**Visvesvaraya Technological University**

### Belgaum-590014



**A Computer Graphics and Visualization**

**Mini-Project Report**

**On**

***“*3D HELICOPTER FIRING ON TARGET*”***

*A Mini-project report submitted in partial fulfilment of the requirements for the award of the degree of* ***Bachelor of Engineering in Computer Science and Engineering*** *of Visvesvaraya Technological University, Belgaum.*

Submitted by:

**ANKUSH M (1DT16CS011)**

Under the Guidance of:

**Mr. ANOOP.G.L**

**(Asst. Prof. Dept of CSE)**

****

**Department of Computer Science and Engineering**

**DAYANADA SAGAR ACADEMY OF TECHNOLOGY AND MANAGAEMENT**

Kanakpura Road,Udayapura, Bangalore,

2018-2019



**DAYANADA SAGAR ACADEMY OF TECHNOLOGY AND MANAGAEMENT,**

Kanakpura Road,Udayapura, Bangalore

**Department of Computer Science & Engineering**

**CERTIFICATE**

This is to certify that the Mini-Project on Computer Graphics and Visualization work entitled **“IMPLEMENTATION OF 3D HELICOPTER FIRING ON TARGET GAME USING OPENGL”** has been successfully carried out by **ANKUSH M (1DT16CS011)** abonafide student of **Dayananda sagar academy of technology and management** in partial fulfilment of the requirements for the award of degree in **Bachelor of Engineering** **in Computer Science and Engineering** of **Visvesvaraya Technological University, Belgaum** during academic year 2018-2019. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of project work for the said degree.

**GUIDES:**

|  |  |  |
| --- | --- | --- |
| **Mr. ANOOP.G.L** | **Mrs. JAHNAVI.SHANKAR** |  |
| **(Asst. Prof. Dept of CSE)** | **(Asst. Prof. Dept of CSE)** |  |

**Dr. C.NANDINI**

**(Prof & HOD, Dept. of CSE)**

**Examiners:**

**1:**

**2: Signature with Date**

### ABSTRACT

We have proposed a 3D view of a helicopter firing which aims at the target and shoots the target which can be any living or non living object.

We can Zoom in and Zoom out and also look around left and right side-view.

We are presenting an island lookalike model in which a helicopter is targeting a human.

Once a target is shot and it is correct a large size red colour circular shape would be visible in and around human.

But in other shots which may not view.

We have tried to portray an animation view for the given scene.

### ACKNOWLEDGEMENT

It gives us immense pleasure to present before you our project titled **‘IMPLEMENTATION OF 3D HELICOPTER FIRING ON TARGET USING OPENGL’.** The joy and satisfaction that accompany the successful completion of any task would be incomplete without the mention of those who made it possible. We are glad to express our gratitude towards our prestigious institution **DAYANANDA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT** for providing us with utmost knowledge, encouragement and the maximum facilities in undertaking this project.

We wish to express a sincere thanks to our respected principal **Dr. B. R. Lakshmikantha**  for all their support.

We express our deepest gratitude and special thanks to **Dr.C.Nandini** , **Prof &** **H.O.D, Dept. Of Computer Science Engineering**, for all her guidance and encouragement.

We sincerely acknowledge the guidance and constant encouragement of our mini- project guides, **Assistant Prof. Mr ANOOP.G.L**

**ANKUSH M (1DT16CS011)**

### TABLE OF CONTENT

### 

1. INTRODUCTION
   1. Computer Graphics
   2. OpenGL Technology
   3. Project Description
   4. Functions Used (OpenGL functions used in the program)
2. REQUIREMENT SPECIFICATION
   1. Hardware Requirements
   2. Software Requirements
3. INTERFACE AND ARCHITECTURE

3.1 Interface

3.2 Architecture (include control flow diagram example included)

1. IMPLEMENTATION (the logic used in the code)
2. SOURCE CODE (only include the important functions and logical functions in a separate file not in the report!!!)
3. SNAPSHOTS (snapshots of the output print this part in color Xerox )
4. FUTURE ENHANCEMENT
5. CONCLUSION
6. REFERENCES (include bibliography in this section too)
7. APPENDIX

10.1 User Manual

10.2 Personal Details

**Chapter-1**

### INTRODUCTION

* 1. **Computer Graphics**

Computer graphics are graphics created using computers and more generally, the representation and manipulation of image data by a computer. "almost everything on computers that is not text or sound".

