost in moving boxes. Lower the distance/cost better the performance of arm movement.

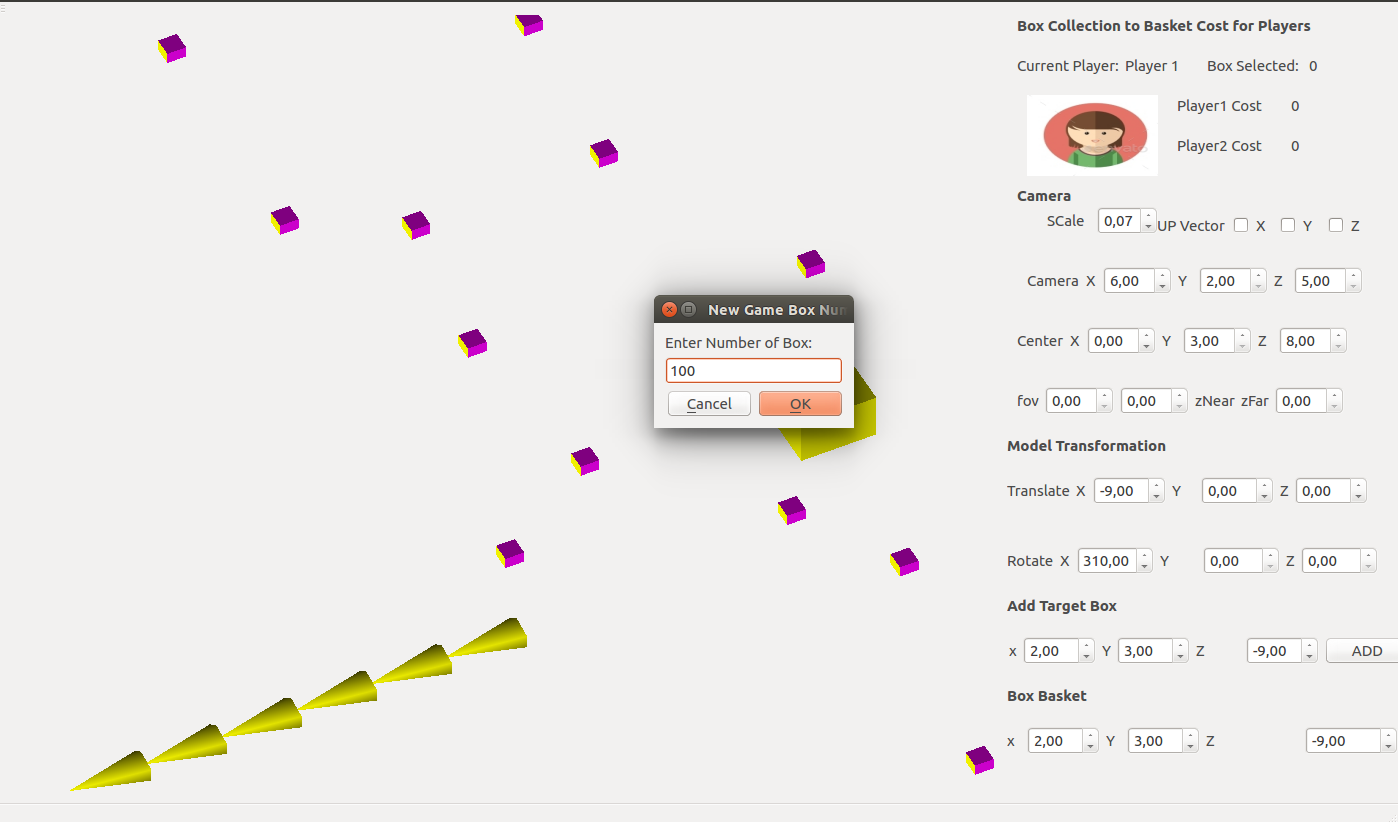
**Results**

**4.1. Demo and Screen shots**

when user runs the application it shows a window with some boxes pre-loaded.

