**Chapter 1**

**INTRODUCTION**

This report contains implementation of ‘**Air Attack Simulation** ‘using a set of OpenGL functions.

This project generates various displays wherein the fighter jet takes off from a runway, flies over a city and lands back at the runway. We are mainly using keyboard and mouse as interface to view the different types of display. The objects are drawn by using GLUT functions. The city background consists of few houses and buildings. The target consists of a building or a moving tank or an enemy fighter jet. On choosing a target, the jet fires a missile and a blast occurs destroying the object. The various options supported in our project are:

* The user can set the target to a building, a battle tank or an enemy fighter jet .
* The user can change the colour of the fighter jet.
* The user can select the time of day to be morning, or night.
* The user can increase the speed ‘X’.
* The user can redisplay the project by pressing ‘R’.
* Full screen display option is enabled by pressing ‘F’.

This project has been developed using **Mandriva Linux 2009/2010** with OpenGL package.

**1.1 Computer Graphics**

Graphics provides one of the most natural means of communicating within a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and effectively. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television. It has the added advantage that, with the computer, we can make pictures not only of concrete real world objects but also of abstract, synthetic objects, such as mathematical surfaces and of data that have no inherent geometry, such as survey results.

Computer graphics started with the display of data on hardcopy plotters and cathode ray tube screens soon after the introduction of computers themselves. It has grown to include the creation, storage, and manipulation of models and images of objects.

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

**1.2 OpenGL Interface**

OpenGL is an application program interface (API) offering various functions to implement primitives, models and images. This offers functions to create and manipulate render lighting, coloring, viewing the models. OpenGL offers different coordinate system and frames. OpenGL offers translation, rotation and scaling of objects.

Most of our applications will be designed to access OpenGL directly through functions in three libraries. They are:

1. Main GL: Library has names that begin with the letter gl and are stored in a library usually referred to as GL.
2. OpenGL Utility Library (GLU): This library uses only GL functions but contains code for creating common objects and simplifying viewing.
3. OpenGL Utility Toolkit (GLUT): This provides the minimum functionality that should be accepted in any modern windowing system.

**OpenGL Overview**

* OpenGL (Open Graphics Library) is the interface between a graphic program and graphics hardware. *It is streamlined*. In other words, it provides low-level functionality. For example, all objects are built from points, lines and convex polygons. Higher level objects like cubes are implemented as six four-sided polygons.
* OpenGL supports features like 3-dimensions, lighting, anti-aliasing, shadows, textures, depth effects, etc.
* *It is system-independent*. It does not assume anything about hardware or operating system and is only concerned with efficiently rendering mathematically described scenes. As a result, it does not provide any windowing capabilities.
* *It is a state machine*. At any moment during the execution of a program there is a current model transformation
* *It is a rendering pipeline*. The rendering pipeline consists of the following steps:
  + Defines objects mathematically.
  + Arranges objects in space relative to a viewpoint.
  + Calculates the colour of the objects.
  + Rasterizes the objects.

**1.2.1 OpenGL Pipeline**

The word pipeline is used to describe a process that can take two or more distinct stages or steps.The Figure 1.1 shows a simplified version of the OpenGL pipeline:

OpenGL

API Calls

Vertex data is usually transformed and lit initially, transform and lighting is a mathematically intensive stage where points used to describe an object’s geometry are recalculated for the given object’s location and orientation. Lighting calculations are performed as well to indicate how bright the colours should be at each vertex. When this stage is complete, the data is fed to the rasterization portion of the pipeline. The rasterizer actually creates the colour image from the geometric, color, and texture data. The image is then placed in the frame buffer. The frame buffer is the memory of the graphics display device, which means the image is displayed on your screen.

**1.2.2. Typical Function format**

Most OpenGL functions follow a naming convention that tells us which library the function is from and often how many and what types of arguments the function takes. All functions have a root that represents the function‟s corresponding OpenGL command. For example, glColor3f has the root Color. The gl prefix represents the gl library, and the 3f suffix means the function takes three floating-point arguments. All OpenGL functions take the following format: <Library prefix><Root command><Optional argument count><Optional argument type>

The Figure 1.2 illustrates the parts of an OpenGL function

**glColor3f(…)**

**1.2.3 Glut,the OpenGL Utility ToolKit**

The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs, which primarily perform system-level I/O with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input. Routines for drawing a number of geometric primitives (both in solid and wireframe mode) are also provided, including cubes, spheres, and the Utah teapot. GLUT even has some limited support for creating pop-up menus. contains rendering commands but is designed to be independent of any window system or operating system.