Now, we can create image using computer that are indistinguishable from photographs from the real objects

The various applications of computer graphics are:

Display of information

Design

Simulation and animation

User interfaces

The phrase “Computer Graphics” was coined in 1960 by William Fetter, a graphic designer for Boeing

Today, we find computer graphics used in various areas that include science, medicine, business, industry, art, entertainment, etc.

The main reason for effectiveness of the interactive computer graphics is the speed with which the user can understand the displayed information

The current trend of computer graphics is to incorporate more physics principles into 3D graphics algorithm to better simulate the complex interactions between objects and lighting environment.

**1.2 OpenGL Technology**

**OpenGL** (**Open G**raphics **L**ibrary) is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and [3D computer graphics](http://en.wikipedia.org/wiki/3D_computer_graphics). The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple [primitives](http://en.wikipedia.org/wiki/Geometric_primitive). OpenGL was developed by [Silicon Graphics Inc](http://en.wikipedia.org/wiki/Silicon_Graphics). (SGI) in 1992 and is widely used in [CAD](http://en.wikipedia.org/wiki/Computer-aided_design), [virtual reality](http://en.wikipedia.org/wiki/Virtual_reality), scientific visualization, information visualization, [flight simulation](http://en.wikipedia.org/wiki/Flight_simulator), and [video games](http://en.wikipedia.org/wiki/Video_game).

OpenGL is a software API to graphics hardware, designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms, Intuitive, procedural interface with c binding, No windowing commands! And No high-level commands for describing models of three-dimensional objects.

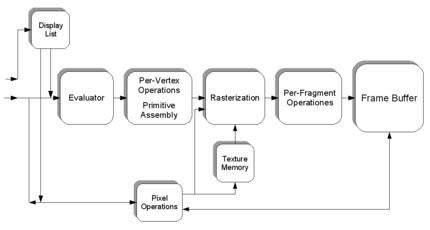
OpenGL serves two main purposes:

1. To hide the complexities of interfacing with different 3D accelerators, by presenting the programmer with a single, uniform API.

2. To hide the differing capabilities of hardware platforms, by requiring that all implementations support the full OpenGL feature set (using software emulation if necessary).

OpenGL has historically been influential on the development of 3D accelerators, promoting a base level of functionality that is now common in consumer-level hardware:

Rasterised points, lines and polygons as basic primitives



Simplified version of the Graphics Pipeline Process; excludes a number of features like blending, VBOs and logic ops

* A transform and lighting pipeline
* Z-buffering
* Texture mapping
* Alpha blending

A brief description of the process in the graphics pipeline could be:

* Evaluation, if necessary, of the polynomial functions which define certain inputs, like NURBS surfaces, approximating curves and the surface geometry.

1. Vertex operations, transforming and lighting them depending on their material. Also clipping non visible parts of the scene in order to produce the viewing volume.
2. Rasterization or conversion of the previous information into pixels. The polygons are represented by the appropriate colour by means of interpolation algorithms.
3. Per-fragment operations, like updating values depending on incoming an game d previously stored depth values, or colour combinations, among others.
4. Lastly, fragments are inserted into the Frame buffer.

Many modern 3D accelerators provide functionality far above this baseline, but these new features are generally enhancements of this basic pipeline rather than radical revisions of it.

As OpenGL is system independent, there are no functions to create windows etc., but there are helper functions for each platform. A very useful thing is GLUT.

**1.3 PROJECT DESCRIPTION:**

WE have proposed a 3D view of a helicopter firing which aims at the target and shoots, the target which can be any living or non-living object. We can Zoom in and Zoom out and also look around left and right side views .We are presenting an island lookalike model in which a helicopter is hitting a human. The human will be shot by helicopter.

Once a target is shot and it is correct a large size red colour circular shape would be visible in and around human. But in other shots which may not visible.

