VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"Jnana Sangama", Belagavi-590018, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY
K.R. Road, V.V. Puram, Bengaluru-560 004

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Computer Graphics Laboratory With Mini Project Report-17CSL68

on

"FLYING THROUGH HOOPS"

Submitted By

1BI17CS119 **RAAGA SHETTY B P**

for the academic year 2019-20

Under the guidance of

Prof. N Thanuja.
Assistant Professor

Prof. Shruthiba A.
Assistant Professor

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"Jnana Sangama", Belagavi-590018, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY
K.R. Road, V.V. Puram, Bengaluru-560 004

Department of Computer Science & Engineering

Certificate

This is to certify that the implementation of **Computer Graphics and Visualization MINI PROJECT (17CSL68)** entitled **"FLYING THROUGH HOOPS"**

has been successfully completed by

1BI17CS119 **RAAGA SHETTY B P**

of VI semester B.E. for the partial fulfillment of the requirements for the Bachelor's degree in **Computer Science & Engineering** of the **Visvesvaraya Technological University** during the academic year **2019-2020**.

Lab In charges:

Prof. N Thanuja.	**Prof. Shruthiba A.**	**Dr. ASHA T.**
Assistant Professor	Assistant Professor	Professor and Head
Dept. of CS&E, BIT	Dept. of CS&E, BIT	Department of CS&E
		Bangalore Institute of Technology

Examiners: 1) 2)

ACKNOWLEDGEMENT

The knowledge & satisfaction that accompany the successful completion of any task would be incomplete without mention of people who made it possible, whose guidance and encouragement crowned my effort with success. I would like to thank all and acknowledge the help I have received to carry out this Mini Project.

I would like to convey my sincere thanks to **Dr. M. U. Aswath**, Principal, BIT and **Dr.Asha T.**, HOD, Department of CS&E, BIT for being kind enough to provide the necessary support to carry out the mini project.

I am most humbled to mention the enthusiastic influence provided by the lab in-charges **Prof. N Thanuja.** and **Prof. Shruthiba A.** , on the project for their ideas, time to time suggestions for being a constant guide and co-operation showed during the venture and making this project a great success.

I would also take this opportunity to thank my friends and family for their constant support and help. I'm very much pleasured to express my sincere gratitude to the friendly co-operation showed by all the **staff members** of Computer Science Department, BIT.

RAAGA SHETTY B P
1BI17CS119

Table of contents

Chapter - 1

INTRODUCTION

1.1 Computer Graphics

Computer graphics is an art of drawing pictures, lines, charts, using computers with the help of programming. Computer graphics image is made up of number of pixels. Pixel is the smallest addressable graphical unit represented on the computer screen.Typically,the term computer graphics refers to several different things :

- The representation and manipulation of image data by a computer.
- The various technologies used to create and manipulate images.
- The sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content.

There are two types of computer graphics :

1. Interactive Computer Graphics

Interactive Compter Graphics is the type of computer graphics which involves two way communication between computer and user.User is given the control over the image by providing an input device that takes the user request to the computer.

Example : Flight simulator used to train pilots,Video game controller.

Advantages:Fuel saving,safety.

2. Non-Interactive Computer Graphics or Passive Graphics

Non-Interactive Compter Graphics is the type of computer graphics in which user does not have any kind of control over the images.Images are displayed with thehelp of a static stored program and according to the instructions that are written in the program.

Example: Screen savers.

1.2 Applications of Computer Graphics

Graphs and charts

An early application for computer graphics is the display of simple data graphs usually plotted on a character printer. Data plotting is one of the most common graphics application. Graphs & charts are used to summarize functional, statistical, mathematical, engineering and economic data for research reports, managerial summaries and other types of publications.

Computer-Aided Design

A major use of computer graphics is in design processes-particularly for engineering and architectural systems.CAD,Computer-Aided Design or CADD, Computer-Aided Drafting and Design methods are now routinely used in the automobiles, aircraft, spacecraft, computers, home appliances.Animations are often used in CAD applications.

Virtual Reality Environments

Animations in virtual-reality environments are often used to train heavy-equipment operators or to analyze the effectiveness of various cabin configurations and control placements. Architectural designs can be examined by taking simulated "walk" through the rooms or around the outsides of buildings to better appreciate the overall effect of a particular design. With a special glove, we can even "grasp" objects in a scene and turn them over or move them from one place to another.

Data Visualisaations

Scientific Visualisations : Producing graphical representations for scientific, engineering and medical data sets and processes is another fairly new application of computer graphics. Business Visualization : used in connection with data sets related to commerce, industry and other nonscientific areas.

Education and Training

Computer generated models of physical,financial,political,social,economic & other systems are often used as educational aids. For some training applications,special hardware systems are designed.Examples of such specialized systems are the simulators for practice sessions ,aircraft pilots,air traffic control personnel.

Image Procesing

The modification or interpretation of existing pictures, such as photographs and TV scans is called image processing.Medical applications also make extensive use of image processing techniques for picture enhancements in tomography and in simulations and surgical operations.It is also used in computed X-ray tomography(CT), position emission tomography(PET),and computed axial tomography(CAT).

Graphical User Interface

It is common now for applications software to provide graphical user interface (GUI). A major component of graphical interface is a window manager that allows a user to display multiple, rectangular screen areas called display windows. Each screen display area can contain a different process, showing graphical or nongraphical information, and various methods can be used to activate a display window.

1.3 OpenGL

Open Graphics Library(OpenGL) is a cross language, cross platform application programming interface(API) for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit(GPU), to achieve hardware-accelerated rendering. Most of our applications will be designed to access OpenGL directly through functions in the three libraries. Functions in main GL libraries have names that begin with the letters gl and are stored in a library usually referred to as GL.

In addition to OpenGL basic(core) library(prefixed with gl), there are a number of associated libraries for handling special operations:-

1) **OpenGL Utility(GLU)** : Prefixed with "glu". It provides routines for setting up viewing and projection matrices, describing complex objects with line and polygon approximations, displaying quadrics and B-splines using linear approximations, processing the surface-rendering operations, and other complex tasks.

2) **Open Inventor** : Provides routines and predefined object shapes for interactive three dimensional applications which are written in C++.

3) **Window-system libraries** : To create graphics we need display window. We cannot create the display window directly with the basic OpenGL functions since it contains only device-independent graphics functions, and window-management operations are device-dependent. However, there are several window-system libraries that supports OpenGL functions for a variety of machines. Eg:- Apple GL(AGL), Windows-to-OpenGL(WGL), PGL, GLX.

4) **OpenGL Utility Toolkit(GLUT)** : Provides a library of functions which acts as interface for interacting with any device specific screen-windowing system, thus making our program device-independent. The GLUT library functions are prefixed with "glut".