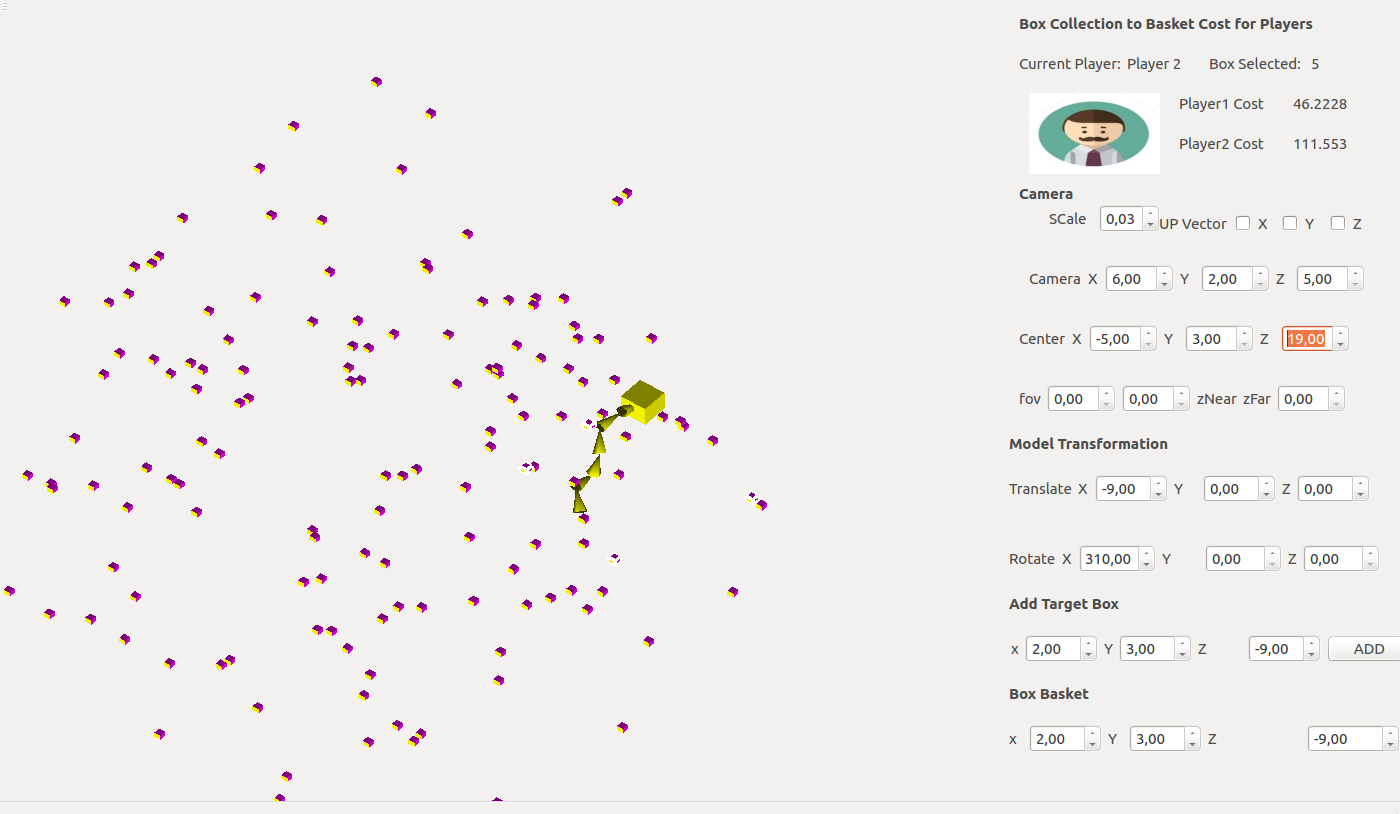


**New Game**

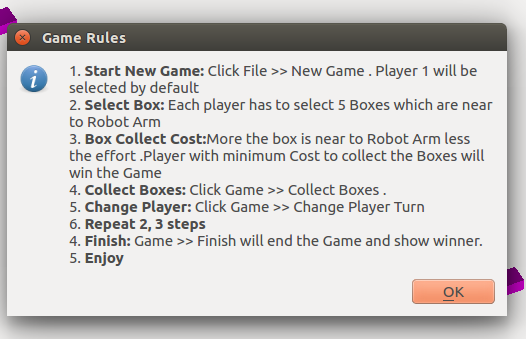
****

to start a new game click on File >> New Game. It will promopt a dialog to know how many objects are needed to be picked up by robot arm.

**Game random boxes**

after clicking on the new game dialog, it will create a random distribution of boxes to be collected by Robot Arm.

