AUGMENTED REALITY 3D MODELER USING KINECT

By

Muhammad Waqar Malik **Registration # 2008-NUST-BIT-37**

Muhammad Abdul Hadi Khan Khosa **Registration # 2007-NUST-BIT-32**

A Project report submitted in partial fulfillment

of the requirement for the degree of

Bachelors in Information Technology

Department of Computing

School of Electrical Engineering & Computer Science

National University of Sciences & Technology

Islamabad, Pakistan

2012

CERTIFICATE

It is certified that the contents and form of thesis entitled **“Augmented Reality 3D Modeler using Kinect”** submitted by *Muhammad Waqar Malik* **(** *Reg# 2008-NUST-BIT-37***)** and *Muhammad Abdul Hadi Khan Khosa* (*Reg# 2007-NUST-BIT-32*) have been found satisfactory for the requirement of the degree.

Advisor: ______________________________

(Mr. Shamyl Bin Mansoor)

Co-Advisor: ______________________________

(Dr. Muhammad Murtaza Khan)

DEDICATION

To Allah the Almighty

&

To my Parents and Faculty

ACKNOWLEDGEMENTS

I am deeply thankful to my advisor and Co-Advisor, Mr. Shamyl Bin Mansoor, and Dr. Muhammad Murtaza Khan and of course Mr. Shahid Razzaq for helping us throughout the course in accomplishing my final project. Their guidance, support and motivation enabled me in achieving the objectives of the project.

TABLE OF CONTENTS

FIGURES

TABLES

ABSTRACT

Kinect device gives the ability to track a human skeleton in 3D space. This allows programmers to do interesting HCI work. Instead of using the mouse pointer in an application like Microsoft Paint, we have created a Kinect based application that gives the user multiple options (a toolbar on the side with selectable options, via hand gestures) to create, manipulate, save, and load 3D models. The view given to the user is an augmented view, i.e. it merges the real work (camera view on Kinect) with the virtual world (application). This gives a very realistic experience to the user and interacting with the virtual objects is fun.

Chapter 1

1 Introduction

1.1 Introduction

Augmented Reality 3D Modeler is software which implements 3D Augmented Reality. The application is able to draw objects in 3D space using a Kinect. It gets input from Kinect and draws 3d objects with the reference of the depth of Kinect as in distance on z-axis. We have drawn our 3D drawing Application in OpenGL and with the help of Kinect sensor we get input from the user and which is further processed by the Kinect and output is shown on the screen.

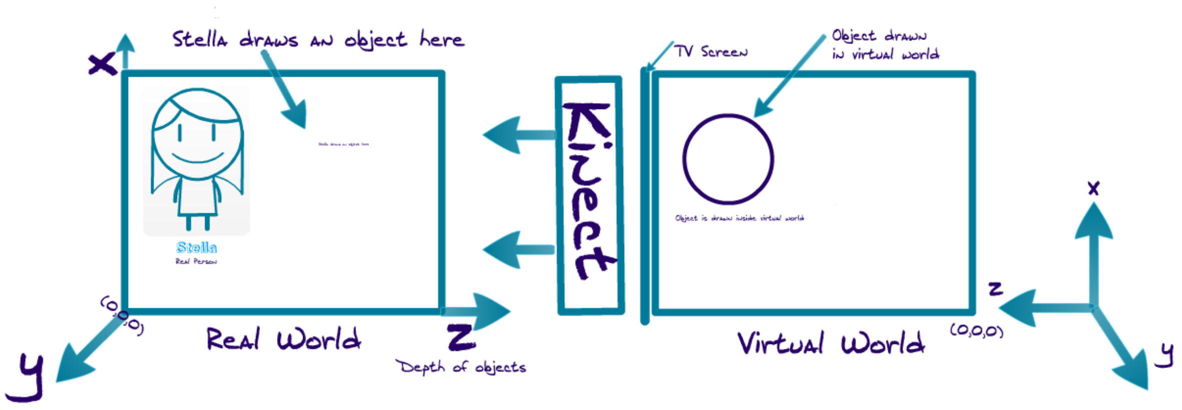

Stella draws an object here
x
Stella
Real Person
(0,0,0)
y
Real World
z
Depth of objects
Kinect
TV Screen
Object drawn in virtual world
Object is drawn inside virtual world
Virtual World
(0,0,0)
x
z
y

Figure 1

This figure explains the main conceptual structure of the application

Kinect device gives the ability to track a human skeleton in 3D space. This allows programmers to do interesting HCI work. Instead of using the mouse pointer in an application like Microsoft Paint, we have created a Kinect based application that gives the user multiple options (a toolbar on the side with selectable option, via hand gestures) to create, manipulate, save, and load 3D models. The view given to the user is an augmented view, i.e. it merges the real work (camera view on Kinect) with the virtual world (application). This gives a very close to realistic experience to the user and interacting with the virtual objects is fun.

1.2 Technologies:

There are different technologies which are necessary to make the application work. This includes some graphic libraries, a programming language and Kinect SDK at software level. At hardware level there is a Kinect sensor for input of 3D object (i.e., Human giving input to application).

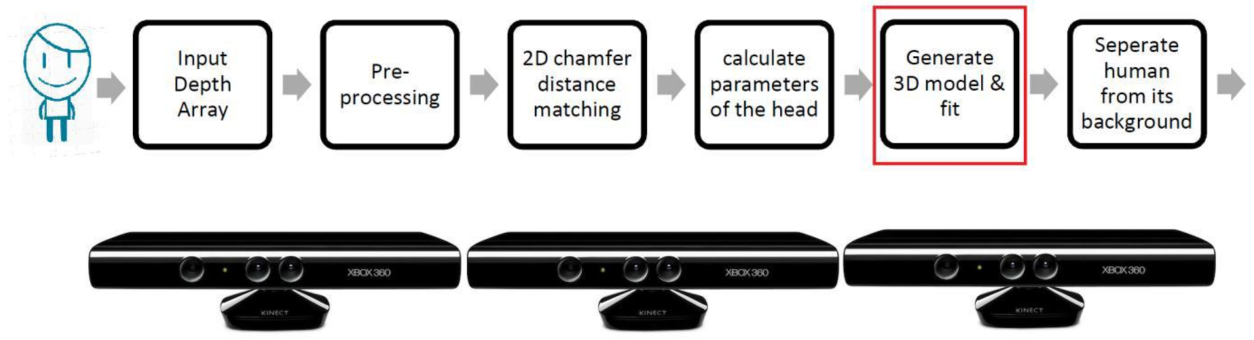

Input Depth Array
Pre-processing
2D chamfer distance matching
calculate parameters of the head
Generate 3D model & fit
Seperate human from its background

Figure 2

Processing of 3D Modeler [Phase-1]

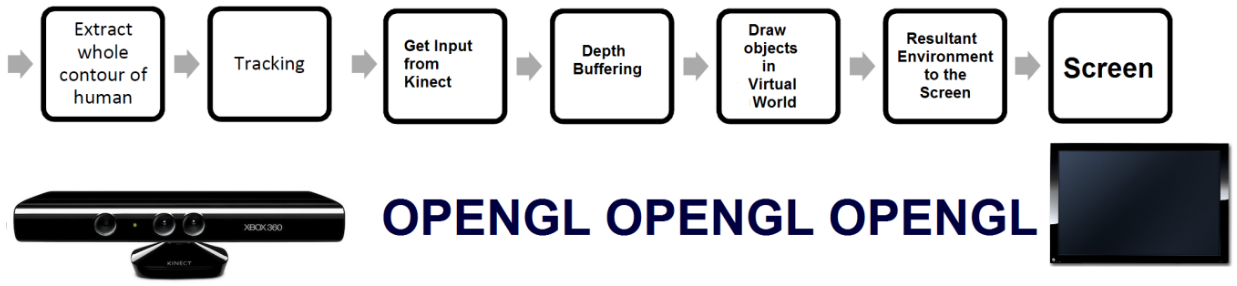

Extract whole contour of human
Tracking
Get Input from Kinect
Depth Buffering
Draw objects in Virtual World
Resultant Environment to the Screen
Screen
OPENGL OPENGL OPENGL

Figure 3

Processing of 3D Modeler [Phase-2]

1.3 Importance

This project is really helpful in junior classes to familiarize children with basic shapes and attracts them to physical activity as input from Kinect involves hand gestures as pointer. For example, a kids can draw a sphere by this application using their hand gestures, the movement of 3D-object as per real world models like dragging across surface of screen.

1.4 Project Goal

The goal of the project is:

- Develop a 3D Drawing Application which can draw some 3D objects e.g. cube and sphere etc.
- Application that can get input from Kinect Sensor.
- Application that can draw basic 3D objects. This application is able to manipulate different objects separately as these objects are in different depths in z-axis.

1.5 Deliverables:

1 Application which can help in drawing 3D objects in 3D space with the help of mouse and keyboard.

2 Reading Z-buffer and manipulation of depth affected by kinect input.

3 Merging of Z-buffer mechanism with 3D Modeler.

4 Merging of Application with Kinect and Z-buffer mechanism.

Chapter 2

2 Literature Review

2.1 Different Related Projects:

2.1.1 A Gesture-Based 3D Drawing Application for the Microsoft Kinect:

Kinect gives us the depth of z-axis, which is useful for the location of the user ("Control") in 3D space. The system can be very useful in the control of 3D applications, which are often difficult to control with conventional devices such as a keyboard or mouse. 3D modeling software, in particular suffer from this problem, which must be put points in 3D space very accurately. And generally aligning different perspectives, where the points are set for the first time in the plane, then corrected in the second. With the 3D input device is simply a matter that can be placed points in a job. In this project, we sought to implement a simple 3D drawing program which allows the controller to draw 3D shapes in the air with his hands, and the presentation of these models to the screen. After that, models can be manipulated or examined from different angles, using various hand signals. The space in front of the camera in the map of the design surface of the 3D display, which means that if the movements of the remote control of the camera, the work is set to a different area of the drawing area.

2.1.2 Comparison:

Building a 3D application gesture drawing closer to our augmented reality 3D Modeler because both have the same basic functions to detect volumetric 3D X, Y, Z coordinates that is a difficult task. There is also a movement of 3D objects in the air, allowing the movement of the hands of the person who is on the display. In our project, and there are some additional features to draw 3D objects based on the application of the notion of depth buffer.

2.1.3 Intangible Canvas (Free-Air Finger Painting on a Projected Canvas):

This project explores the use of open interaction in creating digital art projected on large tables. If recent developments in interactive digital outdoor MobiSpray include art, interactive art, where artists use cell phones to digital graffiti sprayed on the surface. Laser marking, Graffiti Research Labs project, performs a similar task using laser and computer vision instead of cell phones.

Instead of using the physical instrument sensors impregnated, such as the iPhone or Wiimote, using the affordance ZeroTouch sensor allow one to line of sight, and interact with a remote canvas. By placing sensors on a direct line of sight between the artist and the painting expected, the artist can paint the plate with a finger intangible (non-material literally touch it does not). Compared with systems based on the vision of the interaction in the open air or like cave painting Fillon, where three dimensions are used - interact to create three-dimensional graphics 3 - our system is limited to a certain reaction in a two-dimensional plane.

2.1.4 Comparison:

Augmented reality project "intangible canvas", which is useful in exploring types of interaction offered by the precision and outdoor interfaces. Observed in need of pre-activation reactions by

the use of open interaction to control accuracy reactions without clear limits activation controlled and precise positioning.
While on the other hand, the development of our application 3D graphics that can draw applications such as 3D cube and other organizations, and the field becomes Kinect sensor inputs and draw 3D objects in space.

2.1.5 Augmented Reality In-Situ 3D Sketching of Physical Objects:

3D interface to create digital maps in 3D input geometry in 2D or 3D. The three categories of these interfaces have been identified: Marc interpretation devices, and examine the structure and are interested in the interpretation of Mark, which is the interface 2D or 3D maps of continuous paths in 3D geometry. Interpretation in the forums or receive input interfaces in 2D or 3D. These techniques use 2D input to receive input conversion 2D to 3D engineering such as the interpretation of 3D cube drawing sheets Cube.
There have been a number of recent developments in this field, such as system development and system diagram Teddy. Although this approach is based on the ease and accuracy of 2D drawing, the difficulty lies in the ambiguity resolution two-dimensional mapping of the three-dimensional information. Can only approximate existing systems based on 3D shapes covered. The other approach is to get the titles continued correspond directly to 3D free-form 3D lines.
Tracking system tracks the position of operation of the printing device by the user. Using position samples points as points on line free 3D drawing can be recorded and displayed in real time without ambiguity. Various systems have been used and this approach, which uses the system Holosketch keeps track of ultrasound in the virtual reality system, which provides free model haptic force comment for the design of the device when it comes into contact with objects virtual surface and drawing using electromagnetic tracking in a virtual reality environment.
Use the video see-through augmented reality interface similar to Holosketch surface and design, but also allows the user to use the real world as a reference when developing. Augmented reality system must not follow just stick to drawing, but also raises the user's head, so that the point of view of the camera in the virtual world can match the perspective of the user (ie from the perspective of cameras mounted screens). In this way, the cost of food as overlay over live video from cameras, and appear as if drawing in the real world.

2.1.6 Comparison:

In the project " Augmented Reality In-Situ 3D Sketching of Physical Objects" tried to combine manufacturing technologies president mounted and free augmented reality start watching the whole process in this project with the help of the monitoring system so that they can follow a drawing position performed by the user.
Although our project Augmented Reality 3D Modeler There is nothing like a tracking device that appears on the screen just drag and drop various 3D objects previously published on the application tool for us and in 3D can draw different 2D object on the screen using different lines to develop algorithms.
In short, these two applications belong to augmented reality and 3D, but they are very different from each other in the case of their properties and use.

2.1.7 Comparison Table:

Projects Name| Features	**3D-Modeling Software**	**Kinect**	**3-Dimensional Drawing**	**Depth Buffering**
3D Augmented Reality Modeler	Yes	Yes	No (just drag & drop 3D objects)	yes
A Gesture-Based 3D Drawing Application	Yes	Yes	Yes	yes
Intangible Canvas	No	No (its use zero Touch sensor)	Yes	Yes
Augmented Reality In-Situ 3D Sketching	No	No(its use tracking system)	Yes	yes

Table I

2.2 OpenGL

2.2.1 Introduction

Originally created as an alternative to open OpenGL Iris GL and reproducible, who was the owner of the graphics API Silicon Graphics workstations. Although initially OpenGL similar in some respects to the absence IrisGL official specifications and compatibility testing Iris GL unfit for adoption on a larger scale. Mark Segal Kurt Akeley tried and author of the OpenGL 1.0 specification formalize the definition graphics API is not useful and multi-platform 3rd party IMS and support the implementation viable. There was a notable omission from the 1.0 version of the API texture objects. IrisGL taken steps to identify and link all sorts of things, including textures, materials, lights and environments texture. Avoid these items OpenGL to make additional changes to the state with the idea that it can encapsulate collective changes in the supply lists. This philosophy has been the exception rather than the texture of objects (glBindTexture) without separate definition phase is a key element of the API. The OpenGL through a number of revisions were primarily additions where additional API has been extended progressively integrated based in the main body of the API. For example appends the OpenGL 1.1 API glBindTexture base.

