# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Belgaum-590014

**Project Report**

**On**

# "AIRPLANE!"

**Submitted in partial fulfillment of the requirements for the VI Semester**

**Computer Graphics and Visualization Lab(10CSL67)**

**Bachelor of Engineering**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**For the Academic year**

**2012-2013**

**BY**

| | |
|---|---|
| **ANIL S** | **1PE10CS015** |
| **ARPITH K** | **1PE10CS018** |

**Department of Computer Science and Engineering**

**PESIT BANGALORE SOUTH CAMPUS**

**HOSUR ROAD, BENGALURU-560100**

**PESIT BANGALORE SOUTH CAMPUS**
**HOSUR ROAD, BENGALURU-560100**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

## ***CERTIFICATE***

*This is to certify that the project entitle* ***"AIRPANE!"*** *is a bonafide work carried out by* ***ARPITH K*** *and* ***ANIL S*** *bearing USN* ***1PE10CS018 and 1PE10CS015*** respectively , *in Computer Graphics and Visualization Lab (10CSL67) for the 6th Semester in partial fulfillment for the award of Degree of Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belgaum during the year 2012-2013.*

| | |
|---|---|
| ------------------------- | ------------------------- |
| ***Signature of the guide*** | ***Signature of the HOD*** |
| **Mrs Shubha Raj K. B.** | **Dr. Srikanta Murthy.K** |
| Asst Professor, Dept of CSE | HOD, Dept. of CS & E |
| PESIT BSC, Bengaluru. | PESIT BSC, Bengaluru. |

ACKNOWLEDGEMENT

It is beyond the comprehension of mere elegance of words to acknowledge someone who have been the guiding spirit behind this project.

We record our indebtedness to our internal guides **Mrs Shubha Raj K.B**, **Mrs Sarasvati V** and **Mrs Sudha**, Asst. professor who have not only coordinated our work, but also given suggestions from time to time to ensure that the quality of our projects is superlative.

We are grateful to **Dr. Srikanta Murty**, professor and HOD of Computer Science and Engineering department who has seen to it all of us are doing the project well.

With profound reference and high regards, we take this opportunity to thank our Principle/Director **Dr. J. Surya Prasad** for all his help and encouragement throughout the course of development of our project.

We gratefully acknowledge the help lent out by all the staff members of Computer Science and Engineering department of PESIT, BSC at difficult times.

We would like to express our sincere thanks to our friends, who have helped us directly or indirectly to make this work a success.

Anil S
Arpith K

## ABSTRACT

The aim of this project is to create an interesting and creative game using tools like OpenGL and C++ (g++ compiler). Apart from the gameplay, this project also deals with providing a beautiful graphical interface between the user and the system.

In this game, the main objective is to guide the airplane of your choice through a world full of danger before you run out of fuel! A user also has an ability to choose the world of his choice.

A system is put into place to maintain the score of the user. This will enable the user to keep track of the high scores of the player.

# INDEX

# 1. INTRODUCTION

## 1.1 Computer graphics

The term **computer graphics** includes almost everything on computers that is not text or sound. Today almost every computer can do some graphics, and people have even come to expect to control their computer through icons and pictures rather than just by typing.

Computer graphics are created using computers and the representation of image data by a computer specifically with help from specialized graphic hardware and software. The interaction and understanding of computers and interpretation of data has been made easier because of computer graphics. A computer graphic development has had a significant impact on many types of media and has revolutionized animation, movies and the video game industry.

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.

Computer generated imagery can be categorized into several different types: two dimensional (2D), three dimensional (3D), and animated graphics. As technology has improved, 3D computer graphics have become more common, but 2D computer graphics are still widely used. Computer graphics has emerged as a sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content. Over the past decade, other specialized fields have been developed like information visualization, and scientific visualization more concerned with "the visualization of three dimensional phenomena (architectural, meteorological, medical, biological, etc.), where the emphasis is on realistic renderings of volumes, surfaces, illumination sources, and so forth, perhaps with a dynamic (time) component".

## 1.2 OpenGL

Originally developed by Silicon Graphics in the early '90s, OpenGL® has become the most widely-used open graphics standard in the world. OpenGL is a software interface to graphics hardware. This interface consists of about 150 distinct commands that you use to specify the objects and operations needed to produce interactive three-dimensional applications.

**OpenGL** (**Open G**raphics **L**ibrary) is basically a cross-language, multi-platform API for rendering 2D and 3D computer graphics. The API is typically used to interact with a GPU, to achieve hardware-accelerated rendering.

## 1.3 GLUT

GLUT is the OpenGL Utility Toolkit, a window system independent toolkit for writing OpenGL programs. It implements a simple windowing application programming interface (API) for OpenGL.

The GLUT library has C, C++ (same as C), FORTRAN, and Ada programming bindings. The GLUT source code distribution is portable to nearly all OpenGL implementations and platforms. The current version is 3.7. Additional releases of the library are not anticipated.

The GLUT library supports the following functionality:

- Multiple windows for OpenGL rendering.
- Callback driven event processing.
- An `idle' routine and timers.
- Utility routines to generate various solid and wire frame objects.
- Support for bitmap and stroke fonts.
- Miscellaneous window management functions.

## 1.4 FreeGLUT

FreeGLUT is a completely Open Sourced alternative to the OpenGL Utility Toolkit (GLUT) library. GLUT was originally written by Mark Kilgard. Since then, GLUT has been used in a wide variety of practical applications because it is simple, widely available and highly portable.

However, the original GLUT library seems to have been abandoned with the most recent version (3.7) dating back to August 1998.

As with GLUT, FreeGLUT allows the user to create and manage windows containing OpenGL contexts on a wide range of platforms and also read the mouse, keyboard and joystick functions.

## 1.5 SOIL (Simple OpenGL Image Library)

The Simple OpenGL Image Library or SOIL is a public domain image loading library, written in C. The library allows users to load images directly to OpenGL textures. The Simple OpenGL Image Library can load bitmap, jpeg, jpg, png, tga, and dds files.

# 2. SYSTEM REQUIREMENT SPECIFICATION

## 2.1 HARDWARE REQUIREMENT

- CPU: Intel/AMD CPU
- RAM (Main memory): 512 MB
- Hard disk: 10MB of free space
- Hard disk speed (in RPM): 5400 RPM
- Mouse: 2 button mouse
- Keyboard: Standard keyboard with arrow keys
- Monitor: 1366*768 display resolution

## 2.2 SOFTWARE REQUIREMENT

- Operating System: Any Linux based operating system (like Ubuntu) (64bit)
- Code::Blocks with OpenGL and SOIL libraries
- Mouse driver
- Graphic driver

# 3. PROJECT DESCRIPTION

The goal of this game is to fly as far as possible, with only two controls at your disposal, this is one of those “easy to learn, difficult to master” situations. You’ll first select a plane and a scene. Click once to start make your little guy take off from an airport, and it’s on!!

As you fly, right click to begin ascending; the plane descends automatically. So it’s critical to master this bouncy method of flying.

Missiles of varying sizes are shot at you. So, you’ll spend most of your time avoiding these. Keep a close eye at the fuel gauge! You’ll steadily be running out of fuel as you try to gain height. Hit a missile, run out of fuel, or fly into the bottom edge of the screen, and you’ll crash and burn!

As far as the game interface is concerned, you’ll be greeted with a loading screen followed by a splash screen with the name of the developers (that’s Arpith and Anil!). You’ll then see a menu with screen with the following items:

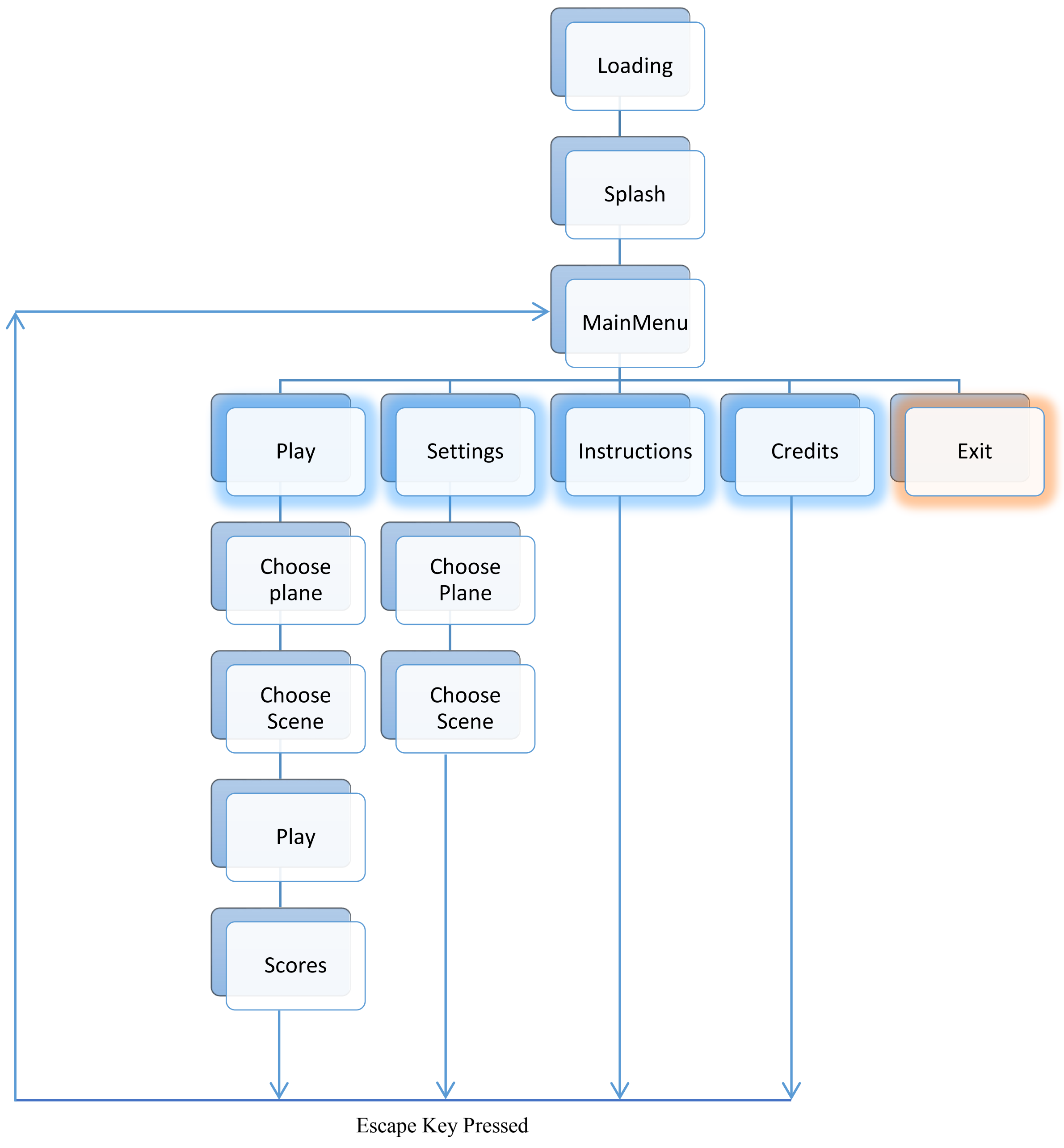
- Play
- Settings
- Instructions
- Credits
- High Scores
- Exit

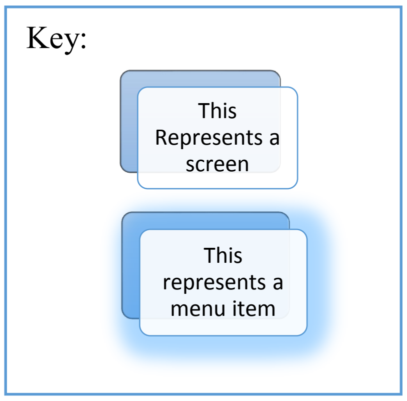
The user can either start the game directly, if he knows how the game works or has played it before, by clicking in the box with “Play” option. Otherwise, he can view the instruction as to the game works by clicking on the “Instruction” button. If the player has changed his mind to play the game sometime later, he can click on “Exit” button which terminates the game.

