VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI

Mini Project Report on

"HELICOPTER GAME"

Submitted in the partial fulfillment for the requirements of Computer Graphics & Visualization Laboratory of 6th semester CSE requirement in the form of the Mini Project work

Submitted By

NISCHITHA V	USN: 1BY18CS415
MANOJ H	USN: 1BY15CS412

Under the guidance of

Mr. SHANKAR R
Assistant Professor, CSE, BMSIT&M

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT
YELAHANKA, BENGALURU - 560064.

2019-2020

BMS INSTITUTE OF TECHNOLOGY &MANAGEMENT
YELAHANKA, BENGALURU – 560064

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project work entitled **"HELICOPTER GAME"**is a bonafide work carried out by **Nischitha V (1BY18CS415) and Manoj H(1BY18CS412)** in partial fulfillment for *Mini Project* during the year 2sss019-2020. It is hereby certified that this project covers the concepts of *Computer Graphics & Visualization*. It is also certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in this report.

Signature of the Guide with date
Mr. SHANKAR R
Assistant Professor
CSE, BMSIT&M

Signature of HOD with date
Dr. Anil G N
Prof & Head
CSE, BMSIT&M

INSTITUTE VISION

To emerge as one of the finest technical institutions of higher learning, to develop engineering professionals who are technically competent, ethical and environment friendly for betterment of the society.

INSTITUTE MISSION

Accomplish stimulating learning environment through high quality academic instruction, innovation and industry-institute interface.

DEPARTMENT VISION

To develop technical professionals acquainted with recent trends and technologies of computer science to serve as valuable resource for the nation/society.

DEPARTMENT MISSION

Facilitating and exposing the students to various learning opportunities through dedicated academic teaching, guidance and monitoring.

PROGRAM EDUCATIONAL OBJECTIVES

1. Lead a successful career by designing, analysing and solving various problems in the field of Computer Science & Engineering.
2. Pursue higher studies for enduring edification.
3. Exhibit professional and team building attitude along with effective communication.
4. Identify and provide solutions for sustainable environmental development.

ACKNOWLEDGEMENT

We are happy to present this project after completing it successfully. This project would not have been possible without the guidance, assistance and suggestions of many individuals. We wouldlike to express our deep sense of gratitude and indebtedness to each and every one who hashelped us make this project a success.

We heartily thank our Principal, Dr. MOHAN BABU G N, BMS Institute of Technology &Management, for his constant encouragement and inspiration in taking up this project.

We heartily thank our Professor and Head of the Department, Dr. ANIL G N, Department of Computer Science and Engineering, BMS Institute of Technology &Management, for his constant encouragement and inspiration in taking up this project.

We gracefully thank our Project Guide, Mr. Shankar R, Assistant Professor, Department of Computer Science and Engineering for his intangible support and forbeing constant backbone for our project.

Special thanks to all the staff members of Computer Science Department for theirhelp and kind co-operation.

Lastly, we thank our parents and friends for the support and encouragement given throughout in completing this precious work successfully.

NISCHITHA V (1BY18CS415)

MANOJ H (1BY18CS412)

ABSTRACT

Automatic Repeat request (ARQ), is an error-control method for data transmission that uses acknowledgements (messages sent by the receiver indicating that it has correctly received a data frame or packet) and timeouts (specified periods of time allowed to elapse before an acknowledgment is to be received) to achieve reliable data transmission over an unreliable service. If the sender does not receive an acknowledgment before the timeout, it usually re-transmits the frame/packet until the sender receives an acknowledgment or exceeds a predefined number of re-transmissions.

The types of ARQ protocols include Stop-and-wait ARQ, Go-Back-N ARQ, and Selective Repeat ARQ / Selective Reject.

- Stop-and-wait ARQ,is a method in telecommunications to send information between two connected devices. It ensures that information is not lost due to dropped packets and that packets are received in the correct order. It is the simplest automatic repeat-request (ARQ) mechanism.
- Go-Back-N ARQ is a specific instance of the automatic repeat request (ARQ) protocol, in which the sending process continues to send a number of frames specified by a window size even without receiving an acknowledgement (ACK) packet from the receiver.
- Selective Repeat ARQ / Selective Reject ARQ is a specific instance of the Automatic Repeat-Request (ARQ) protocol used to solve sequence number dilemma in communications.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 COMPUTER GRAPHICS

Computer graphics are graphics created using computers and, more generally, the representation and manipulation of image data by a computer hardware and software. The development of computer graphics, or simply referred to as CG, has made computers easier to interact with, and better for understanding and interpreting many types of data. Developments in computer graphics have had a profound impact on many types of media and have revolutionized the animation and video game industry. 2D computer graphics are digital images—mostly from two-dimensional models, such as 2D geometric models, text (vector array), and 2D data. 3D computer graphics in contrast to 2D computer graphics are graphics that use a three-dimensional representation of geometric data that is stored in the computer for the purposes of performing calculations and rendering images.

