

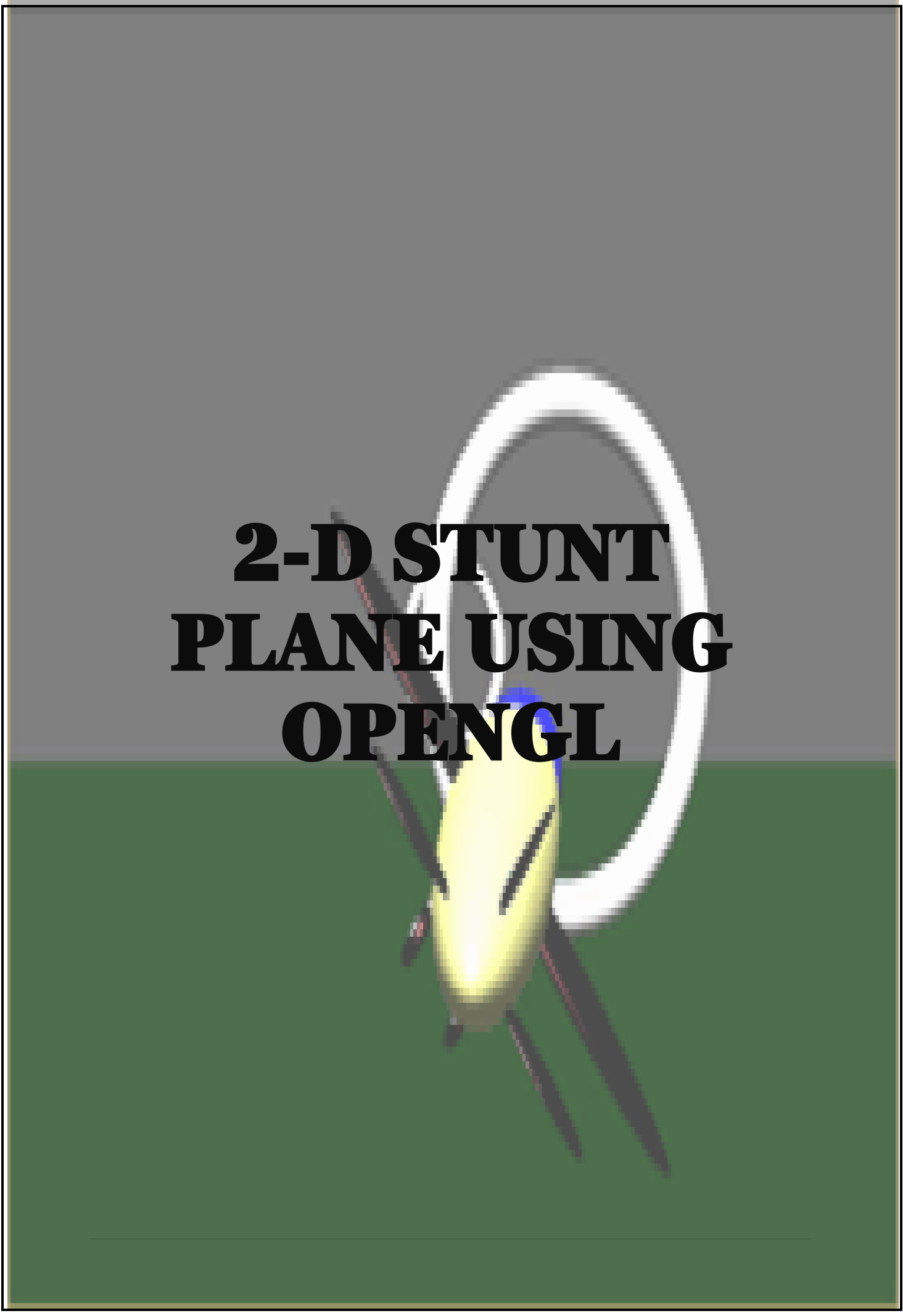
DELHI TECHNOLOGICAL UNIVERSITY
(DEPARTMENT OF APPLIED MATHEMATICS)



COMPUTER GRAPHICS (MC-318)
INNOVATIVE PROJECT

SUBMITTED TO:
Mrs. Trasha Gupta

SUBMITTED BY:
Shubham Kumar (2K18/MC/111)
Akanksha Dhyani (2K18/MC/009)

A 2D graphic of a yellow stunt plane with a blue cockpit and black propeller, performing a loop-the-loop. The plane is at the bottom of the loop, moving upwards. The background is split into a grey upper half and a green lower half, representing sky and ground. The text "2-D STUNT PLANE USING OPENGL" is centered over the loop.