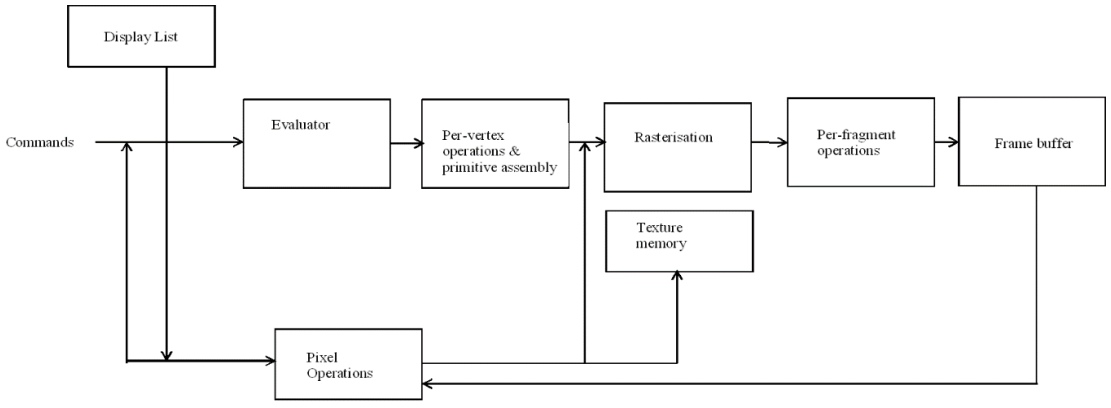


Figure 1.1: Basic block diagram of OpenGL

1.4 Problem Statement

To implement a 3D plane game by using graphics transformation functions, menu options and providing input interaction that allows the user to navigate through various frames displayed before the game begins.To ensure that the frames convey the instructions required in order to help the user play the game. Also, to provide the user options to either increase or decrease the plane speed and to stop or quit the game using menu options.

1.5 Objectives of the Project

- To demonstrate the implementation of a 3D plane game using OpenGL.
- To present the implementation of the OpenGL transformation functions such as translation,rotation and scaling.
- To show the working of lighting and shading concepts.
- To demonstrate the working of menu options.
- To demonstrate the concept of input interaction (Keyboard/Mouse).

1.6 Organisation of the Project

The project FLYING THROUGH HOOPS was organised in a precise and standardized way. Initially, prior outline of our project was made so as to have an idea of how the output must look like.Subsequently,all the basic features such as the use of transformation functions, lighting and shading concepts, menu options, input interaction were analysed.After analysis, the source code was formulated as a paper work. All the required software were downloaded and the source code was executed resulting in the successful implementation of the project.

Chapter-2

SYSTEM REQUIREMENTS

2.1 Hardware Requirements

- Main Processor : Intel Core i5 8th Gen 8250U
- Processor Speed: 1.80 GHz
- RAM Size : 8.00 GB (7.84 GB usable)
- Keyboard : Standard qwerty serial or PS/2 keyboard
- Mouse : Standard serial or PS/2 mouse
- Compatibility : AT/T Compatible
- Cache memory : 8 MB