**Game Rules**

****

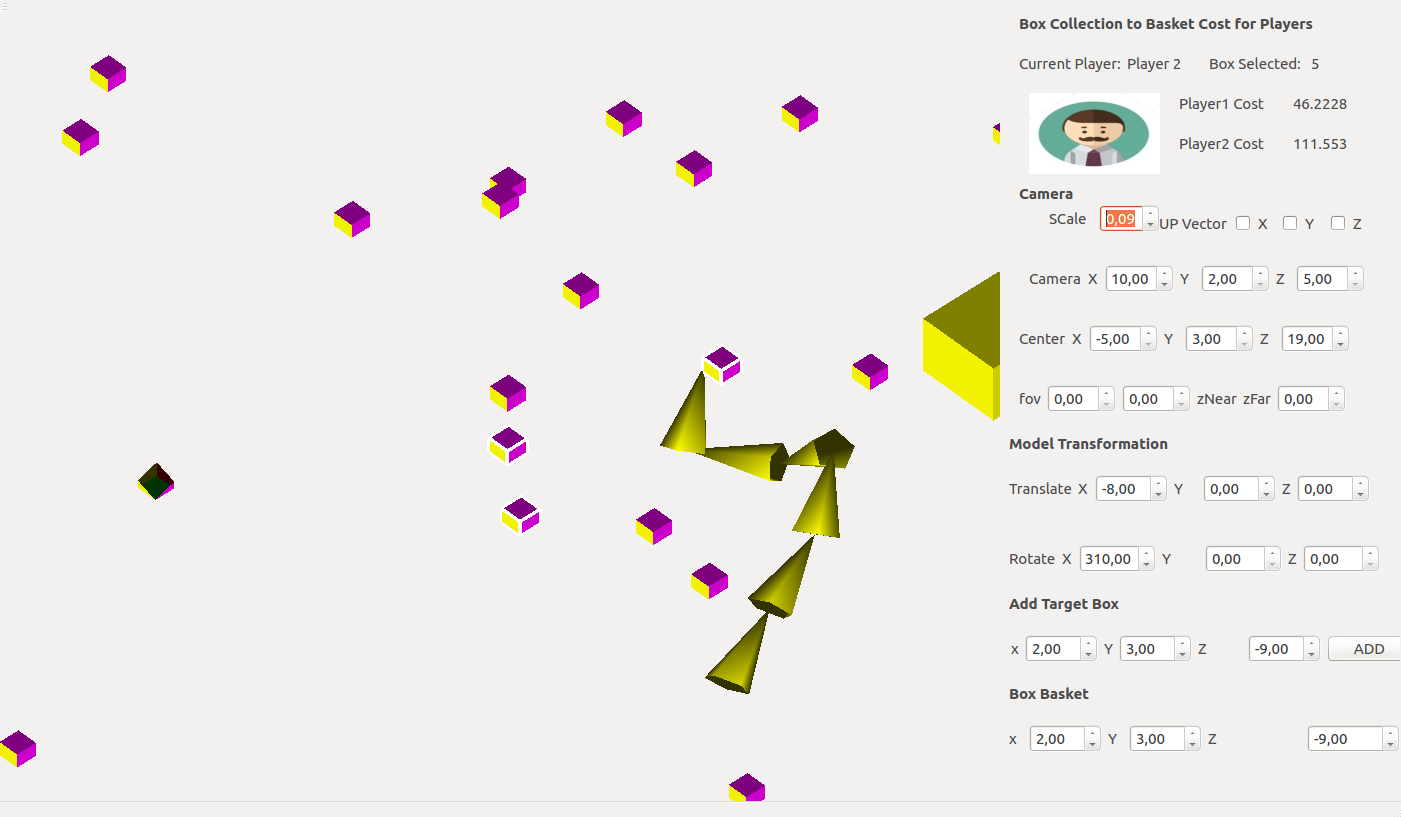
To view the game rule, click on Help >> Rules. Any time player can view the rules.

**Player 1 Box Select by mouse**

****

to select boxes, click on boxes and it will be selected and the highlight color will be white. The Robot arm will also try to catch the box to give player a idea about the distance of the box.

**Close View of Box Selection**

****

Here, we can see a close view of Box selection by mouse. The Selected boxes are all white highlighted.