**1.4 FUNCTIONS USED**

This project is developed using Code::Blocks and this project is implemented by making extensive use of library functions offered by graphics package of OpenGL a summary of those functions follows:

**glBegin( ):**

Specifies the primitives that will be created from vertices presented between glBegin and subsequent glEnd. GL\_POLYGON, GL\_LINE\_LOOP etc.

**glEnd(void) :**

It ends the list of vertices.

**glPushMatrix() :**

*void* ***glPushMatrix****( void )*

glPushMatrixpushes the current matrix stack down by one level, duplicating the current matrix.

**glPopMatrix() :**

*void* ***glPopMatrix(*** *void )*

glPopMatrix pops the top matrix off the stack, destroying the contents of the popped matrix.Initially, each of the stacks contains one matrix, an identity matrix.

**glTranslate() :**

*void* ***glTranslate****( GLdouble x, GLdouble y,GLdouble z )*

Translation is an operation that displaces points by a fixed distance in a given direction. *Parameters x*, *y*, *z* specify the *x*, *y*, and *z* coordinates of a translation vector. Multiplies current matrix by a matrix that translates an object by the given x, y and z-values.

**glClear() :**

*void* ***glClear****(GLbitfield mask)*

glClear takes a single argument that is the bitwise *or* of several values indicating which buffer is to be cleared.

GL\_COLOR\_BUFFER\_BIT, GL\_DEPTH\_BUFFER\_BIT, GL\_ACCUM\_ BUFFER\_BIT, and GL\_STENCIL\_BUFFER\_BIT. Clears the specified buffers to their current clearing values.

**glClearColor() :**

*void* ***glClearColor****(GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha)*

Sets the current clearing color for use in clearing color buffers in RGBA mode. The red, green, blue, and alpha values are clamped if necessary to the range [0,1]. The default clearing color is (0, 0, 0, 0), which is black.

**glMatrixMode() :**

*void* ***glMatrixMode*** *(GLenum mode)*

It accepts three values GL\_MODELVIEW, GL\_PROJECTION and GL\_TEXTURE. It specifies which matrix is the current matrix. Subsequent transformation commands affect the specified matrix.

**glutInitWindowPosition() :**

*void* ***glutInitWindowPosition****(int x, int y);*

This api will request the windows created to have an initial position. The arguments x, y indicate the location of a corner of the window, relative to the entire display.

**glLoadIdentity() :**

*void* ***glLoadIdentity****(void);*

It replaces the current matrix with the identity matrix

**glutInitWindowSize() :**

*void* ***glutInitWindowSize****(int width, int height);*

The api requests windows createdto have an initial size. The arguments width and height indicate the window’s size (in pixels). The initial window size and position are hints and may be overridden by other requests.

**glutInitDisplayMode**

*void* ***glutInitDisplayMode*** *(unsigned int mode );*

Specifies the display mode, normally the bitwise OR-ing of GLUT display mode bit *masks.* This api specifies a display mode (such as RGBA or color-index, or single or double-buffered) for windows.

**glFlush() :**

*void* ***glFlush****(void);*

The glFlush function forces execution of OpenGL functions in finite time.

**glutCreateWindow() :**

*int* ***glutCreateWindow****(char \*name);*

The parameter *name* specifies any name for window and is enclosed in double quotes. This opens a window with the set characteristics like display mode, width, height, and so on. The string name will appear in the title bar of the window system *.* The value returned is a unique integer identifier for the window*.* This identifier can be used for controlling and rendering to multiple windows from the same application.

**glutDisplayFunc() :**

*void* ***glutDisplayFunc****(void (\*func)(void))*

Specifies the new display callback function. The api specifies the function that’s called whenever the contents of the window need to be redrawn. All the routines need to be redraw the scene are put in display callback function.

**glVertex2f**

*void* ***glVertex2f****(GLfloat x,GLfloat y)*

*x* Specifies the x-coordinate of a vertex.

*y* Specifies the y-coordinate of a vertex.

The glVertex function commands are used within glBegin/glEnd pairs to specify point, line, and polygon vertices. The current color, normal, and texture coordinates are associated with the vertex when glVertex is called. When only x and y are specified, z defaults to 0.0 and w defaults to 1.0. When x, y, and z are specified, w defaults to 1.0.