The user controls the contents, structure, and appearance of the objects and of their displayed images by using input devices, such as keyboard, mouse, or touchscreen. Due to close relationships between the input devices and the display, the handling of such devices is included in the study of computer graphics. The advantages of the interactive graphics are many in number. Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D patter-recognition abilities allow us to perceive and process data rapidly and efficiently. In many design, implementation, and construction processes today, the information pictures can give is virtually indispensable. Scientific visualization became an important field in the 1980s when the scientists and engineers realized that they could not interpret the prodigious quantities of data produced in supercomputer runs without summarizing the data and highlighting trends and phenomena in various kinds of graphical representations.

1.2 OPEN GL

OpenGL is the most extensively documented 3D graphics API (Application Program Interface) to date. Information regarding OpenGL is all over the Web and in print. It is impossible to exhaustively list all sources of OpenGL information. OpenGL programs are typically written in C and C++. One can also program OpenGL from Delphi (a Pascal-like language), Basic, Fortran, Ada, and other languages. To compile and link OpenGL programs, one will need OpenGL header files. To run OpenGL programs one may need shared or dynamically loaded OpenGL libraries, or a vendor-specific OpenGL Installable Client Driver (ICD).

OpenGL is a low-level graphics library specification. It makes available to the programmer a small set of geometric primitives - points, lines, polygons, images, and bitmaps. OpenGL provides a set of commands that allow the specification of geometric objects in two or three dimensions, using the provided primitives, together with commands that control how these objects are rendered (drawn).

1.3 GLUT

The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs, which primarily perform system-level I/O with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input. Routines for drawing a number of geometric primitives (both in solid and wireframe mode) are also provided, including cubes, spheres, and cylinders. GLUT even has some limited support for creating pop-up menus. The two aims of GLUT are to allow the creation of rather portable code between operating systems (GLUT is crossplatform) and to make learning OpenGL easier. All GLUT functions start with the glut prefix (for example, glutPostRedisplay marks the current window as needing to be redrawn).

CHAPTER 2

LITERATURE SURVEY

CG (Computer graphics) started with the display of data on hardcopy plotters and cathode ray tube screens soon after the introduction of computer themselves. It includes the creation, storage, and manipulation of models and images of objects. These models include physical, mathematical, engineering, architectural, and even conceptual or abstract structures, natural phenomena, and so on. Computer Graphics today is largely interactive- the user controls the contents, structure, and appearance of objects and their displayed images by using input devices, such as keyboard, mouse or touch sensitive panel on the screen. Bitmap graphics is used for user-computer interaction. A Bitmap is an ones and zeros representation of points (pixels, short for _picture elements') on the screen. Bitmap graphics provide easy-to-use and inexpensive graphics based applications.

The concept of desktop is a popular metaphor for organizing screen space. By means of a window manager, the user can create, position, and resize rectangular screen areas, called windows, that acted as virtual graphics terminals, each running an application. This allowed users to switch among multiple activities just by pointing at the desired window, typically with the mouse. Graphics provides one of the most natural means of communicating with the computer, since our highly developed 2D and 3D pattern – recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. In many design, implementation, and construction processes, the information pictures can give is virtually indispensable.

Computer graphics is the creation and manipulation of pictures with the aid of computers. It is divided into two broad classes:

- Non-Interactive Graphics.
- Interactive Graphics.

2.1 NON – INTERACTIVE GRAPHICS

This is a type of graphics where observer has no control over the pictures

2.2 INTERACTIVE GRAPHICS

This is the type of computer graphics in which the user can control the pictures produced. It involves two-way communication between user and computer. The computer upon receiving signal from the input device can modify the displayed picture appropriately. To the user it appears that the picture changes instantaneously in response to his commands. The following fig. shows the basic graphics system:

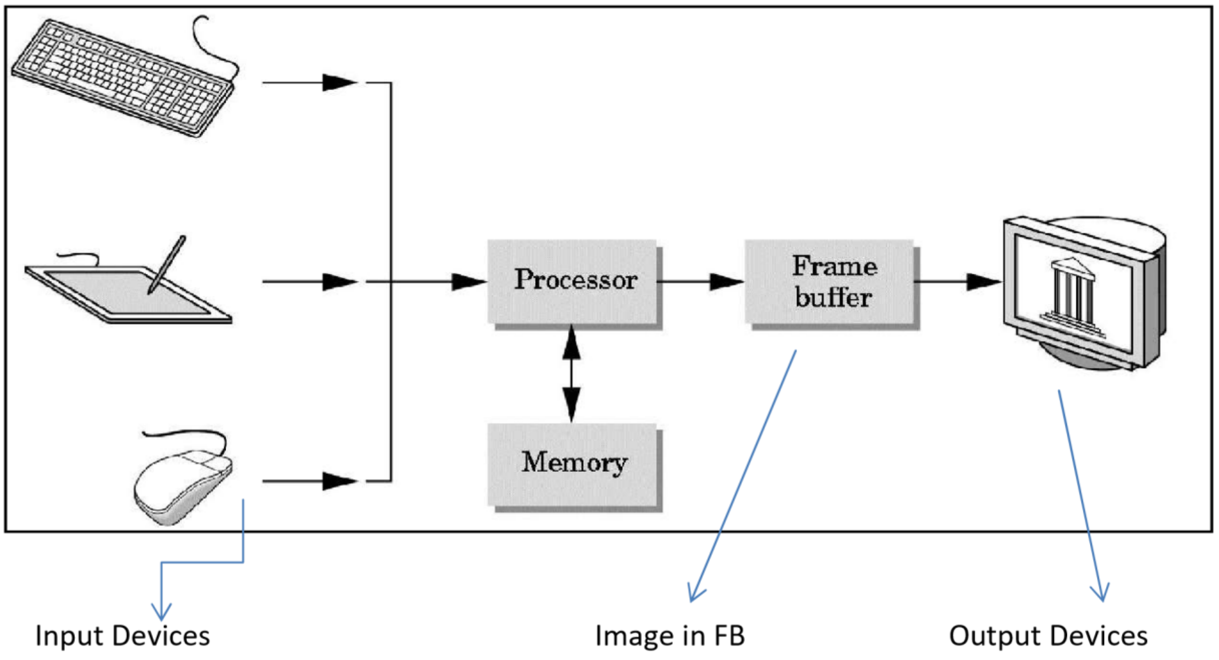


FIGURE 2.2 BASIC GRAPHICS SYSTEM

CHAPTER 3

PROJECT IN DETAIL

3.1 PROBLEM STATEMENT

Computer graphics is no longer a rarity. It is an integral part of all computer user interfaces, and is indispensable for visualizing 2D; 3D and higher dimensional objects .Creating 3D objects, rotations and any other manipulations are laborious process with graphics implementation using text editor. OpenGL provides more features for developing 3D objects with few lines by built in functions. The geometric objects are the building blocks of any individual .Thereby developing, manipulating, applying any transformation, rotation, scaling on them is the major task of any image development.

So in this project the main problem statement is to support 2D gaming with help of computer graphics that is using Open GL.

“helicopter game” using the concepts of Open GL

3.2 BRIEF INTRODUCTION

This project under Computer Graphics and Visualization Laboratory is an implementation of a 2D helicopter game using the OpenGL Graphics Library and GLUT Toolkit. The player input his/her name before starting the game and can control it either using keyboard or mouse at a time and at the end of game it will show how much distance you covered as his/her score. The objective of the game is to fly a helicopter in space with restricted upward and downward motion using either mouse or keyboard, meanwhile walls will move towards player's helicopter and player have to avoid a collision between them. The game will be over if a collision occurs.

3.3 SCOPE AND MOTIVATION

The scope is to use the basic primitives defined in OpenGl library creating complex objects.

In this project we are trying to develop an easy, efficient and cost free method of story simulation of gaming

3.4 PROPOSED SYSTEM

Here in this project helicopter game we have proposed graphical interfaces between user and the system by using the openGl concept. This project deals with 2D picturization of real time process helicopter game and this uses keyboard or an mouse as an interactive to play the game.

CHAPTER 4

REQUIREMENT SPECIFICATION

4.1 HARDWARE REQUIREMENTS

- Processor: INTEL / AMD
- Main memory: 2 GB RAM (Min.)
- Hard Disk: Built-in is sufficient
- Keyboard: QWERTY Keyboard
- Mouse: 2 or 3 Button mouse
- Monitor: 1024 x 768 display resolution

4.2 SOFTWARE REQUIREMENTS

- Programming language – C/C++ using OpenGL
- Operating system – Windows/Linux
- Compiler – C/C++ Compiler (GCC compier)
- IDE – Code blocks
- Functional Requirement – <GL/glut.h>

CHAPTER 5

DESIGN AND IMPLEMENTATION

5.1 DESIGN

The “Hare and Tortoise Story Simulation” is designed using some of the OpenGL inbuilt functions along with some user defined functions. So, this section comprises of the complete explanation of the design of the project using various opengl functions and user defined methods along with its explanation. This chapter is divided into two sections –

- OPENGL Functions – This section throws light on the various openGL functions used and its uses.
- USER – DEFINED Functions – This section demonstrates the various user defined functions and its purpose.

5.1.1 OPEN GL FUNCTIONS:

The different OpenGL functions that are used in the project is described below:

- glClearColor(0.1,0.8,0.1,1.0) :- glClearColor() specifies the red, green, blue, and alpha values used by glClear() to clear the color buffers. Values specified by glClearColor() are clamped to the range 0, 1
- glMatrixMode(GL_MODELVIEW) :- specify which matrix is the current matrix. Specifies which matrix stack is the target for subsequent matrix operations. Three values are accepted: GL_MODELVIEW, GL_PROJECTION, and GL_TEXTURE.
  The initial value is GL_MODELVIEW.
- gluOrtho2D(0,1000,0,500) :- define a 2D orthographic projection matrix. left, right- Specify the coordinates for the left and right vertical clipping planes. bottom, top - Specify the coordinates for the bottom and top horizontal clipping planes.