Consequently, it contains no commands for opening windows or reading events from the keyboard or mouse. Unfortunately, it’s impossible to write a complete graphics program without at least opening a window and most interesting programs require a bit of user input or other services from the operating system or window system. The two aims of GLUT are to allow the creation of rather portable code between operating systems (GLUT is cross platform) and to make learning OpenGL easier.

**1.3 ABOUT THE PROJECT**

The project demonstrates an air to surface attack by a fighter jet. The keys on the keyboard are used for various purposes such as to increase the speed, for redisplaying and for full screen display .The mouse is used to provide a user interactive interface in the form of buttons for selection of the target, selection of day, night and to change the colour of the fighter jet.

**User Interface:**

A set of keys is used to change the following:

* The key ‘x’ is used to increase the speed of the fighter jet, missile, tank and enemy fighter jet.
* The key ‘f’ is used for the full screen display.
* The key ‘r’ is used for redisplaying the project.

We have also provided an option for selecting the options by clicking on the right and left mouse button.

* On clicking the left mouse button, it provides various instructions.
* On clicking the right mouse button, it provides various options such as to select the target, to change the colour of the fighter jet, to select day, night.

**Objective**:

* The aim of the project is to demonstrate that a moving car or a truck passing through various backgrounds supports user interactions.
* As linux doesn’t provide graphics editor, it should be designed in such a way that it provides a very useful graph implementation interface.
* It should be easy to understand, user interactive interface.
* Creation of primitives, i.e polygons
* Providing human interaction through Mouse and keyboard.

**Chapter 2**

**SYSTEM SPECIFICATIONS**

**2.1 Software Requirements**

1. Operating System : LINUX

2. Compiler Used : C/C++

3. OpenGL API and Its Library Functions

**2.2 Hardware Requirements**

1. Processor Speed: 800 MHz

2. RAM Size: 128 MB DDR

3. Keyboard: Standard QWERTY serial

4. Mouse: Standard serial mouse

5. CD-ROM: Speed 48x and above

6. Cache memory: 256 KB

7. Video adapter and monitor with super VGA (800 x 600)or higher resolution.

**Chapter 3**

**DESIGN**

**Chapter 4**

**IMPLEMENTATION**

In this project we are implementing the motion of car/truck with changing backgrounds. We demonstrate the above with the use of open GL functions. We have created the fighter jet using the basic primitives like polygons, lines, etc.

**4.1 MODULES :**

* **FIGHTER JET:**

The fighter jet is generated using glBegin(GL\_POLYGON) and glBegin(GL\_LINES) functions present in openGL. The movement of fighter jet in the background along x and y axis is accomplished by the glTranslated( ) function and glutTimerFunc( ).

The project consists of 4 screens:

* The 1st screen has a runway for the jet to takeoff.
* The 2nd screen consists of a village with a few houses.
* The 3rd screen consists of a city with a few buildings and houses.
* If the user selects a target, then the 3rd screen is replaced with the target screen.
* If the target is a building then a missile is fired at the building and a blast occurs.
* If the target is a battle tank then a missile is fired from the jet at the moving tank and a blast occurs when the missile hits it.
* If the target is an enemy fighter jet then a missile is fired at the jet which crashes and bursts into flames.
* The 4th screen consists of an airbase with a runway where the fighter jet would land on completing these actions.
* Modules in **1st DISPLAY-AIRBASE** are :
* **ROAD** :

This is to generate the runway with separation white strips in between. It is generated using glBegin(GL\_POLYGON); and glColor3f( ) openGL functions. It is appropriately places and continuously translated backwards using glTranslatef() function to simulate the forward movement of the fighter jet.

* Module in **2st DISPLAY-VILLAGE** is :
* **BROWN BACKGROUND, HOUSE** :

This is to generate the brown landscape, and generate the houses using glBegin(GL\_POLYGON) and coloured using glcolor3f( ).These houses are scaled using glScalef( ) and placed at specific positions using using glTranslated().