When the player hits on the Instruction button, another page which describes how the game works appears. Similarly, credits page will show you the name of the developers. Setting page will allow you to choose the plane and environment scene of your wish. Pressing escape at any point of time will bring you to the main page.

This easy to play game can be further developed to include new features. You may unlock new planes as you progress through the game. Each plane may have its own ability from getting invisible or using time warp to a powerful explosive power. Players may left click on the screen to initiate a power-up.

# 4. SYSTEM DESIGN





# 5. IMPLEMENTATION

## 5.1 API'S USED

The following APIs have been used in this project. The API name along with its description is as follows:

**1. glutMainLoop(void);**

It causes the program to begin an event-processing loop.

**2. glutReshapeFunc(void (GLUTCALLBACK *func)(int width, int height));**

The reshape event is generated whenever the window is resized, such as by a user interaction.

**3. glutSwapBuffers(void);**

We can swap the front and back buffers at will from the application programs.

**4. glutStrokeCharacter(void *font, int character);**

Without using any display lists, glutStrokeCharacter renders the character in the named stroke font.

**5. glPushMatrix();**

Set current matrix on the stack

**6. glPopMatrix();**

Pop the old matrix without the transformations.

**7. void glTranslatef(GLfloat *x*, GLfloat *y*, GLfloat *z*);**

glTranslate produces a translation by (*x*, *y*, *z* ).

**8. void glLineWidth(GLfloat *width*);**

Specifies the rasterized width of both aliased and antialiased lines.

**9. void glBindTexture(GLenum *target*, GLuint *texture*);**

Lets you create or use a named texture.

**10. void glTexEnvf(GLenum *target*, GLenum *pname*, GLfloat *param*);**

Specifies a texture environment.

**11. void glEnable(GLenum *cap*);**

Enable server-side GL capabilities

**12. void glDisable(GLenum *cap*);**

Disable server-side GL capabilities

**13. SetFont(string family [], string style [], float size)**

Sets the font used to print character strings.

**14. void glutTimerFunc(unsigned int msecs,void (*func)(int value), value);**

Registers the timer callback func to be triggered in at least msecs milliseconds.

**15. void glutPostRedisplay(void);**

Mark the normal plane of *current window* as needing to be redisplayed.

**16.void glClearColor(GLfloat *red*, GLfloat *green*, GLfloat *blue*, GLfloat *alpha*) ;**

Specifies the red, green, blue, and alpha values used by glClear to clear the color buffers.

**17. void glutSpecialFunc(void (*func)(int key, int x, int y));**
Sets the special keyboard callback for the *current window*.

**18. void glutMainLoop(void);**
Enters the GLUT event processing loop. This routine should be called at most once in a GLUT program.

**19. glColor3f (GLfloat red, GLfloat green, GLfloat blue);**

It sets color to current drawing.

**20. glLoadIdentity (void);**

To initialize the current transform matrix to the identity transform.

**21. glMatrixMode (GLenum mode);**

It switches matrix mode between the two matrices –

- MODEL_VIEW (GL_MODELVIEW)
- PROJECTION (GL_PROJECTION)

**22. glOrtho (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble zNear, GLdouble zFar);**

It establishes as a view volume a parallelepiped that extends from left to right in x, bottom to top in y and near to far in z.

**23. glVertex3f (GLfloat x, GLfloat y, GLfloat z);**

It is used to represent vertex.

**24. glViewport (GLint x, GLint y, GLsizei width, GLsizei height);**

It specifies that the viewport will have lower left corner (x,y) in screen co-ordinates and will be width pixels wide and height pixels high.

**25. glutBitmapCharacter(void *font, int character);**

The character is placed at the present raster position on the display, is measured in pixels and can be altered by the various forms of the function glRasterPos*.

**26. glutCreateWindow(const char *title);**

It creates and opens OpenGL window with the title passed as the argument.

**27. glutDisplayFunc(void (GLUTCALLBACK *func)(void));**

It sends graphics to screen.

**28. glutIdleFunc(void (GLUTCALLBACK *func)(void));**

It is used to increase theta by fixed amount whenever nothing else is happening.

**29. glutInit(int *argcp, char **argv);**

It initiates interaction between windowing system and OpenGL.

**30. glutInitDisplayMode(unsigned int mode);**

This function specifies how the display should be initialized. The constants GLUT_SINGLE and GLUT_RGB, which are ORed together, indicate that a single display buffer should be allocated and the colors are specified using desired amount of red, green and blue.

**31. glutInitWindowSize(int width, int height);**

It sets the size of created window.

**32. glutKeyboardFunc(void (GLUTCALLBACK *func)(unsigned char key, int x, int y));**

The keyboard event is generated when the mouse is in the window and one of the key is pressed or released. This GLUT function is the call back for event generated by pressing a key.

## 5.2 User Defined Functions

**1. void drawString(float x,float y,float z,char* string)**

Prints a string of Bitmap characters onto the screen at position determined by the coordinates (x,y,z)

**3. void draw_fin_text()**

Displays the score and number of missiles dodged, in the finish screen.

**4. void draw_credit_text()**

Displays credits in the credit screen (when page=22)

**5. void draw_high_text()**

Displays high scores and maximum missiles dodged by the player.

**6. void draw_menu_text()**

Draws menu items on the screen

**7. void draw_inst_text()**

Displays instructions onto the screen

**8. void draw_chScene_text()**

Allows you to select the scene (background)

**9. void draw_chPlane_text()**

Allows you to choose the plane of your choise.

**10. void draw_score()**

Displays scores during game play

**11. void select_scene()**

Sets the scene variable to contain the name of the background image file based on the value of ch_scene variable. This variable (ch_scene) is updated in the specialkeys function

**12. void rocket1(int x_cor, int y_cor)**

Draws the first kind of rocket onto the screen at position determined by (x,y) coordinates.

**13. void rocket2(int x_cor, int y_cor)**

Draws the second kind of rocket onto the screen at position determined by (x,y) coordinates.

**14. void rocket3(int x_cor, int y_cor)**

Draws the third kind of rocket onto the screen at position determined by (x,y) coordinates.

**15. void draw_rockets()**

Determines the number of rocket to be displayed on the screen at any point of time can then calls the rocket<X> function. (Here 1<=X<=3)

**16. void draw_chosen_plane()**

Display the chosen plane on the screen where you have to select a polane of your choice.

**17. void drawLogo()**

Displays the college logo in the credit page.

**18. void RenderScene()**

RenderScene is the display callback function which is responsible to display various onscreen elements and call the above functions based on the value of page variable.