- glRasterPos2f(x, y) :- The glRasterPos2 function uses the argument values for x and y while implicitly setting z and w to zero and one. The object coordinates presented by glRasterPos are treated just like those of a glVertex command. They are transformed by the current modelview and projection matrices and passed to the clipping stage.
- glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,string[i]) :- glutBitmapCharacter renders a bitmap character using OpenGL. Without using any display lists, glutBitmapCharacter renders the character in the named bitmap font.
- glutFullScreen() :- glutFullScreen requests that the current window be made full screen. The exact semantics of what full screen means may vary by window system. The intent is to make the window as large as possible and disable any window decorations or borders added the window system.
- glFlush() :- Force execution of GL commands in finite time. glFlush empties all of these buffers, causing all issued commands to be executed as quickly as they are accepted by the actual rendering engine. Though this execution may not be completed in any particular time period, it does complete in finite time.
- glPointSize(4) :- specify the diameter of rasterized points. glPointSize specifies the rasterized diameter of points. The value written to the shading language built-in variable gl_PointSize will be used.
- glBegin(GL_POINTS) :- delimit the vertices of a primitive or a group of like primitives. glBegin and glEnd delimit the vertices that define a primitive or a group of like primitives. glBegin accepts a single argument that specifies in which of ten ways the vertices are interpreted.
- glVertex2i(278,273) :- specify a vertex. glVertex commands are used within glBegin/glEnd pairs to specify point, line, and polygon vertices. The current color, normal, texture coordinates, and fog coordinate are associated with the vertex when glVertex is called.
- glEnd() :- delimit the vertices of a primitive or a group of like primitives. glBegin and glEnd delimit the vertices that define a primitive or a group of like primitives.
- glutSwapBuffers() :- glutSwapBuffers swaps the buffers of the current window if double buffered. Performs a buffer swap on the layer in use for the current window.

Specifically, glutSwapBuffers promotes the contents of the back buffer of the layer in use of the current window to become the contents of the front buffer. An implicit glFlush is done by glutSwapBuffers before it returns. If the layer in use is not double buffered, glutSwapBuffers has no effect.

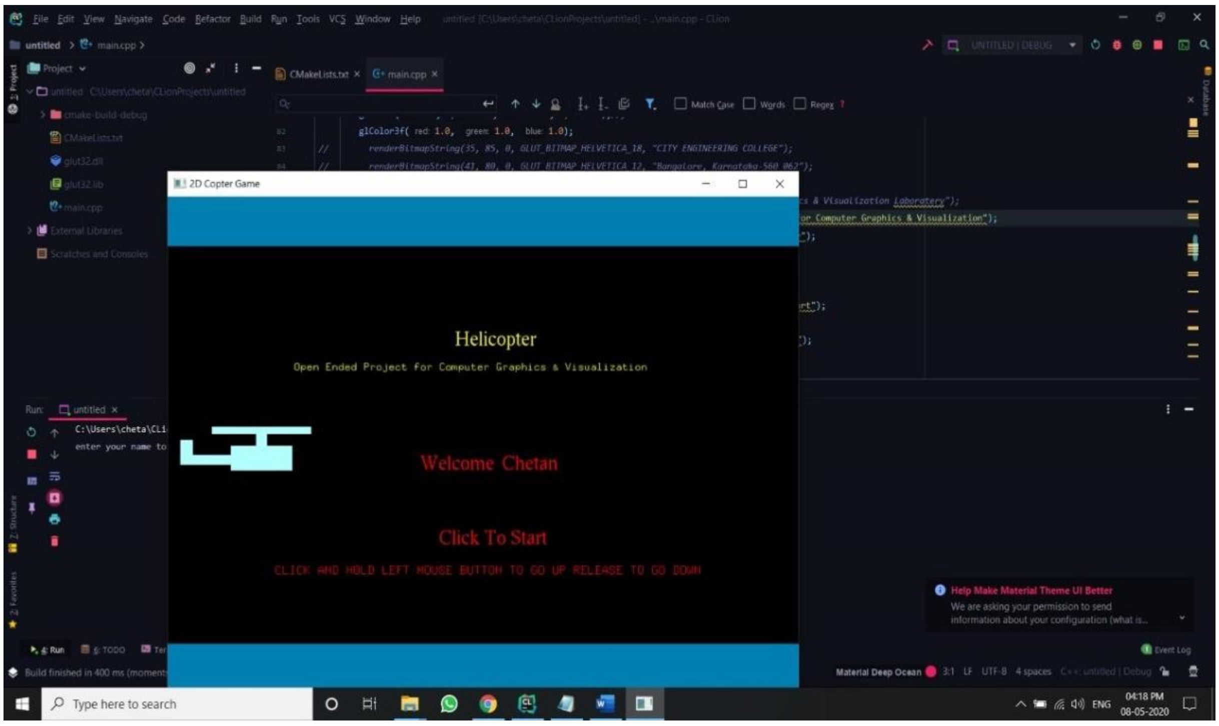
- glPushMatrix() push and pop the current matrix stack. There is a stack of matrices for each of the matrix modes. The current matrix in any mode is the matrix on the top of the stack for that mode.glPushMatrix pushes the current matrix stack down by one, duplicating the current matrix. That is, after a glPushMatrix call, the matrix on top of the stack is identical to the one below it.
- glPopMatrix() :- push and pop the current matrix stack. glPopMatrix pops the current matrix stack, replacing the current matrix with the one below it on the stack.
- glTranslate(a2,0,0) :- multiply the current matrix by a translation matrix. glTranslate produces a translation by x y z . The current matrix is multiplied by this translation matrix, with the product replacing the current matrix.
- glutInit(&argc,argv) :- glutInit is used to initialize the GLUT library. glutInit will initialize the GLUT library and negotiate a session with the window system. During this process, glutInit may cause the termination of the GLUT program with an error message to the user if GLUT cannot be properly initialized. glutInit also processes command line options, but the specific options parse are window system dependent.
- glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB) :- sets the initial display mode. The initial display mode is used when creating top-level windows, subwindows, and overlays to determine the OpenGL display mode for the to-be-created window or overlay.
- glutInitWindowSize(1000,500) :- set the initial window size . Windows created by glutCreateWindow will be requested to be created with the current initial window position and size. The initial value of the initial window size GLUT state is 300 by 300. The initial window size components must be greater than zero.The intent of the initial window position and size values is to provide a suggestion to the window system for a window's initial size and position. The window system is not obligated to use this