* Modules in **3rd DISPLAY-CITY** are :
* **SKYSCRAPER,HOME,BUILDING** :

These modules are used for the construction of different types of structures using glBegin(GL\_POLYGON) and glBegin(GL\_LINE\_LOOP) openGL functions. These created structures are scaled and placed appropriately using glScalef( ) and glTranslatef( ) respectively.

* **ROAD**:

Here a road along with dividing whitestrips is generated using glBegin(GL\_POLYGON) and glColor3f( ) openGL functions. It is appropriately placed using glTranslatef( ) function.

* **PAVEMENT:**

Here a pavement which also doubles up as a footpath is used to separate the road from buildings.It is generated using glBegin(GL\_POLYGON) and glColor3f( ) openGL functions.

* **BLAST\_BUILDING**:

Here a blast polygon is generated and a suitable color is added using glBegin(GL\_POLYGON) and glColor3f( ) openGL functions.

* Modules in **3rd DISPLAY-TANK** background are:
* **TANK**:

Here a battle tank is generated using glBegin(GL\_POLYGON) and glColor3f( ) openGL functions. The wheels for the tank are drawn by using circle\_draw(). It is appropriately placed and continuously translated in the opposite direction of the plane using glTranslatef( ) function.

* **BLAST\_TANK**:

Here a blast polygon is generated using glBegin(GL\_POLYGON) and color is added using glColor3f( ) openGL functions.

* Modules in **3th DISPLAY-PLANE** are:
* **ENEMY\_JET**:

The fighter jet is generated using glBegin(GL\_POLYGON) and glBegin(GL\_LINES) functions present in openGL. The movement of fighter jet in the background along x axis is accomplished by the glTranslated( ) function and glutTimerFunc( ).

* **BLAST**:

Here a blast polygon is generated using glBegin(GL\_POLYGON) and color is added using glColor3f( ) openGL functions. It is rescaled using glScalef( ) function.

* Module in **4th DISPLAY-AIRBASE** is:
* **RUNWAY**:

Here a road with dividing whitestrips is generated using glBegin(GL\_POLYGON) and glColor3f( ) openGL functions. It is appropriately placed and continuously moved backward to simulate the forward movement of the fighter jet while landing using glTranslatef( ) function.

**4.2 Functions**

* **glutMainLoop():**
  + Cause the program to enter an event processing loop. It should be the last statement in main.
* **glutMouseFunc(mouse);**
  + GLUTs mouse interface provides a lot of options for adding mouse interactivy, namely detecting clicks and mouse motion. As in the keyboard version, GLUT provides a way for you to register the function that will be responsible for processing events generated by mouse clicks.
  + The name of this function is *glutMouseFunc*, and it is commonly called in the initialization phase of the application. The syntax is as follows:

void glutMouseFunc(void (\*func)(int button, int state, int x, int y));

* **glutKeyboardFunc(key);**
  + This *OpenGL* will teach you how to add keyboard interaction to your opengl application through the use of glut and glut's keyboard calls. Now glut has many different keyboard calls,
  + But anyway, all you have to edit after that is the 'main' function. Just add the line: glutKeyboardFunc (keyboard);   
    where 'keyboard' is the name of your keyboard function.
* **glutInitWindowPosition() and** **glutInitWindowSize() :**
  + Set the *initial window position* and *size* respectively.
  + Usage:

void glutInitWindowSize(int width, int height);

void glutInitWindowPosition(int x, int y);

* **glutCreateWindow():**
  + Creates a top-level window.
  + Usage:

glutCreateWindow(char \*name);

* **glutDisplayFunc():**
  + Sets the display callback for the *current window*.
  + Usage

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

* + Parameter

func : The new display callback function

* + Description

glutDisplayFunc sets the display callback for the current window.

* **glTranslate();**
  + Displaces points by a fixed distance in a given direction.
  + Usage

void glTranslate[fd](TYPE x, TYPE y, TYPE z)

* Parameter

Displacement values.

* Description

Alters the current matrix by a displacement of(x,y,z).