# 6. SOURCE CODE

```
#include <iostream>
#include <GL/glut.h>
#include "src/SOIL.h"
#include<string.h>
#include <stdio.h>

#define UP 1
#define DOWN 0
#define MAX 10
#define SIZE_MIS_X 55   //determines the size of missiles
#define SIZE_MIS_y 30   //determines the size of missiles
#define MAX_MISSILES 3  //maximum number of missiles in game

using namespace std;

//for choose scene
char scene[50];
int ch_scene=1;

//high scores
int max_dist, max_miss;

//flag to check if you were hit by a missile
int hit_missile=0;

//flad to determint the plane chosen by the user
int plane_choice=2;

//flag for setting
int setting=0;

void* currentfont;

float x_step=-171.0;     //for loading bar movement (in pg=0)
float y_cre=0;           //for credits text moveemnt
float y_pos=0;         //y axis position of plane
float theta=0;         //angle of the plane
bool state;            //state of plane (either going up or down)

int update_mis;

GLfloat x = 0.0f;      //background screen position

GLfloat fuel=98;       //fuel left in plane
GLfloat dist, missiles; //dustance travelled and missiled douged
GLfloat missile_x=250,missile_y[MAX_MISSILES]= {0};//position of missiles
int no_of_missiles=3;   //determines number of missiles in the game
int frames = 5, full = 1;
int i_bck,i_mis1,i_mis2,i_mis3,i_plane,i_inst21,i_cre22,i_sel31,i_sel32,
i_fin4,i_2,i_23;
int i;

GLfloat windowWidth;
GLfloat windowHeight;

GLuint tex_2d, tex_2d_mis[4], tex_2d_plane;
GLuint tex_2d_0, tex_2d_1, tex_2d_2, tex_2d_21, tex_2d_22, tex_2d_31, tex_2d_4,
tex_2d_23;

//determines the current screen
int page=0;
/*****************************************************************
Page no        name
-------        ---------------
0              Loading screen

1              Splash

2              Menu
21             Instructions
22             Credits
23             High scores

31             Choose plane
32             Choose scene
3              Actual game play

4              Finish screen
*****************************************************************/

//class to draw the plane
class plane
{
public:
        float x[MAX], y[MAX], i;
        float ymax, ymin;
        int button;
        plane()
        {
                x[1]=-30, x[2]=30, x[3]=30, x[4]=-30;
                y[1]=15, y[2]=15, y[3]=-15, y[4]=-15;
        }
        void draw_plane()
        {
                glEnable(GL_TEXTURE_2D);
                glEnable(GL_BLEND);
                glColor4f(1.0f, 1.0f, 1.0f, 1.0f);
                if(plane_choice==1)
                {
                        if (i_plane == 0)
                        {
                                tex_2d_plane = SOIL_load_OGL_texture
                                                (
                                                        "res/plane1.png",
                                                        SOIL_LOAD_AUTO,
                                                        SOIL_CREATE_NEW_ID,
                                                        SOIL_FLAG_MULTIPLY_ALPHA
                                                );
                                i_plane = 1;
                        }
                }
                if(plane_choice==2)
                {
                        if (i_plane == 0)
                        {
                                tex_2d_plane = SOIL_load_OGL_texture
                                                (
                                                        "res/plane2.png",
                                                        SOIL_LOAD_AUTO,
                                                        SOIL_CREATE_NEW_ID,
                                                        SOIL_FLAG_MULTIPLY_ALPHA
                                                );
                                i_plane = 1;
                        }
                }
                glBindTexture(GL_TEXTURE_2D, tex_2d_plane);
                glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
GL_MODULATE);
                glBlendFunc(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA);
                glBegin(GL_POLYGON);
                glTexCoord2f(0.0, 0.0);
                glVertex2f(x[1], y[1]);
                glTexCoord2f(1.0, 0.0);
                glVertex2f(x[2], y[2]);
                glTexCoord2f(1.0, 1.0);
                glVertex2f(x[3], y[3]);
                glTexCoord2f(0.0, 1.0);
                glVertex2f(x[4], y[4]);
                glEnd();
                glDisable(GL_TEXTURE_2D);
                glDisable(GL_BLEND);
        }
} plane1;

void keyboard(unsigned char key, int x, int y)
{
        switch (key)
```

```
	{
		//27 is the ASCII value of the ESC key
	case 27:
		x=0.0;
		if(page!=2)
		{

	i_bck=0,i_mis1=0,i_mis2=0,i_mis3=0,i_plane=0,i_sel31=0,i_sel32=0,
i_fin4=0;
			i_23=0;
			missiles=0;
			dist=0;
			setting=0;
			hit_missile=0;
			y_pos=0;
			missile_x=250;
			fuel=98;
			y_cre=0;
			x=0.0;
			cout<<full<<endl;
			cout<<x<<endl;
			page=2;
		}
		else
			//exit game
			exit(0);
		break;

	case 'f':	//full screen
		if (full == 0)
		{
			glutFullScreen();
			full = 1;
		}
		else
		{
			glutReshapeWindow(800, 450);
			glutPositionWindow(320,150);
			full = 0;
		}
	}
}

//Special keys have been used to choose scene (in page 32)
void SpecialKeys(int key, int x, int y)
{
	if(page==32)
	{
		switch (key)
		{
```

```
                    case GLUT_KEY_RIGHT:
                              if(ch_scene<5)
                              {
                                        ch_scene++;
                                        i=0;
                                        cout<<ch_scene<<endl;
                              }
                              break;

                    case GLUT_KEY_LEFT:
                              if(ch_scene>1)
                              {
                                        ch_scene--;
                                        i=0;
                                        cout<<ch_scene<<endl;
                              }
                              break;
                    }
                    glutPostRedisplay();
          }
}

//For glutBitmapCharacter (used only in page 1)
void setFont(void* font)
{
          currentfont=font;
}

void drawString(float x,float y,float z,char* string)
{
          char* c;
          glRasterPos3f(x,y,z);

          for(c=string; *c!='\0'; c++)
          {         glColor3f(0.0,0.0,0.0);
                    glutBitmapCharacter(currentfont,*c);
          }
}

//Determines the action on mouse click event
void Mouse(int button, int m_state, int m_x, int m_y)
{
          if(page==1)
          {
                    if(m_x>620 && m_y>260 && m_state==GLUT_UP)
                    {
                              cout<<m_x<<" "<<m_y<<endl;
                              page=2;
                    }
          }
```

```
else if(page==2)
{
        if(full==0)
        {
                if(m_state==GLUT_UP)
                        cout<<m_x<<" "<<m_y<<endl;
                if(m_y>92 && m_y<116 && m_state==GLUT_UP)
                {
                        cout<<"play"<<endl;
                        page=31;
                }
                if(m_y>126 && m_y<149 && m_state==GLUT_UP)
                {
                        cout<<"Settings"<<endl;
                        page=31;
                        setting=1;
                }
                if(m_y>158 && m_y<182 && m_state==GLUT_UP)
                {
                        cout<<"Instructions"<<endl;
                        page=21;
                }
                if(m_y>193 && m_y<216 && m_state==GLUT_UP)
                {
                        cout<<"Credit"<<endl;
                        page=22;
                }
                if(m_y>226 && m_y<256 && m_state==GLUT_UP)
                {
                        cout<<"High"<<endl;
                        page=23;
                        //exit(0);
                }
                if(m_y>260 && m_state==GLUT_UP)
                {
                        cout<<"exit"<<endl;
                        exit(0);
                }
        }
        else
        {
                if(state==GLUT_UP)
                        cout<<m_x<<" "<<m_y<<endl;
                if(m_y>154 && m_y<195 && m_state==GLUT_UP)
                {
                        cout<<"play"<<endl;
                        page=31;
                }
                if(m_y>213 && m_y<251 && m_state==GLUT_UP)
                {
```

```
                                        cout<<"Settings"<<endl;
                                        page=31;
                                        setting=1;
                                }
                                if(m_y>269 && m_y<310 && m_state==GLUT_UP)
                                {
                                        cout<<"Instructions"<<endl;
                                        page=21;
                                }
                                if(m_y>329 && m_y<366 && m_state==GLUT_UP)
                                {
                                        cout<<"Credit"<<endl;
                                        page=22;
                                }
                                if(m_y>387 && m_y<425 && m_state==GLUT_UP)
                                {
                                        cout<<"High"<<endl;
                                        page=23;
                                }
                                if(m_y>430 && m_state==GLUT_UP)
                                {
                                        cout<<"exit"<<endl;
                                        exit(0);
                                }

                        }

                }
                else if(page==31)
                {
                        if(button==GLUT_LEFT_BUTTON && m_state==GLUT_UP)
                        {
                                if(full==0)
                                {
                                        cout<<m_x<<" "<<m_y<<endl;
                                        if(m_y>168 && m_y<205)
                                        {
                                                plane_choice=1;
                                                i_plane=0;
                                                glutPostRedisplay();
                                        }
                                        if(m_y>242 && m_y<287)
                                        {
                                                plane_choice=2;
                                                i_plane=0;
                                                glutPostRedisplay();
                                        }
                                        if(m_y>300)
                                        {
                                                cout<<"next"<<endl;
                                                i_plane=0;
                                                page=32;
                                        }
                                }
                                else
                                {
                                        if(m_y>307 && m_y<347)
                                        {
                                                plane_choice=1;
                                                i_plane=0;
                                                glutPostRedisplay();
                                        }
                                        if(m_y>416 && m_y<463)
                                        {
                                                plane_choice=2;
                                                i_plane=0;
                                                glutPostRedisplay();
                                        }
                                        if(m_y>470)
                                        {
                                                cout<<"next"<<endl;
                                                i_plane=0;
                                                page=32;
                                        }
                                }
                        }
                }
                else if(page==32)
                {
                        if(button==GLUT_LEFT_BUTTON && m_state==GLUT_UP)
                        {
                                if(full==0)
                                {
                                        if(m_y>300)
                                        {
                                                cout<<"next"<<endl;
                                                i_plane=0;
                                                if(setting==1)
                                                {
                                                        setting=0;
                                                        page=2;
                                                }
                                                else
                                                        page=3;
                                        }
                                }
                                else
                                {
                                        if(m_y>470)