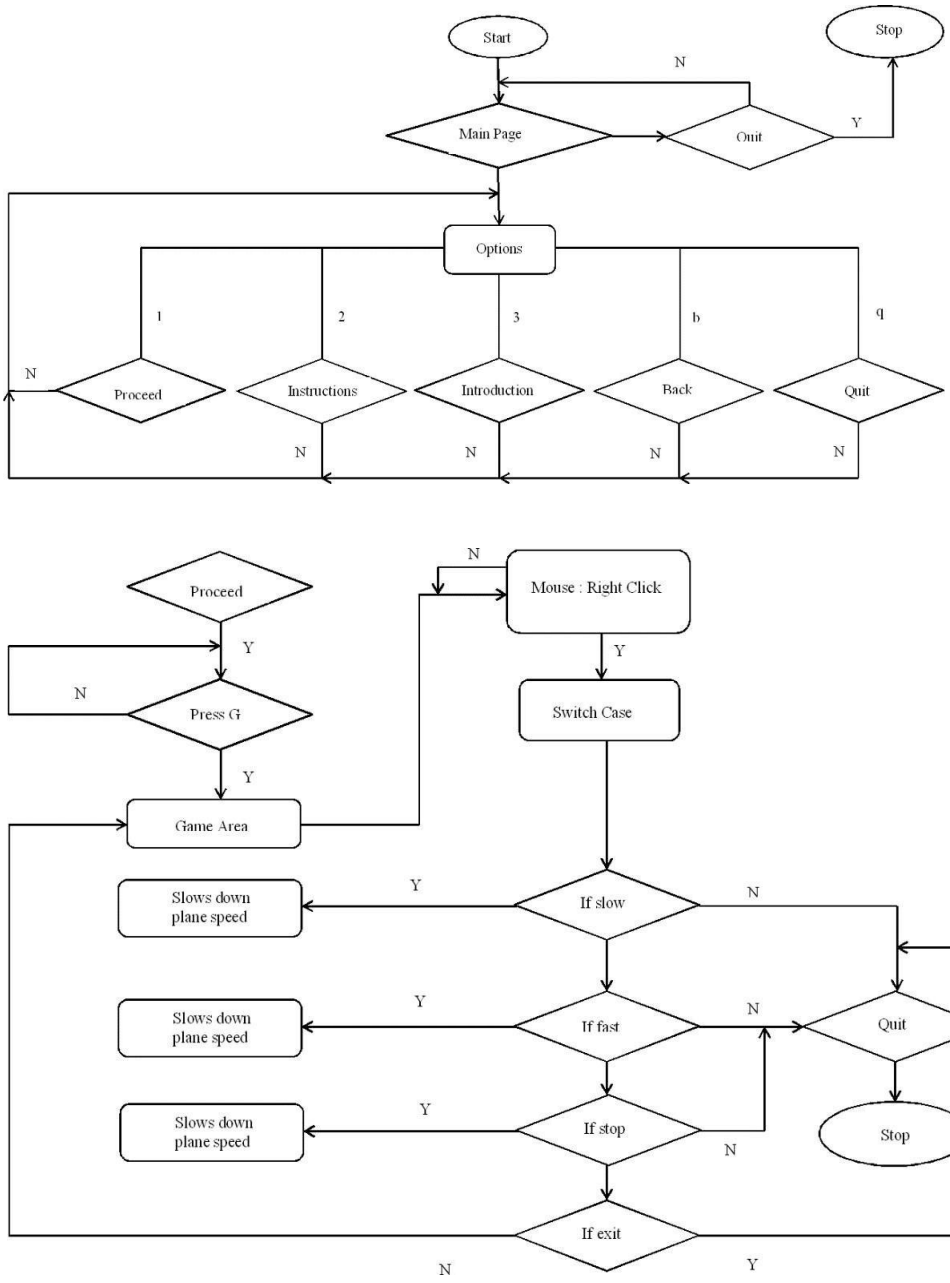
2.2 Software Requirements

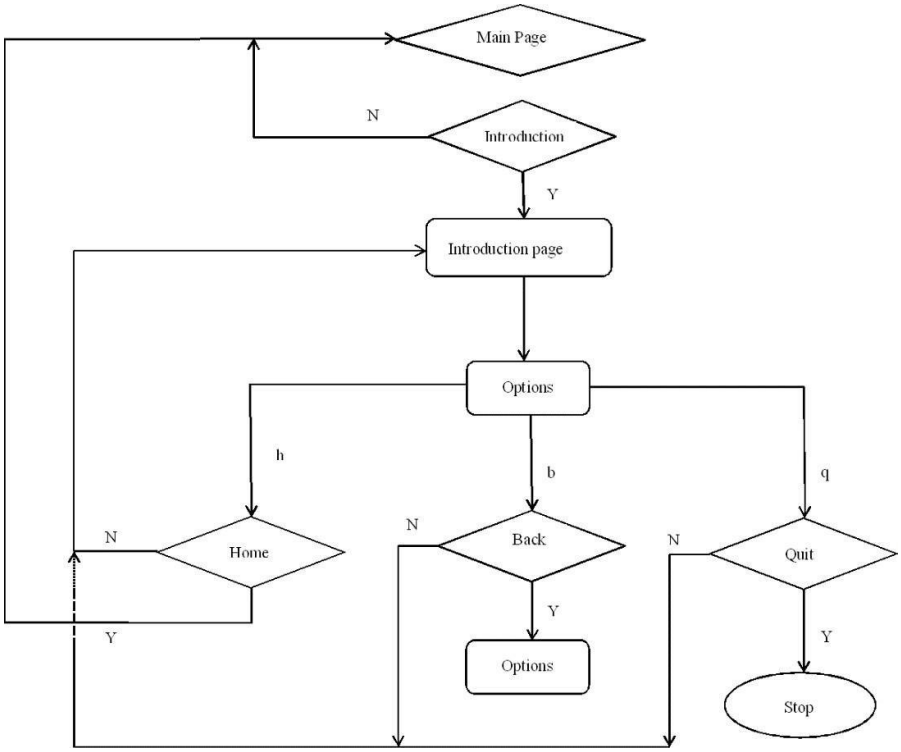
- Operating System: Windows 10 and Linux (Fedora)
- Hypervisor used : VMware workstation
- Compiler used : g++
- Language used : C++ language
- Editor : gedit Text Editor
- Toolkit : GLUT Toolkit

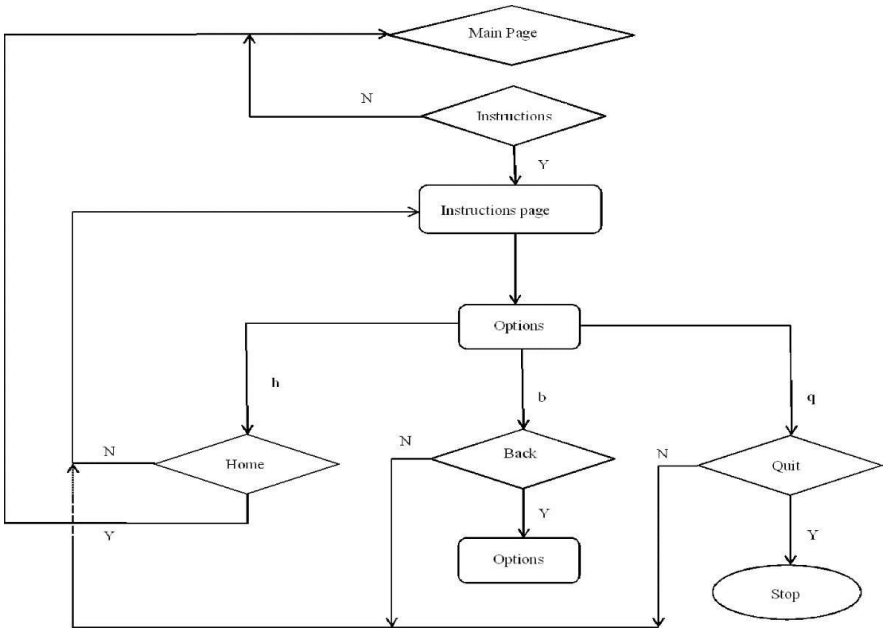
Chapter - 3

DESIGN

3.1 Flow Diagram




Main Page
N
Introduction
Y
Introduction page
Options
h
b
q
N
Home
N
Back
N
Quit
Y
Y
Y
Options
Stop


Main Page
N
Instructions
Y
Instructions page
Options
h
b
q
N
Home
N
Back
N
Quit
Y
Y
Y
Options
Stop

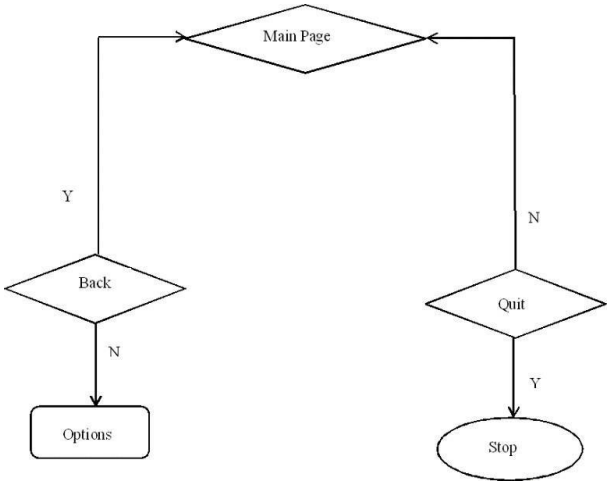


Figure 3.1: Flow diagram

3.2 Description of Flow Diagram

The description of the flow diagram is as follows:

Step1: Start

Step2: The user has to press any key to continue.

Step3: In the next frame, the user can select from five options : proceed,instructions,introduction,back,quit.

- Proceed : Displays the game start page.
- Instructions : Displays the instructions of the game.
- Introduction : Displays the basic introduction of the game.
- Back : Returns to the main page.
- Quit : Ends the game.

Step4: In the game start page, the user has to press the letter G to enter into the game area.

Step5: In the game area, the player can start playing the game by passing the plane through the hoops.This frame has menu options :

- Slow : Slows down the speed of the plane.
- Fast : Increases the plane speed.
- Stop : Stops the plane.
- Exit : Quits the game.