information. A GLUT program should use the window's reshape callback to determine the true size of the window.

- glutCreateWindow("HARE AND TORTOISE") :- creates a top-level window. The name will be provided to the window system as the window's name. The intent is that the window system will label the window with the name.Implicitly, the current window is set to the newly created window.
- glutPositionWindow(50,50) :- requests a change to the position of the current window. . For top-level windows, the x and y parameters are pixel offsets from the screen origin. For subwindows, the x and y parameters are pixel offsets from the window's parent window origin.
- glutDisplayFunc(display1) :- registers the callback function (or event handler) for handling window-paint event. The OpenGL graphic system calls back this handler when it receives a window re-paint request. In the example, we register the function display() as the handler.
- glutKeyboardFunc(NormalKey) :- glutKeyboardFunc sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The key callback parameter is the generated ASCII character. The state of modifier keys such as Shift cannot be determined directly; their only effect will be on the returned ASCII data.
- glutMainLoop() :- glutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary any callbacks that have been registered.

CHAPTER 6

INTERPRETATION OF RESULT



2D Copter Game

Helicopter

Open Ended Project for Computer Graphics & Visualization

Welcome Chetan

Click To Start

CLICK AND HOLD LEFT MOUSE BUTTON TO GO UP RELEASE TO GO DOWN

FIGURE 6.2: This is our first screen of our project when it starts to run

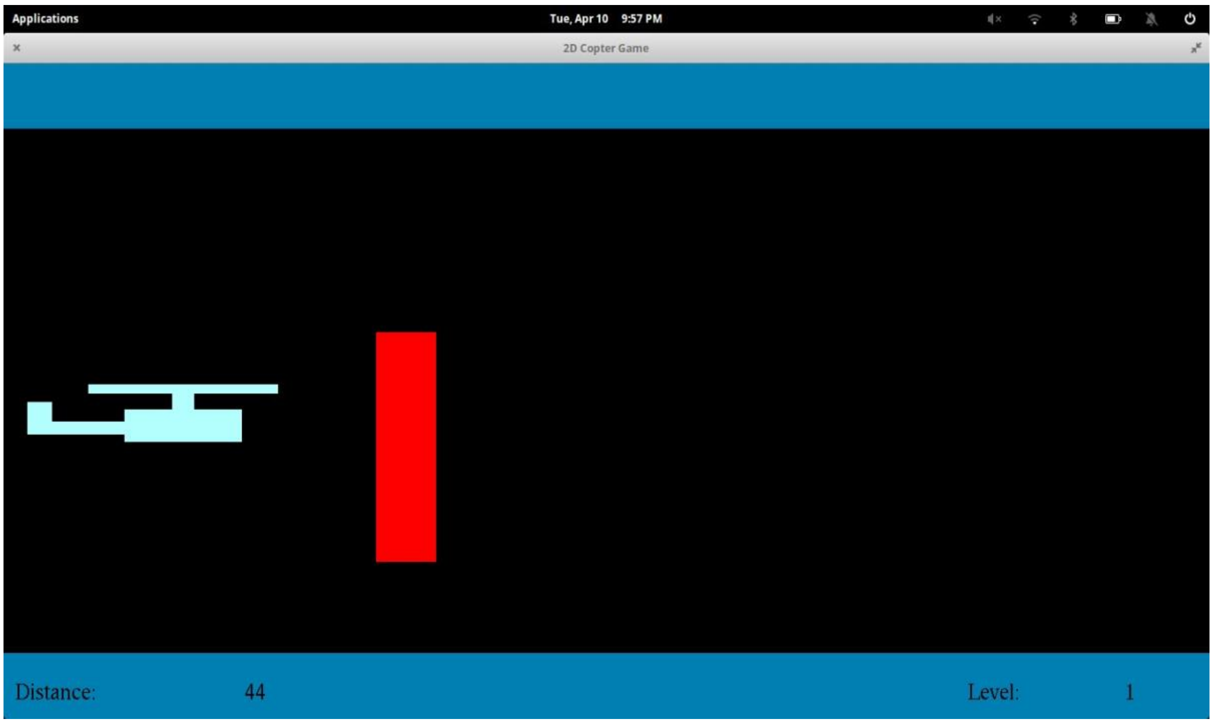


FIGURE 6.2: This is the second page of project where it displays the movement of the helicopter begins.