                                        {
```

```
                                                cout<<"next"<<endl;
                                                i_plane=0;
                                                if(setting==1)
                                                {
                                                        setting=0;
                                                        page=2;
                                                }
                                                else
                                                        page=3;
                                        }
                                }
                        }
                }
                else if(page==3)
                {
                        if(button==GLUT_LEFT_BUTTON && m_state==GLUT_DOWN)
                        {
                                state=UP;
                                cout<<"Going Up!"<<endl;
                        }
                        else if(button==GLUT_LEFT_BUTTON && m_state==GLUT_UP)
                        {
                                state=DOWN;
                                cout<<"Going Down"<<endl;
                        }
                }
}

//draw text in finifh screen (page=4)
void draw_fin_text()
{
                char string[6][40];
                int i,lengthOfString;

                strcpy(string[0],"WOW!!");
                sprintf(string[1],"You just travelled %d meters",(int)dist);
                sprintf(string[2],"and douged %d missiles",(int)missiles);
                sprintf(string[3],"before you ran out of fuel!!");
                sprintf(string[4],"before you were hit by one!!");

                //update high scores;
                if((int)dist>max_dist)
                {
                   max_dist=dist;
                   cout<<max_dist<<" "<<dist<<endl;
                }
            if(missiles>max_miss)
            {
                   max_miss=missiles;
                   //cout<<max_miss<<endl;
```

```
	}

		glLineWidth(4);
		glPushMatrix();
		glTranslatef(-105,55,0);
		glScalef(0.3,0.3,0.3);
		lengthOfString = (int)strlen(string[0]);
		for(i=0; i<lengthOfString; i++)
		{
			glColor3f(1,1,1);
			glutStrokeCharacter(GLUT_STROKE_ROMAN,string[0][i]);
		}
		glPopMatrix();

		glLineWidth(3);
		glPushMatrix();
		glTranslatef(-155,35,0);
		glScalef(0.1,0.1,0.1);
		lengthOfString = (int)strlen(string[1]);
		for(i=0; i<lengthOfString; i++)
		{
			glColor3f(1,1,1);
			glutStrokeCharacter(GLUT_STROKE_ROMAN,string[1][i]);
		}
		glPopMatrix();

		glPushMatrix();
		glTranslatef(-155,20,0);
		glScalef(0.1,0.1,0.1);
		lengthOfString = (int)strlen(string[2]);
		for(i=0; i<lengthOfString; i++)
		{
			glColor3f(1,1,1);
			glutStrokeCharacter(GLUT_STROKE_ROMAN,string[2][i]);
		}
		glPopMatrix();

		if(!hit_missile)
		{
			glPushMatrix();
			glTranslatef(-155,5,0);
			glScalef(0.1,0.1,0.1);
			lengthOfString = (int)strlen(string[3]);
			for(i=0; i<lengthOfString; i++)
			{
				glColor3f(1,1,1);
				glutStrokeCharacter(GLUT_STROKE_ROMAN,string[3][i]);
			}
			glPopMatrix();
		}
```

```
        else
        {
                glPushMatrix();
                glTranslatef(-155,5,0);
                glScalef(0.1,0.1,0.1);
                lengthOfString = (int)strlen(string[4]);
                for(i=0; i<lengthOfString; i++)
                {
                        glColor3f(1,1,1);
                        glutStrokeCharacter(GLUT_STROKE_ROMAN,string[4][i]);
                }
                glPopMatrix();
        }
}

//draw text in credit screen (page=22)
void draw_credit_text()
{
        char string[5][50];
        int i,lengthOfString;

        strcpy(string[3],"PES Institute of Technology, BSC");
        strcpy(string[2],"Thank you!");
        strcpy(string[1],"Arpith (1PE10CS018)");
        strcpy(string[0],"Anil S (1PE10CS015)");
        strcpy(string[4],"Special thanks to: Prof Sarasvathi V");


        glLineWidth(1);

        glPushMatrix();
        glTranslatef(-105,140-y_cre,0);
        glScalef(0.1,0.1,0.1);
        lengthOfString = (int)strlen(string[3]);
        for(i=0; i<lengthOfString; i++)
        {
                glColor3f(1,1,1);
                glutStrokeCharacter(GLUT_STROKE_ROMAN,string[3][i]);
        }
        glPopMatrix();

        glPushMatrix();
        glTranslatef(-35,110-y_cre,0);
        glScalef(0.1,0.1,0.1);
        lengthOfString = (int)strlen(string[2]);
        for(i=0; i<lengthOfString; i++)
        {
                glColor3f(1,1,1);
                glutStrokeCharacter(GLUT_STROKE_ROMAN,string[2][i]);
        }
```

```
		glPopMatrix();

		glPushMatrix();
		glTranslatef(-160,-90+y_cre,0);
		glScalef(0.1,0.1,0.1);
		lengthOfString = (int)strlen(string[1]);
		for(i=0; i<lengthOfString; i++)
		{
			glColor3f(1,1,1);
			glutStrokeCharacter(GLUT_STROKE_ROMAN,string[1][i]);
		}
		glPopMatrix();

		glPushMatrix();
		glTranslatef(25,-90+y_cre,0);
		glScalef(0.1,0.1,0.1);
		lengthOfString = (int)strlen(string[0]);
		for(i=0; i<lengthOfString; i++)
		{
			glColor3f(1,1,1);
			glutStrokeCharacter(GLUT_STROKE_ROMAN,string[0][i]);
		}
		glPopMatrix();

		glPushMatrix();
		glTranslatef(-125,-110+y_cre,0);
		glScalef(0.1,0.1,0.1);
		lengthOfString = (int)strlen(string[4]);
		for(i=0; i<lengthOfString; i++)
		{
			glColor3f(1,1,1);
			glutStrokeCharacter(GLUT_STROKE_ROMAN,string[4][i]);
		}
		glPopMatrix();

}


void draw_high_text()
{
	char string_high[10][20];
		int lengthOfString,i;

		strcpy(string_high[0],"High Scores: ");
		sprintf(string_high[1],"Distance: %d",max_dist);
		sprintf(string_high[2],"Missiles: %d",max_miss);

		glLineWidth(3);
		glPushMatrix();
		glTranslatef(0,30,0);
```

```
        glScalef(0.2,0.2,0.2);
        lengthOfString = (int)strlen(string_high[0]);
        for(i=0; i<lengthOfString; i++)
        {
                glColor3f(1,1,1);
                glutStrokeCharacter(GLUT_STROKE_ROMAN,string_high[0][i]);
        }
        glPopMatrix();

      glLineWidth(2);
        glPushMatrix();
        glTranslatef(70,0,0);
        glScalef(0.1,0.1,0.1);
        lengthOfString = (int)strlen(string_high[1]);
        for(i=0; i<lengthOfString; i++)
        {
                glColor3f(1,1,1);
                glutStrokeCharacter(GLUT_STROKE_ROMAN,string_high[1][i]);
        }
        glPopMatrix();

        glPushMatrix();
        glTranslatef(70,-15,0);
        glScalef(0.1,0.1,0.1);
        lengthOfString = (int)strlen(string_high[2]);
        for(i=0; i<lengthOfString; i++)
        {
                glColor3f(1,1,1);
                glutStrokeCharacter(GLUT_STROKE_ROMAN,string_high[2][i]);
        }
        glPopMatrix();
}

//draw text in page 2 (menu screen)
void draw_menu_text()
{

        char string_menu[10][20];
        int lengthOfString,i;

        strcpy(string_menu[0],"Play!");
        strcpy(string_menu[1],"Settings");
        strcpy(string_menu[2],"Instructions");
        strcpy(string_menu[3],"Credit");
        strcpy(string_menu[4],"High Scores");
        strcpy(string_menu[5],"Exit..");

        glLineWidth(1);
        glPushMatrix();
        glTranslatef(-120,50,0);
```

```
glScalef(0.1,0.1,0.1);
lengthOfString = (int)strlen(string_menu[0]);
for(i=0; i<lengthOfString; i++)
{
        glColor3f(1,1,1);
        glutStrokeCharacter(GLUT_STROKE_ROMAN,string_menu[0][i]);
}
glPopMatrix();

glPushMatrix();
glTranslatef(-120,35,0);
glScalef(0.1,0.1,0.1);
lengthOfString = (int)strlen(string_menu[1]);
for(i=0; i<lengthOfString; i++)
{
        glColor3f(1,1,1);
        glutStrokeCharacter(GLUT_STROKE_ROMAN,string_menu[1][i]);
}
glPopMatrix();

glPushMatrix();
glTranslatef(-120,20,0);
glScalef(0.1,0.1,0.1);
lengthOfString = (int)strlen(string_menu[2]);
for(i=0; i<lengthOfString; i++)
{
        glColor3f(1,1,1);
        glutStrokeCharacter(GLUT_STROKE_ROMAN,string_menu[2][i]);
}
glPopMatrix();

glPushMatrix();
glTranslatef(-120,5,0);
glScalef(0.1,0.1,0.1);
lengthOfString = (int)strlen(string_menu[3]);
for(i=0; i<lengthOfString; i++)
{
        glColor3f(1,1,1);
        glutStrokeCharacter(GLUT_STROKE_ROMAN,string_menu[3][i]);
}
glPopMatrix();

glPushMatrix();
glTranslatef(-120,-10,0);
glScalef(0.1,0.1,0.1);
lengthOfString = (int)strlen(string_menu[4]);
for(i=0; i<lengthOfString; i++)
{
        glColor3f(1,1,1);
        glutStrokeCharacter(GLUT_STROKE_ROMAN,string_menu[4][i]);
```

```
		}
		glPopMatrix();

		glPushMatrix();
		glTranslatef(-120,-25,0);
		glScalef(0.1,0.1,0.1);
		lengthOfString = (int)strlen(string_menu[5]);
		for(i=0; i<lengthOfString; i++)
		{
			glColor3f(1,1,1);
			glutStrokeCharacter(GLUT_STROKE_ROMAN,string_menu[5][i]);
		}
		glPopMatrix();
}

//draw text in page 21 (instruction screen)
void draw_inst_text()
{
		char string[15][120];
		int i,lengthOfString;

		strcpy(string[0],"Instructions:");
		strcpy(string[1],"The main objectiove of this game is to go as far");
		strcpy(string[2],"as possible in your plane, without hitting the");
		strcpy(string[3],"missiles.");
		strcpy(string[4],"Press right mouse button to increase altitude!");
		strcpy(string[5],"Leaving it will automatically take you down.");
		strcpy(string[6],"Keep a close eye on the fuel guage though :P");
		strcpy(string[7],"Have FUN!!!");
		glLineWidth(3);

		glPushMatrix();
		glTranslatef(-40,50,0);
		glScalef(0.3,0.3,0.3);
		lengthOfString = (int)strlen(string[0]);
		for(i=0; i<lengthOfString; i++)
		{
			glColor3f(1,1,1);
			glutStrokeCharacter(GLUT_STROKE_ROMAN,string[0][i]);
		}
		glPopMatrix();

		glLineWidth(1);
		int y_pos_21=20;
		for(int k_t=1; k_t<=7; k_t++)
		{
			glPushMatrix();
			glTranslatef(-40,y_pos_21-=10,0);
			glScalef(0.06,0.06,0.06);
			lengthOfString = (int)strlen(string[k_t]);
```

```
                for(i=0; i<lengthOfString; i++)
                {
                        glColor3f(1,1,1);

        glutStrokeCharacter(GLUT_STROKE_ROMAN,string[k_t][i]);
                }
                glPopMatrix();
        }

}

//draw text in page 32 (choose plane)
void draw_chScene_text()
{
        char string[15][120];
        int i,lengthOfString;


        strcpy(string[0],"Choose Scene");
        strcpy(string[1],"Use arrow keys");
        strcpy(string[2],"Next");

        glLineWidth(2);
        glPushMatrix();
        glTranslatef(-130,55,0);
        glScalef(0.3,0.3,0.3);
        lengthOfString = (int)strlen(string[0]);
        for(i=0; i<lengthOfString; i++)
        {
                glColor3f(1,1,1);
                glutStrokeCharacter(GLUT_STROKE_ROMAN,string[0][i]);
        }
        glPopMatrix();

        glLineWidth(3);
        glPushMatrix();
        glTranslatef(-75,35,0);
        glScalef(0.15,0.15,0.15);
        lengthOfString = (int)strlen(string[1]);
        for(i=0; i<lengthOfString; i++)
        {
                glColor3f(1,1,1);
                glutStrokeCharacter(GLUT_STROKE_ROMAN,string[1][i]);
        }
        glPopMatrix();

        glLineWidth(2);
        glPushMatrix();
        glTranslatef(-25,-75,0);
        glScalef(0.1,0.1,0.1);
        lengthOfString = (int)strlen(string[2]);
        for(i=0; i<lengthOfString; i++)
        {
                glColor3f(1,1,1);
                glutStrokeCharacter(GLUT_STROKE_ROMAN,string[2][i]);
        }
        glPopMatrix();

}

//draw text in page 31 (choose plane)
void draw_chPlane_text()
{
        char string[15][120];
        int i,lengthOfString;


        strcpy(string[0],"Choose Plane");
        strcpy(string[1],"Paper Plane!");
        strcpy(string[2],"Military X991+");
        strcpy(string[3],"Next");

        glLineWidth(2);
        glPushMatrix();
        glTranslatef(-120,55,0);
        glScalef(0.3,0.3,0.3);
        lengthOfString = (int)strlen(string[0]);
        for(i=0; i<lengthOfString; i++)
        {
                glColor3f(1,1,1);
                glutStrokeCharacter(GLUT_STROKE_ROMAN,string[0][i]);
        }
        glPopMatrix();

        glLineWidth(1);
        int y_pos_21=40;
        for(int k_t=1; k_t<=2; k_t++)
        {
                glPushMatrix();
                glTranslatef(50,y_pos_21-=30,0);
                glScalef(0.1,0.1,0.1);
                lengthOfString = (int)strlen(string[k_t]);
                for(i=0; i<lengthOfString; i++)
                {
                        glColor3f(1,1,1);

        glutStrokeCharacter(GLUT_STROKE_ROMAN,string[k_t][i]);
                }
                glPopMatrix();
        }
        glLineWidth(2);
        glPushMatrix();
        glTranslatef(-30,-75,0);
        glScalef(0.15,0.15,0.15);
        lengthOfString = (int)strlen(string[3]);
        for(i=0; i<lengthOfString; i++)
        {
                glColor3f(1,1,1);
                glutStrokeCharacter(GLUT_STROKE_ROMAN,string[3][i]);
        }
        glPopMatrix();

}

//print the score in page 3 (prints the game scores during game play)
void draw_score()
{
        int length;
        char score_text[15];
        strcpy(score_text,"Distance: ");
        glLineWidth(1);
        glPushMatrix();
        glTranslatef(85,82,0);
        glScalef(0.08,0.08,0.08);
        length = (int)strlen(score_text);
        for(i=0; i<length; i++)
        {
                glColor3f(1,1,1);
                glutStrokeCharacter(GLUT_STROKE_ROMAN,score_text[i]);
        }
        glPopMatrix();

        char dist_text_val[15];