Step6: Stop

Chapter - 4

IMPLEMENTATION

4.1 Built in Functions

1. **glutInit( )**

   Usage: glutInit (&argc,argv);

   Description: glutInit will initialize the GLUT library and negotiate a session with the window system.

2. **glutInitDisplayMode( )**

   Usage: glutInitDisplayMode (unsigned int mode);

   Description: A number of options for the display window, such as buffering and a choice of color modes can be set with the glutInitDisplayMode function. Arguments for this routine are assigned sumbolic GLUT constants. The values of the constants passed to this function are combined using a logical OR operation.

3. **glutCreateWindow( )**

   Usage: glutCreateWindow (char *name);

   Description: We can state that a display window is to be created on the screen with a given caption for the title bar. That is accomplished with the glutCreateWindow() function – where the single argument for this function ccan be any character string that we want to use for the display-window title.

4. **glutDisplayFunc( )**

   Usage: glutDisplayFunc (void(*func)(void));

   Description: We create a picture using OpenGL functions and pass the picture definition to the GLUT routine glutDisplayFunc, which assigns our picture to the display window. Example: suppose we have the OpenGL code for describing a line segment in a procedure called lineSegment.

   Then the following function call passes the line-segment description to the display window: glutDisplayFunc(lineSegment);

5. glutMainLoop( )

Usage: glutMainLoop ( );

Description: This is one more GLUT function to complete the window-processing operations. After execution of the statement, all display windows that we have created, including their graphic content, will be activated. This function must be the last one in our program. It displays the initial graphics and puts the program into an infinite loop that checks for input from devices such as a mouse or keyboard.

6. glMatrixMode( )

The two most important matrices are the model-view and projection matrix. At many times, the state includes values for both of these matrices, which are initially set to identity matrices. There is only a single set of functions that can be applied to any type of matrix. Select the matrix to which the operations apply by first set in the matrix mode, a variable that is set to one type of matrix and is also part of the state.

7. glTranslate(GLfloat X, GLfloat Y, GLfloat Z)

glTranslate produces a translation by x y z. If the matrix mode is either GL_MODEL_VIEW or GL_PROJECTION, all objects drawn after a call to glTranslate are translated.

8. glRotatef(GLdouble angle, GLdouble X, GLdouble Y, GLdouble Z)

glRotatef produces a rotation of angle degrees around the vector x y z. If the matrix mode is either GL_MODEL_VIEW or GL_PROJECTION, all objects drawn after glRotatef is called are rotated. Use glPushMatrix() and glPopmatrix() to save and restore the unrotated coordinate system.

9. glPushMatrix()

There is a stack of matrices for each of the matrix mode. In GL_MODELVIEW mode, the stack depth is atleast 32. In other modes, GL_COLOR, GL_PROJECTION, and GL_TEXTURE, the depth is atleast 2. The current matrix in any mode is matrix on the top of the stack for that mode.

10. glPopMatrix()

glPopMatrix pops the current matrix stack, replacing the current matrix with the one below it on the stack. Initially, each of the stack contains one matrix, an identity matrix. It is an error to push a full matrix stack or pop a matrix stack that contains only a single matrix. In either case, the error flag is set and no other changeis made to GL state.

11. glutSwapBuffers( )

Usage: void glutSwapBuffers(void);

Description: Performs a buffer swap on the layer in use for the current window. Specifically, glutSwapBuffers promotes the contents of the front buffer. The contents of the back buffer then become undefined.

12. glPointSize(GLfloat size)

glPointSize specifies the rasterized diameter of points. This value will be used rasterize points. Otherwise, the value written to the shading language built-in variable gl-PointSize will be used. The point size specified by glPointSize is always returned when GL_POINT_SIZE is queried.

13. glutKeyboardFunc( )

Usage: void glutKeyboardFunc(void(*func)(unsigned char key, int x, int y)

Func: The new keyboard callback function

Description: 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.

14. glLineWidth(GLfloat width)

Parameters: width- Specifies the width of rasterized lines.(initial value =1)

Description: glLineWidth specifies the rasterized width of lines. The actual width is determined by rounding the supplied width to the nearest integer. i pixels are filled in each column that is rasterized, where I is the rounded value of width.

4.2 User-Defined Function

1. **void plane( ) -** Used to draw the plane and specify the color for different parts.
2. **void drawPillar1( ) -** Used to mention the specification of each pillar in the scene.
3. **void environment(int n) -** Used to draw the terrain, buildings, rings.
4. **void output(int x, int y, char *string, void *font) -** String holds the content that can be displayed on the screen. Integer type x and y are the variables that hold the values of the x and y positions on the graphical window.
5. **void menu(int val) -** Used to select options like Fast, Slow, Stop and Exit.
6. **void draw( ) -** Controls the rotation, movements, speed of the scene. Also shows the timer on the right top corner.
7. **void front( ) -** Used to specify the information to be displayed in the front screen along with the color associated with each line. Also displays a statement to be followed in order to navigate to the next page.
8. **void menus( ) -** Used to display various options to the user before starting the 3D plane game – options to proceed, view instructions, introduction and navigate back or go home.
9. **void menuset( ) -** Includes glMatrixMode, glLoadIdentity, glOrtho and glClear functions. Which mainly sets the projection type (mode) and other viewing parameters that we need. This specifies that an orthogonal projection is to be used to map from world coordinates onto the screen.
10. **void key(unsigned char key, int x, int y) -** Used to handle the keyboard interaction and to navigate through the frames. Also includes the glutSwapBuffers( ) function to facilitate the same.