OpenGL 2.0 integrates the most important of the OpenGL Shading Language (GLSL also called), and C like language that can be programmed processing steps and a shaded line.
OpenGL 3.0 adds the notion of context compatible with previous versions and compatibility context. In

direct mode, some functions do not work GL old because they are old / outdated and should not be used. There are better and more moderne.Depuis GL 1.5, is added VBO that are more effective. Initialization

Before using OpenGL in the program, you must create for the first time. Because OpenGL platform, and there is no standard method to create the platform OpenGL works differently.

There are two phases OpenGL initialization. The first step is to create a framework OpenGL, and the second phase is to download all the necessary functions to use OpenGL. Some non-C / C + + integration of these links in one.

2.3 Depth Buffer

2.3.1 Introduction

The depth buffer stores a depth value for each pixel. A depth buffer operates by combining depth, or distance from the plane View (generally the cutting plane in the vicinity), for each pixel in the window. At first, the depth values defined for all pixels in the largest possible number (usually takes up the plane) using glClear () with GL_DEPTH_BUFFER_BIT. Then draw the objects in the scene in any order.

Usually measured in terms of the depth distance from the eye, so pixels with values greater buffer depth is overwritten b The depth buffer stores a depth value for each pixel. A depth buffer operates by combining depth, or distance from the plane View (generally the cutting plane in the vicinity), for each pixel in the window. At first, the depth values defined for all pixels in the largest possible number (usually takes up the plane) using glClear () with GL_DEPTH_BUFFER_BIT. Then draw the objects in the scene in any order. Usually measured in terms of the depth distance from the eye, so pixels with values greater buffer depth is overwritten by pixels with smaller values. It is simply an agreement useful, however, and can modify the behavior of the buffer as described in depth "depth test." Sometimes called the Z-buffer depth buffer. It is simply an agreement useful, however, and can modify the behavior of the buffer as described in depth "depth test." Sometimes called the Z-buffer depth buffer (z to execute the X and Y values of the measured horizontal and vertical movement on the screen, and the value of the distance measurements Z perpendicular to the screen).

2.3.2 Logical Operations, Blending and Dithering

Once in the past all the tests described in the previous section, along with the contents of the current buffer to color in several ways. The easiest way, which is also the default is to replace the existing values. In addition, if you use RGBA mode and we want to be partially transparent or anti-aliasing, you can average value with the value already in the buffer (mixture). On systems with a small number of colors available, you can calibrate the color values to increase the number of colors available for the price of the loss of resolution. In the last step, you can use arbitrary bitwise logical operations to combine part incoming pixel already written.

2.3.2.1 Blending

Mixing a selected portion combines R, G, B, and alpha values with those already stored in pixels on the site. Different methods can be applied to a mixture, and mix that appears depends on the value of the alpha value contained (if any) are stored in pixels.

2.3.2.2 Dithering

On systems with a small number of color a small plane, you can improve the color resolution at the expense of the juxtaposition of this spatial resolution of the image colors. Dither halftone as is the case in the newspapers. Although the New York Times and only two colors - black and white - can show images that represent shades of gray with groups of black and white points. Compare files recorded newsprint (no shades of gray) with the original image (gray) net loss of spatial resolution. Likewise, you may box systems with a small number of bit values abandoned Red Blue neighboring pixels Walokhaddr to collect a wide range of colors. Frequency process that occurs depends on the hardware and OpenGL allows you to do is turn it on and off. In fact, in some devices, you can set the frequency to anything, which makes sense if the device already has high color accuracy. To enable or disable dithering, pass GL_DITHER glEnable () and glDisable (). Dithering is enabled by default.

2.3.2.3 Logical Operations

Last operation is part of the logic of the process, such as OR, XOR, or reflection, which is applied to the values contained in the first part (source) (destination) and / or those who are currently in the color buffer. These fragments are particularly useful type machines bitwise BLT, who was the master copy rectangle drawing data from one place in one context to another, from the window of a window memory, processor or memory. In general, do not copy writing data directly to memory, but allows arbitrary logical operations data and the data already exists, it replaces the existing data with the results.

2.3.2.4 Depth Test

Each pixel of the screen, the depth buffer monitors the distance between the viewpoint and the object pixel occupation. Then, if the test of passing a specific depth, and the depth to replace a value stored previously in the depth buffer.

2.3.3 Enabling depth buffering

Follow these steps to enable depth buffer:.

- The current framebuffer, maybe default framebuffer or an FBO, must have a depth buffer.
- User must enable Depth testing using glEnable(GL_DEPTH_TEST). Depth testing is disabled by default.
- Using glDepthFunc() depth method must be set.
- Set the depth mask using glDepthMask().
- Set remap of depth using glDepthRange().

Clean depth buffer before and after using it.

2.4 Kinect

2.4.1 Introduction

Take the end of 2010, has attracted a lot of attention to Microsoft's Kinect in the scientific community. As originally designed games Kinect device, the player uses his body to control games. Quickly released open source drivers and applications many interesting. Kinect combines an ordinary camera with an image of "flight time" infrared camera, and give each user an RGB image of the scene in front of the camera, and the depth of the scene.

This makes it easy to do all sorts of things that are very difficult to do with an RGB camera, such as split objects. The process of international recognition of Atomic Energy (IAEA) it is possible to get rid of this technology. Infrared camera and works by sending pulses of infrared light, and measuring the time required for the light of the return of the device.

There is a slight horizontal shift between the sender and the camera, which means that not only parts of the image by the camera looks at lighting. Fans Kinect used for all kinds of interesting applications, such as building a 3D reconstruction of the scene and keep fingerprints to control the application to display images or PowerPoint presentation control with hand movements. In this project, we Kinect for something else.

Kinect is an input device to detect the movement of video game Xbox 360. Depends on the style webcam add the device to the Xbox 360, users can control and interact with the Xbox 360 without the need to touch and control games through natural user interface using gestures and voice commands. The project aims to expand the audience for the Xbox 360 and beyond its basic model.

Kinect is a horizontal bar connected to a small base of the main engine and adapted to be in a position longitudinal above or below the screen. The device is characterized by "RGB sensor, camera and microphone depth or performance of goods", which offers a full range of 3D motion capture of the human body, facial recognition and voice recognition. Absolutely, speech recognition is not only available in Japan, and the United Kingdom, Canada and the United States. Work on the continent of Europe in the spring of 2011. Detection of Kinect for Xbox 360 accuracy allows all audio sources and remove ambient noise, allowing for things like headphones without collective thinking on Xbox Live.

Depth sensor consists of optical sensors for data to CMOS monochrome laser infrared video-related 3D for all inputs to the ambient light conditions. Detection range of the sensor depth adjustable, and the program is able to calibrate sensors automatically depending on the Kinect game and the player's physical environment to get a piece of furniture or other obstacles. Described by Microsoft as members of the main innovation of Kinect and technology program allows gesture recognition, facial recognition and voice recognition. According to the information available to retailers, Kinect is able to track at one time up to six people, including two active players for motion analysis with a feature extraction of 20 joints per player. However, he noted that many of the PrimeSense camera people can "see" (but not players) by a number of people may occur in the field of view of the camera.

Chapter 3

3 Technologies and Methodologies

3.1 3.1 Requirements:

3.1.1 Modules:

3.1.1.1 3D Modeler:

3D Modeler is an application which avails user with the functionalities like drawing using hand and body gestures. Interface parameters of the application are given below:

- **i.** Top menu is on the top of main screen. it consists of five buttons create, edit, move, save and load.
- **ii.** 3D objects menu is on the left side of the screen. It has different 3D objects like cube, sphere and triangle.
- **iii.** Drawing console is simple console where user can draw different 3D object selected from the 3D object menu. Drawing console is like a convas or simple plan where all the drawing will occur.

When Application starts first of all there is a loading screen. After this main screen of the application is displayed with blank console and two menus, top menu and 3D object menu as describe above.

At start both menus are disabled or hidden with show and hide option gestures with kinect, waving hand across extreme left side or top of the screen. 3D objects are selected by hand movement. To draw an object user need to select it from object menu. Objects will be drawn on drawing console, when user selects edit option in top menu then he/she can change the size of object by using specified gesture, in this case spreading both arms perpendicular to the body and keeping both hands open and straight; when user closes his/her arms in front of torso/chest size decreases or doing it otherwise increases the size of the object. When user selects move button then he/she can move an object in the drawing console by using hand movement. Through save option he/she can save the changes made during the drawing. This will be saved in a text file as in the form of numbers and text.

3.1.1.2 Depth buffering:

Depth buffering is done by OpenGL itself. All we intended to do is to customize it. Using glReadPixels(); function. Using frame buffer and get depth values indirect.

- glSelectBuffer(64, buff);
- glGetIntegerv(GL_VIEWPORT, view);
- glRenderMode(GL_SELECT);
- glInitNames();

- glMatrixMode(GL_MODELVIEW);

3.1.1.3 Kinect Integration:

3.1.1.3.1 Kinect Input:

As described in the 3D Modeler part: input is gesture based and hand movement works as pointer.

3.1.1.3.2 Input Manipulation

.

3.2 Use Case Diagram:

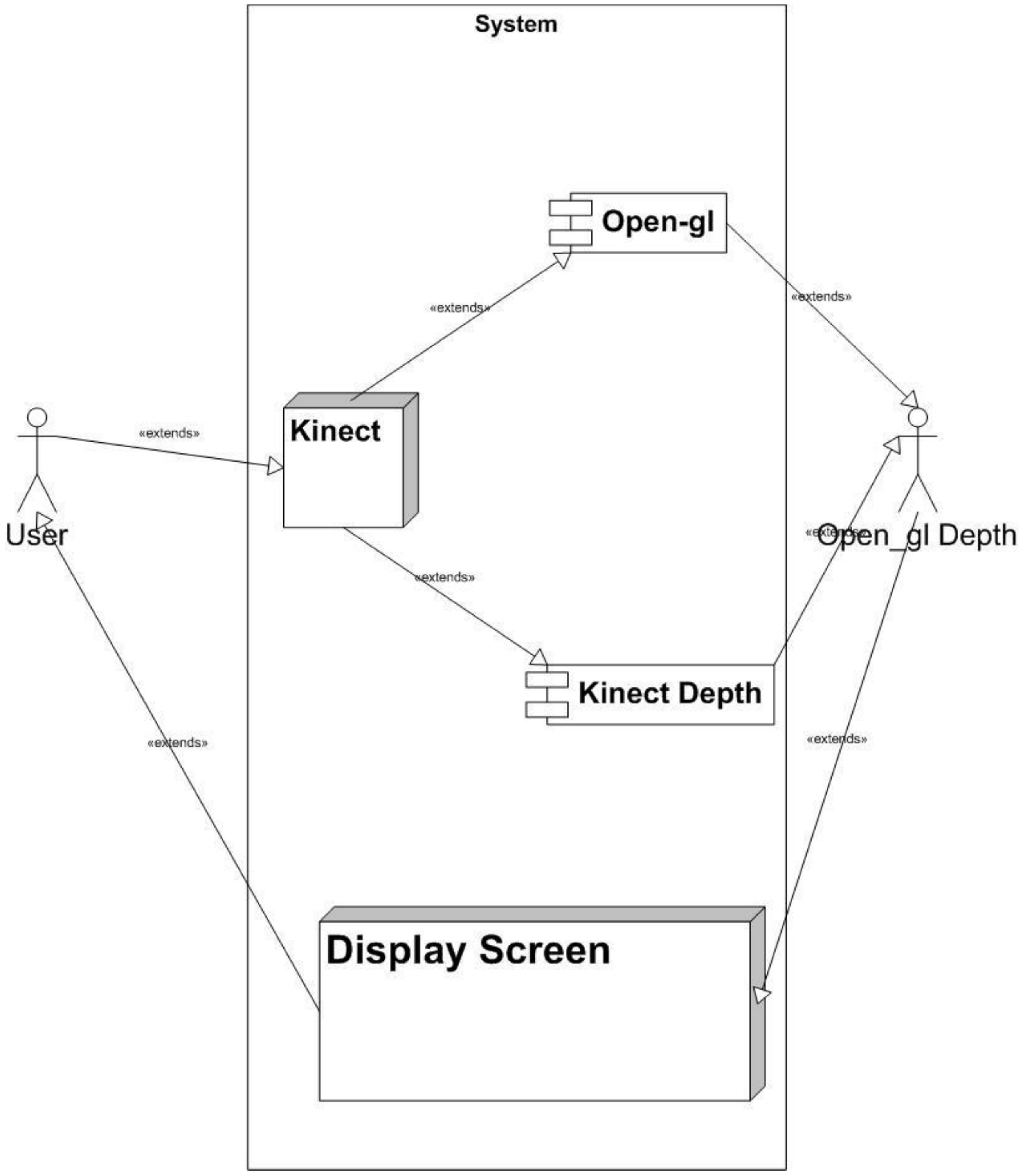


Figure 4

Use case diagram

3.3 System Architecture:

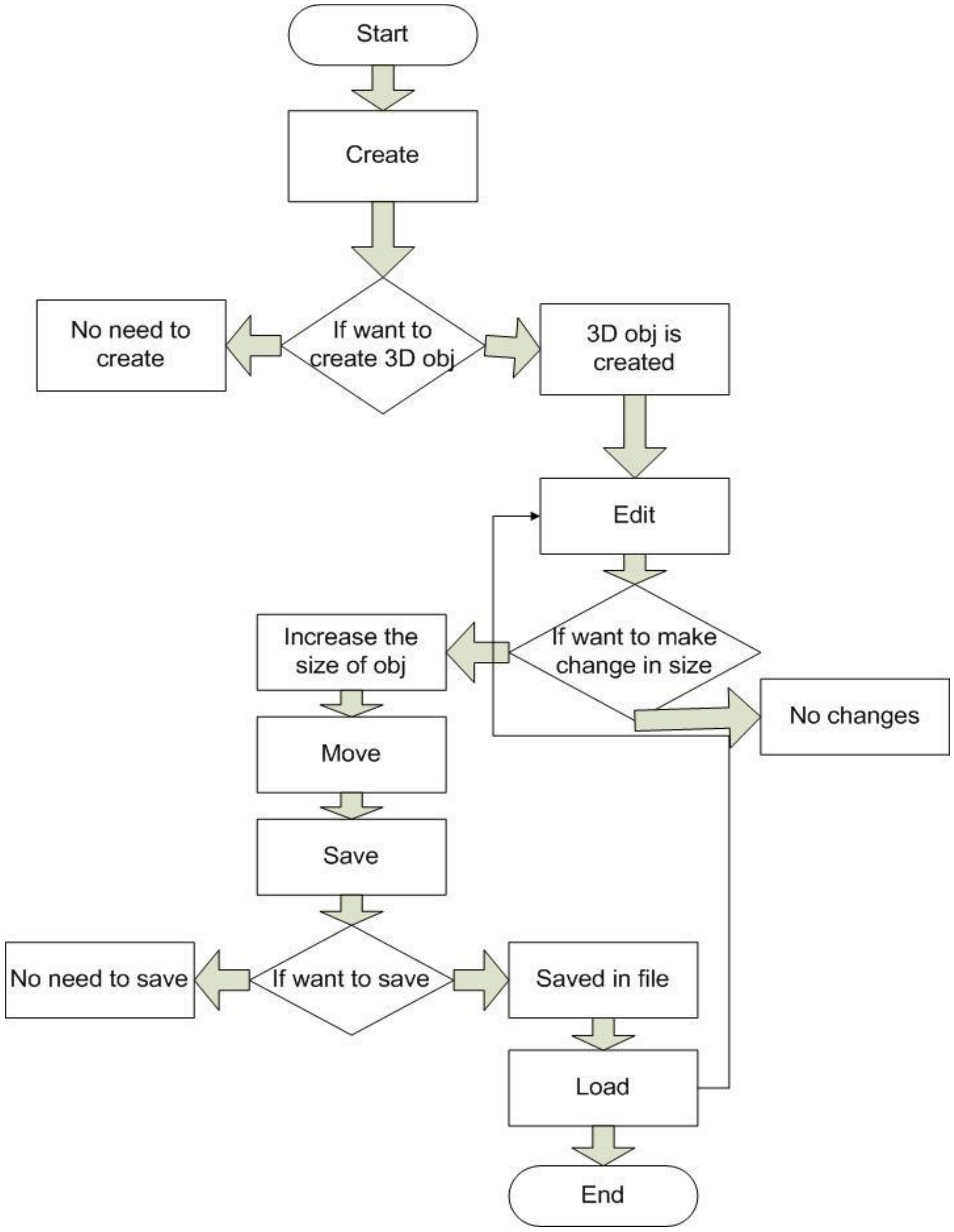


Figure 5

System Architecture

3.4 Challenges Faced:

Many challenges faced during our work. We have divided these challenges in two pieces which are as follows:

3.4.1 3D Modeler:

Movement of 3D objects in the drawing console was one of the big challenge. we had done it by dragging the object from 3D object menu to the drawing console but it had problem in integration with kinect because kinect doesn't support dragging.