**Player 1 Box Collected**

****

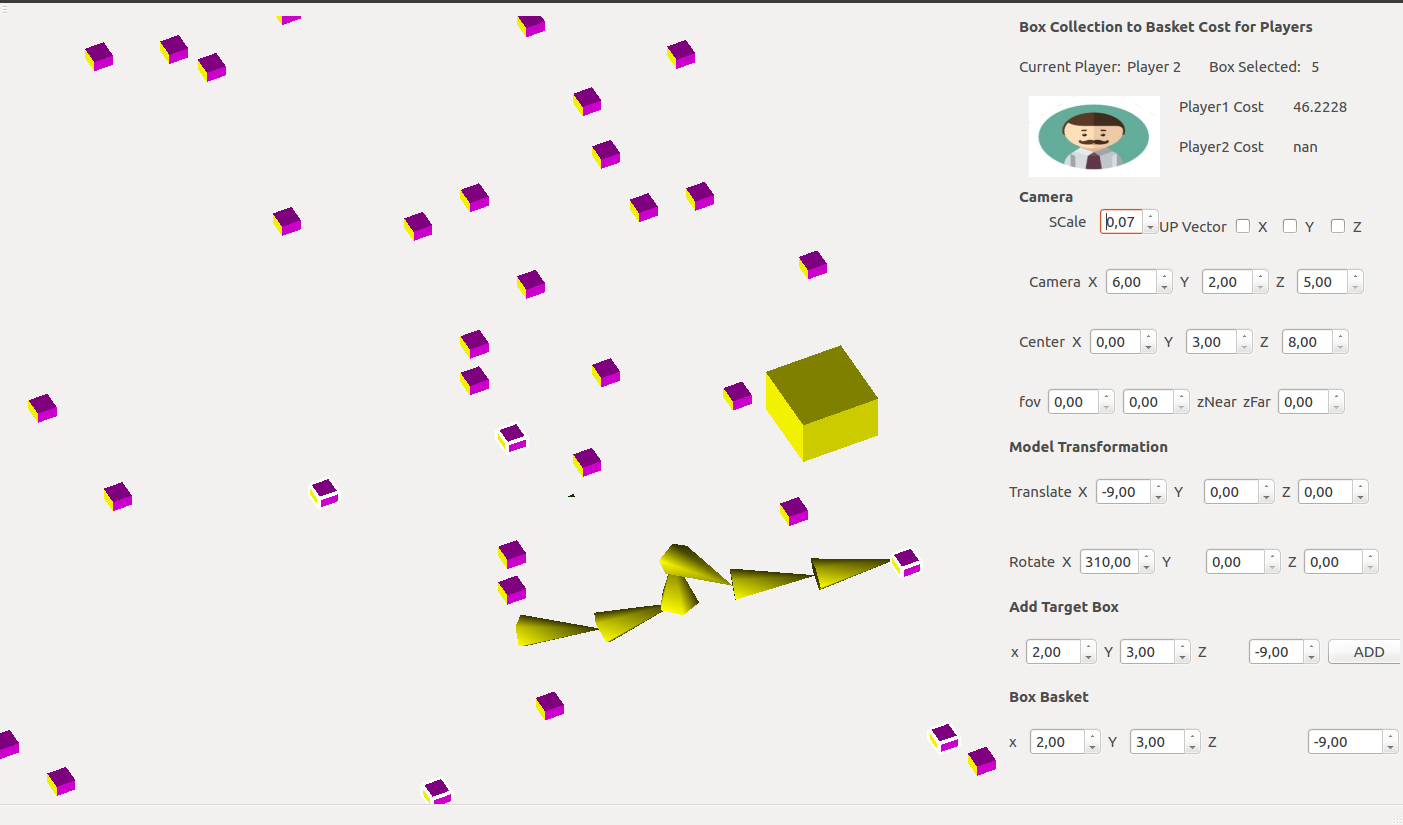
to select boxes, click on boxes and it will be selected and the highlight color will be white. The Robot arm will also try to catch the box to give player a idea about the distance of the box.

**Player 1 changed to Player 2**

****

to change the player, click on Game >> Change Player turn. It will change the player and load the appropriate setting.