4.3 Pseudocode

```
void output(int x, int y, char *string,void *font)
{
     int len, i;
     glRasterPos2f(x, y);
     len = (int) strlen(string);
     for (i = 0; i < len; i++)
    {
         glutBitmapCharacter(font, string[i]);
     }
}

static void resize(int width, int height)
{
     const float ar = (float) width / (float) height;
     glViewport(0, 0, width, height);
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     glFrustum(-ar, ar, -1.0, 1.0, 2.0, 1000.0);
     glMatrixMode(GL_MODELVIEW);
     glLoadIdentity();
}

void  drawPillar1()
{
     glColor3d(0.4,0.2,0.2);
    glPushMatrix();
    glTranslated(0,1.55,0);
    glScaled(2,0.05,1.5);
    glutSolidCube(1);
    glPopMatrix();
    glColor3d(0.4,0.2,0.2);
    glPushMatrix();
    glTranslated(0,1.65,0);
    glScaled(1.8,0.05,1.3);
    glutSolidCube(1);
    glPopMatrix();

    /// ROD
    glColor3d(0,0,0);
```

```
    glPushMatrix();
    glTranslated(0.07,1.99,-0.4);
    glScaled(0.003,0.7,0.003);
    glutSolidCube(1);
    glPopMatrix();
    glPushMatrix();
    glTranslated(0.15,1.99,-0.4);
    glScaled(0.003,0.7,0.003);
    glutSolidCube(1);
    glPopMatrix();

    ///small pillars
    glPushMatrix();
    ///ROD
    glPopMatrix();

    /// Circle
    glColor3d(1,0,0);
    glPushMatrix();
    glTranslated(0,2.1,-0.44);
    glScaled(0.35,0.35,0.01);
    glutSolidSphere(1,50,50);
    glPopMatrix();
    glColor3d(0,0,0);
    glPushMatrix();
    glTranslated(-0.18,1.9,-0.45);
    glScaled(0.01,0.5,0.01);
    glutSolidCube(1);
    glPopMatrix();
}

void plane()
{
    const double t = glutGet(GLUT_ELAPSED_TIME) / 1000.0;
    double a = t*90.0;
    /// Main body
    glColor3d(0.5,1,0);
    glPushMatrix();
    glTranslated(0,0,0);
    glScaled(3,0.4,0.5);
    glutSolidSphere(1,30,30);
```

```
    glPopMatrix();
    glColor3d(0,0,0);
    glPushMatrix();
    glTranslated(1.7,0.1,0);

    glScaled(1.5,0.7,0.8);
    glRotated(40,0,1,0);
    glutSolidSphere(0.45,30,30);
    glPopMatrix();

     ///Right
    glColor3d(0.8,1,0);
    glPushMatrix();
    glTranslated(0.4,0,1.5);
    glRotated(-30,0,1,0);
    glScaled(0.7,0.1,3);
    glRotated(10,0,1,0);
    glutSolidCube(1);
    glPopMatrix();

    ///left
    glColor3d(0.8,1,0);
    glPushMatrix();
    glTranslated(0.4,0,-1.5);
    glRotated(30,0,1,0);
    glScaled(0.7,0.1,3);
    glRotated(-10,0,1,0);
    glutSolidCube(1);
    glPopMatrix();
    glPopMatrix();

    /// upper
    glColor3d(0.8,1,0);
    glPushMatrix();
    glTranslated(-2.7,0.5,0);
    glRotated(45,0,0,1);
    glScaled(0.8,2,0.1);
    glRotated(-20,0,0,1);
    glutSolidCube(0.5);
    glPopMatrix();
}
```

```
void house(int n,int R,int G)
{
    for(int i=0;i<n;i++){
    glPushMatrix();
    glTranslated(0,0.8+i,0);
    glPopMatrix(); } }

void environment(int n)
{
    /// Ground
    glColor3d(0,0.5,0.1);
    glPushMatrix();
    glTranslated(0,0,0);
    glScaled(EN_SIZE*2,0.3,EN_SIZE*2);
    glutSolidCube(1);
    glPopMatrix();
    glColor3d(0,1,0.1);
    glPushMatrix();
    glTranslated(torusPosX[n],torusPosY[n],0);
    glScaled(0.3,0.3,0.3);
    glutSolidTorus(1,3,30,30);
    glPopMatrix();
    for(int i=-(EN_SIZE/2)+1;i<(EN_SIZE/2);i+=2){
    for(int j=-(EN_SIZE/2)+1;j<(EN_SIZE/2);j+=2){
    if(tola[i+(EN_SIZE/2)+1][j+(EN_SIZE/2)+1]!=0){
    glPushMatrix();
    glTranslated(i,0,j);
    house(tola[i+(EN_SIZE/2)+1][j+(EN_SIZE/2)+1],i,j);
    glPopMatrix();
}

void draw()
{
    double t = glutGet(GLUT_ELAPSED_TIME) / 1000.0;
    double a = t*90.0;
    TIME = t;

    ///Plane
    if(rotX>11)rotX=11;
    if(rotX<-11)rotX=-11;
    if(rotZ>10)rotZ=10;
```

```
if(rotZ<-15)rotZ=-15;
glPushMatrix();
glTranslated(0,1,0);
glRotated(90,0,1,0);
glRotated(rotX,1,0,0);
glRotated(rotY,0,1,0);
glRotated(rotZ,0,0,1);
plane();
glPopMatrix();

///Environment
if(tX>=4.1)tX=4.1;
if(tX<=-4.1)tX=-4.1;
if(tY>0.1)tY= 0.1;
if(tY<-15)tY=  -15;
glPushMatrix();
glTranslated(tX,tY,tZ);
environment(2);
glPopMatrix();
glPushMatrix();
glTranslated(tX,tY,tZ6);
environment(2);
glPopMatrix();
tZ+=speed;
tZ1+=speed;
tZ2+=speed;
tZ3+=speed;
tZ4+=speed;
tZ5+=speed;
tZ6+=speed;
if(tZ>=20)tZ=-110;
if(tZ1>=20)tZ1=-110;
if(tZ2>=20)tZ2=-110;
if(tZ3>=20)tZ3=-110;
if(tZ4>=20)tZ4=-110;
if(tZ5>=20)tZ5=-110;
if(tZ6>=20)tZ6=-110;
if(rotX>0)rotX-=angleBackFrac;
if(rotX<0)rotX+=angleBackFrac;
if(rotY<0)rotY+=angleBackFrac;
if(rotZ>0)rotZ-=angleBackFrac;
```

```
    if(rotZ<0)rotZ+=angleBackFrac;
    speed += 0.0002;
    if(speed>=0.7)speed=0.7;
}

void drawBitmapText(char *str,float x,float y,float z)
{
    char *c;
    glRasterPos3f(x,y+8,z);
    for (c=str; *c != '\0'; c++)
   {
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_10, *c);
} }
void front()
{
      glClear(GL_COLOR_BUFFER_BIT);
      glColor3f(0,0,1);
      output(250,600," BANGALORE INSTITUTE OF TECHNOLOGY",fonts[3]);
      glColor3f(0.7,0,1);
      output(250,570,"DEPARTMENT OF COMPUTER SCIENCE &
      ENGINEERING.",fonts[0]);
      glColor3f(0.8,0.1,0.2);
      output(280,500,"FLYING THROUGH HOOPS",fonts[2]);
      glColor3f(1.0,0.0,1.0);
      output(400,400,"SUBMITTED BY :",fonts[0]);
      glColor3f(0.3,0.5,0.8);
      output(130,350,"RAAGA SHETTY B P",fonts[3]);
      output(130,310,"NEHA RAVISHANKAR",fonts[3]);
      output(180,300,"",fonts[3]);
      output(680,350,"1BI17CS119",fonts[0]);
      output(680,310,"1BI17CS099",fonts[0]);
      glColor3f(0.6,0.25,0.0);
      output(300,100,"[ PRESS ANY KEY TO CONTINUE ]",fonts[3]);
}

void menuset()
{
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      glOrtho(0, 1000, 0.0, 750,-2000,1500);
      glMatrixMode(GL_MODELVIEW);
```

```
    glClear( GL_DEPTH_BUFFER_BIT | GL_COLOR_BUFFER_BIT);
}

static void display(void)
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    if(f==0)
    {
        menuset();
        front();
        glutSwapBuffers();
    }
    else if(f==1)
    {
       menuset();
       menus();
       glutSwapBuffers();
    }
    else if(f==3)
    {
        menuset();
        instructions();
        glutSwapBuffers();
    }
    else if(f==4)
    {
        menuset();
        intro();
        glutSwapBuffers();
    }
    else
    {
    glClearColor(1.0,0.8,0.3,0.0);
    }
const double t = glutGet(GLUT_ELAPSED_TIME) / 1000.0;
double a = t*90.0;
double aa=a;
if(!rot){
a=0; }
glLoadIdentity();
gluLookAt( 0.0, 4.5, 10.0,0, 4, 0,0, 1.0f, 0.0f);
```

```
if(START){
glPushMatrix();
glTranslated(0,0,0);
glScaled(zoom,zoom,zoom);
glRotated(a,0,1,0);
draw();
glPopMatrix();
drawStrokeText("UP: W, DOWN: S, LEFT: A, RIGHT: D, MAIN MENU: M",-
8,0.9,0);
drawStrokeText("TIME:",3,0,0);
int mod,number=0;
while(TIME){
mod=TIME%10;
number=number*10+mod;
TIME/=10;
} } }
else{
glPushMatrix();
glTranslated(0,3,0);

glRotated(aa,0,1,0);
glScaled(1.5,1.5,1.5);
plane();
glPopMatrix();
drawStrokeText("Press G to Start",-1,-1,0);
drawStrokeText2("Plane Game",-2,0,0);
}
glutSwapBuffers();
}

void key( unsigned char key, int x, int y )
{
float frac = 0.3;
float rotFrac = 1;
      if(f==0)
            f=1;
      else if(f==1)
      {
            switch(key)
            {
            case '1':f=2;break;
```

```
            case '2':f=3;break;
            case '3':f=4;break;
            case 'b':
            case 'B':f=0;break;
            case 'q':
            case 'Q':exit(0);
            }
      }
      else if(f==2)
      {
            switch(key)
            {
            case 'q':
            case 'Q':exit(0);break;
            case 'b':
            case 'B':f=1;break;
            case 'h':
            case 'H':f=0;break;
            case 'r': rot=true;
                  break;
            case 't': rot=false;
                  break;
            case 'z': zoom+=0.05;
                  break;
             case 'Z': zoom-=0.05;
            case 'w': tY-=frac;
                  rotZ+=rotFrac;
                  break;
            }
      }
  resize( 800,500 );
  idle( );
}

void menu(int val)
{
 switch(val)
 {
   case  1:if(speed==0)
               speed=0.5;
         else
```

```
            speed-=0.15;
          break;
  case 2:if(speed==0)
            speed=0.5;
          else
           speed+=0.25;
          break;
  case 3:speed=0;
          break;
  case 4:exit(0);
          break;
 }
}

/* Program entry point */
int main(int argc, char *argv[])
{
    glutInit(&argc, argv);
    glutInitWindowPosition(0,0);
    glutInitWindowSize(2000,1000);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
    glutCreateWindow("Flying Through Hoops");

    int sub_menu;
    sub_menu=glutCreateMenu(menu);
    glutAddMenuEntry("Slow",1);
    glutAddMenuEntry("Fast",2);
    glutCreateMenu(menu);
    glutAddSubMenu("Plane Speed",sub_menu);
    glutAddMenuEntry("Stop",3);
    glutAddMenuEntry("Exit",4);
    glutAttachMenu(GLUT_RIGHT_BUTTON);

    glutReshapeFunc(resize);
    glutDisplayFunc(display);
    glutKeyboardFunc(key);
    glutIdleFunc(idle);
    glClearColor(1.0,0.8,0.3,0.0);
    glEnable(GL_CULL_FACE);
    glCullFace(GL_BACK);
    glEnable(GL_DEPTH_TEST);
    glDepthFunc(GL_LESS);
```

```
    glEnable(GL_LIGHT0);
    glEnable(GL_NORMALIZE);
    glEnable(GL_COLOR_MATERIAL);
    glEnable(GL_LIGHTING);
    glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, high_shininess);
    glutMainLoop();
    return EXIT_SUCCESS;
}
```