2-D STUNT PLANE USING OPENGL

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 Trasha Gupta**, 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 gratefully acknowledge the help lent out by all the staff members of Mathematics and Computing department of Delhi Technological University 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.

ABSTRACT

The aim of this project is to create an interesting and creative game using tools like OpenGL and C++ (g++ compiler) and showcase the importance of Computer Graphics. 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

Sl. No.	CONTENTS	Page No.
	Acknowledgement	i
	Abstract	
1.	Introduction to	2
	OpenGL	2
	1.1 Computer graphics	2
	1.2 OpenGL	3
	1.3 GLUT	3
	1.4 Free GLUT	
2.	1.5 Simple OpenGL Image Library	4
	System Requirement Specification	4
	2.1 Hardware Requirement	
3.	2.2 Software	5
4.	Requirement Project	7
5.	Description	8
	System Design	10
6.	Implementation	12
	4.1 APIs Used	
7.	4.2 User Defined	48
8.	Functions Source code	53
	Sample output	
	Bibliography	

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 Graphics Library) 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 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.

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

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 is 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 will spend most of your time avoiding these. Keep a close eye at the fuel gauge! You will 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 will 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 Shubham and Akanksha!). You will then see a menu with screen with the following items:

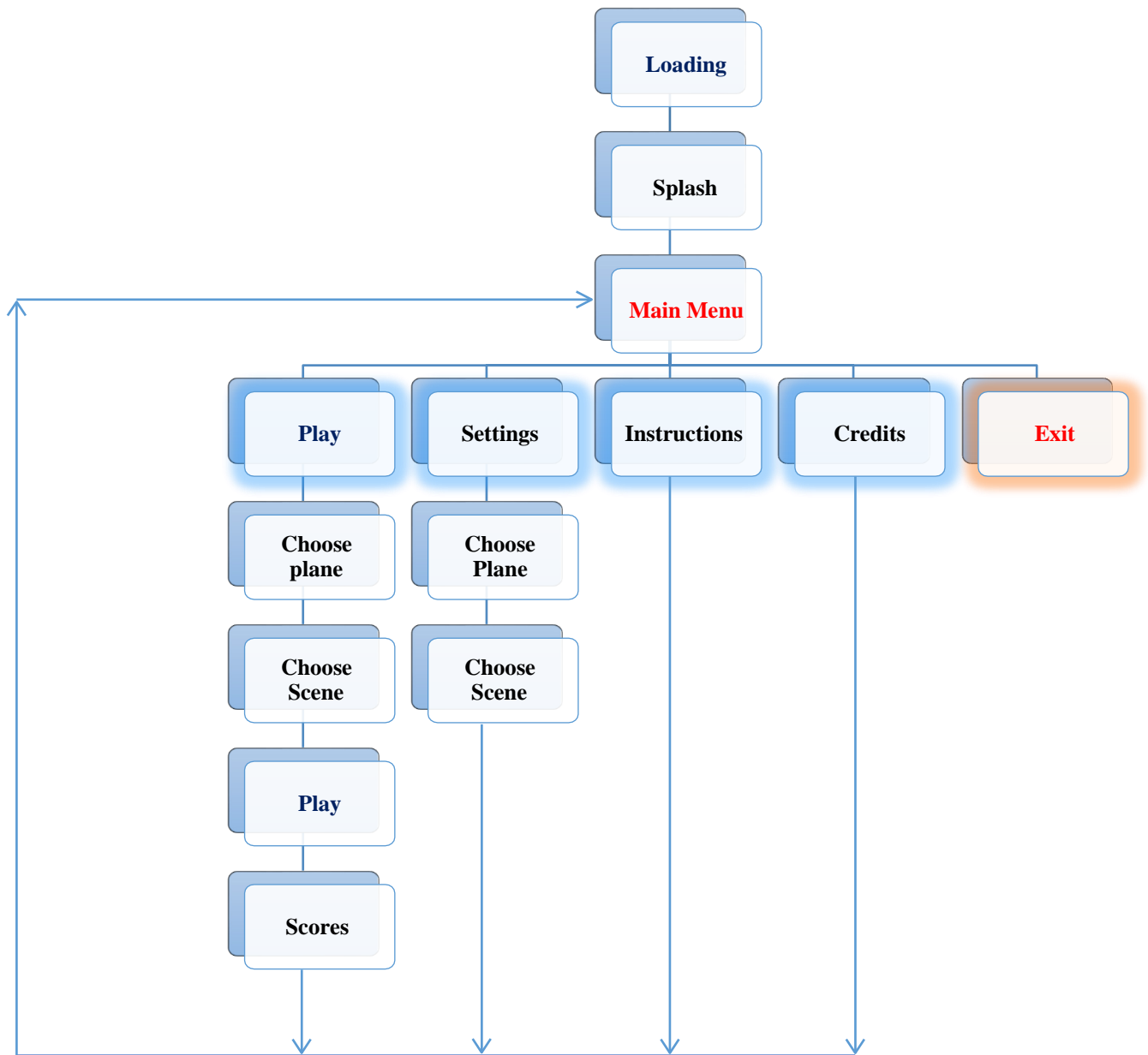
- Play
- Settings
- Instructions
- Credits
- High Scores
- Exit