**glutBitMapCharacter() :**

*void* ***glutBitmapCharacter****(void \*font, int character);*

Without using any display lists, glutBitmapCharacter renders the character in the named bitmap font.

**glRasterPos2f() :**

*void* ***glRasterPos2f****(GLfloat x,GLfloat y);*

*x* Specifies the x-coordinate for the current raster position.

*y* Specifies the y-coordinate for the current raster position.

OpenGL maintains a 3-D position in window coordinates. This position, called the raster position, is maintained with sub pixel accuracy. It is used to position pixel and bitmap write operations. The current raster position consists of three window coordinates (x, y, z), a clip coordinate w value, an eye coordinate distance, a valid bit, and associated color data and texture coordinates. The w coordinate is a clip coordinate, because w is not projected to window coordinates

**Void keyboard (unsigned char key, int x, int y)**

This function specifies the keys to be pressed to manipulate the object displayed and the output window.

**glutMainLoop(void) :**

glutMainLoop enters the GLUT event processing loop. These routines should be called at most once in the GLUT program. Once called, these routine will never exit. It will call all the necessary call back functions that have been registered.

**Chapter-2**

### REQUIREMENTS SPECIFICATION

**2.1 Hardware requirements**:

* Pentium 4 or Equivalent Processor.
* 256 MB RAM
* 5 MB Disk Space
* Intel GMA 850 or better display adaptor
* Mouse and Keyboard input devices

**2.2 Software requirements**:

* Microsoft Windows XP / Linux 2.6.27 or later
* OpenGL and GLUT libraries
* Code::Blocks IDE

**Platform and Tools:**

1. The project has been carried out in Linux platform.
2. Built using Code Blocks IDE with glut libraries.
3. Keyboard interface is required for this project.

**Chapter-3**

### INTERFACE AND ARCHITECTURE

**MOUSE FUNCTIONS:**

MOUSE\_LEFT\_BUTTON: It is used to stop the movement of torus.

MOUSE\_MIDDLE\_BUTTON:is used to do nothing.

MOUSE\_RIGHT\_BUTTON:calls idlefunc with parameter NULL.

**KEYBOARD FUNCTIONS:**

KEY(‘a’):used to spin torso to left.

KEY(‘s’):used to spin torso to right.

KEY(‘q’):used to zoomout.

KEY(‘z’):used to zoomin

KEY(‘x’):used to hit the target.

KEY(‘m’):used to increase the speed of torso.

KEY(‘b’):used to decrease the speed of torso/to pause the entire simulation.

KEY(‘8’):used to move the helicopter upwards.

KEY(‘2’):used to move the helicopter downwards.

KEY(‘4’):used to move the helicopter to the left.

KEY(‘6’):used to move the helicopter to the right.

KEY(‘7’):used to move the helicopter to the front.

KEY(‘9’):used to move the helicopter to the back.

KEY(‘d’):used to change the background to daylight.

KEY(‘n’):used to change the background to night.

**Chapter-4**

### IMPLEMENTATION

**Implementation:** implementation of the project is the most important part of the project. The implement phase deals with the quality, performance, accuracy and debugging. The end derivable is the final game.

This game is implemented using keyboard and mouse interfaces which are as follows:

**Keyboard Function :**

***void inputKey(unsigned char, int x, int y);***

This function is used to make responds to the keys that are pressed from the keyboard. In this project, these keyboard keys are used to switch on the kit, to choose different gates, etc.

**Mouse Function :**

***void mouse(int btn, int state, int x,int y);***

This function is used to make responds to the buttons that are pressed from the mouse. In this project, these buttons are used for rotating purpose

**Display Function :**

***void display();***

In this function, rotation,translation and scaling is done by pushing and popping the matrix and using the inbuilt glutSolid function. We draw the helicopter,human and clouds.

**Main Function :**

***int main(int argc, char \*argv[]);***

In this function, we are creating the window, enabling the mouse and keyboard functions, initializing the display mode and also enables the 3D view.