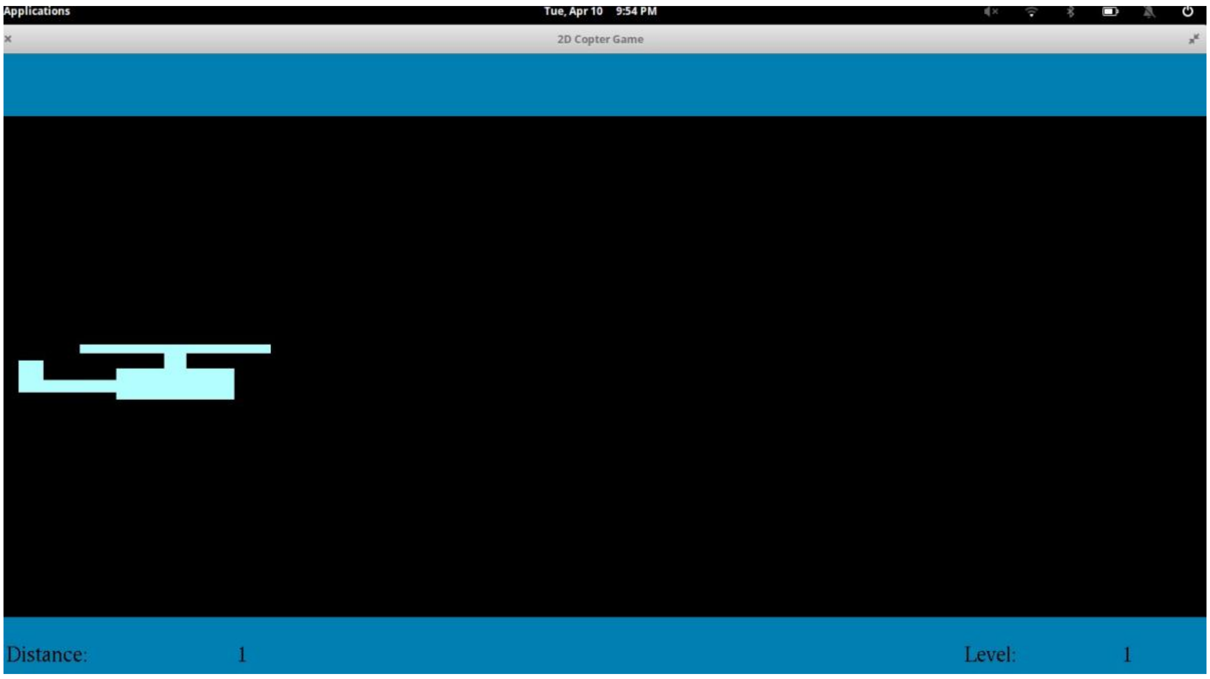
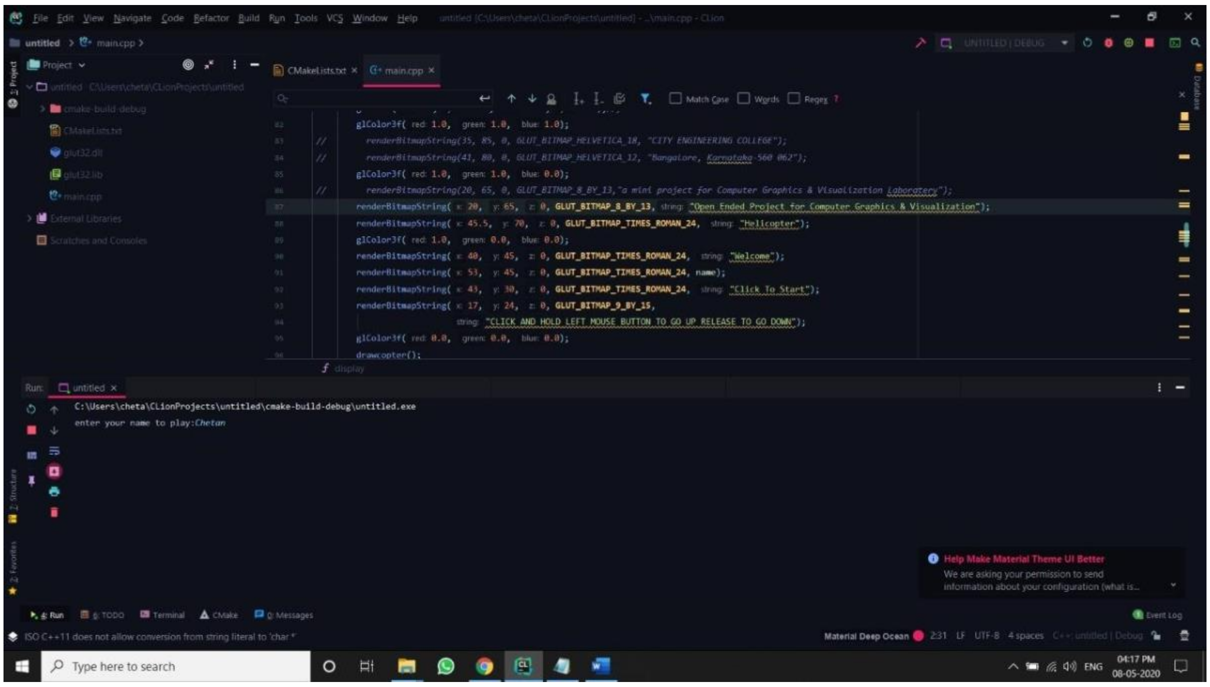