3.4.2 Depth Buffering

Depth Buffering is hardware dependent. So it seems nearly impossible to develop a single application which can run as desired on all kind of Systems or PC's. By the name of PC's, we mean GPU's which are primarily concerned with Graphics rendering and Depth Buffering.

In Market there are mainly 3 vendors which make GPU's. We'll discuss each of them briefly.

3.4.2.1 Intel:

Intel makes basic graphics processors which come as default GPU's. As a matter of fact Intel is not a choice for graphics processing among gaming or graphics designing society. It is due to poor performance of Intel's graphics cards. We tested our first application on this GPU. As expected, it did not give any depth reading.

3.4.2.2 NVIDIA:

NVIDIA makes best graphics processors which used to be the priority GPU's. NVIDIA's GPU's are high performance GPU's and very expensive in market. NVIDIA used to be the first choice of gamers. We tested our first application on this GPU. Unexpectedly, it did not give any depth reading either.

3.4.2.3 AMD/ATI:

ATI which is now part of AMD makes best graphics processors which are the priority GPU's. AMD's GPU's are high performance GPU's and relatively cheap in market. AMD is now the first choice of gamers. We tested our first application on this GPU. Unexpectedly, it did work and gave exact depth reading.

3.5 Tools and Technologies:

3.5.1 Language:

Currently we are using C/C++ as we have to use pointers to read depth buffer.

3.5.2 Library:

We are using OpenGL library to draw objects and handle depth of overlapped objects. OpenGL is preferred library because it is open source and relatively easy to manipulate.

3.5.3 IDE:

We are using Microsoft Visual Studio 2010 with Service Pack 1 updates.

3.5.4 Hardware:

We are testing our application on two kinds of systems right now. As for our requirement there are 3 main components of a system we are concerned about.

3.5.4.1 Intel core 2 Duo and core 2 Quad with Intel Chip Set Family

This PC did not not meet our requirements.

3.5.4.1.1 Processor:

Core 2 Duo 2.1 GHz T8100 Series Dell

3.5.4.1.2 GPU:

Intel 954 Chip Set Family

3.5.4.1.3 RAM:

4 GB DDR-II

3.5.4.2 Intel core i5 with AMD Radeon HD 4870

3.5.4.2.1 This PC was giving our required results until now.

3.5.4.2.2 Processor:

Core i5 2.56 GHz

3.5.4.2.3 GPU:

AMD ATI Radeon HD 4870

3.5.4.2.4 RAM:

4 GB

Chapter 4

4 Methodology

4.3 Classes Documentation

4.3.1 check Struct Reference

4.3.1.1 Public Attributes

- bool **triangle**
- bool **sphere**
- bool **cube**

4.3.1.2 Brief Summary

Defination of Checks weather trinagle clicked sphere clicked or cube clicked

Defined at 130 of file SkeletalViewer.cpp.

4.3.1.3 Member Data Documentation

4.3.1.3.1 bool check::cube

Defined at 134 of file SkeletalViewer.cpp.

4.3.1.3.2 bool check::sphere

Defined at 133 of file SkeletalViewer.cpp.

4.3.1.3.3 bool check::triangle

Defined at 132 of file SkeletalViewer.cpp.

4.3.1.3.4 Information of this struct was gathered from file below:

- Parent Directory/**SkeletalViewer.cpp**

4.3.2 CSkeletalViewerApp Class Reference

#include <SkeletalViewer.h>

4.3.2.1 *Public Member Functions*

- **CSkeletalViewerApp** ()
- **~CSkeletalViewerApp** ()
- HRESULT **Nui_Init** ()
- HRESULT **Nui_Init** (OLECHAR *instanceName)
- void **Nui_UnInit** ()
- void **Nui_GotDepthAlert** ()
- void **Nui_GotColorAlert** ()
- void **Nui_GotSkeletonAlert** ()
- void **Nui_Zero** ()
- void **Nui_BlankSkeletonScreen** (HWND hWnd, bool getDC)
- void **Nui_DoDoubleBuffer** (HWND hWnd, HDC hDC)
- void **Nui_DrawSkeleton** (NUI_SKELETON_DATA *pSkel, HWND hWnd, int WhichSkeletonColor)
- void **Nui_DrawSkeletonId** (NUI_SKELETON_DATA *pSkel, HWND hWnd, int WhichSkeletonColor)
- void **Nui_DrawSkeletonSegment** (NUI_SKELETON_DATA *pSkel, int numJoints,...)
- void **Nui_EnableSeatedTracking** (bool seated)
- void **Nui_SetApplicationTracking** (bool applicationTracks)
- void **Nui_SetTrackedSkeletons** (int skel1, int skel2)
- RGBQUAD **Nui_ShortToQuad_Depth** (USHORT s)
- LRESULT CALLBACK **WndProc** (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
- void CALLBACK **Nui_StatusProc** (HRESULT hrStatus, const OLECHAR *instanceName, const OLECHAR *uniqueDeviceName)
- int **MessageBoxResource** (UINT nID, UINT nType)

4.3.2.2 *Static Public Member Functions*

- static LRESULT CALLBACK **MessageRouter** (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
- static void CALLBACK **Nui_StatusProcThunk** (HRESULT hrStatus, const OLECHAR *instanceName, const OLECHAR *uniqueDeviceName, void *pUserData)

4.3.2.3 *Public Attributes*

- HWND **m_hWnd**
- HINSTANCE **m_hInstance**

4.3.2.4 *Brief Summary*

Defined at 25 of file SkeletalViewer.h.

4.3.2.5 Constructor and Destructor Description

4.3.2.5.1 *CSkeletalViewerApp::CSkeletalViewerApp ()*

: Constructor

Defined at 625 of file SkeletalViewer.cpp.

4.3.2.5.2 *CSkeletalViewerApp::~CSkeletalViewerApp ()*

: Destructor

Defined at 640 of file SkeletalViewer.cpp.

4.3.2.6 Member Function Description

4.3.2.6.1 *int CSkeletalViewerApp::MessageBoxResource (UINT nID, UINT nType)*

: Display a MessageBox with a string table table loaded string

Defined at 913 of file SkeletalViewer.cpp.

4.3.2.6.2 *LRESULT CALLBACK CSkeletalViewerApp::MessageRouter (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam) [static]*

Defined at 765 of file SkeletalViewer.cpp.

4.3.2.6.3 *void CSkeletalViewerApp::Nui_BlankSkeletonScreen (HWND hWnd, bool getDC)*

Defined at 542 of file NuiImpl.cpp.

4.3.2.6.4 *void CSkeletalViewerApp::Nui_DoDoubleBuffer (HWND hWnd, HDC hDC)*

Defined at 756 of file NuiImpl.cpp.

4.3.2.6.5 *void CSkeletalViewerApp::Nui_DrawSkeleton (NUI_SKELETON_DATA * pSkel, HWND hWnd, int WhichSkeletonColor)*

Defined at 654 of file NuiImpl.cpp.

4.3.2.6.6 *void CSkeletalViewerApp::Nui_DrawSkeletonId (NUI_SKELETON_DATA * pSkel, HWND hWnd, int WhichSkeletonColor)*

Defined at 713 of file NuiImpl.cpp.

*4.3.2.6.7 void CSkeletalViewerApp::Nui_DrawSkeletonSegment (NUI_SKELETON_DATA * pSkel, int numJoints, ...)*

Defined at 556 of file NuiImpl.cpp.

4.3.2.6.8 void CSkeletalViewerApp::Nui_EnableSeatedTracking (bool seated)

4.3.2.6.9 void CSkeletalViewerApp::Nui_GotColorAlert ()

: Handle new color data

Defined at 460 of file NuiImpl.cpp.

4.3.2.6.10 void CSkeletalViewerApp::Nui_GotDepthAlert ()

: Handle new depth data

Defined at 491 of file NuiImpl.cpp.

4.3.2.6.11 void CSkeletalViewerApp::Nui_GotSkeletonAlert ()

: Handle new skeleton data

Defined at 771 of file NuiImpl.cpp.

4.3.2.6.12 HRESULT CSkeletalViewerApp::Nui_Init ()

: Initialize Kinect

Defined at 189 of file NuiImpl.cpp.

*4.3.2.6.13 HRESULT CSkeletalViewerApp::Nui_Init (OLECHAR * instanceName)*

: Initialize Kinect by instance name @*instanceName Takes the anme of instance as an argument

Defined at 161 of file NuiImpl.cpp.

4.3.2.6.14 void CSkeletalViewerApp::Nui_SetApplicationTracking (bool applicationTracks)

Defined at 860 of file NuiImpl.cpp.

4.3.2.6.15 void CSkeletalViewerApp::Nui_SetTrackedSkeletons (int skel1, int skel2)

Defined at 872 of file NuiImpl.cpp.

4.3.2.6.16 RGBQUAD CSkeletalViewerApp::Nui_ShortToQuad_Depth (USHORT s)

: Get the player colored depth value : parameter to get the depth

Defined at 842 of file NuiImpl.cpp.

*4.3.2.6.17 void CALLBACK CSkeletalViewerApp::Nui_StatusProc (HRESULT hrStatus, const OLECHAR * instanceName, const OLECHAR * uniqueDeviceName)*

: Callback to handle Kinect status changes : status of head : checking instance : checking device name

Defined at 128 of file NuiImpl.cpp.

*4.3.2.6.18 void CALLBACK CSkeletalViewerApp::Nui_StatusProcThunk (HRESULT hrStatus, const OLECHAR * instanceName, const OLECHAR * uniqueDeviceName, void * pUserData) [static]*

Defined at 117 of file NuiImpl.cpp.

4.3.2.6.19 void CSkeletalViewerApp::Nui_UnInit ()

: Uninitialize Kinect

Defined at 310 of file NuiImpl.cpp.

4.3.2.6.20 void CSkeletalViewerApp::Nui_Zero ()

: Zero out member variables

Defined at 86 of file NuiImpl.cpp.

4.3.2.6.21 LRESULT CALLBACK CSkeletalViewerApp::WndProc (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

: Handle windows messages

Defined at 791 of file SkeletalViewer.cpp.

4.3.2.7 Member Data Documentation

4.3.2.7.1 HINSTANCE CSkeletalViewerApp::m_hInstance

Defined at 56 of file SkeletalViewer.h.

4.3.2.7.2 HWND CSkeletalViewerApp::m_hWnd

Defined at 55 of file SkeletalViewer.h.

4.3.2.8 Information of this class was gathered from file belows:

- Parent Directory/**SkeletalViewer.h**
- Parent Directory/**NuiImpl.cpp**
- Parent Directory /**SkeletalViewer.cpp**

4.3.3 DrawDevice Class Reference

#include <DrawDevice.h>

4.3.3.1 Public Member Functions

- **DrawDevice** ()
- virtual **~DrawDevice** ()
- HRESULT **EnsureResources** ()
- void **DiscardResources** ()
- bool **Initialize** (HWND hwnd, ID2D1Factory *pD2DFactory, int sourceWidth, int sourceHeight, int Stride)
- bool **Draw** (BYTE *pBits, unsigned long cbBits)

4.3.3.2 Brief Summary

Defined at 17 of file DrawDevice.h.

4.3.3.3 Constructor and Destructor Description

4.3.3.3.1 DrawDevice::DrawDevice ()

: Constructor

Defined at 25 of file DrawDevice.cpp.

4.3.3.3.2 DrawDevice::~DrawDevice () [virtual]

: Destructor

Defined at 39 of file DrawDevice.cpp.

4.3.3.4 Member Function Description

4.3.3.4.1 void DrawDevice::DiscardResources ()

: Dispose Direct2d resources

Defined at 92 of file DrawDevice.cpp.

4.3.3.4.2 bool DrawDevice::Draw (BYTE * pBits, unsigned long cbBits)

DrawFrame: Draw the video frame.

Defined at 122 of file DrawDevice.cpp.

4.3.3.4.3 HRESULT DrawDevice::EnsureResources ()

: Ensure necessary Direct2d resources are created

Defined at 48 of file DrawDevice.cpp.

*4.3.3.4.4 bool DrawDevice::Initialize (HWND hwnd, ID2D1Factory * pD2DFactory, int sourceWidth, int sourceHeight, int Stride)*

: Set the window to draw to, video format, etc.

Defined at 101 of file DrawDevice.cpp.

4.3.3.5 Information of this class was gathered from file belows:

- Parent Directory/**DrawDevice.h**
- Parent Directory/**DrawDevice.cpp**

4.3.4 kinect_code Class Reference

#include <kinect_code.h>

4.3.4.1 Public Member Functions

- **kinect_code** (void)
- void **setvalues** (void)
- void **checkcoordinates** (void)
- void **kinect_mouse** (void)
- **~kinect_code** (void)

4.3.4.2 Brief Summary

: position in x-axis : position in y-axis : position in z-axis : stored previous position in x-axis : stored previous position in y-axis : stored previous position in z-axis : change in position : flag to check the click

Defined at 13 of file kinect_code.h.

4.3.4.3 Constructor and Destructor Description

4.3.4.3.1 kinect_code::kinect_code (void)

Defined at 12 of file kinect_code.cpp.

4.3.4.3.2 kinect_code::~kinect_code (void)

Defined at 186 of file kinect_code.cpp.

4.3.4.4 Member Function Description

4.3.4.4.1 void kinect_code::checkcoordinates (void)

Defined at 39 of file kinect_code.cpp.

4.3.4.4.2 void kinect_code::kinect_mouse (void)

Defined at 63 of file kinect_code.cpp.

4.3.4.4.3 void kinect_code::setvalues (void)

Defined at 22 of file kinect_code.cpp.