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.

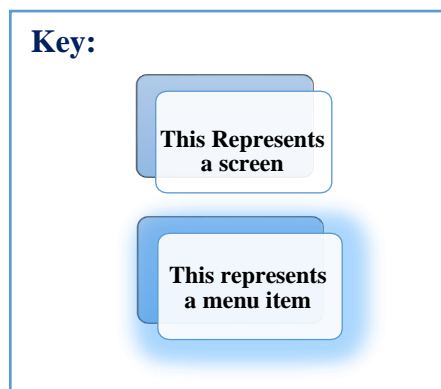
When the player hits on the Instruction button, another page which describes how the game works appears. Similarly, credits page will show you the name of the developers. Setting page will allow you to choose the plane and environment scene of your wish. Pressing escape at any point of time will bring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

SYSTEM DESIGN



Escape Key Pressed



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:

- **glutMainLoop(void):** It causes the program to begin an event-processing loop.
- **glutReshapeFunc(void (GLUTCALLBACK *func)(int width, int height)):** The reshape event is generated whenever the window is resized, such as by a user interaction.
- **glutSwapBuffers(void):** We can swap the front and back buffers at will from the application programs.
- **glutStrokeCharacter(void *font, int character):** Without using any display lists, glutStrokeCharacter renders the character in the named stroke font.
- **glPushMatrix():** Set current matrix on the stack
- **glPopMatrix():** Pop the old matrix without the transformations.
- **void glTranslatef(GLfloat x, GLfloat y, GLfloat z):** glTranslate produces a translation by (x, y, z).
- **void glLineWidth(GLfloat width):** Specifies the rasterized width of both aliased and antialiased lines.
- **void glBindTexture(GLenum target, GLuint texture):** Lets you create or use a named texture.
- **void glTexEnvf(GLenum target, GLenum pname, GLfloat param):** Specifies a texture environment.
- **void glEnable(GLenum cap):** Enable server-side GL capabilities
- **void glDisable(GLenum cap):** Disable server-side GL capabilities
- **SetFont(string family [], string style [], float size):** Sets the font used to print character strings.
- **void glutTimerFunc(unsigned int msec,void (*func)(int value), value):** Registers the timer callback func to be triggered in at least msec milliseconds.
- **void glutPostRedisplay(void):** Mark the normal plane of current window as needing to be redisplayed.
- **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.
- **void glutSpecialFunc(void (*func)(int key, int x, int y)):** Sets the special keyboard callback for the current window.
- **void glutMainLoop(void):** Enters the GLUT event processing loop. This routine should be called at most once in a GLUT program.
- **glColor3f (GLfloat red, GLfloat green, GLfloat blue):** It sets color to current drawing.
- **glLoadIdentity (void):** To initialize the current transform matrix to the identity transform.
- **glMatrixMode (GLenum mode):** It switches matrix mode between the two