FIGURE 6.3: This page displays that the helicopter crossed the block.



```
glColor3f( red: 1.0,  green: 1.0,  blue: 1.0);
//  renderBitmapString(35, 85, 0, GLUT_BITMAP_HELVETICA_18, "CITY ENGINEERING COLLEGE");
//  renderBitmapString(41, 80, 0, GLUT_BITMAP_HELVETICA_12, "Bangalore, Karnataka-560 062");
glColor3f( red: 1.0,  green: 1.0,  blue: 0.0);
//  renderBitmapString(20, 65, 0, GLUT_BITMAP_8_BY_13,"a mini project for Computer Graphics & Visualization Laboratery");
renderBitmapString( x: 20,  y: 65,  z: 0, GLUT_BITMAP_8_BY_13, string: "Open Ended Project for Computer Graphics & Visualization");
renderBitmapString( x: 45.5,  y: 70,  z: 0, GLUT_BITMAP_TIMES_ROMAN_24,  string: "Helicopter");
glColor3f( red: 1.0,  green: 0.0,  blue: 0.0);
renderBitmapString( x: 40,  y: 45,  z: 0, GLUT_BITMAP_TIMES_ROMAN_24,  string: "Welcome");
renderBitmapString( x: 53,  y: 45,  z: 0, GLUT_BITMAP_TIMES_ROMAN_24, name);
renderBitmapString( x: 43,  y: 30,  z: 0, GLUT_BITMAP_TIMES_ROMAN_24,  string: "Click To Start");
renderBitmapString( x: 17,  y: 24,  z: 0, GLUT_BITMAP_9_BY_15,
                    string: "CLICK AND HOLD LEFT MOUSE BUTTON TO GO UP RELEASE TO GO DOWN");
glColor3f( red: 0.0,  green: 0.0,  blue: 0.0);
drawcopter();
```

```
C:\Users\cheta\CLionProjects\untitled\cmake-build-debug\untitled.exe
enter your name to play:Chetan
```