**Player 2 select box**

****

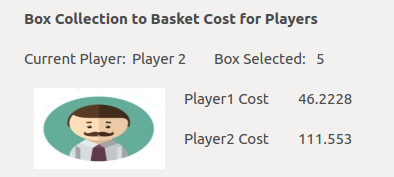
to select boxes, click on boxes and it will be selected and the highlight color will be white. The Robot arm will also try to catch the box to give player a idea about the distance of the box.

**Player 2 box Collected**

****

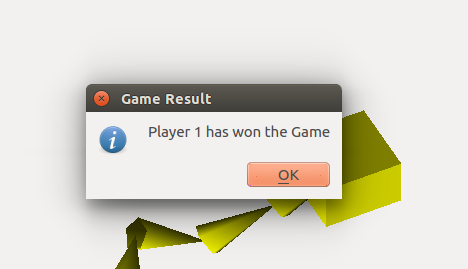
to collect the boxes to basket, click on Game >> collect Boxes. It will calculate the total cost needed to collect all these boxes to basket and update player to cost in Score. To transfer the arm to basket position linear interpolation is used.

**Cost of Player 1 and Player 2 to move the boxes**

****

player 1 has lower cost comparing to player 2. So player 1 will win the game.

**Game Finish & Result**

****

to finish the game and find the result of the game, click on menu Game >> finish. It will show a dialog named Game Result and show which player has won the game.

**Kinect**

kinect can capture RGB-D and RGB images. It operates on 30frames/second. The depth images is particularly useful in detecting gesture.

****



More Close the object, more brighter it is. We detected the gesture and tried to apply in robot arm.

**Test**

As the number of possible tests for our software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding software bugs (errors or other defects).

Software testing can provide objective, independent information about the quality of software and risk of its failure because of these we used testing for our components. While developing our software, we used various unit testing to find errors. Also, we used mock, stabs.

**5. Conclusion and Future works**

Our Robot can find the object in 3D which is with within its range. Though it has some limitations due to our lack of experience, still it can be used to plan and study robot arm. We have learnt significant knowledge in robot arm movement by doing this project.

**Project in a brief:**

We designed a 6-segment robot arm with triangles. ThisRobot Arm should have a movement with three angles and it should be able to pick a object and move it. OpenGL, Qt, Eigen, C++ were our tools for this purpose.

**Incidence**

We review some important incidents in our project that were more important

* We made some models but we couldn’t load and implement them correctly. We got a sample from our teacher to solve this problem. Again to solve our problem better we use the sample in different way and we change it up to we can use it.
* We setup our project in Java and run some sample with JOGL but as we couldn’t find useful sources and libraries in Java we had to switch to C++ and that taught us if we want to do the same project again we never use Java

**Improvement**

This project can be improved by many ways. For simplicity we choose to implement, the n-segment triangle to act as robot arm where each segment is of same type and each segment has same degree of freedom. Different type of segment with different degree of freedom can be used to construct the robot arm . We also drew the robot arm by using primitives. .obj model could be used to load the model. We also did not create surrounding environment for robot arm like Room, Table. Surrounding environment can be created to a more natural view. Currently, the animation of robot arm movement is very fast. This can be slow down and smoother by using more interpolation points between initial and final position and different types of interpolation techniques.

**6. Bibliography**

1. ***http://cerbero.uab.cat/moodle2/course/***
2. ***https://www.opengl.org/***
3. ***https://www.khronos.org/opengl/***
4. **https://developer.nvidia.com/opengl/**
5. **http://www.qt.io/**
6. **https://bitbucket.org/samirmenon/scl-manips-v2/wiki/vision/kinect**
7. **http://faltastico.tumblr.com/post/90660911262/giving-kinect-libfreenect-a-spin-on-ubuntu-14-04**
8. **http://eigen.tuxfamily.org/index.php?title=Main\_Page**
9. **http://en.wikipedia.org/wiki/Euclidean\_distance**
10. **http://en.wikipedia.org/wiki/Jacobian\_matrix\_and\_determinant**
11. **http://en.wikipedia.org/wiki/Inverse\_kinematics**
12. **http://www.3dkingdoms.com/ik.htm**
13. **http://en.wikipedia.org/wiki/Trilinear\_interpolation**
14. **https://www.opengl.org/wiki/Skeletal\_Animation**
15. **http://zetcode.com/gui/qt4/**
16. **https://github.com/michaelforney/inverse\_kinematics**
17. **http://www.sjbaker.org/steve/omniv/opengl\_lighting.html**