4.3.4.5 Information of this class was gathered from file belows:

- Parent Directory/**kinect_code.h**
- Parent Directory/**kinect_code.cpp**

4.3.5 temp::points Struct Reference

#include <kinect.h>

4.3.5.1 Public Attributes

- float **x**
- float **y**
- float **z**

4.3.5.2 Brief Summary

Defined at 5 of file kinect.h.

4.3.5.3 Member Data Documentation

4.3.5.3.1 float temp::points::x

Defined at 6 of file kinect.h.

4.3.5.3.2 float temp::points::y

Defined at 7 of file kinect.h.

4.3.5.3.3 float temp::points::z

Defined at 8 of file kinect.h.

4.3.5.4 Information of this struct was gathered from file below:

- Parent Directory/**kinect.h**

4.3.6 temp Class Reference

#include <kinect.h>

4.3.6.1 Classes

- struct **points**

4.3.6.2 Static Public Attributes

- static **points pnt** [2] = {0.0}
- static bool **tracked** = false

4.3.6.3 Brief Summary

Defined at 2 of file kinect.h.

4.3.6.4 Member Data Documentation

4.3.6.4.1 temp::points temp::pnt = {0.0} [static]

::pnt[2] Temporay Storage Of points of both hands

Defined at 11 of file kinect.h.

4.3.6.4.2 bool temp::tracked = false [static]

::tracked Store the tracking state of skeleton

Defined at 12 of file kinect.h.

4.3.6.5 Information of this class was gathered from file belows:

- Parent Directory/**kinect.h**
- Parent Directory/**SkeletalViewer.cpp**

4.3.7 vertex Struct Reference

4.3.7.1 Public Attributes

- int **xpos**
- int **ypos**
- int **zpos**

4.3.7.2 Brief Summary

Defination of vertex

Defined at 119 of file SkeletalViewer.cpp.

4.3.7.3 Member Data Documentation

4.3.7.3.1 int vertex::xpos

Defined at 121 of file SkeletalViewer.cpp.

4.3.7.3.2 int vertex::ypos

Defined at 122 of file SkeletalViewer.cpp.

4.3.7.3.3 int vertex::zpos

Defined at 123 of file SkeletalViewer.cpp.

4.3.7.4 Information of this struct was gathered from file below:

- D:/Study/FYP/Submission/Documentation/**SkeletalViewer.cpp**

4.4 File Documentation

4.4.1 Parent Directory/DrawDevice.cpp File Reference

#include "stdafx.h"
#include "DrawDevice.h"

4.4.2 Functions

- LONG **Width** (const RECT &r)
- LONG **Height** (const RECT &r)

4.4.3 Function Documentation

4.4.3.1 LONG Height (const RECT & r) [inline]

Defined at 17 of file DrawDevice.cpp.

4.4.3.2 LONG Width (const RECT & r) [inline]

Defined at 12 of file DrawDevice.cpp.

4.4.4 Parent Directory/DrawDevice.h File Reference

#include <d2d1.h>
#include <d2d1helper.h>
#include <dwrite.h>

4.4.4.1 Classes

- class **DrawDevice**

4.4.4.2 Functions

- template<class Interface > void **SafeRelease** (Interface *&pInterfaceToRelease)

4.4.4.3 Function Documentation

*4.4.4.3.1 template<class Interface > void SafeRelease (Interface *& pInterfaceToRelease) [inline]*

Defined at 51 of file DrawDevice.h.

4.4.5 Parent Directory/kinect.h File Reference

4.4.5.1 Classes

- class **temp**
- struct **temp::points**

4.4.6 Parent Directory/kinect_code.cpp File Reference

```
#include "StdAfx.h"
#include "kinect_code.h"
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
```

4.4.7 Parent Directory/kinect_code.h File Reference

4.4.7.1 Classes

- class **kinect_code**

4.4.8 Parent Directory/NuiImpl.cpp File Reference

```
#include "stdafx.h"
#include "SkeletalViewer.h"
#include "resource.h"
#include "kinect.h"
#include <mmsystem.h>
#include <assert.h>
#include <strsafe.h>
```

4.4.9 Parent Directory/resource.h File Reference

4.4.9.1 Defines

- #define **IDC_MYICON** 2
- #define **IDD_SKELETALVIEWER_DIALOG** 102
- #define **IDS_APP_TITLE** 103
- #define **IDD_ABOUTBOX** 103
- #define **IDM_ABOUT** 104
- #define **IDM_EXIT** 105
- #define **IDI_SKELETALVIEWER** 107
- #define **IDC_SKELETALVIEWER** 109
- #define **IDD_APP** 110
- #define **IDR_MAINFRAME** 128
- #define **IDS_APPTITLE** 129
- #define **IDS_ERROR_APP_INSTANCE** 130
- #define **IDS_ERROR_DRAWDEVICE** 131
- #define **IDS_ERROR_NUIINIT** 132
- #define **IDS_ERROR_SKELETONTRACKING** 133
- #define **IDS_ERROR_DEPTHSTREAM** 134
- #define **IDS_ERROR_VIDEOSTREAM** 135
- #define **IDS_ERROR_SETTRACKED** 136
- #define **IDS_ERROR_IN_USE** 140
- #define **IDS_ERROR_NUICREATE** 150
- #define **IDC_DEPTHVIEWER** 1001
- #define **IDC_SKELETALVIEW** 1002
- #define **IDC_VIDEOVIEW** 1003
- #define **IDC_FPS** 1004
- #define **IDC_STATUS** 1005
- #define **IDB_RECONNECT** 1007
- #define **IDC_CAMERAS** 1008
- #define **IDC_APPTRACKING** 1009
- #define **IDC_TRACK0** 1010
- #define **IDC_TRACK1** 1011
- #define **IDC_STATIC** -1

4.4.9.2 Define Documentation

4.4.9.2.1 #define IDB_RECONNECT 1007

Defined at 30 of file resource.h.

4.4.9.2.2 #define IDC_APPTRACKING 1009

Defined at 32 of file resource.h.

4.4.9.2.3 #define IDC_CAMERAS 1008

Defined at 31 of file resource.h.

4.4.9.2.4 #define IDC_DEPTHVIEWER 1001

Defined at 25 of file resource.h.

4.4.9.2.5 #define IDC_FPS 1004

Defined at 28 of file resource.h.

4.4.9.2.6 #define IDC_MYICON 2

Defined at 5 of file resource.h.

4.4.9.2.7 #define IDC_SKELETALVIEW 1002

Defined at 26 of file resource.h.

4.4.9.2.8 #define IDC_SKELETALVIEWER 109

Defined at 12 of file resource.h.

4.4.9.2.9 #define IDC_STATIC -1

Defined at 35 of file resource.h.

4.4.9.2.10 #define IDC_STATUS 1005

Defined at 29 of file resource.h.

4.4.9.2.11 #define IDC_TRACK0 1010

Defined at 33 of file resource.h.

4.4.9.2.12 #define IDC_TRACK1 1011

Defined at 34 of file resource.h.

4.4.9.2.13 #define IDC_VIDEOVIEW 1003

Defined at 27 of file resource.h.

4.4.9.2.14 #define IDD_ABOUTBOX 103

Defined at 8 of file resource.h.

4.4.9.2.15 #define IDD_APP 110

Defined at 13 of file resource.h.

4.4.9.2.16 #define IDD_SKELETALVIEWER_DIALOG 102

Defined at 6 of file resource.h.

4.4.9.2.17 #define IDI_SKELETALVIEWER 107

Defined at 11 of file resource.h.

4.4.9.2.18 #define IDM_ABOUT 104

Defined at 9 of file resource.h.

4.4.9.2.19 #define IDM_EXIT 105

Defined at 10 of file resource.h.

4.4.9.2.20 #define IDR_MAINFRAME 128

Defined at 14 of file resource.h.

4.4.9.2.21 #define IDS_APP_TITLE 103

Defined at 7 of file resource.h.

4.4.9.2.22 #define IDS_APPTITLE 129

Defined at 15 of file resource.h.

4.4.9.2.23 #define IDS_ERROR_APP_INSTANCE 130

Defined at 16 of file resource.h.

4.4.9.2.24 #define IDS_ERROR_DEPTHSTREAM 134

Defined at 20 of file resource.h.

4.4.9.2.25 #define IDS_ERROR_DRAWDEVICE 131

Defined at 17 of file resource.h.

4.4.9.2.26 #define IDS_ERROR_IN_USE 140

Defined at 23 of file resource.h.

4.4.9.2.27 #define IDS_ERROR_NUICREATE 150

Defined at 24 of file resource.h.

4.4.9.2.28 #define IDS_ERROR_NUIINIT 132

Defined at 18 of file resource.h.

4.4.9.2.29 #define IDS_ERROR_SETTRACKED 136

Defined at 22 of file resource.h.

4.4.9.2.30 #define IDS_ERROR_SKELETONTRACKING 133

Defined at 19 of file resource.h.

4.4.9.2.31 #define IDS_ERROR_VIDEOSTREAM 135

Defined at 21 of file resource.h.

4.4.10 Parent Directory/SkeletalViewer.cpp File Reference

```
#include "stdafx.h"
#include <strsafe.h>
#include "SkeletalViewer.h"
#include "resource.h"
#include <stdio.h>
#include <stdlib.h>
#include "GL/glut.h"
#include "kinect.h"
#include <conio.h>
#include <math.h>
#include <iostream>
#include <sstream>
#include "tga.h"
#include <io.h>
#include <fcntl.h>
```

4.4.10.1 Classes

- struct **vertex**
- struct **check**

4.4.10.2 Defines

- #define **INSTANCE_MUTEX_NAME** L"SkeletalViewerInstanceCheck"

4.4.10.3 Functions

- void **init** ()
- void **sphere_move** (int x, int y)
- void **triangle_move** (int x, int y)
- void **cube_move** (int x, int y)
- void **cube_scale** ()
- void **tri_scale** ()
- void **sph_scale** ()
- void **myMouseFunction** (int button, int state, int mouseX, int mouseY)
- void **myKeyboardFunction** (unsigned char key, int mouseX, int mouseY)
- void **output** (GLfloat x, GLfloat y, char *text)
- void **Reshape** (int width, int height)
- void **timer** (int val)
- void **display** ()
- void **drawCube** ()
- void **drawmouse** ()
- void **save_work** ()
- void **load_work** ()
- void **kinect_mouse** ()
- void **Draw_line** ()
- void **setvalues** ()

- void **checkcoordinates** ()
- void **increase_scale** ()
- void **decrease_scale** ()
- void **manage_scale** ()
- void **line_init** ()
- void **tri_init** ()
- void **cube_init** ()
- void **sphere_init** ()
- DWORD WINAPI **MyThreadFunction** (LPVOID lpParam)
- int APIENTRY **_tWinMain** (HINSTANCE hInstance, HINSTANCE hPrevInstance, LPTSTR lpCmdLine, int nCmdShow)
- void **upmenu** ()
- void **sidemenu** ()
- void **drawtriangle** ()
- void **drawcube1** ()
- void **drawcube** ()
- void **drawcheckbox** ()
- void **drawCircleOutline** ()
- void **drawsphere** ()

4.4.10.4 Variables

- char * **first_string**
- float **right_xposition**
- float **right_yposition**
- float **right_zposition**
- float **right_prev_xposition**
- float **right_prev_yposition**
- float **right_prev_zposition**
- float **left_xposition**
- float **left_yposition**
- float **left_zposition**
- float **left_prev_xposition**
- float **left_prev_yposition**
- float **left_prev_zposition**
- int **increment**
- bool **clicked**
- bool **stop_menu** = false
- char **g_SelectedColor** = 'w'
- int **g_Width**
- int **g_Height**
- float **light_diffuse** [] = { 0.8, 0.8, 0.8, 1.0 }
- float **light_ambient** [] = { 0.1, 0.1, 0.1, 1.0 }
- float **light_specular** [] = { 0.5, 0.5, 0.5, 1.0 }
- float **light_position** [] = { 300.0, 100.0, 0.0, 1.0 }
- float **screenMaterial** [4] = { 1.0, 1.0, 1.0, 1.0 }
- bool **create** = false

- bool **kinect_scale** = false
- bool **load** = false
- bool **movebtn** = false
- bool **scale** = false
- bool **move_flag** = false
- bool **save** = false
- bool **t_move** = false
- bool **c_move** = false
- bool **sp_move** = false
- bool **c_upmenu** = false
- bool **c_sidemenu** = false
- bool **line_display** = false
- int **upint** = 0
- int **sideint** = 0
- int **createid** = 8
- int **editid** = 7
- int **moveid** = 6
- int **saveid** = 5
- bool **myt_scale** = false
- bool **myc_scale** = false
- bool **mysp_scale** = false
- float **t_scale** = 1.2
- float **c_scale** = 1.2
- float **s_scale** = 1.2
- struct **vertex tri_vertex** [4]
- struct **vertex cube_vertex** [4]
- struct **vertex sphere_vertex** [1]
- struct **vertex line_vertex** [2]
- struct **check mycheck** [1]
- **CSkeletalViewerApp g_skeletalViewerApp**
- int **cx**
- int **cy**
- HINSTANCE **g_hInstance**
- int **g_nCmdShow**
- char * **test**
- bool **left_hand_check** = false
- int **move_count** = 1
- float **first_difference**
- float **second_difference**
- bool **kinect_scale2** = false
- int **frame_check** = 0

4.4.10.5 Define Documentation

4.4.10.5.1 #define INSTANCE_MUTEX_NAME L"SkeletalViewerInstanceCheck"

Defined at 465 of file SkeletalViewer.cpp.

4.4.10.6 Function Documentation

4.4.10.6.1 int APIENTRY _tWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, LPTSTR lpCmdLine, int nCmdShow)

: Entry point for the application

Defined at 484 of file SkeletalViewer.cpp.

4.4.10.6.2 void checkcoordinates ()

coordinates checking for the kinect mouse weather it is inside screen or outside tge screen

: Defination of Check Coordinates function which Check the Kinect Mouse Coordinates

Defined at 1936 of file SkeletalViewer.cpp.

4.4.10.6.3 void cube_init ()

Defination of Initilization of Cube Function

Defined at 431 of file SkeletalViewer.cpp.

4.4.10.6.4 void cube_move (int x, int y)

- x coordinate - y coordinate move the cube to x ,y coordinate

: Defination of Cube Move function which Move the cube to x , y coordinates - xcoordinates - ycoordinates

Defined at 1582 of file SkeletalViewer.cpp.

4.4.10.6.5 void cube_scale ()

increase or decrease the size of cube

: Defination of Cube scale function which increase or decrease the size of Cube

Defined at 1544 of file SkeletalViewer.cpp.

4.4.10.6.6 void decrease_scale ()

Decrease the size of selected object

: Defination of Decrease size Function

Defined at 1274 of file SkeletalViewer.cpp.

4.4.10.6.7 void display ()

display function that draw the any thing shown on the screen

: Defination of Diplay function which Display all stuff on the screen

Defined at 1810 of file SkeletalViewer.cpp.

4.4.10.6.8 void Draw_line ()

Draw the line below the button when it is clicked

: Defination of Draw Line function which Draw Line on the screen

Defined at 1784 of file SkeletalViewer.cpp.

4.4.10.6.9 void drawcheckbox ()

: Defination of Draw Check Box function which Draw Check Box on the screen

Defined at 1710 of file SkeletalViewer.cpp.