* **glScaled();**
* Non-rigid body transformation by which we can make an object bigger or smaller**.**
* Usage

void glScale[fd] (TYPE sx, TYPE sy,TYPE sz );

* Parameter

Fixed point, direction of scaling, scale factor

* Description

Alters the current matrix by a scaling of(sx,sy,sz);

**4.2.1 Running the Program and Display**

* glutDisplayFunc (void (\* func)(void)) is the most important event callback function. Whenever GLUT determines the contents of the window need to be redisplayed, the callback function registered by glutDisplayFunc()is executed.Therefore, all the routines needed to redraw the scene should be put in the display callback function. If the program changes the contents of the window, sometimes we will have to call **glutPostRedisplay**(void),which gives **glutMainLoop()** a nudge to call the registered displaycallback at itsnext opportunity.
* Similarly we can register for keyboard events with the followingfunctions:glutKeyboardFunc(void (\*func)(unsigned char key, int x, int y));

glutSpecialFunc(void (\*func)(int key, int x, int y));

* **glutMainLoop**(void):All windows that have been created are shown when this function is called, and rendering to those windows is now effective. Event processing begins, and the registered display callback is triggered. Once this loop is entered, it is never exited**!**

We can specify a function that‟s to be executed if no other events are pending - for example, when the event loop would otherwise be idle - with **glutIdleFunc**(void (\* func)(void)). This routine takes a pointer to the function as its only argument. Pass in NULL (zero) to disable the execution of the function.

**4.2.2 Other Important APIs**

* **glMatrixMode**(GLenum Mode)subroutine sets the current matrix mode. The Mode parameter can assume one of the following three values: **GL\_MODELVIEW** applies subsequent matrix operations to the model view matrix stack. **GL\_PROJECTION** applies subsequent matrix operations to the projection matrix stack. **GL\_TEXTURE** applies subsequent matrix operations to the texture matrix stack.
* **glPushMatrix** or **glPopMatrix** Subroutines: There is a stack of matrices for each of the matrix modes. In **GL\_MODELVIEW** mode, the stack depth is at least 32. In the other two modes, **GL\_PROJECTION** and **GL\_TEXTURE**, the depth is at least 2. The current matrix in any mode is the matrix on the top of the stack for that mode. The **glPushMatrix** subroutine pushes the current matrix stack down by one, duplicating the current matrix. That is, after a **glPushMatrixcall**, the matrix on the top of the stack is identical to the one below it. The **glPopMatrix** subroutine pops the current matrix stack, replacing the current matrix with the one below it on the stack.
* The glLoadIdentity subroutine replaces the current matrix with the identity matrix.
* glTranslatef(GLfloat x, GLfloat y, GLfloat z)subroutine moves the coordinate system origin to the point specified by (X,Y,Z). The translation vector is used to compute a 4 x 4 translation matrix as follows:

The current matrix is multiplied by this translation matrix, with the product replacing the current matrix.

* **glRotatef**(GLfloatAngle, GLfloatX, GLfloatY, GLfloatZ) subroutine computes a matrix that performs a counterclockwise rotation of Angle degrees about the vector from the origin through the point (X, Y, Z). The current matrix is multiplied by this rotation matrix, with the product replacing the current matrix.
* **glutBitmapCharacter**(void \*font, int character) subroutine outputs the given character in the given font at the current location on the window.
* **glEnable**(GLenum capability) or **glDisable(**GLenum capability) **Subroutines:** enable and disable various capabilities or settings for rendering. Both **glEnable** and **glDisable** take a single argument, capability, which may assume values for texturing, blending, hidden-surface removal, culling, etc.
* **glBegin**(GLenum mode) or glEnd Subroutine: delimit the vertices that define a primitive or group of like primitives. The glBegin subroutine accepts a single argument that specifies which of 10 ways the vertices will be interpreted, such as points, lines, quads, polygons, etc.

**Chapter 5**

**TESTING AND RESULTS**

The full designing and creating of Need For Speed 2012 has been executed under Ubuntu operating system , this platform provides a and satisfies the basic need of a good compiler. Using GL/glut.h library and built in functions make it easy to design good graphics package such as this walking robot.

Testing involves unit testing, module testing and system testing.

**UNIT TESTING**

Here the individual components are tested to ensure that they operate correctly. Each component is tested independently, without other system components.