matrices –

- ✓ **MODEL_VIEW (GL_MODELVIEW)**
- ✓ **PROJECTION (GL_PROJECTION)**

- **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.
- **glVertex3f (GLfloat x, GLfloat y, GLfloat z):** It is used to represent vertex.
- **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.
- **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*.
- **glutCreateWindow(const char *title):** It creates and opens OpenGL window with the title passed as the argument.
- **glutDisplayFunc(void (GLUTCALLBACK *func)(void)):** It sends graphics to screen.
- **glutIdleFunc(void (GLUTCALLBACK *func)(void)):** It is used to increase theta by fixed amount whenever nothing else is happening.
- **glutInit(int *argcp, char **argv):** It initiates interaction between windowing system and OpenGL.
- **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.
- **glutInitWindowSize(int width, int height):** It sets the size of created window.
- **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

- **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)
- **void draw_fin_text():** Displays the score and number of missiles dodged, in the finish screen.
- **void draw_credit_text():** Displays credits in the credit screen (when page=22)
- **void draw_high_text():** Displays high scores and maximum missiles dodged by the player.
- **void draw_menu_text():** Draws menu items on the screen

- **void draw_inst_text():** Displays instructions onto the screen
- **void draw_chScene_text():** Allows you to select the scene (background)
- **void draw_chPlane_text():** Allows you to choose the plane of your choice.
- **void draw_score():** Displays scores during game play
- **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
- **void rocket1(int x_cor, int y_cor):** Draws the first kind of rocket onto the screen at position determined by (x,y) coordinates.
- **void rocket2(int x_cor, int y_cor):** Draws the second kind of rocket onto the screen at position determined by (x,y) coordinates.
- **void rocket3(int x_cor, int y_cor):** Draws the third kind of rocket onto the screen at position determined by (x,y) coordinates.
- **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 $1 \leq X \leq 3$)
- **void draw_chosen_plane():** Display the chosen plane on the screen where you have to select a plane of your choice.
- **void drawLogo():** Displays the college logo in the credit page.
- **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.