        sprintf(dist_text_val,"%d",(int)dist);
        glPushMatrix();
        glTranslatef(130,82,0);
        glScalef(0.08,0.08,0.08);
        length = (int)strlen(dist_text_val);
        for(i=0; i<length; i++)
        {
                glColor3f(1,1,1);
                glutStrokeCharacter(GLUT_STROKE_ROMAN,dist_text_val[i]);
        }
        glPopMatrix();

        char missiles_text[15];
        strcpy(missiles_text,"Missiles: ");
        glPushMatrix();
        glTranslatef(85,72,0);
        glScalef(0.08,0.08,0.08);
        length = (int)strlen(missiles_text);
        for(i=0; i<length; i++)
        {
                glColor3f(1,1,1);
                glutStrokeCharacter(GLUT_STROKE_ROMAN,missiles_text[i]);
        }
        glPopMatrix();

        char mis_text_val[15];
        sprintf(mis_text_val,"%d",(int)missiles);
        glPushMatrix();
        glTranslatef(130,72,0);
        glScalef(0.08,0.08,0.08);
        length = (int)strlen(mis_text_val);
        for(i=0; i<length; i++)
        {
                glColor3f(1,1,1);
                glutStrokeCharacter(GLUT_STROKE_ROMAN,mis_text_val[i]);
        }
        glPopMatrix();

        char fuel_text[15];
        strcpy(fuel_text,"Fuel:  %");
        glPushMatrix();
        glTranslatef(-149,82,0);
        glScalef(0.08,0.08,0.08);
        length = (int)strlen(fuel_text);
        for(i=0; i<length; i++)
        {
                glColor3f(1,1,1);
                glutStrokeCharacter(GLUT_STROKE_ROMAN,fuel_text[i]);
        }
        glPopMatrix();

        char fuel_text_val[15];
        sprintf(fuel_text_val,"%d",(int)fuel);
        glPushMatrix();
        glTranslatef(-125,82,0);
        glScalef(0.08,0.08,0.08);
        length = (int)strlen(fuel_text_val);
        for(i=0; i<length; i++)
        {
                glColor3f(1,1,1);
                glutStrokeCharacter(GLUT_STROKE_ROMAN,fuel_text_val[i]);
        }
        glPopMatrix();
}

//select the scene
```

```
void select_scene()
{
        switch(ch_scene)
        {
        case 1:
                sprintf(scene,"res/scene1.jpg");
                break;
        case 2:
                sprintf(scene,"res/scene2.png");
                break;
        case 3:
                sprintf(scene,"res/scene3.png");
                break;
        case 4:
                sprintf(scene,"res/scene4.jpg");
                break;
        case 5:
                sprintf(scene,"res/scene5.jpg");
                break;
        }
}

void rocket1(int x_cor, int y_cor)
{
        glEnable(GL_TEXTURE_2D);
        glEnable(GL_BLEND);
        glColor4f(1.0f, 1.0f, 1.0f, 1.0f);
        if (i_mis1 == 0)
        {
                tex_2d_mis[1] = SOIL_load_OGL_texture
                                (
                                        "res/rocket2.png",
                                        SOIL_LOAD_AUTO,
                                        SOIL_CREATE_NEW_ID,
                                        SOIL_FLAG_MULTIPLY_ALPHA
                                );
                i_mis1 = 1;
        }
        glBindTexture(GL_TEXTURE_2D, tex_2d_mis[1]);
        glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
GL_MODULATE);
        glBlendFunc(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA);
        glBegin(GL_POLYGON);
        glTexCoord2f(0.0, 1.0);
        glVertex2f(x_cor, y_cor);
        glTexCoord2f(1.0, 1.0);
        glVertex2f(x_cor+SIZE_MIS_X, y_cor);
        glTexCoord2f(1.0, 0.0);
        glVertex2f(x_cor+SIZE_MIS_X, y_cor+SIZE_MIS_y-10);
        glTexCoord2f(0.0, 0.0);
        glVertex2f(x_cor, y_cor+SIZE_MIS_y-10);
        glEnd();
        glDisable(GL_TEXTURE_2D);
        glDisable(GL_BLEND);
}

void rocket2(int x_cor, int y_cor)
{
        glEnable(GL_TEXTURE_2D);
        glEnable(GL_BLEND);
        glColor4f(1.0f, 1.0f, 1.0f, 1.0f);
        if (i_mis2 == 0)
        {
                tex_2d_mis[2] = SOIL_load_OGL_texture
                                (
                                        "res/rocket3.png",
                                        SOIL_LOAD_AUTO,
                                        SOIL_CREATE_NEW_ID,
                                        SOIL_FLAG_MULTIPLY_ALPHA
                                );
                i_mis2 = 1;
        }
        glBindTexture(GL_TEXTURE_2D, tex_2d_mis[2]);
        glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
GL_MODULATE);
        glBlendFunc(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA);
        glBegin(GL_POLYGON);
        glTexCoord2f(0.0, 1.0);
        glVertex2f(x_cor, y_cor);
        glTexCoord2f(1.0, 1.0);
        glVertex2f(x_cor+SIZE_MIS_X, y_cor);
        glTexCoord2f(1.0, 0.0);
        glVertex2f(x_cor+SIZE_MIS_X, y_cor+SIZE_MIS_y-10);
        glTexCoord2f(0.0, 0.0);
        glVertex2f(x_cor, y_cor+SIZE_MIS_y-10);
        glEnd();
        glDisable(GL_TEXTURE_2D);
        glDisable(GL_BLEND);
}

void rocket3(int x_cor, int y_cor)
{
        glEnable(GL_TEXTURE_2D);
        glEnable(GL_BLEND);
        glColor4f(1.0f, 1.0f, 1.0f, 1.0f);
        if (i_mis3 == 0)
        {
                tex_2d_mis[3] = SOIL_load_OGL_texture
                                (
                                        "res/rocket4.png",
                                        SOIL_LOAD_AUTO,
                                        SOIL_CREATE_NEW_ID,
                                        SOIL_FLAG_MULTIPLY_ALPHA
                                );
                i_mis3 = 1;
        }
        glBindTexture(GL_TEXTURE_2D, tex_2d_mis[3]);
        glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
GL_MODULATE);
        glBlendFunc(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA);
        glBegin(GL_POLYGON);
        glTexCoord2f(0.0, 1.0);
        glVertex2f(x_cor, y_cor);
        glTexCoord2f(1.0, 1.0);
        glVertex2f(x_cor+SIZE_MIS_X, y_cor);
        glTexCoord2f(1.0, 0.0);
        glVertex2f(x_cor+SIZE_MIS_X, y_cor+SIZE_MIS_y-10);
        glTexCoord2f(0.0, 0.0);
        glVertex2f(x_cor, y_cor+SIZE_MIS_y-10);
        glEnd();
        glDisable(GL_TEXTURE_2D);
        glDisable(GL_BLEND);
}

void draw_rockets()
{
        if(missile_x>200)
        {
                no_of_missiles=rand()%MAX_MISSILES+1;
        }
        if(missile_x>=195 && missile_x<=200)
        {
                for(int k=1; k<=no_of_missiles; k++)
                        missile_y[k]=-101+rand()%165;
        }
        switch(no_of_missiles)
        {
        case 1:
                rocket1(missile_x,missile_y[1]);
                break;
        case 2:
                rocket1(missile_x,missile_y[1]);
                rocket2(missile_x,missile_y[2]);
                break;
        case 3:
                rocket1(missile_x,missile_y[1]);
                rocket2(missile_x,missile_y[2]);
                rocket3(missile_x,missile_y[3]);
                break;
        case 4:
                rocket1(missile_x,missile_y[1]);
                rocket3(missile_x,missile_y[2]);
                rocket2(missile_x,missile_y[3]);
                rocket3(missile_x,missile_y[4]);
                break;
        default:
        case 5:
                rocket1(missile_x,missile_y[1]);
                rocket3(missile_x,missile_y[2]);
                rocket2(missile_x,missile_y[3]);
                rocket3(missile_x,missile_y[4]);
                rocket1(missile_x,missile_y[5]);
                break;

        }
}

void draw_chosen_plane()
{
        glColor3f(1,1,1);
        glBegin(GL_LINE_LOOP);
        glVertex2f(-150,-40);
        glVertex2f(-30,-40);
        glVertex2f(-30,40);
        glVertex2f(-150,40);
        glEnd();

        glPushMatrix();
        glTranslatef(-90,0,0);
        glScalef(1.6,1.6,0);
        plane1.draw_plane();
        glPopMatrix();
}

void drawLogo()
{
        glEnable(GL_TEXTURE_2D);
        glEnable(GL_BLEND);
        glColor4f(1.0f, 1.0f, 1.0f, ((y_cre/100.0)));
        cout<<y_cre<<endl;
                tex_2d_plane = SOIL_load_OGL_texture
                            (
                                    "res/logo.png",
                                    SOIL_LOAD_AUTO,
                                    SOIL_CREATE_NEW_ID,
                                    SOIL_FLAG_MULTIPLY_ALPHA
                            );

        glBindTexture(GL_TEXTURE_2D, tex_2d_plane);
```

```
	glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
GL_MODULATE);
	glBlendFunc(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA);
	glBegin(GL_POLYGON);
	glTexCoord2f(0.0, 1.0);
	glVertex2f(-30, -100);
	glTexCoord2f(1.0, 1.0);
	glVertex2f(30, -100);
	glTexCoord2f(1.0, 0.0);
	glVertex2f(30, -35);
	glTexCoord2f(0.0, 0.0);
	glVertex2f(-30, -35);
	glEnd();
	glDisable(GL_TEXTURE_2D);
	glDisable(GL_BLEND);
}

void RenderScene()
{
	if(page==0)
	{
		glClear(GL_COLOR_BUFFER_BIT);

		//load .png image
		glEnable(GL_TEXTURE_2D);
		glColor4f(1.0f, 1.0f, 1.0f, 1.0f);
		tex_2d_0 = SOIL_load_OGL_texture
			(
				"res/loading.png",
				SOIL_LOAD_RGBA,
				SOIL_CREATE_NEW_ID,
				SOIL_FLAG_NTSC_SAFE_RGB
			);
		glBindTexture(GL_TEXTURE_2D, tex_2d_0);
		glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
GL_MODULATE);


		glBegin(GL_POLYGON);
		glTexCoord2f(0.0, 1.0);
		glVertex2f(-190.0f, -190.0f);
		glTexCoord2f(1.0, 1.0);
		glVertex2f(190.0f, -190.0f);
		glTexCoord2f(1.0, 0.0);
		glVertex2f(190.0f, 190.0f);
		glTexCoord2f(0.0, 0.0);
		glVertex2f(-190.0f, 190.0f);
		glEnd();
		glDisable(GL_TEXTURE_2D);
```

```
                //draw loading bar
                glColor3f(1,1,1);
                glBegin(GL_POLYGON);
                glVertex2f(-171.0f, -4.4f);
                glVertex2f(0.0f+x_step, -4.4f);
                glVertex2f(0.0f+x_step, 1.3f);
                glVertex2f(-171.0f, 1.3f);
                glEnd();

                glutSwapBuffers();
                glFlush();
        }
        if(page==1)
        {
                glClear(GL_COLOR_BUFFER_BIT);

                //load .png image
                glEnable(GL_TEXTURE_2D);
                glColor4f(1.0f, 1.0f, 1.0f, 1.0f);
                tex_2d_1 = SOIL_load_OGL_texture
                        (
                                "res/paper.jpg",
                                SOIL_LOAD_RGBA,
                                SOIL_CREATE_NEW_ID,
                                SOIL_FLAG_NTSC_SAFE_RGB
                        );
                glBindTexture(GL_TEXTURE_2D, tex_2d_1);
                glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
GL_MODULATE);


                glBegin(GL_POLYGON);
                glTexCoord2f(0.0, 1.0);
                glVertex2f(-178.0f, -100.0f);
                glTexCoord2f(1.0, 1.0);
                glVertex2f(178.0f, -100.0f);
                glTexCoord2f(1.0, 0.0);
                glVertex2f(178.0f, 100.0f);
                glTexCoord2f(0.0, 0.0);
                glVertex2f(-178.0f, 100.0f);
                glEnd();
                glDisable(GL_TEXTURE_2D);

                setFont(GLUT_BITMAP_TIMES_ROMAN_24);
                glColor3f(0.0,0.0,0.0);
                if(!full)
                {
                        drawString(-129.0,65.0,0.0,"PES Institute of Technology");
                        setFont(GLUT_BITMAP_HELVETICA_18);
                        glColor3f(0.0,0.0,0.0);
                        drawString(-90.0,55.0,0.0,"Made By:");
                        setFont(GLUT_BITMAP_HELVETICA_18);
                        glColor3f(0.0,0.0,0.0);
                        drawString(-110.0,45.0,0.0,"Arpith K - 1PE10CS018");
                        setFont(GLUT_BITMAP_HELVETICA_18);
                        glColor3f(0.0,0.0,0.0);
                        drawString(-105.0,35.0,0.0,"Anil S - 1PE10CS015");
                        setFont(GLUT_BITMAP_HELVETICA_18);
                        glColor3f(0.0,0.0,0.0);
                        drawString(130.0,-50.0,0.0,"Next!");
                }
                else
                {
                        drawString(-105.0,65.0,0.0,"PES Institute of Technology");
                        setFont(GLUT_BITMAP_HELVETICA_18);
                        glColor3f(0.0,0.0,0.0);
                        drawString(-80.0,55.0,0.0,"Made By:");
                        setFont(GLUT_BITMAP_HELVETICA_18);
                        glColor3f(0.0,0.0,0.0);
                        drawString(-95.0,45.0,0.0,"Arpith K - 1PE10CS018");
                        setFont(GLUT_BITMAP_HELVETICA_18);
                        glColor3f(0.0,0.0,0.0);
                        drawString(-92.0,35.0,0.0,"Anil S - 1PE10CS015");
                        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
                        glColor3f(0.0,0.0,0.0);
                        drawString(125.0,-50.0,0.0,"Click Here!");
                        setFont(GLUT_BITMAP_HELVETICA_18);
                        glColor3f(0.0,0.0,0.0);
                        drawString(135.0,-55.0,0.0,"Next!");
                }


                glutSwapBuffers();
        }
        if(page==2)
        {
                glClear(GL_COLOR_BUFFER_BIT);

                //load .jpg image
                glEnable(GL_TEXTURE_2D);
                glColor4f(1.0f, 1.0f, 1.0f, 1.0f);
                if(i_2==0)
                {
                        tex_2d_2 = SOIL_load_OGL_texture
                                (