**MODULE TESTING**

Module is a collection of dependent components such as procedures and functions. Since the module encapsulates related components can be tested with our other system modules. The testing process is concerned with finding errors which results from erroneous function calls from the main function to various individual functions.

**SYSTEM TESING**

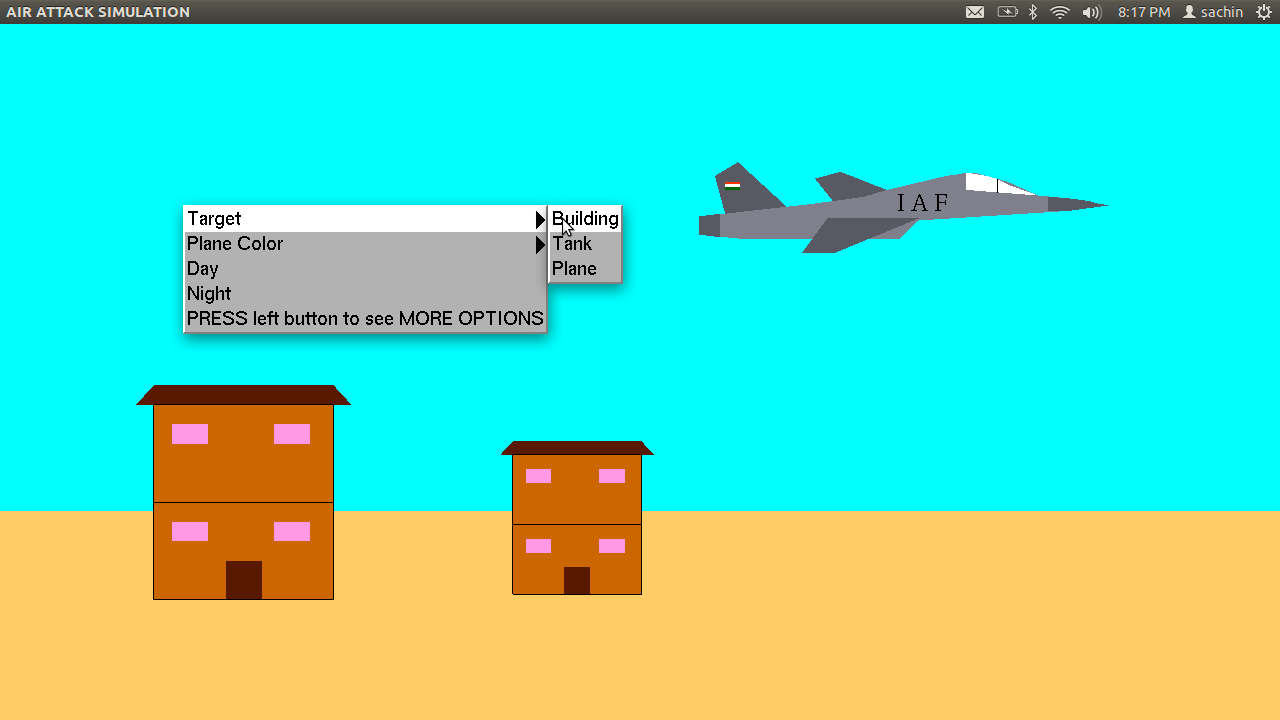
The Modules are integrated to make up the entire system. The testing process is concerned with finding errors with the results from unanticipated interactions between module and system components. It is also concerned with validating that the system meets its functional and non-functional requirements.