4.4.10.6.10 void drawCircleOutline ()

: Defination of Draw Sphere function which Draw Sphere on the screen

Defined at 1741 of file SkeletalViewer.cpp.

4.4.10.6.11 void drawCube ()

Draw the cube on the screen

4.4.10.6.12 void drawcube ()

: Defination of Draw Cube function which Draw Cube on the screen

Defined at 1685 of file SkeletalViewer.cpp.

4.4.10.6.13 void drawcube1 ()

: Defination of Draw Cube function which Draw Cube on the screen

Defined at 1661 of file SkeletalViewer.cpp.

4.4.10.6.14 drawmouse ()

Draw the mouse pointer for kinect on the screen

: Defination of Draw Mouse function which Draw Kinect Mouse on the screen

Defined at 1862 of file SkeletalViewer.cpp.

4.4.10.6.15 void drawsphere ()

: Defination of Draw Sphere function which Draw Sphere on the screen

Defined at 1772 of file SkeletalViewer.cpp.

4.4.10.6.16 void drawtriangle ()

: Defination of Draw Triangle function which Draw triangle on the screen

Defined at 1641 of file SkeletalViewer.cpp.

4.4.10.6.17 void increase_scale ()

Increase the size of selected object

: Defination of Increase size Function

Defined at 1296 of file SkeletalViewer.cpp.

4.4.10.6.18 void init (void)

Initilization function for opengl when window initilized

of Initilization Function of openGL

Defined at 925 of file SkeletalViewer.cpp.

4.4.10.6.19 void kinect_mouse ()

handle fuction for kinect mouse

: Defination of Kinect Mouse Handler

Defined at 1963 of file SkeletalViewer.cpp.

4.4.10.6.20 void line_init ()

Defination of Initilization of line Function

Defined at 397 of file SkeletalViewer.cpp.

4.4.10.6.21 void load_work ()

Load the work

Defination of Load Work Function

Defined at 357 of file SkeletalViewer.cpp.

4.4.10.6.22 void manage_scale ()

Model function for increase ofr decrease size of object

: Defination of Manage scale function which decide weather call increase scale or decrease scale called

Defined at 2190 of file SkeletalViewer.cpp.

4.4.10.6.23 void myKeyboardFunction (unsigned char key, int mouseX, int mouseY)

get the key when any key pressed from keyboard and perform the task accordingly

: Defination of Keyboard Function of openGL : which key was presses : where was mouse pointer in x-axis when key was pressed : where was mouse pointer in y-axis when key was pressed

Defined at 1224 of file SkeletalViewer.cpp.

4.4.10.6.24 void myMouseFunction (int button, int state, int mouseX, int mouseY)

left or right button state of button - x coordinate of click - y coordinate of click get the coordinates when mouse clicked perform the task accordingly

: Defination of Mouse Function of openGL : which button was presses, left/right/middle : sate of button, still pressed or released : position of mouse pointer in x-axis : position of mouse pointer in y-axis

Defined at 1006 of file SkeletalViewer.cpp.

4.4.10.6.25 DWORD WINAPI MyThreadFunction (LPVOID lpParam)

Defined at 549 of file SkeletalViewer.cpp.

*4.4.10.6.26 void output (GLfloat x, GLfloat y, char * text)*

- xcoordinate of text - ycoordinate of text pointer to text to be displayed display the text on x,y coordinate

: Function which is used to draw text on screen xcoordinate ycoordinate Text to be displayed

Defined at 2255 of file SkeletalViewer.cpp.

4.4.10.6.27 void Reshape (int width, int height)

Reshape function of opengl when window size increase or decreased

: Defination of Reshape Function of openGL

Defined at 1319 of file SkeletalViewer.cpp.

4.4.10.6.28 void save_work ()

Save the work

Defination of Save Work Function

Defined at 377 of file SkeletalViewer.cpp.

4.4.10.6.29 void setvalues ()

set the values of left and righjt hand taken from kinect after some manuplation

: Defination of Set Values function which Set the coordinates of Kinect Mouse

Defined at 1890 of file SkeletalViewer.cpp.

4.4.10.6.30 void sidemenu ()

: Defination of sidemenu Function which draw the side menu

Defined at 1435 of file SkeletalViewer.cpp.

4.4.10.6.31 void sph_scale ()

increase or decrease the size of sphere

: Defination of Sphere scale function which increase or decrease the size of sphere

Defined at 1495 of file SkeletalViewer.cpp.

4.4.10.6.32 void sphere_init ()

Defination of Initilization of Sphere Function

Defined at 455 of file SkeletalViewer.cpp.

4.4.10.6.33 void sphere_move (int x, int y)

- x coordinate - y coordinate move the sphere to x ,y coordinate

: Defination of Sphere Move function which Move the Sphere to x , y coordinates - xcoordinates - ycoordinates

Defined at 1608 of file SkeletalViewer.cpp.

4.4.10.6.34 void timer (int val)

Timer function of opengl how many tinmes called the function in second

: Defination of Timer Function : value of timer in milli seconds

Defined at 1340 of file SkeletalViewer.cpp.

4.4.10.6.35 void tri_init ()

Defination of Initilization of Triangle Function

Defined at 411 of file SkeletalViewer.cpp.

4.4.10.6.36 void tri_scale ()

increase or decrease the size of Triangle

: Defination of Triangle scale function which increase or decrease the size of Triangle

Defined at 1504 of file SkeletalViewer.cpp.

4.4.10.6.37 void triangle_move (int x, int y)

- x coordinate - y coordinate move the triangle to x ,y coordinate

: Defination of Triangle Move function which Move the Triangle to x , y coordinates - xcoordinates - ycoordinates

Defined at 1620 of file SkeletalViewer.cpp.

4.4.10.6.38 void upmenu ()

: Defination of upmenu Function which draw the upper menu

Defined at 1349 of file SkeletalViewer.cpp.

4.4.10.7 Variable Documentation

4.4.10.7.1 bool c_move = false

Defined at 307 of file SkeletalViewer.cpp.

4.4.10.7.2 float c_scale = 1.2

Defined at 318 of file SkeletalViewer.cpp.

4.4.10.7.3 bool c_sidemenu = false

Defined at 309 of file SkeletalViewer.cpp.

4.4.10.7.4 bool c_upmenu = false

Defined at 309 of file SkeletalViewer.cpp.

4.4.10.7.5 bool clicked

Store the click state through kinect

Defined at 103 of file SkeletalViewer.cpp.

4.4.10.7.6 bool create = false

Different flags used in opengl for checking of differents values in opengl

Defined at 305 of file SkeletalViewer.cpp.

4.4.10.7.7 int createid = 8

Defined at 313 of file SkeletalViewer.cpp.

4.4.10.7.8 struct vertex cube_vertex[4]

vertices for cube

Defined at 330 of file SkeletalViewer.cpp.

4.4.10.7.9 int cx

Defined at 466 of file SkeletalViewer.cpp.

4.4.10.7.10 int cy

Defined at 467 of file SkeletalViewer.cpp.

4.4.10.7.11 int editid = 7

Defined at 313 of file SkeletalViewer.cpp.

4.4.10.7.12 float first_difference

difference which stores current difference between two hand Difference which stores previous difference between two hand scale2 used as aflag check how long position stable after this position discarded

Defined at 2181 of file SkeletalViewer.cpp.

4.4.10.7.13 char first_string*

Store the message stings to be displyed on screen

Defined at 44 of file SkeletalViewer.cpp.

4.4.10.7.14 int frame_check = 0

Defined at 2184 of file SkeletalViewer.cpp.

4.4.10.7.15 int g_Height

Defined at 138 of file SkeletalViewer.cpp.

4.4.10.7.16 HINSTANCE g_hInstance

Defined at 475 of file SkeletalViewer.cpp.

4.4.10.7.17 int g_nCmdShow

Defined at 476 of file SkeletalViewer.cpp.

4.4.10.7.18 char g_SelectedColor = 'w'

Defined at 136 of file SkeletalViewer.cpp.

4.4.10.7.19 CSkeletalViewerApp g_skeletalViewerApp

Defined at 463 of file SkeletalViewer.cpp.

4.4.10.7.20 int g_Width

Defined at 137 of file SkeletalViewer.cpp.

4.4.10.7.21 int increment

Store the how many times u or s pressed

Defined at 97 of file SkeletalViewer.cpp.

4.4.10.7.22 bool kinect_scale = false

Defined at 305 of file SkeletalViewer.cpp.

4.4.10.7.23 bool kinect_scale2 = false

Defined at 2183 of file SkeletalViewer.cpp.

4.4.10.7.24 bool left_hand_check = false

Defined at 1955 of file SkeletalViewer.cpp.

4.4.10.7.25 float left_prev_xposition

Store the previous coordinates of left hand (10 frames earlier)

Defined at 89 of file SkeletalViewer.cpp.

4.4.10.7.26 float left_prev_yposition

Defined at 90 of file SkeletalViewer.cpp.

4.4.10.7.27 float left_prev_zposition

Defined at 91 of file SkeletalViewer.cpp.

4.4.10.7.28 float left_xposition

Store the current coordinates of left hand

Defined at 78 of file SkeletalViewer.cpp.

4.4.10.7.29 float left_yposition

Defined at 79 of file SkeletalViewer.cpp.

4.4.10.7.30 float left_zposition

Defined at 80 of file SkeletalViewer.cpp.

4.4.10.7.31 float light_ambient[] = { 0.1, 0.1, 0.1, 1.0 }

Defined at 296 of file SkeletalViewer.cpp.

4.4.10.7.32 float light_diffuse[] = { 0.8, 0.8, 0.8, 1.0 }

Light Parameters

Defined at 295 of file SkeletalViewer.cpp.

4.4.10.7.33 float light_position[] = { 300.0, 100.0, 0.0, 1.0 }

Defined at 298 of file SkeletalViewer.cpp.

4.4.10.7.34 float light_specular[] = { 0.5, 0.5, 0.5, 1.0 }

Defined at 297 of file SkeletalViewer.cpp.

4.4.10.7.35 bool line_display = false

Defined at 309 of file SkeletalViewer.cpp.

4.4.10.7.36 struct vertex line_vertex[2]

vertices for Line

Defined at 342 of file SkeletalViewer.cpp.

4.4.10.7.37 bool load = false

Defined at 305 of file SkeletalViewer.cpp.

4.4.10.7.38 int move_count = 1

Defined at 1956 of file SkeletalViewer.cpp.

4.4.10.7.39 bool move_flag = false

Defined at 305 of file SkeletalViewer.cpp.

4.4.10.7.40 bool movebtn = false

Defined at 305 of file SkeletalViewer.cpp.

4.4.10.7.41 int moveid = 6

Defined at 313 of file SkeletalViewer.cpp.

4.4.10.7.42 bool myc_scale = false

Defined at 316 of file SkeletalViewer.cpp.

4.4.10.7.43 struct check mycheck[1]

Checking of selected object

Defined at 348 of file SkeletalViewer.cpp.

4.4.10.7.44 bool mysp_scale = false

Defined at 316 of file SkeletalViewer.cpp.

4.4.10.7.45 bool myt_scale = false

Defined at 316 of file SkeletalViewer.cpp.

4.4.10.7.46 float right_prev_xposition

Store the previous coordinates of right hand (10 frames earlier)

Defined at 65 of file SkeletalViewer.cpp.

4.4.10.7.47 float right_prev_yposition

Defined at 66 of file SkeletalViewer.cpp.

4.4.10.7.48 float right_prev_zposition

Defined at 67 of file SkeletalViewer.cpp.

4.4.10.7.49 float right_xposition

Store the current coordinates of right hand

Defined at 54 of file SkeletalViewer.cpp.

4.4.10.7.50 float right_yposition

Defined at 55 of file SkeletalViewer.cpp.

4.4.10.7.51 float right_zposition

Defined at 56 of file SkeletalViewer.cpp.

4.4.10.7.52 float s_scale = 1.2

Defined at 318 of file SkeletalViewer.cpp.

4.4.10.7.53 bool save = false

Defined at 305 of file SkeletalViewer.cpp.

4.4.10.7.54 int saveid = 5

Defined at 313 of file SkeletalViewer.cpp.

4.4.10.7.55 bool scale = false

Defined at 305 of file SkeletalViewer.cpp.

4.4.10.7.56 float screenMaterial[4] = { 1.0, 1.0, 1.0, 1.0 }

Defined at 299 of file SkeletalViewer.cpp.

4.4.10.7.57 float second_difference

Defined at 2182 of file SkeletalViewer.cpp.

4.4.10.7.58 int sideint = 0

Defined at 311 of file SkeletalViewer.cpp.

4.4.10.7.59 bool sp_move = false

Defined at 307 of file SkeletalViewer.cpp.

4.4.10.7.60 struct vertex sphere_vertex[1]

vertices for sphere

Defined at 336 of file SkeletalViewer.cpp.

4.4.10.7.61 bool stop_menu = false

Store the menu display state through kinect

Defined at 110 of file SkeletalViewer.cpp.

4.4.10.7.62 bool t_move = false

Defined at 307 of file SkeletalViewer.cpp.

4.4.10.7.63 float t_scale = 1.2

Defined at 318 of file SkeletalViewer.cpp.

4.4.10.7.64 char test*

4.4.10.7.65 Test:

which purpose is testing

Defined at 1803 of file SkeletalViewer.cpp.

4.4.10.7.66 struct vertex tri_vertex[4]

vertices for triangle

Defined at 324 of file SkeletalViewer.cpp.

4.4.10.7.67 int upint = 0

Defined at 311 of file SkeletalViewer.cpp.

4.4.11 Parent Directory/SkeletalViewer.h File Reference

#include "resource.h"
#include "NuiApi.h"
#include "DrawDevice.h"

4.4.11.1 Classes

- class **CSkeletalViewerApp**

4.4.11.2 Defines

- #define **SZ_APPDLG_WINDOW_CLASS** _T("SkeletalViewerAppDlgWndClass")
- #define **WM_USER_UPDATE_FPS** WM_USER
- #define **WM_USER_UPDATE_COMBO** WM_USER+1
- #define **WM_USER_UPDATE_TRACKING_COMBO** WM_USER+2

4.4.11.3 Define Documentation

4.4.11.3.1 #define SZ_APPDLG_WINDOW_CLASS _T("SkeletalViewerAppDlgWndClass")

Defined at 15 of file SkeletalViewer.h.

4.4.11.3.2 #define WM_USER_UPDATE_COMBO WM_USER+1

Defined at 17 of file SkeletalViewer.h.

4.4.11.3.3 #define WM_USER_UPDATE_FPS WM_USER

Defined at 16 of file SkeletalViewer.h.

4.4.11.3.4 #define WM_USER_UPDATE_TRACKING_COMBO WM_USER+2

Defined at 18 of file SkeletalViewer.h.

4.4.12 Parent Directory/stdafx.cpp File Reference

#include "stdafx.h"

4.4.13 Parent Directory/stdafx.h File Reference

#include "targetver.h"
#include <windows.h>
#include <ole2.h>
#include <stdlib.h>
#include <malloc.h>
#include <memory.h>
#include <tchar.h>
#include <d2d1.h>
#include <d2d1helper.h>
#include <dwrite.h>

4.4.13.1 Defines

- #define **WIN32_LEAN_AND_MEAN**

4.4.13.2 Define Documentation

4.4.13.2.1 #define WIN32_LEAN_AND_MEAN

Defined at 13 of file stdafx.h.