                                        "res/plane_menu.jpg",
                                        SOIL_LOAD_RGBA,
                                        SOIL_CREATE_NEW_ID,
                                        SOIL_FLAG_NTSC_SAFE_RGB
                                );
```

```
				i_2=1;
			}
			glBindTexture(GL_TEXTURE_2D, tex_2d_2);
			glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
GL_MODULATE);


			glBegin(GL_POLYGON);
			glTexCoord2f(0.0, 1.0);
			glVertex2f(-178.0f, -100.0f);
			glTexCoord2f(1.0, 1.0);
			glVertex2f(178.0f, -100.0f);
			glTexCoord2f(1.0, 0.0);
			glVertex2f(178.0f, 100.0f);
			glTexCoord2f(0.0, 0.0);
			glVertex2f(-178.0f, 100.0f);
			glEnd();
			glDisable(GL_TEXTURE_2D);

			draw_menu_text();
			glutSwapBuffers();
		}
		if(page==21)
		{
			glClear(GL_COLOR_BUFFER_BIT);

			glEnable(GL_TEXTURE_2D);

			glColor4f(1.0f, 1.0f, 1.0f, 1.0f);

			if (i_inst21 == 0)
			{
				tex_2d_21 = SOIL_load_OGL_texture
					(
						"res/instructions2.png",
						SOIL_LOAD_RGBA,
						SOIL_CREATE_NEW_ID,
						SOIL_FLAG_NTSC_SAFE_RGB
					);
				i_inst21 = 1;
			}

			glBindTexture(GL_TEXTURE_2D, tex_2d_21);

			glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
GL_MODULATE);

			glBegin(GL_POLYGON);
			glTexCoord2f(0.0, 1.0);
			glVertex2f(-178.0f, -100.0f);
```

```
                glTexCoord2f(1.0, 1.0);
                glVertex2f(178.0f, -100.0f);
                glTexCoord2f(1.0, 0.0);
                glVertex2f(178.0f, 100.0f);
                glTexCoord2f(0.0, 0.0);
                glVertex2f(-178.0f, 100.0f);
                glEnd();
                glDisable(GL_TEXTURE_2D);

                draw_inst_text();
                glutSwapBuffers();
        }
        if(page==22)
        {
                glClear(GL_COLOR_BUFFER_BIT);

                //load .png image
                glEnable(GL_TEXTURE_2D);
                glColor4f(1.0f, 1.0f, 1.0f, 1.0f);
                if(i_cre22==0)
                {
                        tex_2d_22 = SOIL_load_OGL_texture
                                (
                                        "res/cre.jpg",
                                        SOIL_LOAD_RGBA,
                                        SOIL_CREATE_NEW_ID,
                                        SOIL_FLAG_NTSC_SAFE_RGB
                                );
                        i_cre22=1;
                }
                glBindTexture(GL_TEXTURE_2D, tex_2d_22);
                glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
GL_MODULATE);


                glBegin(GL_POLYGON);
                glTexCoord2f(0.0, 1.0);
                glVertex2f(-178.0f, -100.0f);
                glTexCoord2f(1.0, 1.0);
                glVertex2f(178.0f, -100.0f);
                glTexCoord2f(1.0, 0.0);
                glVertex2f(178.0f, 100.0f);
                glTexCoord2f(0.0, 0.0);
                glVertex2f(-178.0f, 100.0f);
                glEnd();
                glDisable(GL_TEXTURE_2D);
                drawLogo();
                draw_credit_text();
                glutSwapBuffers();
        }
        if(page==23)
        {
                glClear(GL_COLOR_BUFFER_BIT);

                //load .jpg image
                glEnable(GL_TEXTURE_2D);
                glColor4f(1.0f, 1.0f, 1.0f, 1.0f);
                if(i_23==0)
                {
                        tex_2d_23 = SOIL_load_OGL_texture
                                  (
                                          "res/scene3.png",
                                          SOIL_LOAD_RGBA,
                                          SOIL_CREATE_NEW_ID,
                                          SOIL_FLAG_NTSC_SAFE_RGB
                                  );
                        i_23=1;
                }
                glBindTexture(GL_TEXTURE_2D, tex_2d_23);
                glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
GL_MODULATE);


                glBegin(GL_POLYGON);
                glTexCoord2f(0.0, 1.0);
                glVertex2f(-178.0f, -100.0f);
                glTexCoord2f(1.0, 1.0);
                glVertex2f(178.0f, -100.0f);
                glTexCoord2f(1.0, 0.0);
                glVertex2f(178.0f, 100.0f);
                glTexCoord2f(0.0, 0.0);
                glVertex2f(-178.0f, 100.0f);
                glEnd();
                glDisable(GL_TEXTURE_2D);

                draw_high_text();
                glutSwapBuffers();
        }
        if(page==31)
        {
                glClear(GL_COLOR_BUFFER_BIT);


                glEnable(GL_TEXTURE_2D);

                glColor4f(1.0f, 1.0f, 1.0f, 1.0f);

                if (i_sel31 == 0)
                {
                        tex_2d_31 = SOIL_load_OGL_texture
```

```
                                        (
                                                "res/bck_plane.jpg",
                                                SOIL_LOAD_RGBA,
                                                SOIL_CREATE_NEW_ID,
                                                SOIL_FLAG_NTSC_SAFE_RGB
                                        );
                                i_sel31 = 1;
                        }

                        glBindTexture(GL_TEXTURE_2D, tex_2d_31);

                        glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
GL_MODULATE);

                        glBegin(GL_POLYGON);
                        glTexCoord2f(0.0, 1.0);
                        glVertex2f(-178.0f, -100.0f);
                        glTexCoord2f(1.0, 1.0);
                        glVertex2f(178.0f, -100.0f);
                        glTexCoord2f(1.0, 0.0);
                        glVertex2f(178.0f, 100.0f);
                        glTexCoord2f(0.0, 0.0);
                        glVertex2f(-178.0f, 100.0f);
                        glEnd();
                        glDisable(GL_TEXTURE_2D);

                        draw_chPlane_text();
                        draw_chosen_plane();

                        glutSwapBuffers();
                }
                if(page==32)
                {
                        glClear(GL_COLOR_BUFFER_BIT);

                        glEnable(GL_TEXTURE_2D);

                        glColor4f(1.0f, 1.0f, 1.0f, 1.0f);

                        select_scene();

                        if (i == 0)
                        {
                                tex_2d = SOIL_load_OGL_texture
                                        (
                                                scene,
                                                SOIL_LOAD_RGBA,
                                                SOIL_CREATE_NEW_ID,
                                                SOIL_FLAG_NTSC_SAFE_RGB
                                        );
```

```
				i = 1;
			}

			glBindTexture(GL_TEXTURE_2D, tex_2d);

			glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
GL_MODULATE);

			glBegin(GL_POLYGON);
			glTexCoord2f(0.0, 1.0);
			glVertex2f(-178.0f, -100.0f);
			glTexCoord2f(1.0, 1.0);
			glVertex2f(178.0f, -100.0f);
			glTexCoord2f(1.0, 0.0);
			glVertex2f(178.0f, 100.0f);
			glTexCoord2f(0.0, 0.0);
			glVertex2f(-178.0f, 100.0f);
			glEnd();
			glDisable(GL_TEXTURE_2D);

			draw_chScene_text();

			glutSwapBuffers();
		}
		if(page==3)
		{
			glClear(GL_COLOR_BUFFER_BIT);

			glEnable(GL_TEXTURE_2D);
			glColor4f(1.0f, 1.0f, 1.0f, 1.0f);
			if (i_bck == 0)
			{
				tex_2d = SOIL_load_OGL_texture
					(
						scene,
						SOIL_LOAD_RGBA,
						SOIL_CREATE_NEW_ID,
						SOIL_FLAG_NTSC_SAFE_RGB
					);
				i_bck = 1;
			}
			glBindTexture(GL_TEXTURE_2D, tex_2d);
			glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
GL_MODULATE);
			glBegin(GL_POLYGON);
			glTexCoord2f(0.0, 1.0);
			glVertex2f(-190.0f+x, -100.0f);
			glTexCoord2f(1.0, 1.0);
			glVertex2f(890.0f+x, -100.0f);
			glTexCoord2f(1.0, 0.0);
		glVertex2f(890.0f+x, 100.0f);
		glTexCoord2f(0.0, 0.0);
		glVertex2f(-190.0f+x, 100.0f);
		glEnd();
		glDisable(GL_TEXTURE_2D);

		//fuel indicator outline
		glColor3f(0,0,0);
		glLineWidth(3);
		glBegin(GL_LINE_LOOP);
		glVertex2f(-50-100,80);
		glVertex2f(50-100,80);
		glVertex2f(50-100,70);
		glVertex2f(-50-100,70);
		glEnd();

		//fuel indicator
		glColor3f(1,1,1);
		glBegin(GL_POLYGON);
		glVertex2f(-49-100,79);
		glVertex2f(-49+fuel-100,79);
		glVertex2f(-49+fuel-100,71);
		glVertex2f(-49-100,71);
		glEnd();

		//seprator--to seprate score and game screen
		glLineWidth(1);
		glColor3f(1,1,1);
		glBegin(GL_LINES);
		glVertex2f(-200,64);
		glVertex2f(200,64);
		glEnd();

		draw_score();

		draw_rockets();

		glPushMatrix();
		glTranslatef(-130,y_pos,0);
		glRotatef(theta,0,0,1);
		plane1.draw_plane();
		glPopMatrix();
		glutSwapBuffers();
	}
	if(page==4)
	{
		glClear(GL_COLOR_BUFFER_BIT);

		//load .png image
		glEnable(GL_TEXTURE_2D);
```

```
                glColor4f(1.0f, 1.0f, 1.0f, 1.0f);
                if(i_fin4==0)
                {
                        tex_2d_4 = SOIL_load_OGL_texture
                                  (
                                          "res/finish.png",
                                          SOIL_LOAD_RGBA,
                                          SOIL_CREATE_NEW_ID,
                                          SOIL_FLAG_NTSC_SAFE_RGB
                                  );
                        i_fin4=1;
                }
                glBindTexture(GL_TEXTURE_2D, tex_2d_4);
                glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
GL_MODULATE);


                glBegin(GL_POLYGON);
                glTexCoord2f(0.0, 1.0);
                glVertex2f(-178.0f, -100.0f);
                glTexCoord2f(1.0, 1.0);
                glVertex2f(178.0f, -100.0f);
                glTexCoord2f(1.0, 0.0);
                glVertex2f(178.0f, 100.0f);
                glTexCoord2f(0.0, 0.0);
                glVertex2f(-178.0f, 100.0f);
                glEnd();
                glDisable(GL_TEXTURE_2D);
                //drawText();
                draw_fin_text();
                glutSwapBuffers();
        }

}

void TimerFunction(int value)
{
        if(page==0)
        {
                if(x_step<171.0)
                        x_step+=6.0;
                else
                        page=1;
        }
        if(page==22)
        {
                if(y_cre<80)
                        y_cre+=0.5;
        }
        if(page==3)
```

```
	{
		if(fuel==98)
			x=0;
		x-=0.3;
		cout<<x<<endl;
		if(missile_x<-210)
			missile_x=250;
		missile_x-=1;
		if(state==UP)
		{
			if(fuel>0)
				fuel-=.1;
		}

		dist+=0.1;

		//update number of missiles douged
		{
			if(missile_x<-90)
			{
				if(update_mis==0)
				{
					missiles+=no_of_missiles;
					update_mis=1;
				}
			}
			else
				update_mis=0;
		}

		//plane position
		if(state==UP)
		{
			if(fuel>0)
			{
				if(y_pos<=90)
					y_pos++;
				if(theta<0)
					theta+=.3;
				else
					theta+=.1;
			}
			else
				y_pos--;
		}
		else
		{
			if(y_pos>=-100)
				y_pos--;
			else
```

```
                    {
                              y_pos=0;
                              page=4;
                    }
                    if(theta>0)
                              theta-=.3;
                    else
                              theta-=.1;

          }

          //check for collision
          if(missile_x<-110)
          {
                    cout<<"Possibility of crash"<<endl;
                    for(int m=1; m<=no_of_missiles; m++)
                    {
                              if(missile_y[m]>y_pos-3 && missile_y[m]<y_pos+3)
                              {
                                        cout<<"Crash"<<endl;
                                        hit_missile=1;
                                        for(int m1=1; m1<=no_of_missiles; m1++)
                                                  missile_x=200;
                                        y_pos=0;
                                        x=0;
                                        page=4;
                                        break;
                              }
                    }

          }
     }

     glutPostRedisplay();
     glutTimerFunc(frames,TimerFunction, 1);
}

void myinit(void)
{
     glClearColor(0.0f, 0.8f, 0.0f, 1.0f);
     glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
GL_MODULATE);
     glEnable(GL_TEXTURE_2D);
     glEnable(GL_BLEND);
}

void Resize(int w, int h)
{
     GLfloat aspectRatio;
```

```
	if(h == 0)
		h = 1;

	glViewport(0, 0, w, h);

	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();

	aspectRatio = (GLfloat)w / (GLfloat)h;
	if (w <= h)
	{
		windowWidth = 100;
		windowHeight = 100 / aspectRatio;
		glOrtho (-100.0, 100.0, -windowHeight, windowHeight, 1.0, -1.0);
	}
	else
	{
		windowWidth = 100 * aspectRatio;
		windowHeight = 100;
		glOrtho (-windowWidth, windowWidth, -100.0, 100.0, 1.0, -1.0);
	}

	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
}


int main(int argc, char* argv[])
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA);
	glutInitWindowSize(800,450);
	glutCreateWindow("Airplane!");
	//glutPositionWindow(320,150);
	glutFullScreen();
	myinit();
	glutKeyboardFunc(keyboard);
	glutSpecialFunc(SpecialKeys);
	glutMouseFunc(Mouse);
	glutDisplayFunc(RenderScene);
	glutTimerFunc(33, TimerFunction, 1);
	glutReshapeFunc(Resize);
	glutMainLoop();
	return 0;
}
```