**5.1 Test Cases**

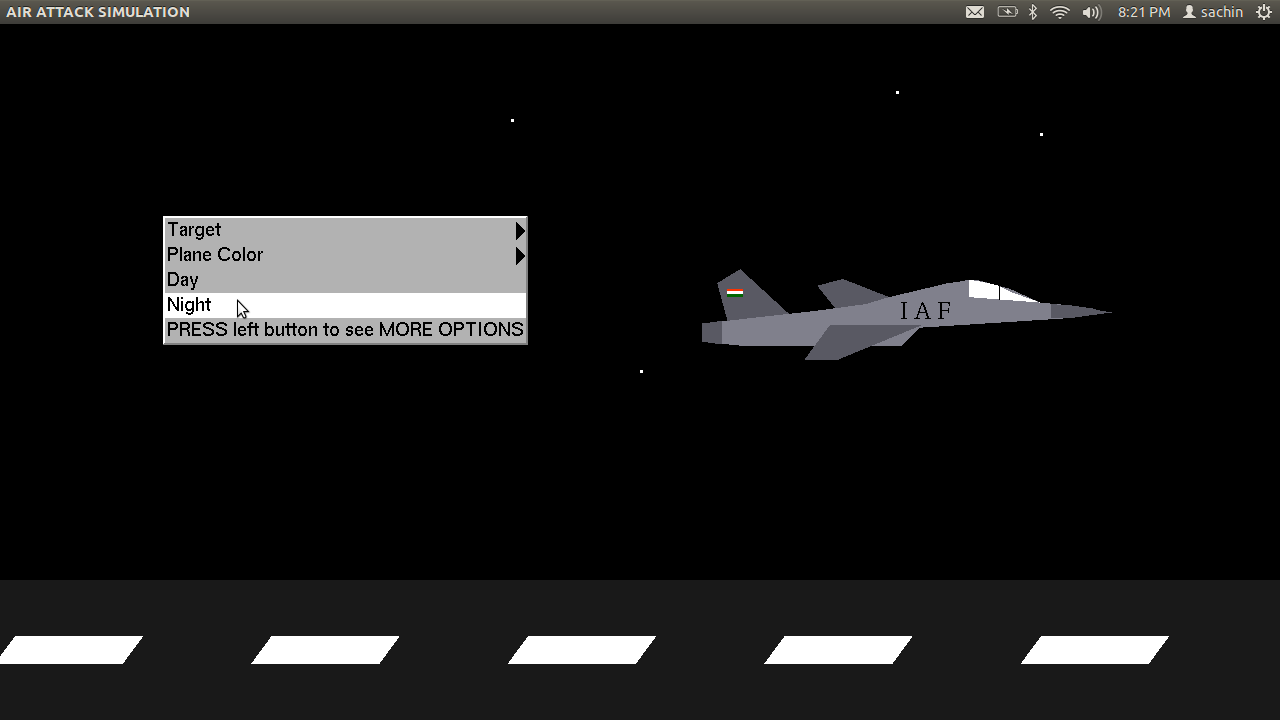
The table depices the test case result

**Chapter 6**