4.4.14 Parent Directory/targetver.h File Reference

```
#include <SDKDDKVer.h>
```

4.4.15 Parent Directory/tga.cpp File Reference

```
#include "stdafx.h"
#include "tga.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>
#include <gl/gl.h>
```

4.4.15.1 Functions

- int **checkSize** (int x)
- unsigned char * **getRGBA** (FILE *s, int size)
- unsigned char * **getRGB** (FILE *s, int size)
- unsigned char * **getGray** (FILE *s, int size)
- char * **getData** (FILE *s, int sz, int iBits)
- int **returnError** (FILE *s, int error)
- int **loadTGA** (char *name, int id)

4.4.15.2 Variables

- GLenum **texFormat**

4.4.15.3 Function Documentation

4.4.15.3.1 int checkSize (int x)

: Make sure its a power of 2. : size in bytes

Defined at 29 of file tga.cpp.

4.4.15.3.2 char getData (FILE * s, int sz, int iBits)*

: Gets the image data for the specified bit depth. @*s holds the address of the file in memory : holds the size of the file

Defined at 145 of file tga.cpp.

4.4.15.3.3 unsigned char getGray (FILE * s, int size)*

: Gets the grayscale image data. Used as an alpha channel. @*s holds the address of the file in memory : holds the size of the file

Defined at 118 of file tga.cpp.

4.4.15.3.4 unsigned char getRGB (FILE * s, int size)*

: Reads in RGB data for a 24bit image. @*s holds the address of the file in memory : holds the size of the file

Defined at 81 of file tga.cpp.

4.4.15.3.5 unsigned char getRGBA (FILE * s, int size)*

: Reads in RGBA data for a 32bit image. @*s holds the address of the file in memory : holds the size of the file

Defined at 45 of file tga.cpp.

4.4.15.3.6 *int loadTGA (char * name, int id)*

: Loads up a targa file. Supported types are 8,24 and 32 uncompressed images. id is the texture ID to bind too. @*s holds the address of the file in memory : holds the size of the file

Defined at 171 of file tga.cpp.

4.4.15.3.7 *int returnError (FILE * s, int error)*

: Called when there is an error loading the .tga file. @*s holds the address of the file in memory : holds the size of the erronious file

Defined at 161 of file tga.cpp.

4.4.15.4 Variable Documentation

4.4.15.4.1 *GLenum texFormat*

Defined at 23 of file tga.cpp.

4.4.16 Parent Directory/tga.h File Reference

4.4.16.1 Defines

- #define **TGA_FILE_NOT_FOUND** 13 /* file was not found */
- #define **TGA_BAD_IMAGE_TYPE** 14 /* color mapped image or image is not uncompressed */
- #define **TGA_BAD_DIMENSION** 15 /* dimension is not a power of 2 */
- #define **TGA_BAD_BITS** 16 /* image bits is not 8, 24 or 32 */
- #define **TGA_BAD_DATA** 17 /* image data could not be loaded */

4.4.16.2 Functions

- int **loadTGA** (char *name, int id)

4.4.16.3 Define Documentation

4.4.16.3.1 #define TGA_BAD_BITS 16 / image bits is not 8, 24 or 32 */*

Defined at 15 of file tga.h.

4.4.16.3.2 #define TGA_BAD_DATA 17 / image data could not be loaded */*

Defined at 16 of file tga.h.

4.4.16.3.3 #define TGA_BAD_DIMENSION 15 / dimension is not a power of 2 */*

Defined at 14 of file tga.h.

4.4.16.3.4 #define TGA_BAD_IMAGE_TYPE 14 / color mapped image or image is not uncompressed */*

Defined at 13 of file tga.h.

4.4.16.3.5 #define TGA_FILE_NOT_FOUND 13 / file was not found */*

Error Codes

Defined at 12 of file tga.h.

4.4.16.4 Function Documentation

*4.4.16.4.1 int loadTGA (char * name, int id)*

: Loads up a targa file. Supported types are 8,24 and 32 uncompressed images. id is the texture ID to bind too. *s holds the address of the file in memory : holds the size of the file

Defined at 171 of file tga.cpp.

Chapter 5

Discussion

This project was hardware dependent and there are limitations in hardware. These limitations in hardware cause problems while testing the firmware. We had to face a lot of difficulties while performing functions such as depth buffering. For example Nvidia and Intel GPU's do not have good support with reading depth buffer for Open GL. The presence of a specific kind of hardware causes the problem of performing all the test cases.

As this is a research based project we faced a lot of difficulties during the development of different modules. First of all there was a problem to measure the depth of 3D object. There are several techniques for doing this but they were not suitable with the functionality of this project. After a lot of research and discussion we came up with solution that we can write its value frame buffer and read from there. Also in depth buffering there are many hardware compatibility related issues.

Another problem we face yet is that the movement of 3D objects in the drawing console. Previously we had done it by dragging the object from 3D object menu to the drawing console. but it had problem in integration with kinect because kinect doesn't support dragging. We came up with solution by selecting the object with mouse cursor and then click the cursor on the drawing console through this an object can easily draw. This is done by event handling in openly. And this technique fulfills the requirements of kinect.

Here are some snapshots of the version 1.0 of 3D Modeler:

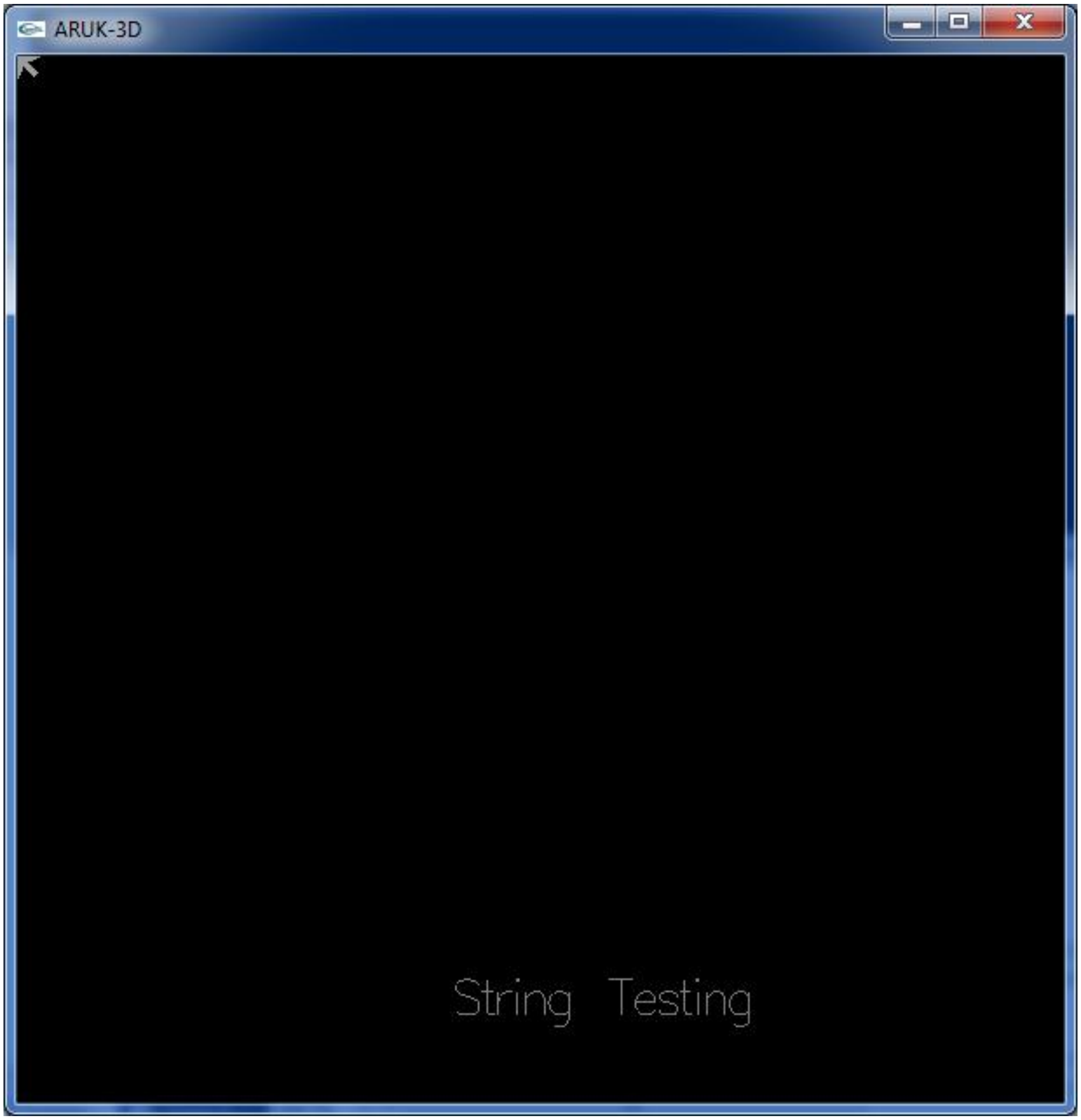


Figure 6

Main Screen of ARUK-3D

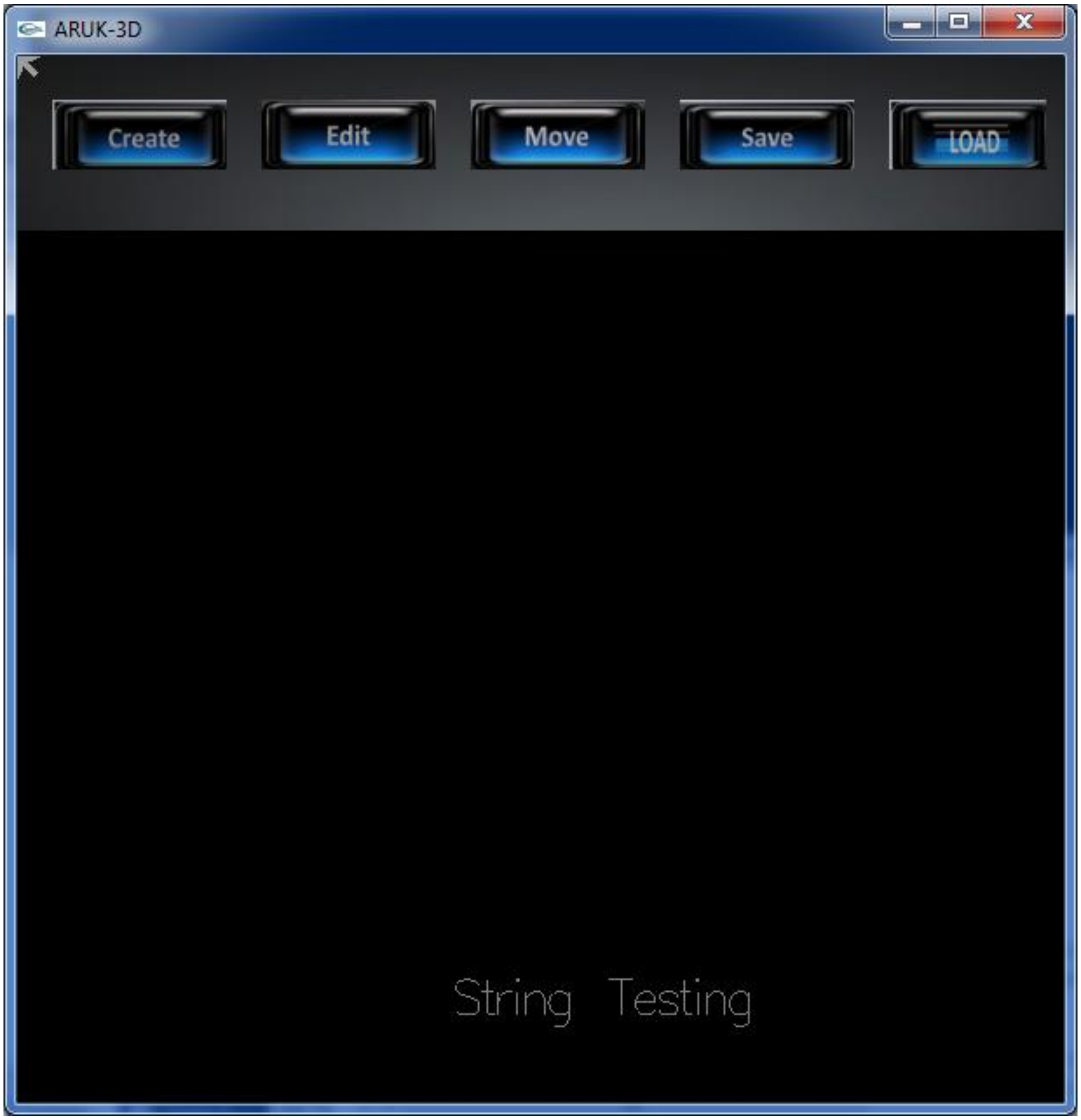


Figure 7

Upper Menu of ARUK-3D

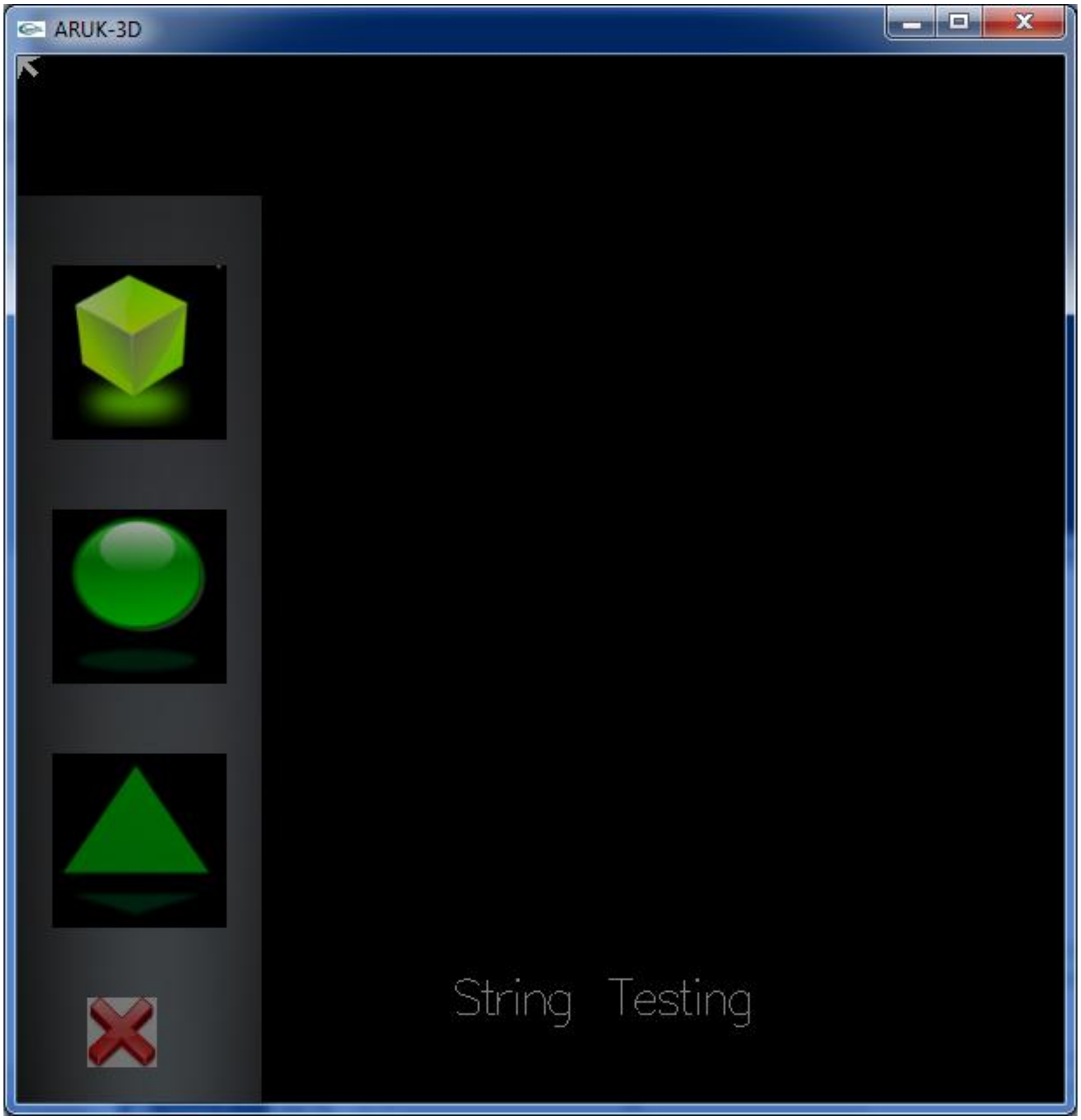


Figure 8

Side Menu of ARUK-3D

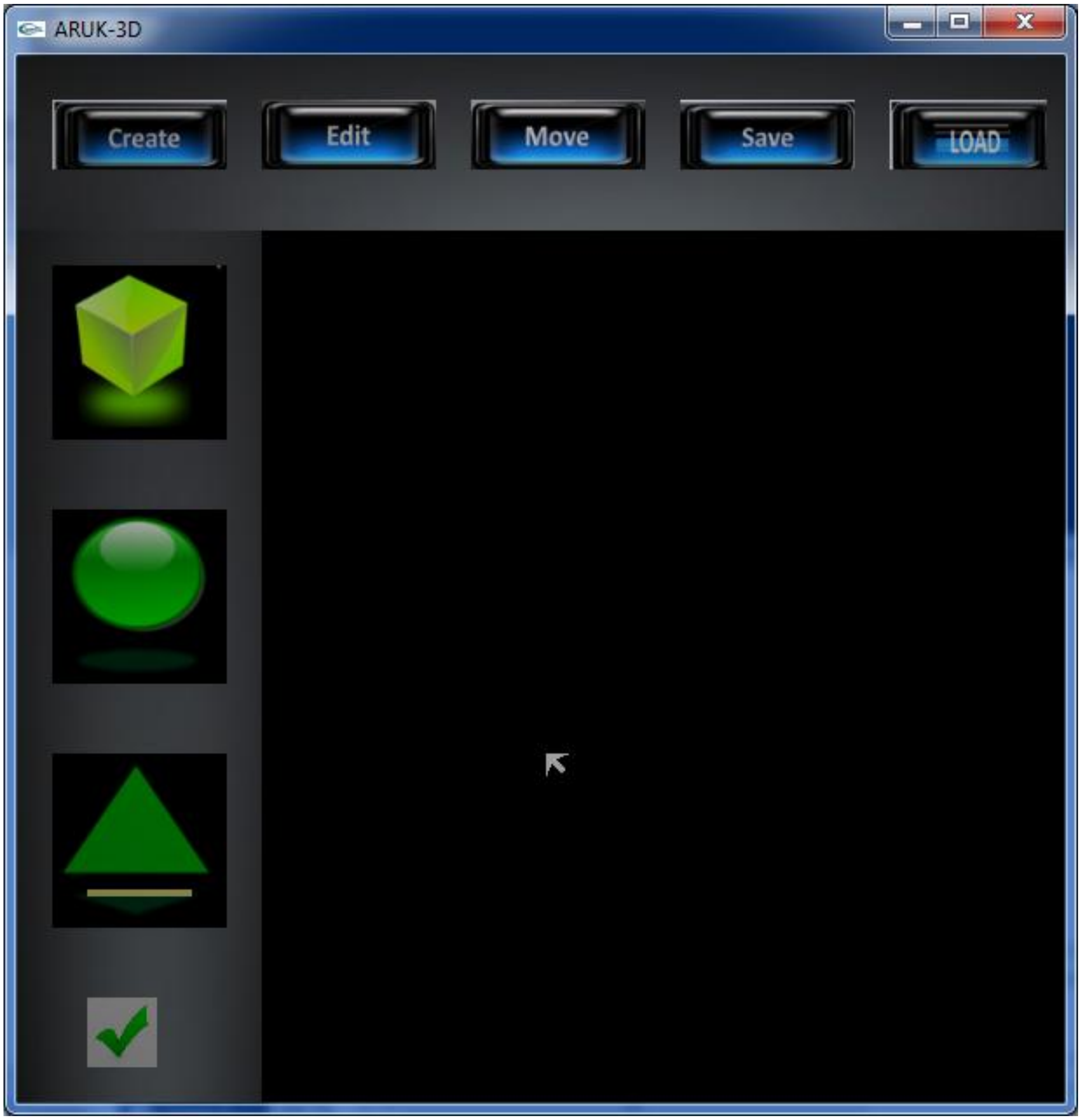


Figure 9

Complete Menus

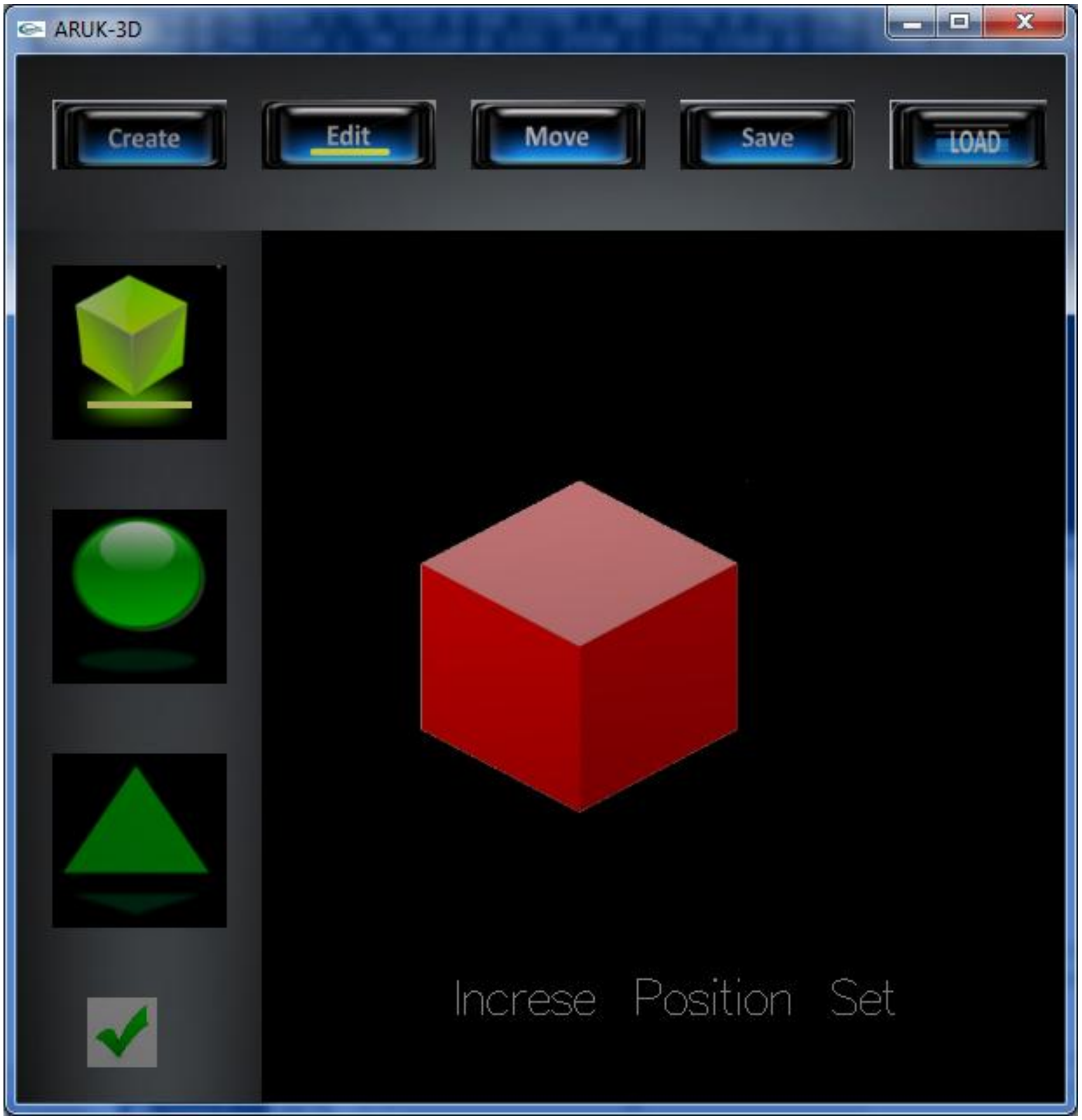


Figure 10

3D Object Manipulation

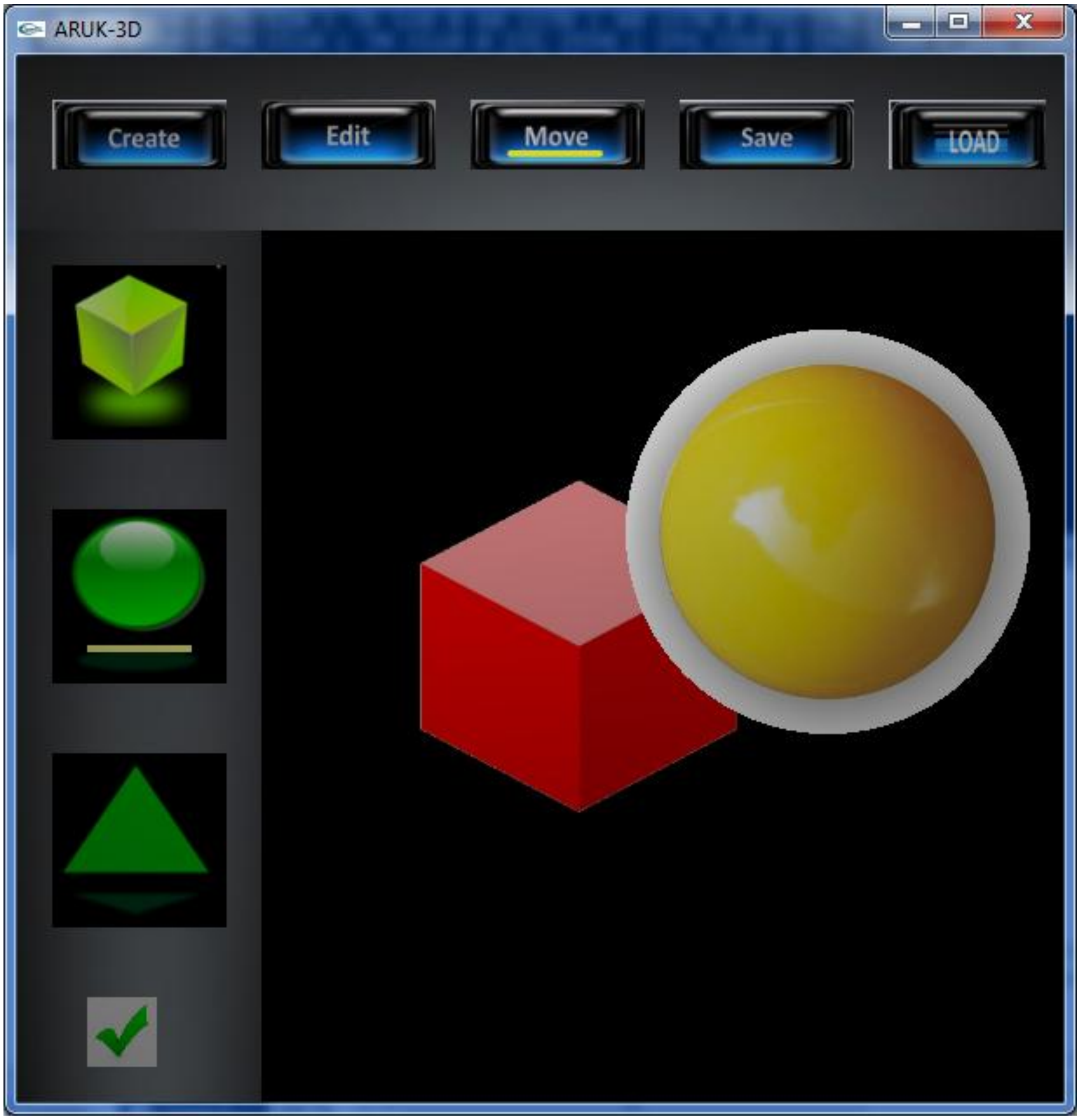


Figure 11

Editing Multiple 3D Objects

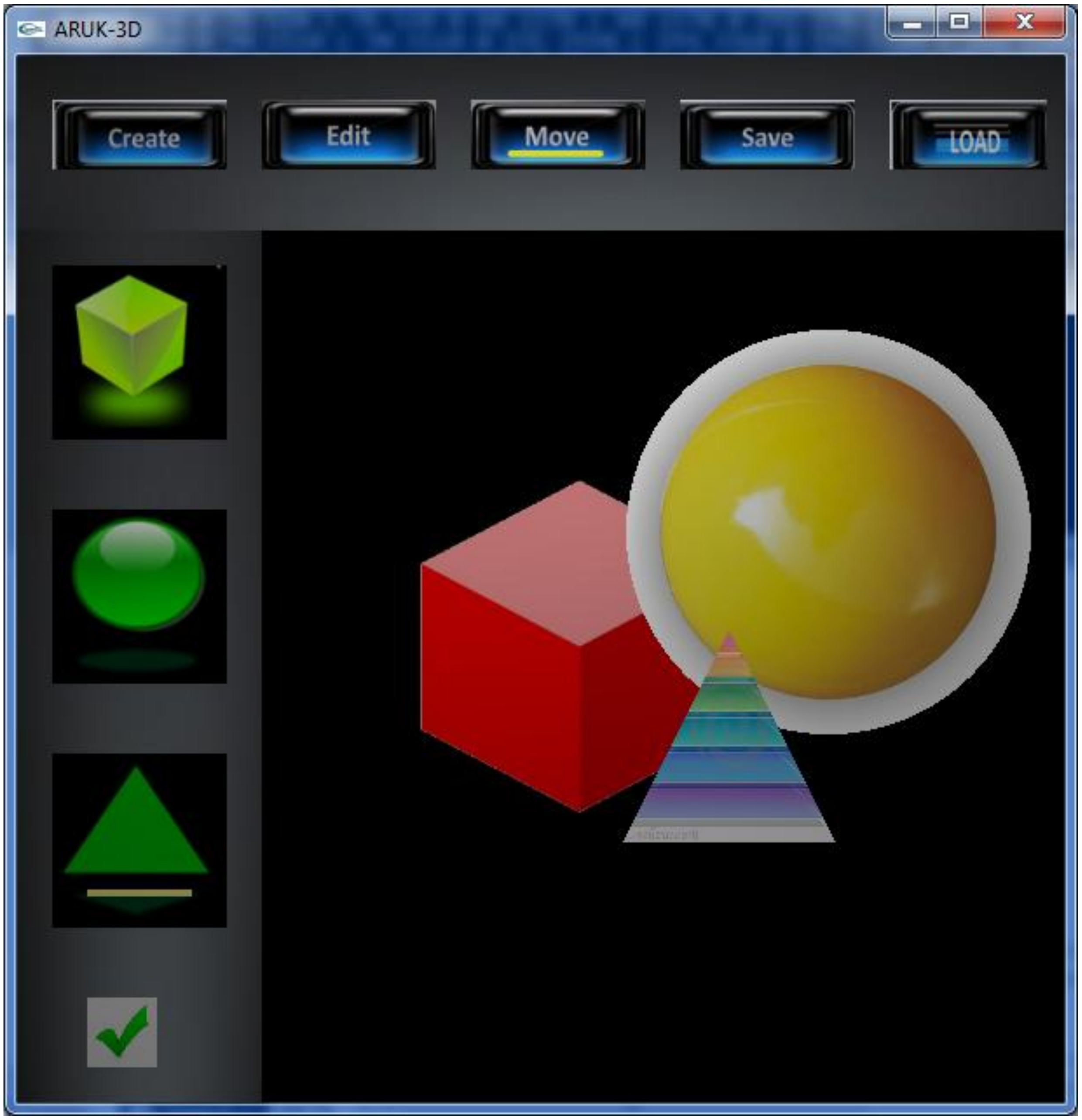
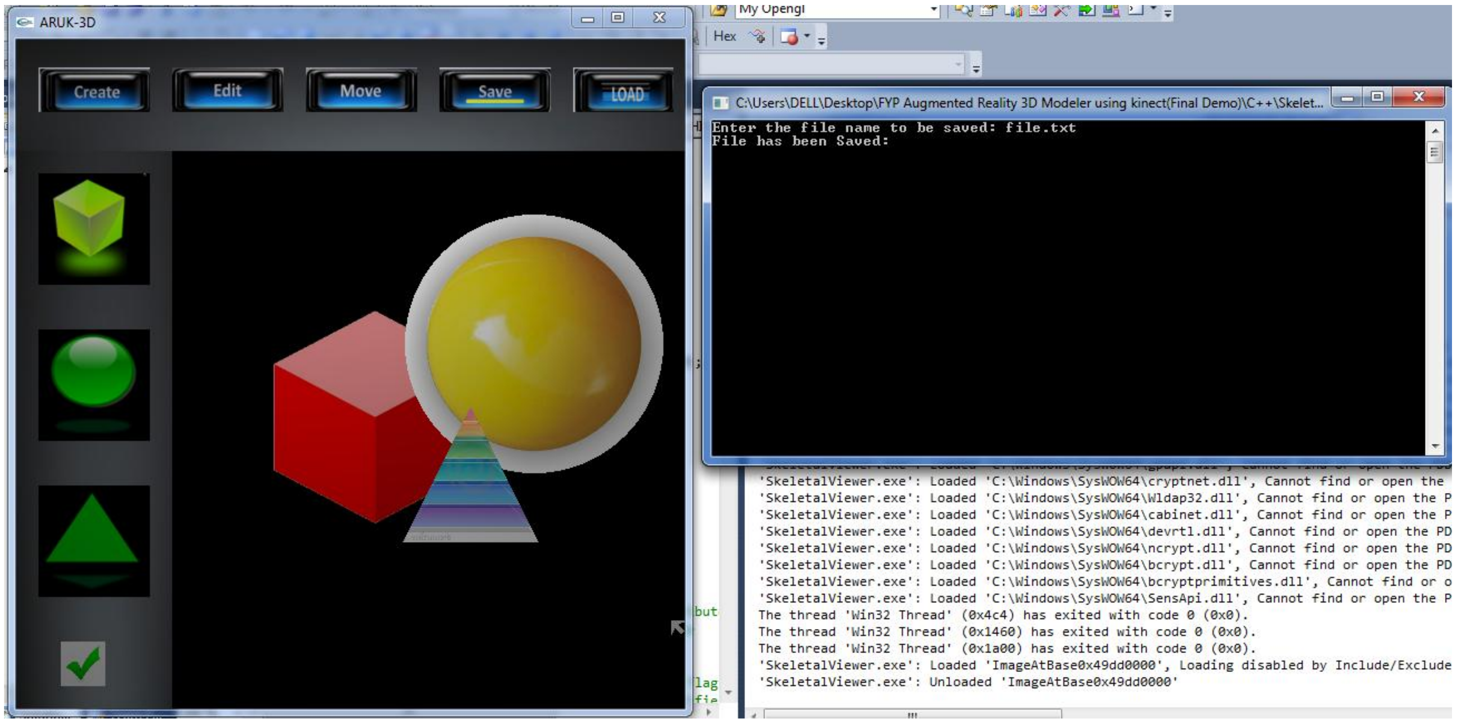


Figure 12

Manipulation of multiple Objects

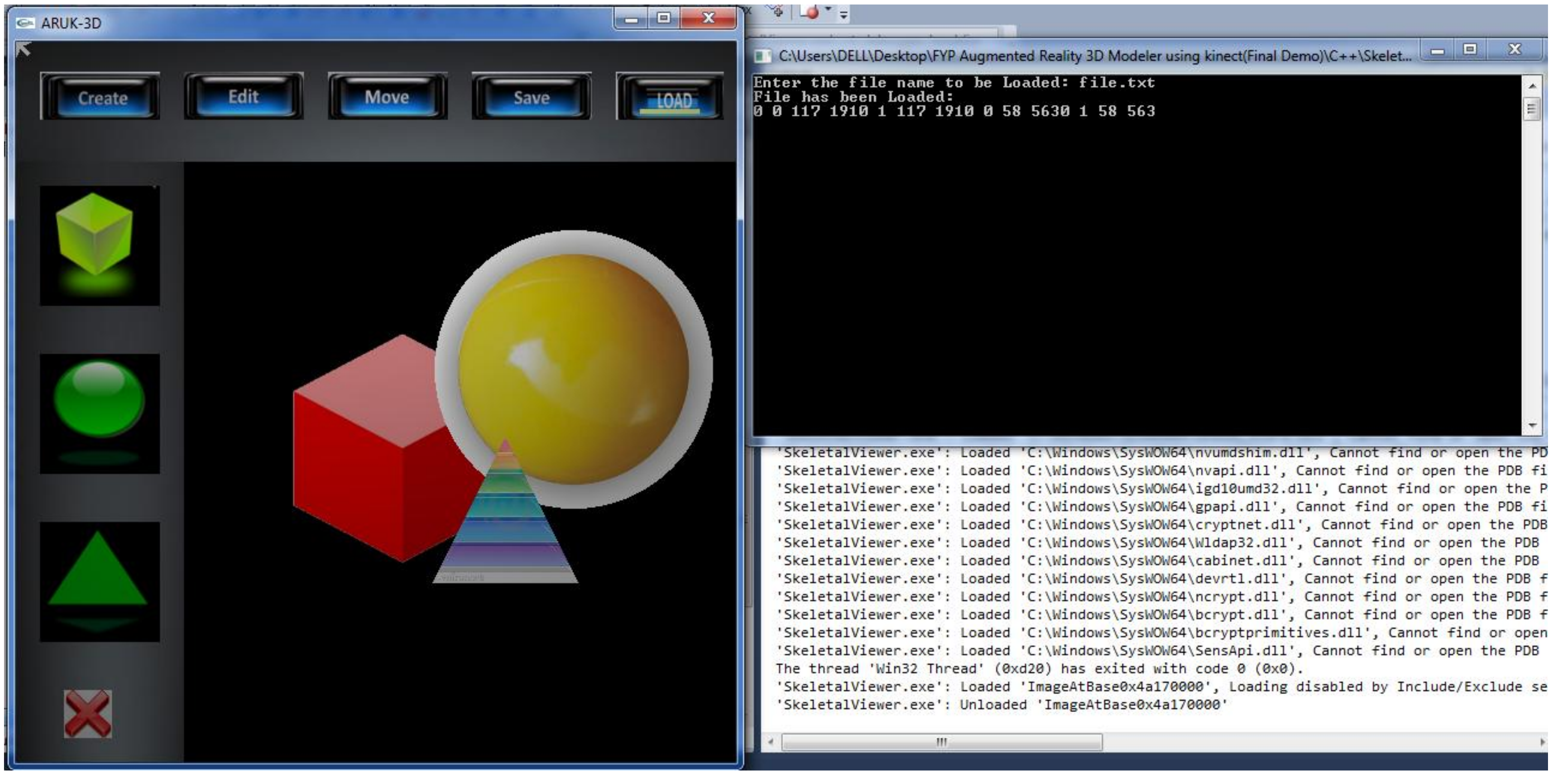

ARUK-3D
Create
Edit
Move
Save
LOAD
My Opengl
Hex
C:\Users\DELL\Desktop\FYP Augmented Reality 3D Modeler using kinect(Final Demo)\C++\Skelet...

```
Enter the file name to be saved: file.txt
File has been Saved:
```

```
'SkeletalViewer.exe': Loaded 'C:\Windows\SysWOW64\cryptnet.dll', Cannot find or open the
'SkeletalViewer.exe': Loaded 'C:\Windows\SysWOW64\Wldap32.dll', Cannot find or open the P
'SkeletalViewer.exe': Loaded 'C:\Windows\SysWOW64\cabinet.dll', Cannot find or open the P
'SkeletalViewer.exe': Loaded 'C:\Windows\SysWOW64\devrtl.dll', Cannot find or open the PD
'SkeletalViewer.exe': Loaded 'C:\Windows\SysWOW64\ncrypt.dll', Cannot find or open the PD
'SkeletalViewer.exe': Loaded 'C:\Windows\SysWOW64\bcrypt.dll', Cannot find or open the PD
'SkeletalViewer.exe': Loaded 'C:\Windows\SysWOW64\bcryptprimitives.dll', Cannot find or o
'SkeletalViewer.exe': Loaded 'C:\Windows\SysWOW64\SensApi.dll', Cannot find or open the P
The thread 'Win32 Thread' (0x4c4) has exited with code 0 (0x0).
The thread 'Win32 Thread' (0x1460) has exited with code 0 (0x0).
The thread 'Win32 Thread' (0x1a00) has exited with code 0 (0x0).
'SkeletalViewer.exe': Loaded 'ImageAtBase0x49dd0000', Loading disabled by Include/Exclude
'SkeletalViewer.exe': Unloaded 'ImageAtBase0x49dd0000'
```

Figure 13

Saving a File


ARUK-3D
Create
Edit
Move
Save
LOAD
C:\Users\DELL\Desktop\FYP Augmented Reality 3D Modeler using kinect(Final Demo)\C++\Skelet...

```
Enter the file name to be Loaded: file.txt
File has been Loaded:
0 0 117 1910 1 117 1910 0 58 5630 1 58 563
```

```
'SkeletalViewer.exe': Loaded 'C:\Windows\SysWOW64\nvumdshim.dll', Cannot find or open the PD
'SkeletalViewer.exe': Loaded 'C:\Windows\SysWOW64\nvapi.dll', Cannot find or open the PDB fi
'SkeletalViewer.exe': Loaded 'C:\Windows\SysWOW64\igd10umd32.dll', Cannot find or open the P
'SkeletalViewer.exe': Loaded 'C:\Windows\SysWOW64\gpapi.dll', Cannot find or open the PDB fi
'SkeletalViewer.exe': Loaded 'C:\Windows\SysWOW64\cryptnet.dll', Cannot find or open the PDB
'SkeletalViewer.exe': Loaded 'C:\Windows\SysWOW64\Wldap32.dll', Cannot find or open the PDB
'SkeletalViewer.exe': Loaded 'C:\Windows\SysWOW64\cabinet.dll', Cannot find or open the PDB