# 7. SCREENSHOTS

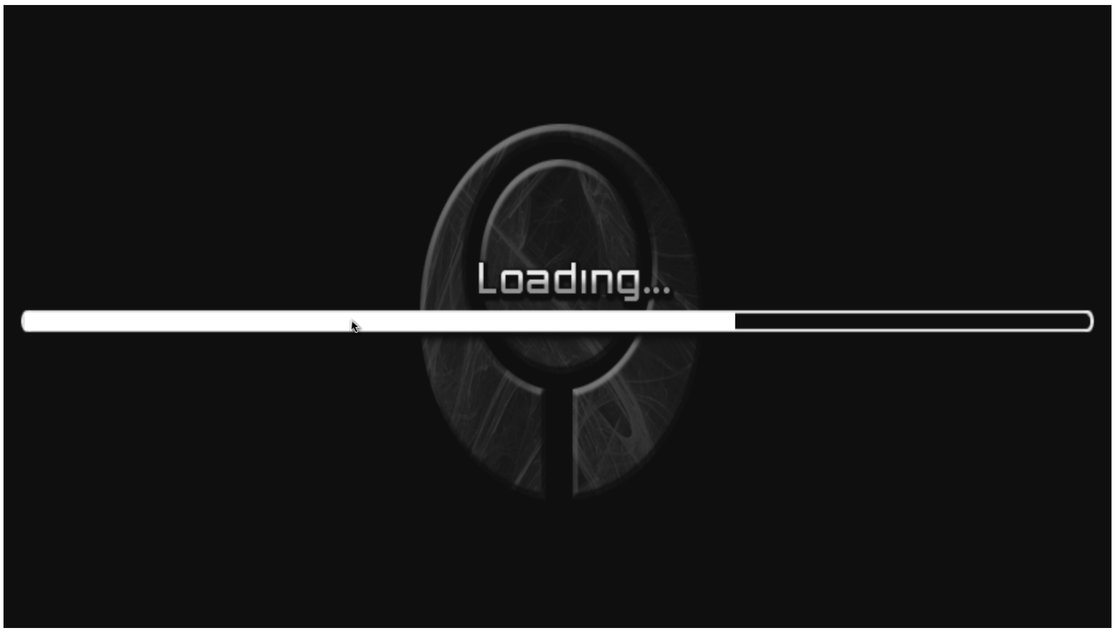


**Figure 7.1: Loading Page**

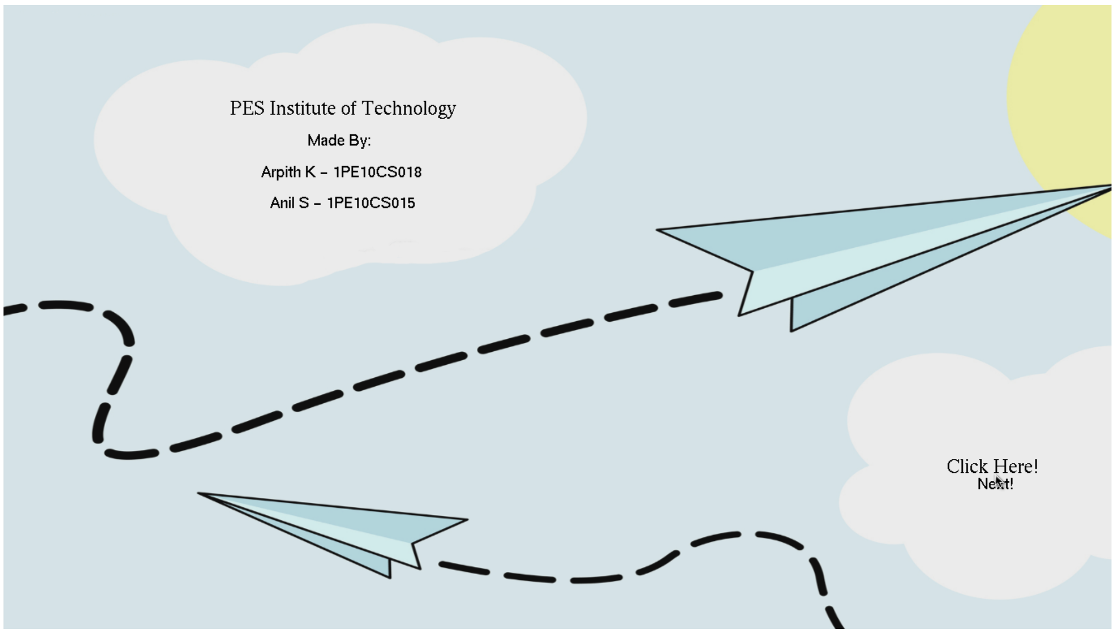


**Figure 7.2: Splash Screen**

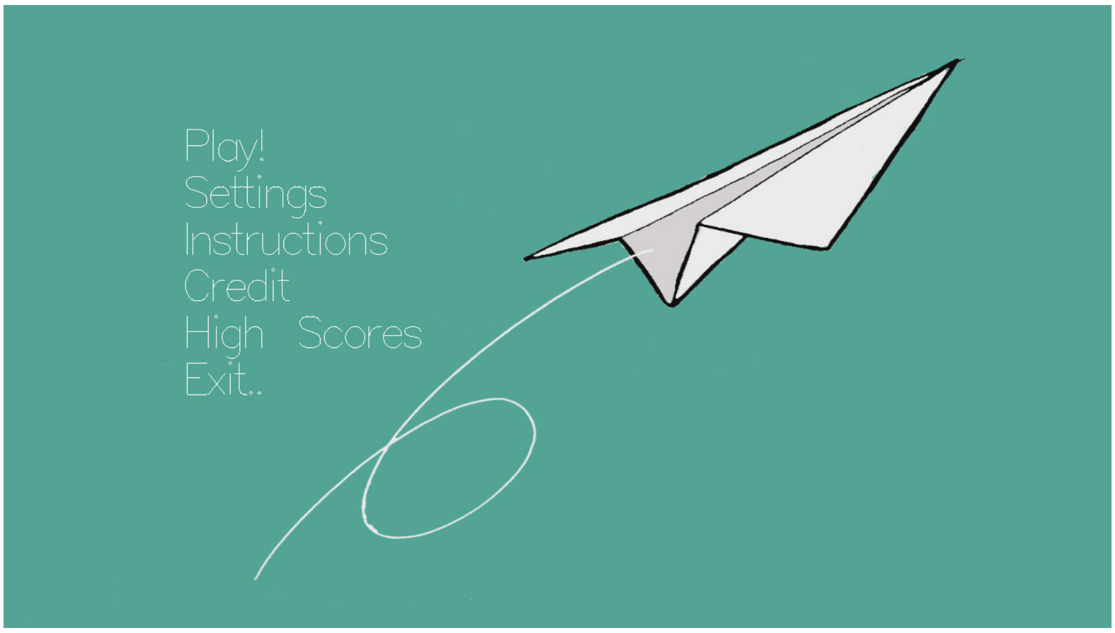


**Figure 7.3: Main Menu**

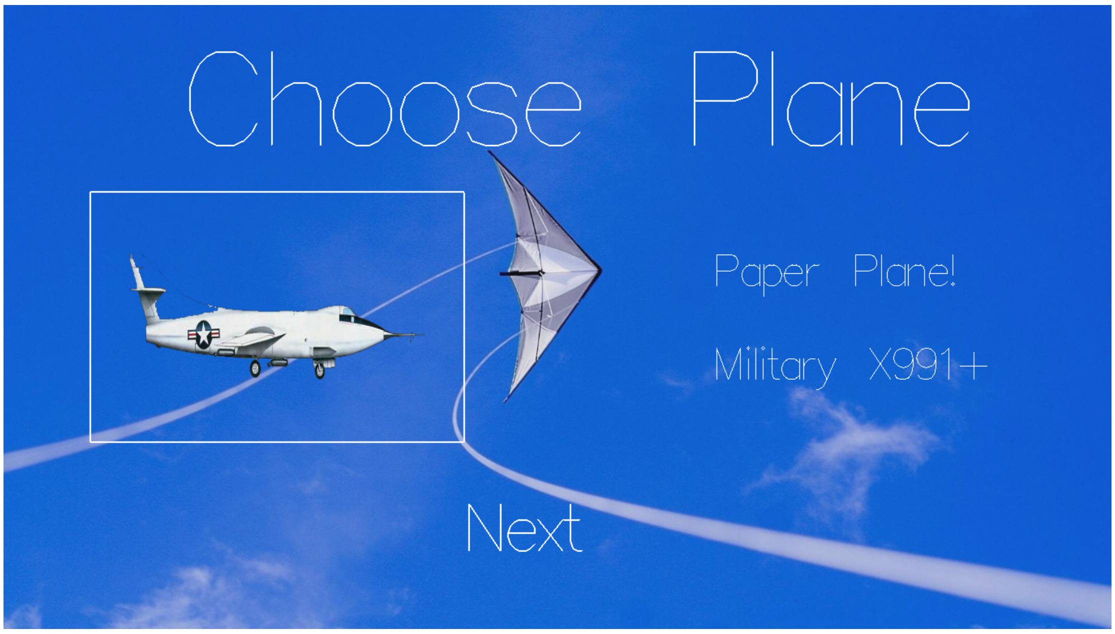


**Figure 7.4: Choose plane**

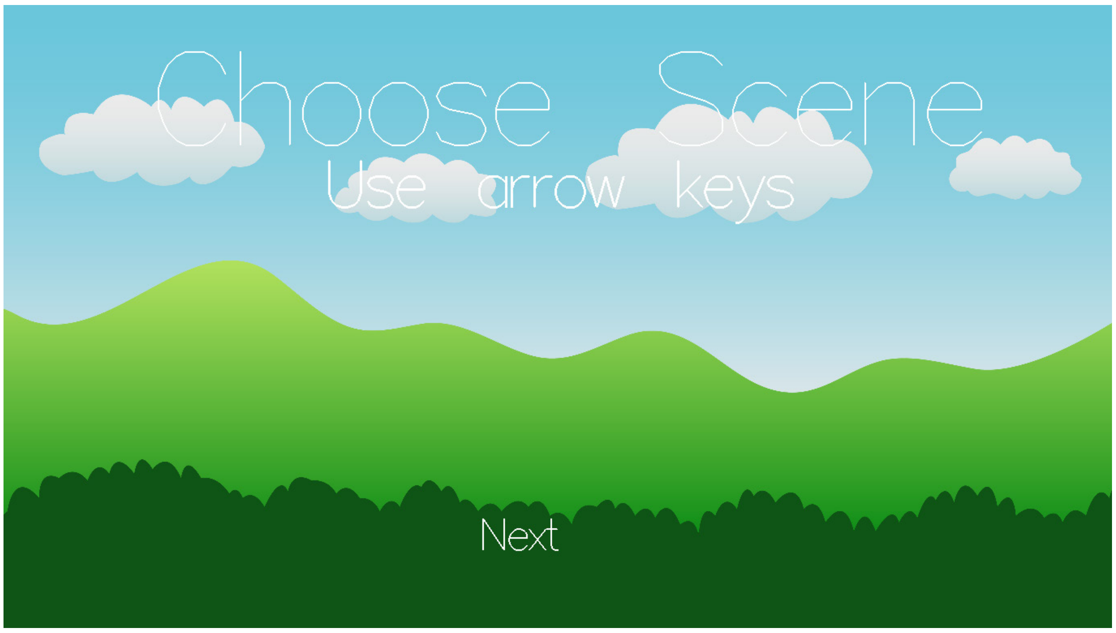


**Figure 7.5: Choose scenery**

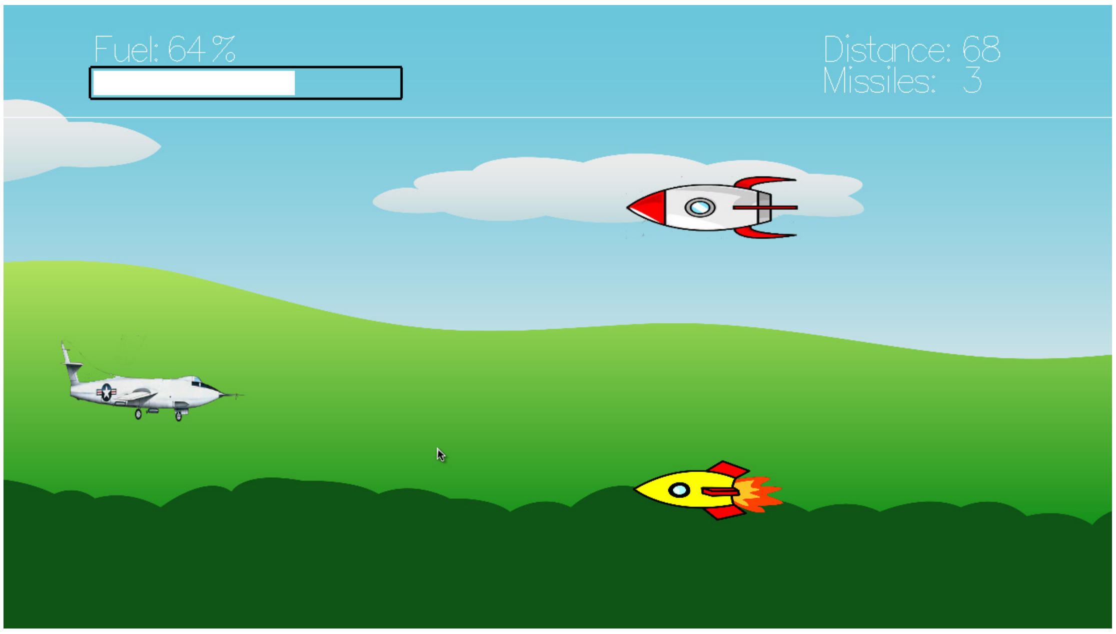


**Figure 7.6: Actual gameplay**

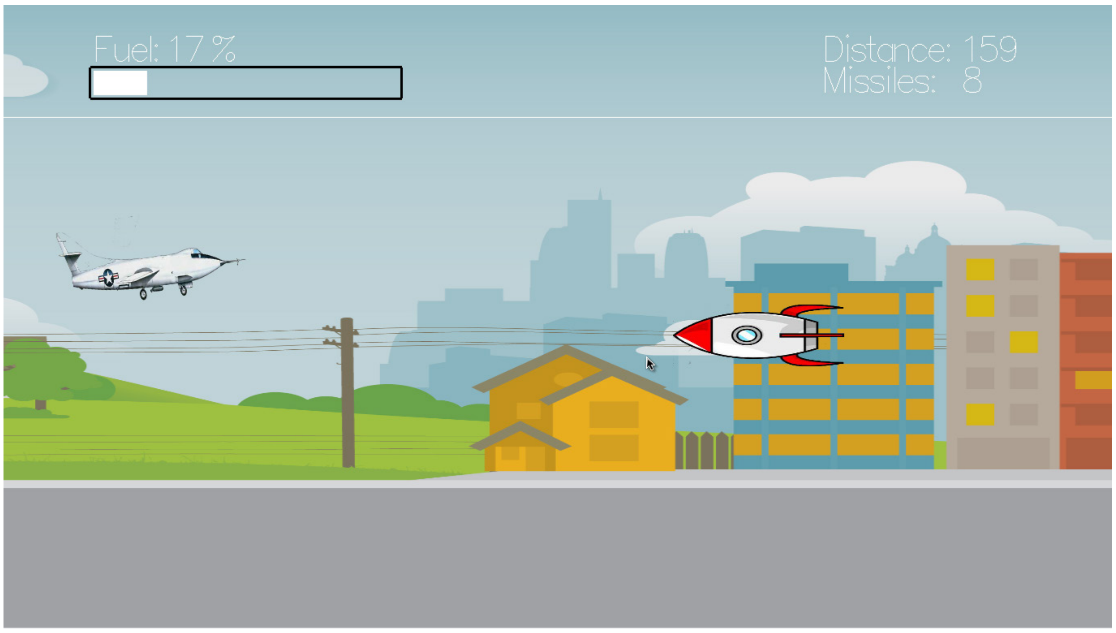


**Figure 7.7: Gameplay with a different scenery**

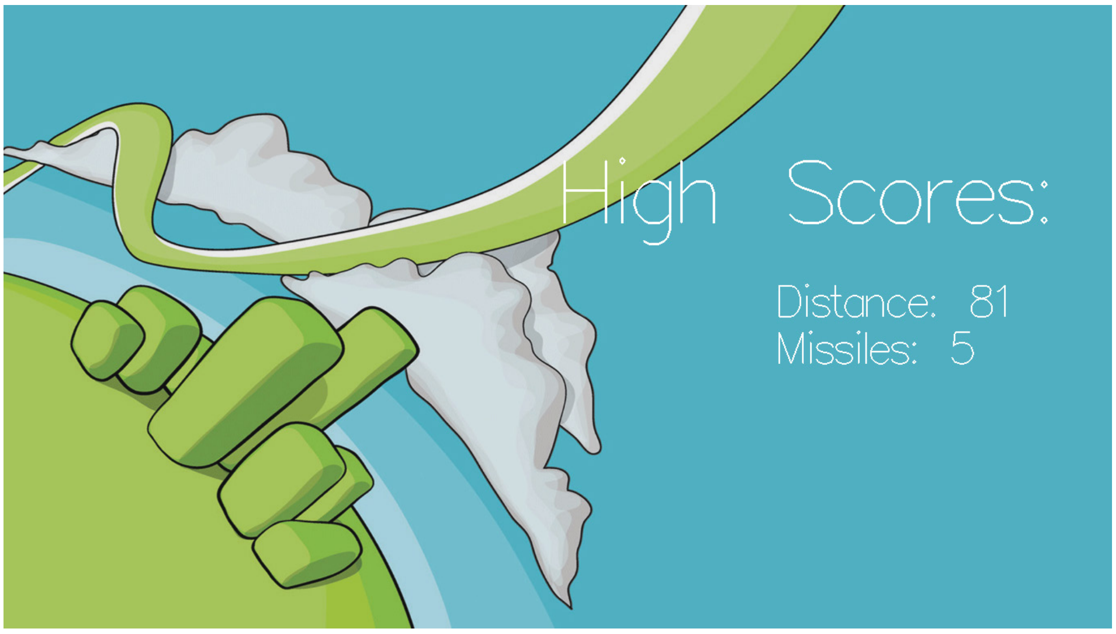
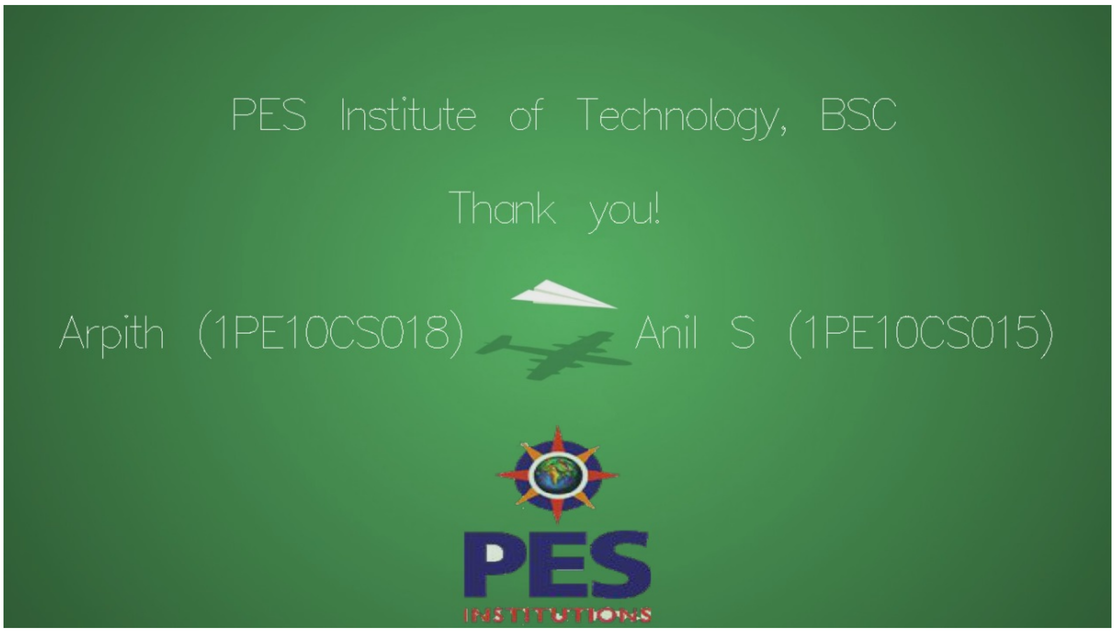


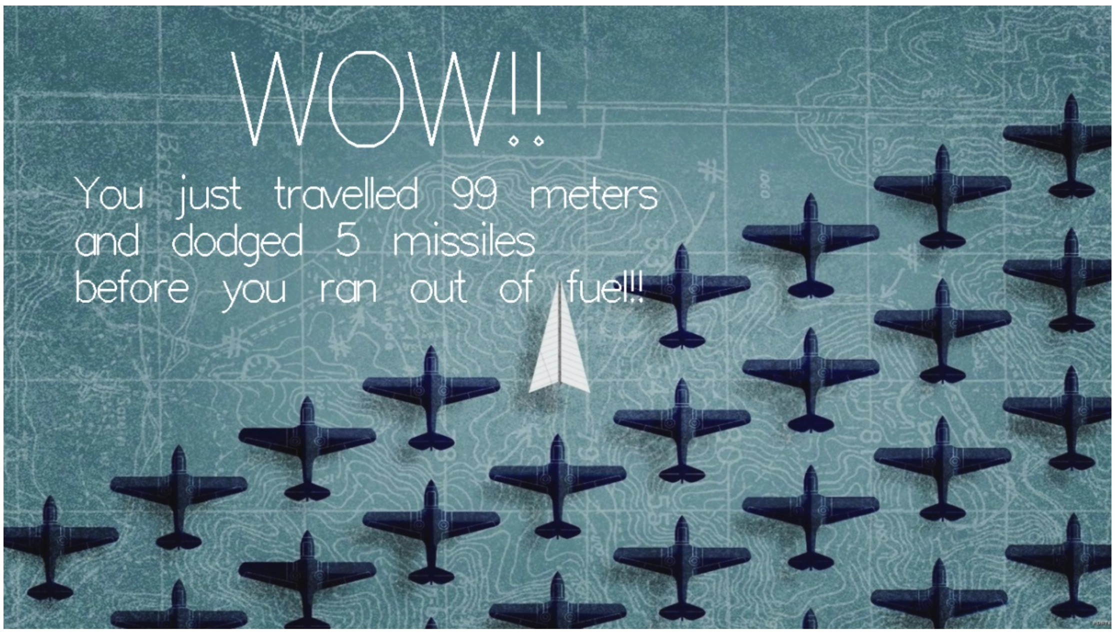
**Figure 7.8: High Scores**


PES Institute of Technology, BSC

Thank you!

Arpith (1PE10CS018) Anil S (1PE10CS015)

**Figure 7.9: Credits Page**


WOW!!

You just travelled 99 meters
and dodged 5 missiles
before you ran out of fuel!!

**Figure 7.10: Final page displaying the distance and number of missiles dodged.**

# 8. BIBLIOGRAPHY

**Books referred:**

- Edward Angel: Interactive Computer Graphics A Top-Down Approach with OpenGL, 5th Edition, Pearson Education, 2008.

**SOIL libraries:**

- Lonesock.net/soil.html

**Other Webpages:**

- freeglut.sourceforge.net/
- Opengl.org/documentation/
- Opengl.org/discussion_boards/
- Opengl-tutorial.org
- Nehe.gamedev.net/
- en.wikipedia.org/wiki/OpenGL