**Chapter-5**

### SOURCE CODE

static void keypress(unsigned char key, int x, int y)

{ //calls the key functions

switch(key)

{

case 'a': spinleft(); break;

case 'd': glClearColor(2.6,2.6,0.0,0.0); break;

case 'n': glClearColor(0.0,0.0,0.0,0.0); break;

case 's': spinright(); break;

case 'q': zoomout(); break;

case 'z': zoomin(); break;

case 'p': if(p == 0)p = 1; else p=0; break;

case '8': moveup(); break;

case '2': movedown(); break;

case 'h': lookleft(); break;

case 'k': lookright(); break;

case 'x': if(fire == 0)fire = 1;

else fire=0; break;

case 'm': speed(); break;

case 'b': slow(); break;

case '4': moveleft(); break;

case '6': moveright(); break;

case '7': movefront(); break;

case '9': moveback(); break;

case GLUT\_KEY\_RIGHT: for(int i=0;i<=n;i++)

{

torso();

glTranslatef(0.0,0.0,1.0);

torso();

}

break;

}

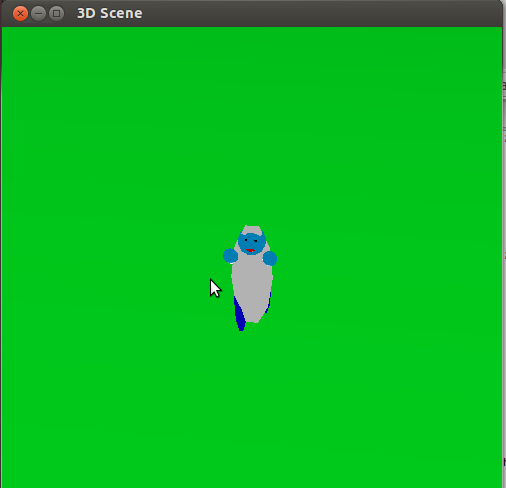
glutPostRedisplay();

}

**Chapter-6**

### 

### SNAPSHOTS

**SAMPLE OUTPUT**

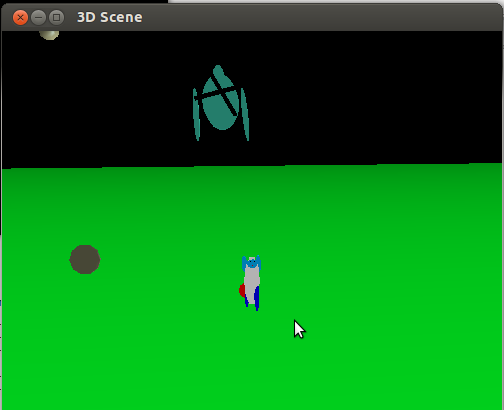
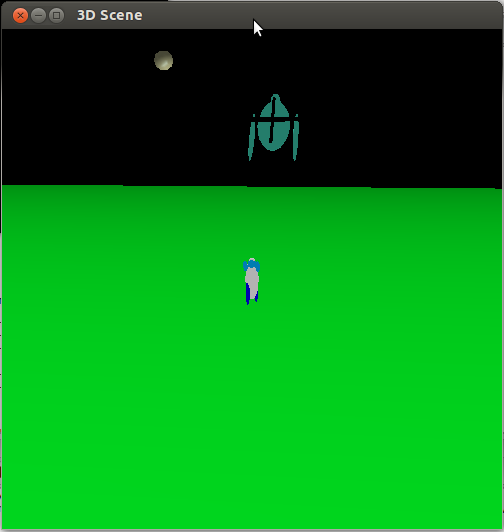
****Fig6.1 View of Torso