'SkeletalViewer.exe': Loaded 'C:\Windows\SysWOW64\devrtl.dll', Cannot find or open the PDB f
'SkeletalViewer.exe': Loaded 'C:\Windows\SysWOW64\ncrypt.dll', Cannot find or open the PDB f
'SkeletalViewer.exe': Loaded 'C:\Windows\SysWOW64\bcrypt.dll', Cannot find or open the PDB f
'SkeletalViewer.exe': Loaded 'C:\Windows\SysWOW64\bcryptprimitives.dll', Cannot find or open
'SkeletalViewer.exe': Loaded 'C:\Windows\SysWOW64\SensApi.dll', Cannot find or open the PDB
The thread 'Win32 Thread' (0xd20) has exited with code 0 (0x0).
'SkeletalViewer.exe': Loaded 'ImageAtBase0x4a170000', Loading disabled by Include/Exclude se
'SkeletalViewer.exe': Unloaded 'ImageAtBase0x4a170000'
```

Figure 14

Loading a File

Chapter 6

Conclusion

We have designed a 3D Modeler through which students can learn by creating different 3D models using the Kinect device. This 3D Modeler is simple to use and requires a little time to familiarize with. It provides different shapes which the students can manipulate to design different 3D models.

Experimental version of 3D modeler application is completed. Firstly we operated it with the help of keyboard and mouse. Then we integrated Kinect sensor with our application. Now application is operated with hands gesture movement using Kinect. A minute input is still requires keyboard to give name for saving and loading files.

ARUK-3D is capable of creating, resizing, moving, saving and loading objects using Kinect.

References

http://glprogramming.com/red/chapter05.html

http://glprogramming.com/red/chapter10.html

http://glprogramming.com/red/chapter10.html

http://www.opengl.org/wiki/Depth_Buffer

http://www.opengl.org/wiki/Hardware_specifics:_NVidia

http://www.opengl.org/wiki/Hardware_specifics:_Intel

http://www.opengl.org/wiki/Hardware_specifics:_ATI

http://www.sjbaker.org/steve/omniv/love_your_z_buffer.html

http://stackoverflow.com/questions/2211303/glreadpixels-really-slow-better-solution-to-get-opengl-coordinates-from-mouse

http://stackoverflow.com/questions/tagged/shader?page=5&sort=newest

http://eonstrife.wordpress.com/2007/06/02/taking-a-screenshot-from-an-opengl-application/

http://www.opengl.org/discussion_boards/ubbthreads.php?ubb=showflat&Number=246720
http://www.songho.ca/opengl/gl_pbo.html

http://developer.meego.com/api/1.2/opengles-1.1/glDepthMask.html

http://www.opengl.org/resources/faq/technical/depthbuffer.htm

http://www.3dkingdoms.com/selection.html

http://pyopengl.sourceforge.net/documentation/manual/glReadBuffer.3G.html

http://www.gamedev.net/page/resources/_/technical/opengl/opengl-frame-buffer-object-201-r2333

http://lists.apple.com/archives/mac-opengl/2008/Feb/msg00063.html

http://www.gamedev.net/topic/455434-how-to-readwrite-depth-buffer/

http://www.swiftless.com/tutorials/opengl/fullscreen.html