Chapter - 5

SNAPSHOTS

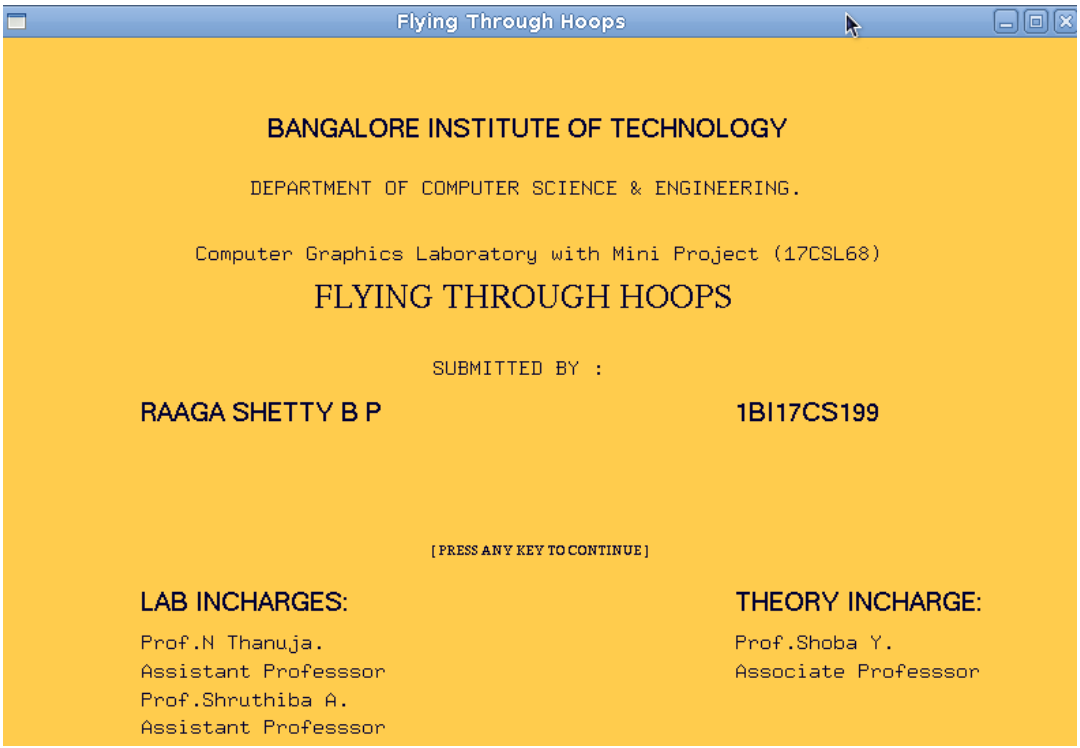


Fig. First page.