SCREENSHOTS



Figure 1: Loading Page

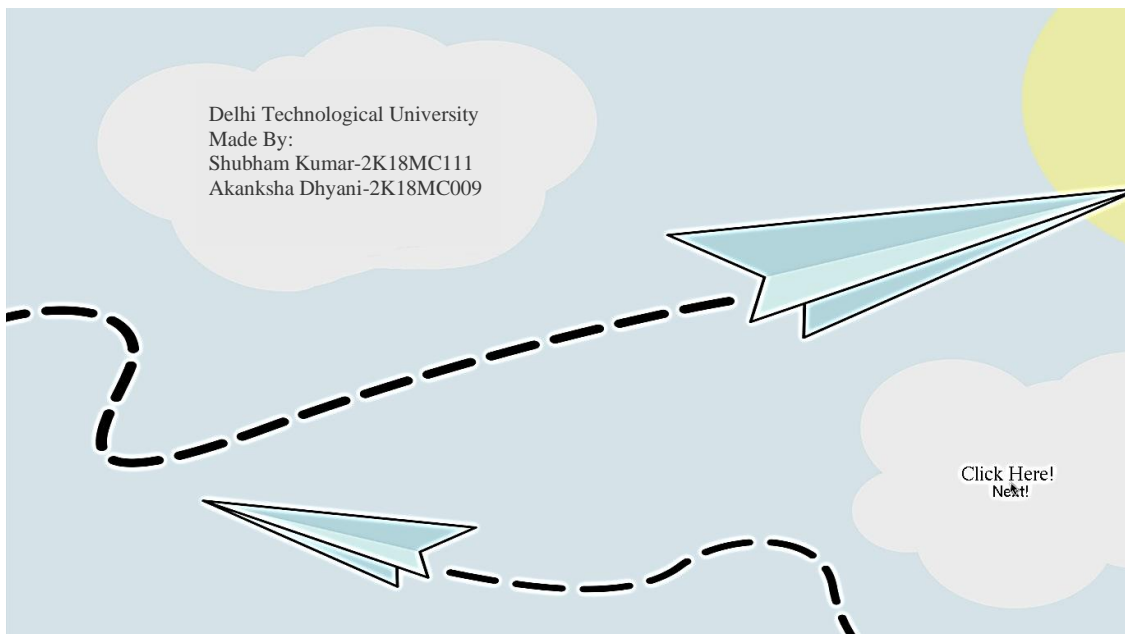


Figure 2: Splash Screen



Figure 3: Main Menu

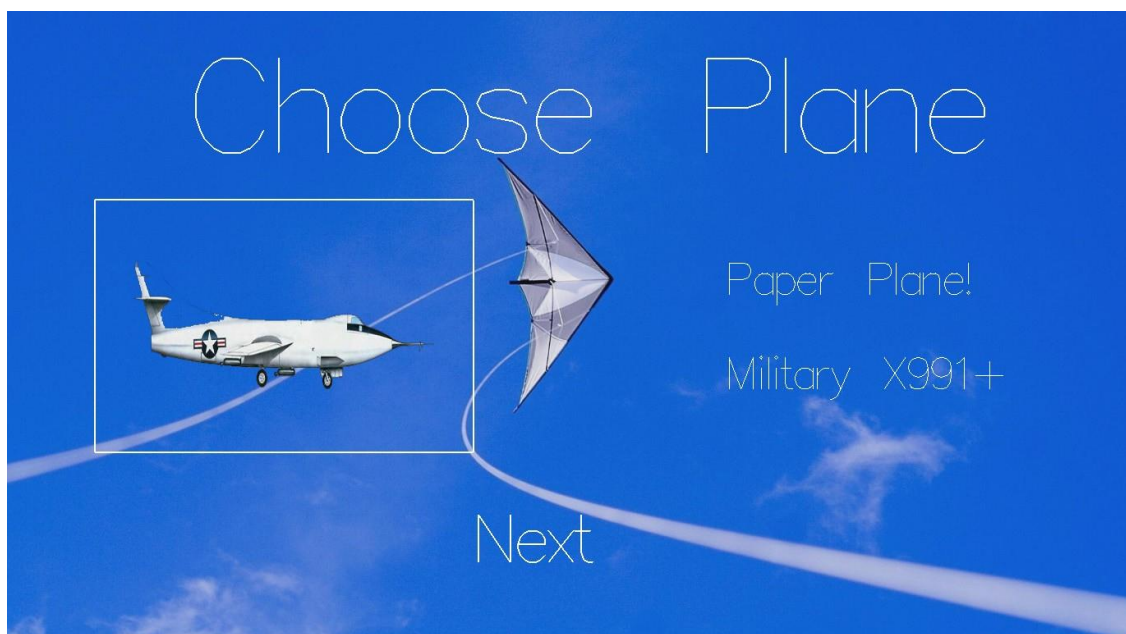


Figure 4: Choose plane



Figure 5: Choose scenery

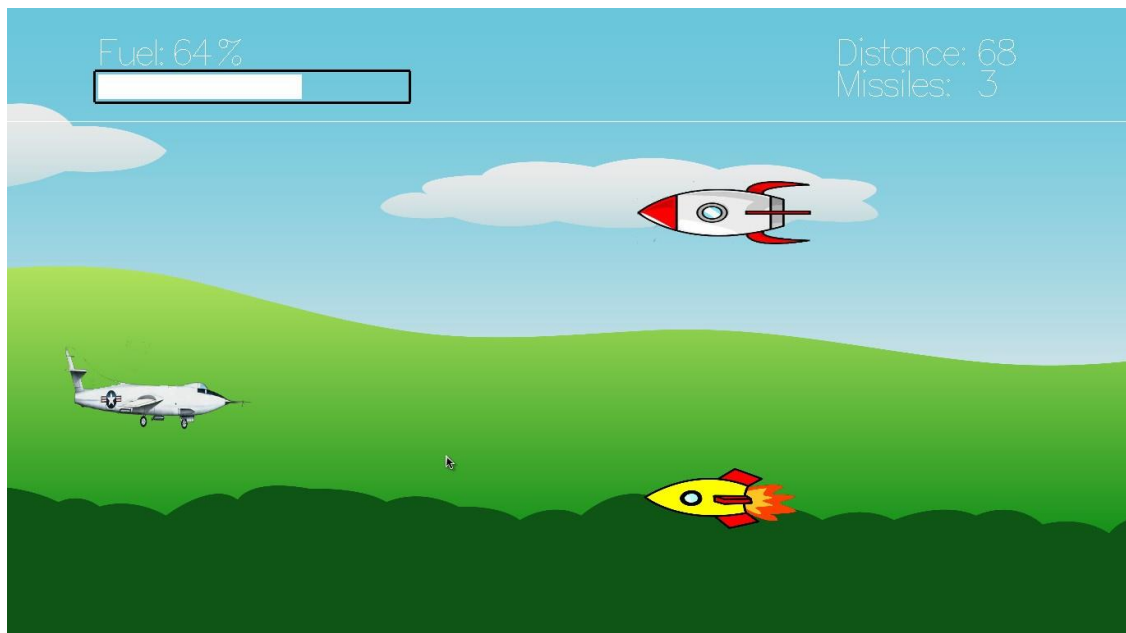


Figure 6: Actual gameplay

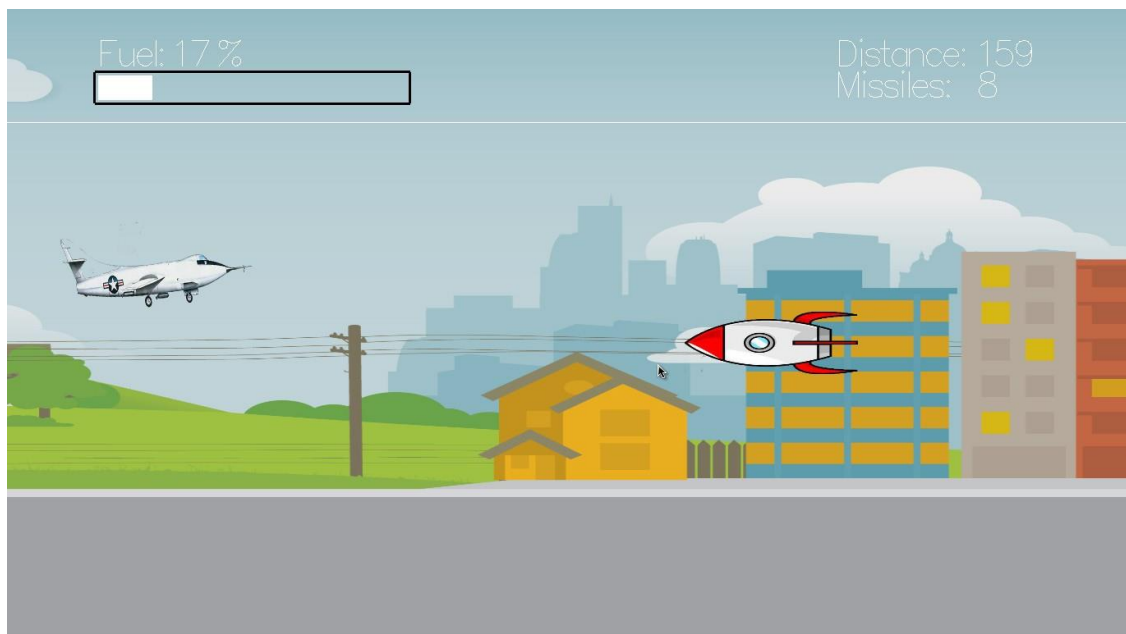


Figure 7: Gameplay with a different scenery