Fig6.2 View of Firing

Fig6.3 Front View

****

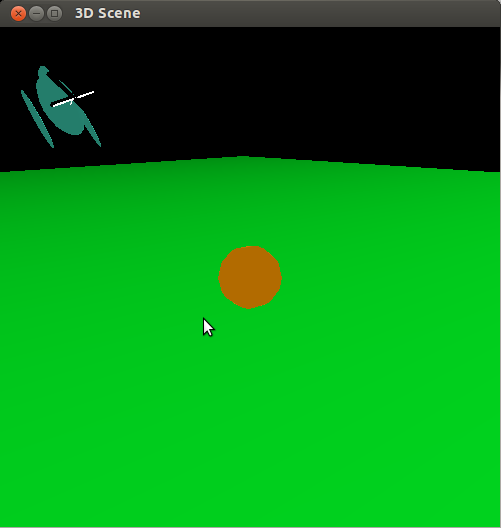
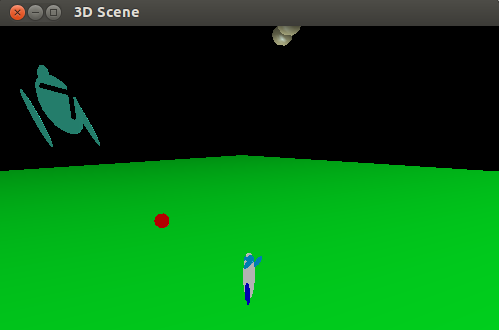
****

fig6.4 View of Torso getting Fired

****Fig6.5 View of Shooting

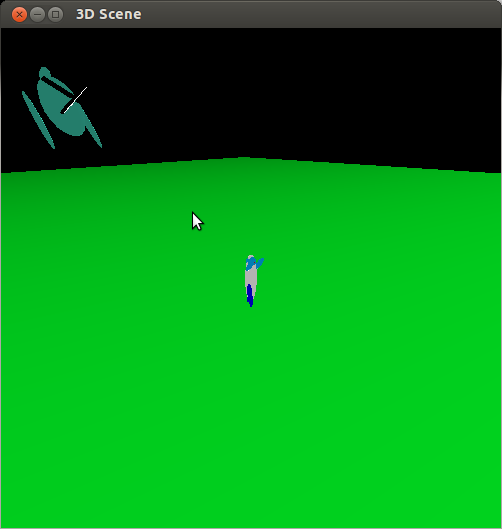
****

fig6.6 Side View

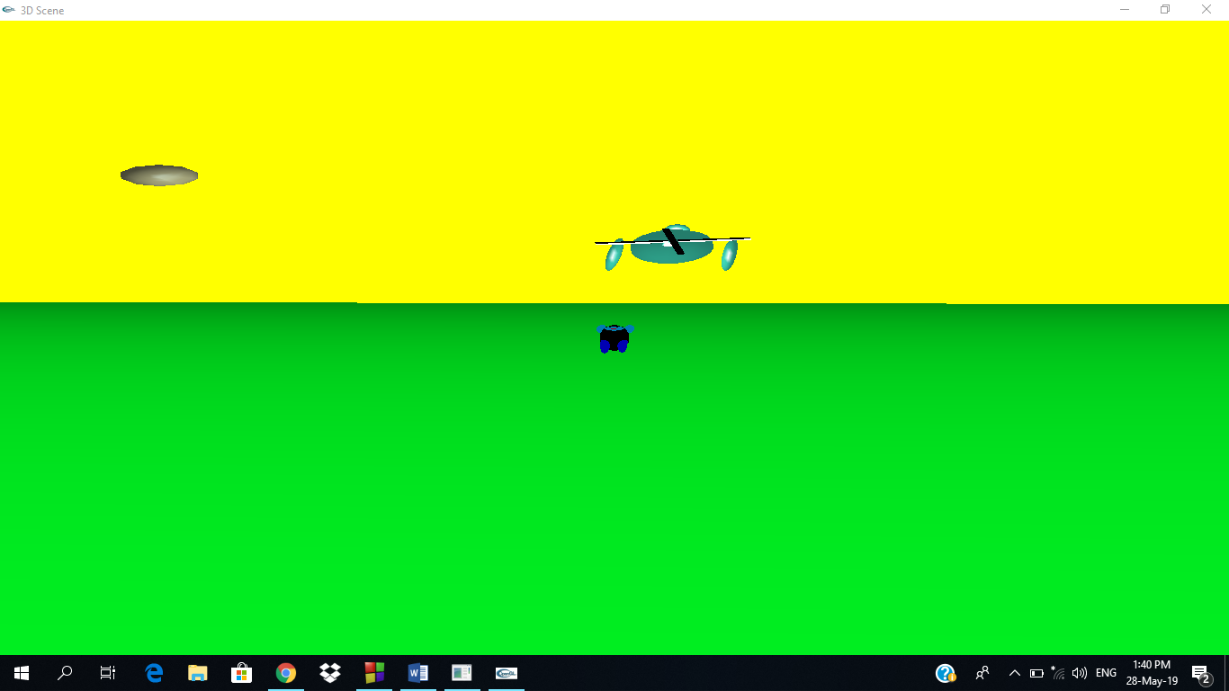
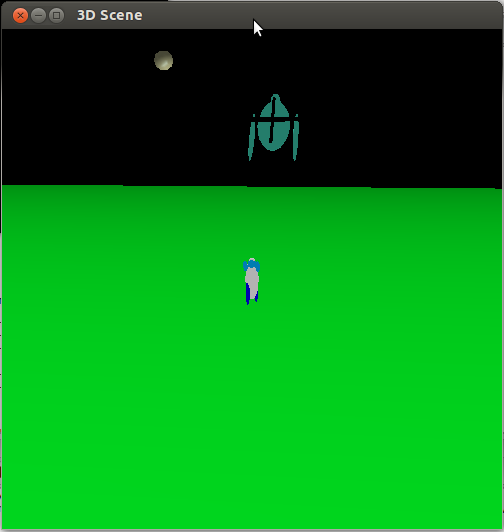
****