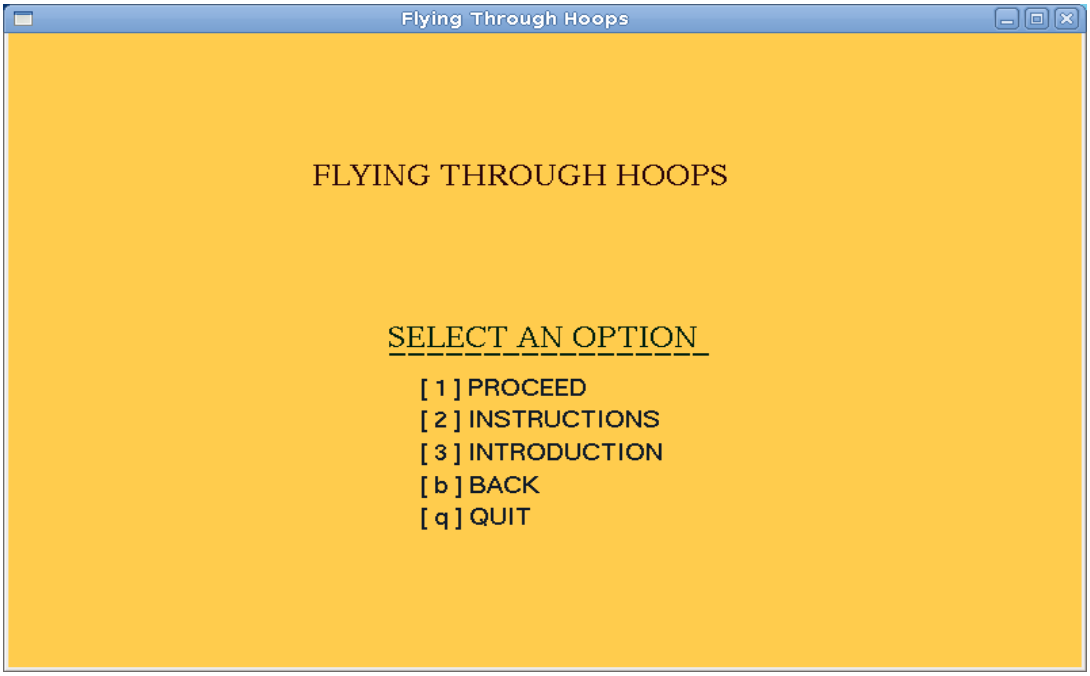


Fig. Options given to the user.

Fig. Game start page.

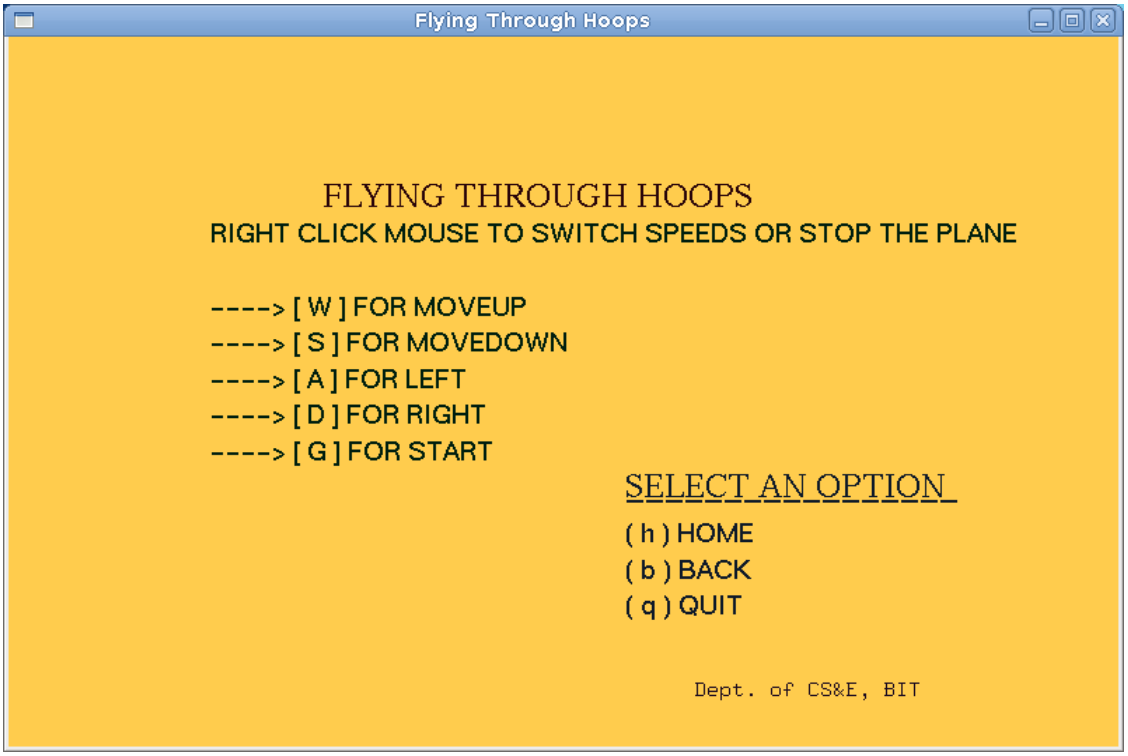


Fig. Instruction Page.