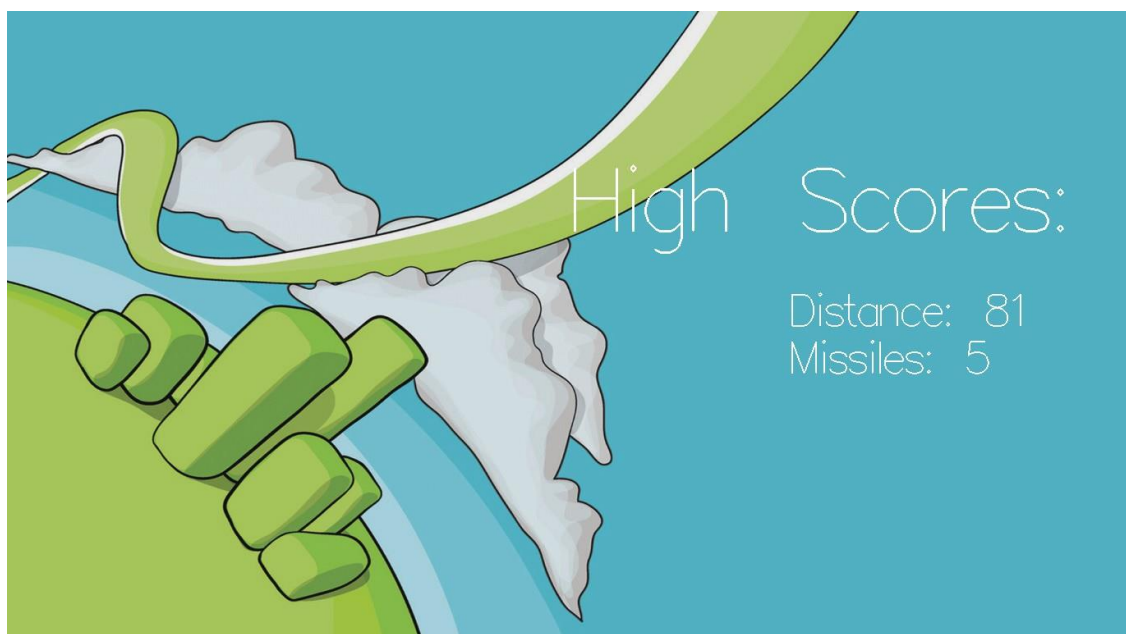


Figure 8: High Scores

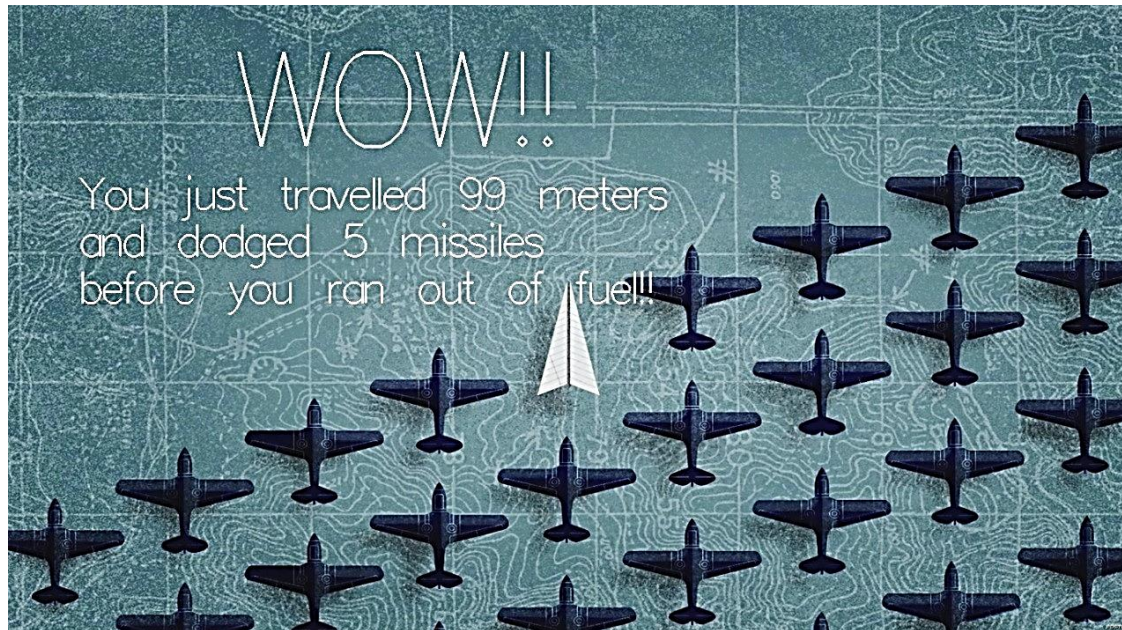


Figure 9: Final page displaying the distance and number of missiles dodged.

CONCLUSION

Stunt Plane is designed and implemented using a graphics software system called OpenGL which has become a widely accepted standard for developing graphic applications. Using OpenGL functions, users can create geometrical objects and can use translation, rotation, scaling with respect to the coordinate system. The development of this project has enabled us to improve accuracy, problem solving skills while providing a fun and interactive experience to the player.

FILE LINK

- <https://github.com/Shubhamkr2211/Stunt-Plane>

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