FIGURE 6.4: This is the last page of the project

Implementation of code

#include<stdlib.h>

#include<GL/glut.h>

#include<time.h>

//#include<dos.h>

#include<stdio.h>

//#include<conio.h>

//#include<windows.h>

float bspd=0.02;

```
char name[25];

  float b1x=50.0,b1y=0;

  float hm=0.0;


int i=0,sci=1;float scf=1;

char scs[20],slevel[20];

//to store score_string using itoa() and level as well

  int level=1,lflag=1,wflag=1;

  void init(void)

{

srand(time(0));
b1y=(rand()%45)+10;
  glClearColor (0.0, 0.0, 0.0, 0.0);  glShadeModel (GL_SMOOTH);

glLoadIdentity    ();        glOrtho(0.0,
100.0, 0.0, 100.0, -1.0 , .0);  }
void drawcopter()
{                          glColor3f(0.7,1.0,1.0);
glRectf(10,49.8,19.8,44.8);//body
glRectf(2,46,10,48);//tail
glRectf(2,46,4,51);//tail                    up
glRectf(14,49.8,15.8,52.2);//propeller
stand
glRectf(7,53.6,22.8,52.2);//propeller
}

  void renderBitmapString(float x,float y,float z,void *font,char*string)

{
  char *c;  glRasterPos3f(x, y,z);  for(c=string; *c != '\0'; c++)
{  glutBitmapCharacter(font, *c);
}

}               void
display(void)
{

glClear(GL_COLOR_BUFFER_BIT);

//GameOver Checking

if(

(i==730||i==-700)

 ( ((int)b1x==10||(int)b1x==7||(int)b1x==4||(int)b1x==1)

&&(int)b1y<53+(int)hm&&(int)b1y+35>53+(int)hm)



(((int)b1x==9||(int)b1x==3||(int)b1x==6)
&&(int)b1y<45+(int)hm&&(int)b1y+35>45+(int)hm)
( ((int)b1x==0) && (int)b1y<46+(int)hm&&(int)b1y+35>46+(int)hm))
 {  glColor3f(0.0,0.0,1.0);
glRectf(0.0,0.0,100.0,10
0.0);
glColor3f(1.0,0.0,0.0);
renderBitmapString(40,70,0,GLUT_BITMAP_HELVETICA_18,"GAMEOVER!!!");
glColor3f(1.0,1.0,1.0);
renderBitmapString(25,58,0,GLUT_BITMAP_TIMES_ROMAN_24,"You");
renderBitmapString(45,58,0,GLUT_BITMAP_TIMES_ROMAN_24,"scored:");
renderBitmapString(70,58,0,GLUT_BITMAP_TIMES_ROMAN_24,scs);
```

```
glutSwapBuffers();  glFlush();  printf("\nGAME OVER\n\n");  printf("%s\You
scored  %s" ,name,scs);  printf("\n\nClose the console window to exit...\n");
//get
ch();
exit(
0);  }
else if(wflag==1)//Welcome Screen

{

wflag=0;
glColor3f(0.0,0.5,0.7);
glRectf(0.0,0.0,100.0,10.0);/
/ceil
glRectf(0.0,100.0,100.0,90.0
);//floor
glColor3f(1.0,1.0,1.0);
renderBitmapString(35,85,0,GLUT_BITMAP_HELVETICA_18,"CITY              ENGINEERING
COLLEGE");        renderBitmapString(41,80,0,GLUT_BITMAP_HELVETICA_12,"Bangalore,
Karnataka-560
062");  glColor3f(1.0,1.0,0.0);  renderBitmapString(20,65,0,GLUT_BITMAP_8_BY_13,"a mini
projectforComputerGraphics&VisualizationLaboratery");
renderBitmapString(45.5,70,0,GLUT_BITMAP_TIMES_ROMAN_24,"Helicopter");
glColor3f(1.0,0.0,0.0);
renderBitmapString(40,45,0,GLUT_BITMAP_TIMES_ROMAN_24,"Welcome");
renderBitmapString(53,45,0,GLUT_BITMAP_TIMES_ROMAN_24,name);
renderBitmapString(43,30,0,GLUT_BITMAP_TIMES_ROMAN_24,"Click          To          Start");
renderBitmapString(17,24,0,GLUT_BITMAP_9_BY_15,"CLICK    AND    HOLD    LEFT    MOUSE
BUTTON TO GO UP RELEASE TO GO DOWN");
glColor3f(0.0,0.0,0.0);
drawcopter();
```

```
glutSwapBuffers();
glFlush();
} else
{
if(sci%50!=0&&lflag!=
1)
{
lflag=
1;
}

glPushMatrix();
glColor3f(0.0,0.5,0.7);
glRectf(0.0,0.0,100.0,10.0
);
glRectf(0.0,100.0,100.0,9
0.0);
glColor3f(0.0,0.0,0.0);

renderBitmapString(1,3,0,GLUT_BITMAP_TIMES_ROMAN_24,"Distance:");
sprintf(slevel,"%d",level);
renderBitmapString(80,3,0,GLUT_BITMAP_TIMES_ROMAN_24,"Level:");
renderBitmapString(93,3,0,GLUT_BITMAP_TIMES_ROMAN_24,slevel);
scf+=0.025;          sci=(int)scf;  sprintf(scs,"%d",sci);
renderBitmapString(20,3,0,GLUT_BITMAP_TIMES_ROMAN_24,scs);
glTranslatef(0.0,hm,0.0);  drawcopter();  //code for helicopter
//if wall move towards left & get out of projection volume  if(b1x<-
10)
{
b1x=
50;
b1y=(rand()%25)+20;
//10 for selling+10 for giving enough space

// block bottom limit 0+20 & top limit 24+20=44

}     else     b1x-=bspd;
//withi
glTranslatef(b1x,-
hm,0.0);
glColor3f(1.0,0.0,0.0);
glRectf(b1x,b1y,b1x+5,b1y+35);//blo
ck          1                    glPopMatrix();
glutSwapBuffers();  glFlush();
}

}

void moveHeliU(void)

{

hm+=0.05;   i++;
glutPostRedispl
ay();
}

void
moveHeliD()
{ hm-=0.05;
i--;

glutPostRedisplay();

}  void mouse(int button, int state, int
x, int y)
```

```
{     switch
(button)
{

case
GLUT_LEFT_BUTTON:     if
(state    ==    GLUT_DOWN)
glutIdleFunc(moveHeliU);
else if (state == GLUT_UP)
glutIdleFunc(moveHeliD);
break;  default: break;
}

}

void keys(unsigned char key,int x,int y)

{

if(key=='w')                glutIdleFunc(moveHeliU);
if(key=='m') glutIdleFunc(moveHeliD);
}  int main(int argc, char**
argv)  {  printf("enter your
name      to      play:      ");
scanf("%s",name);
glutInit(&argc, argv);
glutInitDisplayMode  (GLUT_DOUBLE  |  GLUT_RGB);
glutInitWindowSize              (800,              600);
glutInitWindowPosition                        (200,200);
glutCreateWindow ("2D Copter Game");
init();

glutDisplayFunc(displ
ay);
glutMouseFunc(mous
e);
glutKeyboardFunc(ke
ys);    glutMainLoop();
return 0;
 }
```

CONCLUSION:

With the help of computer graphics we have created a helicopter game. It helps user to access it easily and user friendly to. This game is enjoyable and fun too. It helped us to learn computer garphics , user interface, screen management. This helped us in gaining more knowledge with the subject.

FUTURE ENHANCEMENT

- Making game even more complex than this.
- Upgrading it to 3D effect of playing.
- Adding more planes with it.
- Improving for the realistiv feel.
- Making for few online scoring method.

BIBILOGRAPHY

- Prescribed VTU CG textbook-James D Folkey Computer Graphics with opengl
- https://en.wikipedia.org/
- www.OpenGL.org
- https://learnopengl.com/