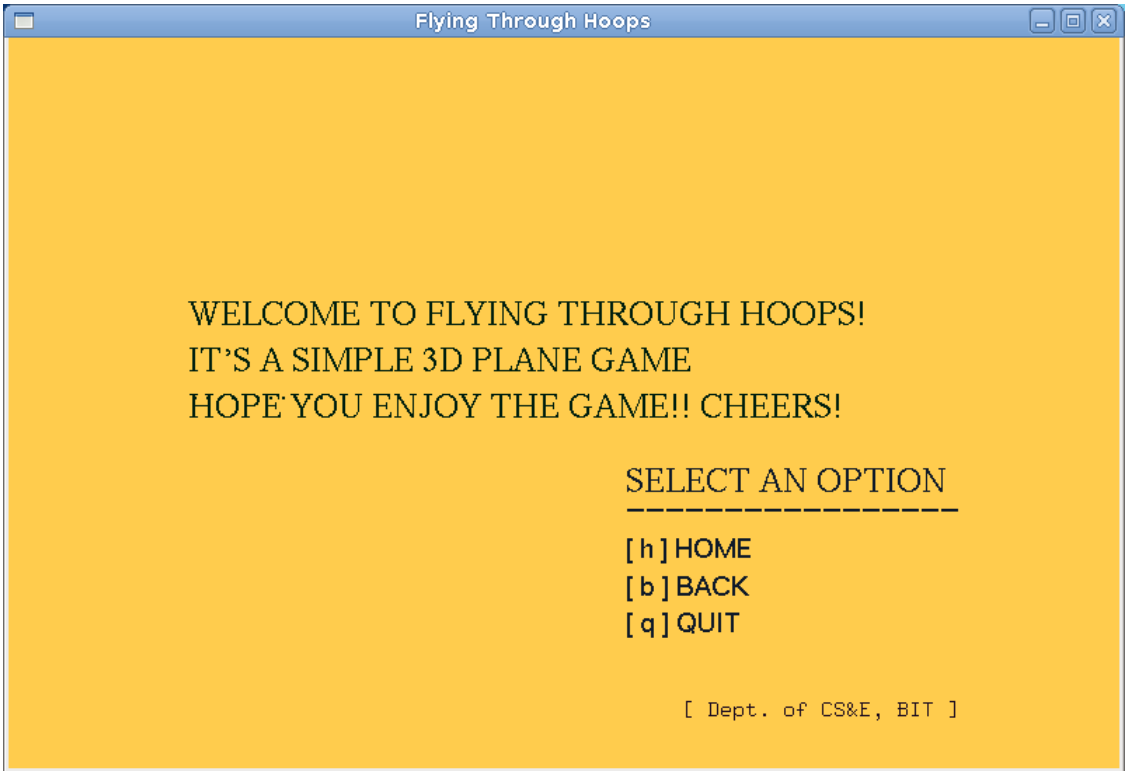


Fig. Introduction Page.

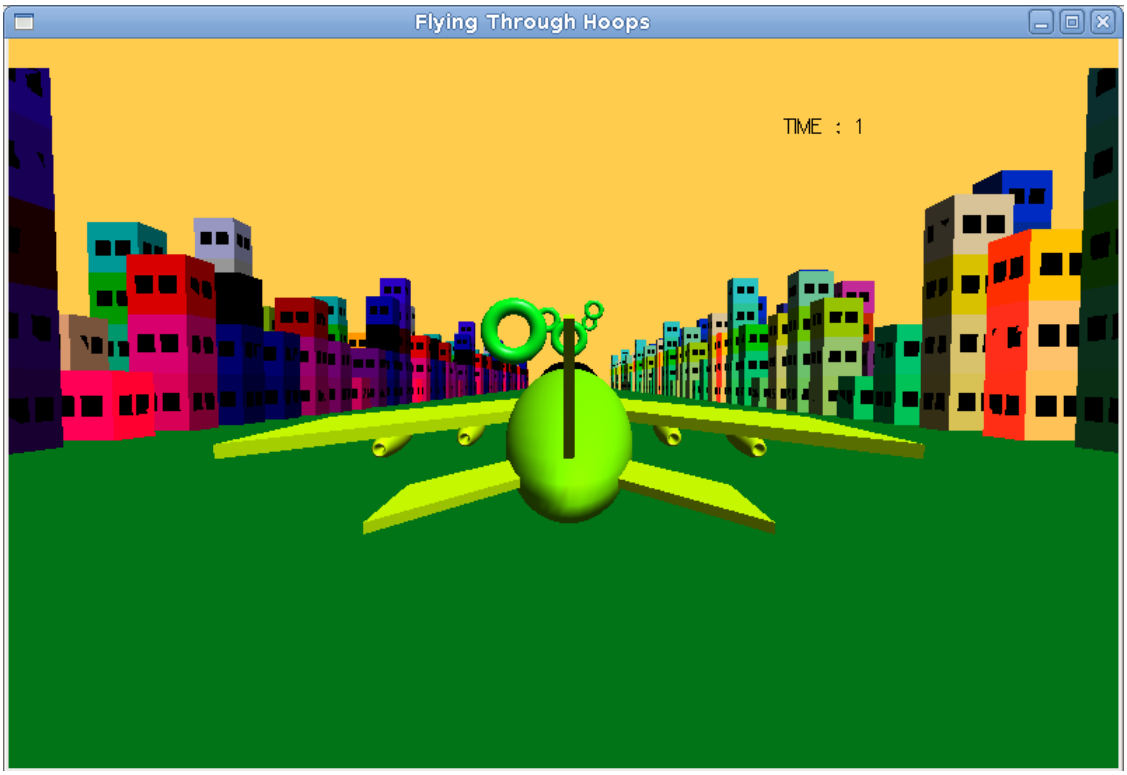


Fig. Game Scene – 1.

Fig. Game Scene – 2.

Fig. Menu Options.

Chapter - 6

CONCLUSION

"FLYING THROUGH HOOPS" is an animated 3D plane game which comes under the theme of animated movies.Through this project,we have learnt to implement the OpenGL transformation fuctions,lighting and shading concepts,menu options,input interaction(Keyboard/Mouse) and various other OpenGL functionalities.

Thus, in this project we have required a lot of knowledge about various techniques in OpenGL programming.

We would like to emphasize the importance of this project to many other perspectives of technical, mathematical, graphical and software concepts which we were unaware of.

Future Enhancement

- In future the same project can be enhanced in such a way that we can interact more with the project. As of now, we can only change the speed and view the different frames with information before the game begins.
- A score can be generated when the player misses to go through the hoop and we can also provide options to the player to select a different plane color.
- More features can be included and can be modified in a more versatile way,for example by including textures and other effects that can be used to improve the look and feel of the game.

BIBLIOGRAPHY

Reference Books

[1] Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version, 3rd/4th Edition, Pearson Education, 2011

[2] Edward Angel: Interactive Computer Graphics- A Top Down approach with OpenGL, 5th Edition, Pearson Education

[3] James D Foley, Andries Van Dam, Steven K Feiner, John F Huges Computer Graphics with OpenGL, Pearson Education

[4] Macro Cantu: Mastering Delphi

Websites

[1] https://www.opengl.org/

[2] http://stackoverflow.com/questions/ opengl

[3] https://geeksforgeeks.org/

[4] https://tutorialspoint.com/