Fig6.7 Day View

****

**Fig6.8 Night View**

**Chapter-7**

### FUTURE ENHANCEMENT

1. More number of designs can be implemented.
2. We can add number of other objects.
3. Can be modified further to make a game.
4. Sound and music can be added.
5. Torso can be destroyed.
6. Many number of torso can be made and will be shot to destroy with a time limit.
7. Torso can increase its size.

**Chapter-8**

### 

### CONCLUSION

We have been successful in our attempt at implementing “3D HELICOPTER FIRING ON TARGET”.

This project was developed in Ubuntu 10.04 which is a Linux based OS as it is a programmer friendly OS. It was also tested on Windows XP, Windows 7 and Mac OS X making it a truly cross-platform project

3D SCENE, we tried to implement some of the graphics functions in openGL, which results in the graphics application. This project views a 3d chase of a helicopter in field. The camera angle can be changed and also zoomed in and out. We can make the man run with the helicopter and pause. Camera can be zoomed in an zoom out as well as rotated sideways.. When the man moves the camera focuses on the man and we will see the clouds moving backwards, giving the illusion that the man is moving front.

Here the keys like Q, Z, A and S are used to perform certain operations like zooming in, zooming out, rotating camera to left and right respectively. Here we have used some of the function display(), mouse(), myinit(), myReshape(), keypress(). The functions come under the header file GL/glut.h…

This project makes use of various functions provided by OpenGL. Doing this project has helped us understand and implement the concepts of OpenGL. In this project we have successfully designed the 3D HELICOPTER FIRING ON TARGET.

**Chapter-9**

### REFERENCES

**Books:**

[1] Interactive Computer Graphics a Top-Down Approach with OpenGL -Edward Angel

[2] Computer Graphics using OpenGL by F.S. Hill Jr. & Stephen M. Kelley Jr

**Websites:**

[1] [http://www.opengl.org](http://www.opengl.org/) [Official Website]

[3] <http://www.glprogramming.com/red/> [The Official Guide to Learning OpenGL]

[4] <http://www.opengl.org/resources/libraries/glut/glut_downloads.php> [GLUT]

[5] <http://pyopengl.sourceforge.net/documentation/manual> [Documentation]

[6] <http://stackoverflow.com/questions/tagged/opengl> [Questions]

**Chapter-10**

### 

### APPENDIX

**10.1 User Manual**

This game is about hitting a target. Here helicopter hits the target.

The target is a human which should be shot by helicopter using key ‘x’.

One can also use key ‘a’ to spin target to left and key ‘s’ to spin target to right.

All the instruction will be displayed once you run the program.

**10.2 Personal Details**

**NAME:** ANKUSH.M

**USN**: 1DT16CS011

**DEPARTMENT:** COMPUTER SCIENCE AND ENGINEERING

**SEMESTER AND SECTION:** 6TH SEMESTER AND SECTION ‘A’

**COLLEGE:** DAYANANDA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT

**EMAIL ID:** ankushmahesh@outlook.com