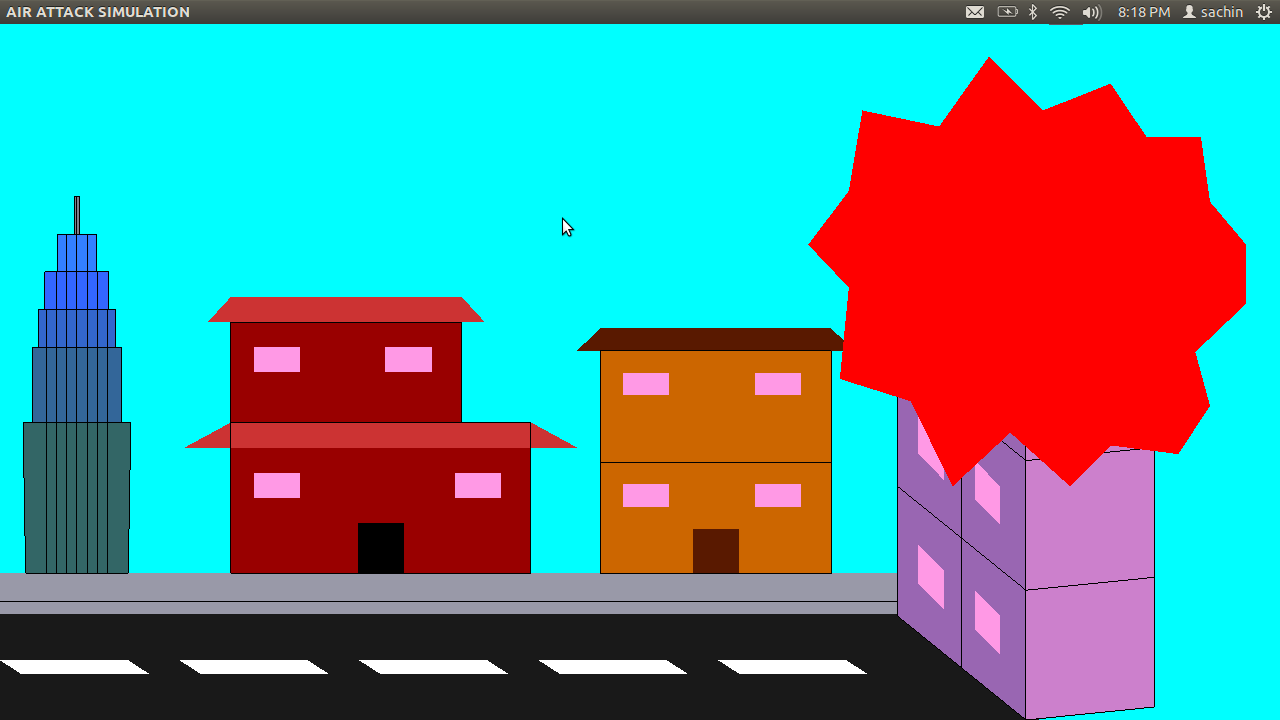
**SNAPSHOT**



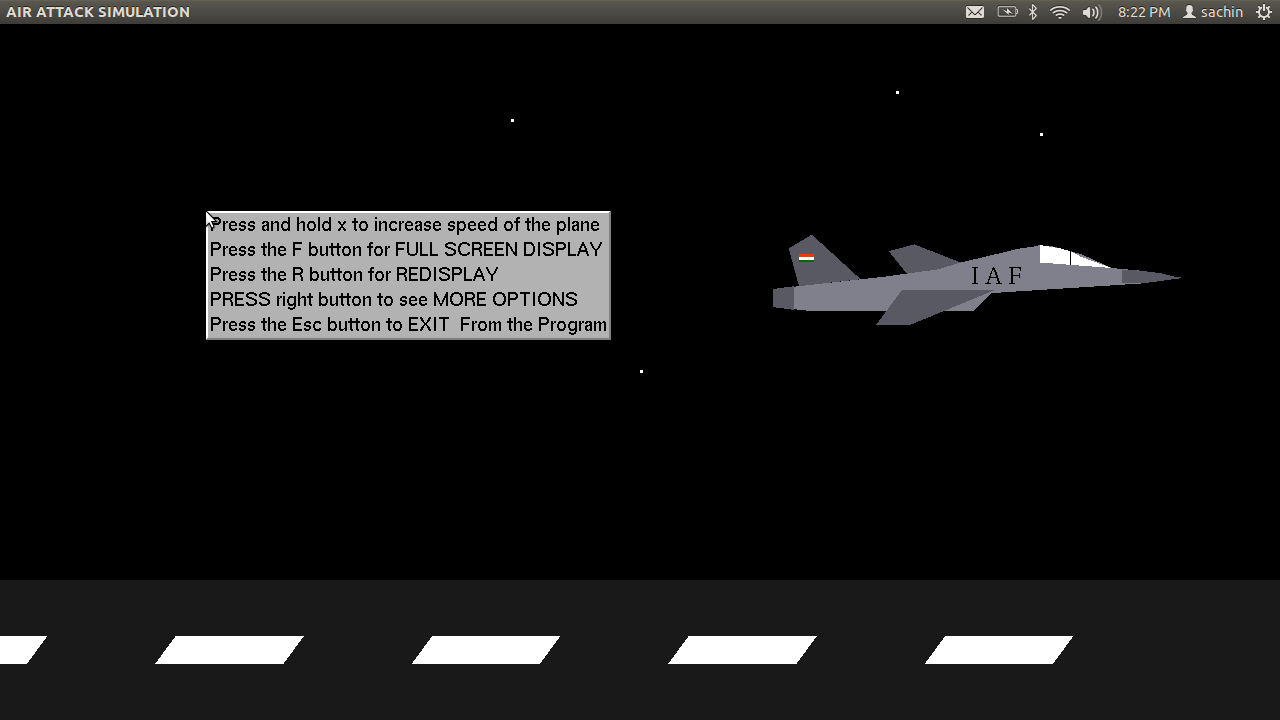
**Fig 6.1** starting page of the Project



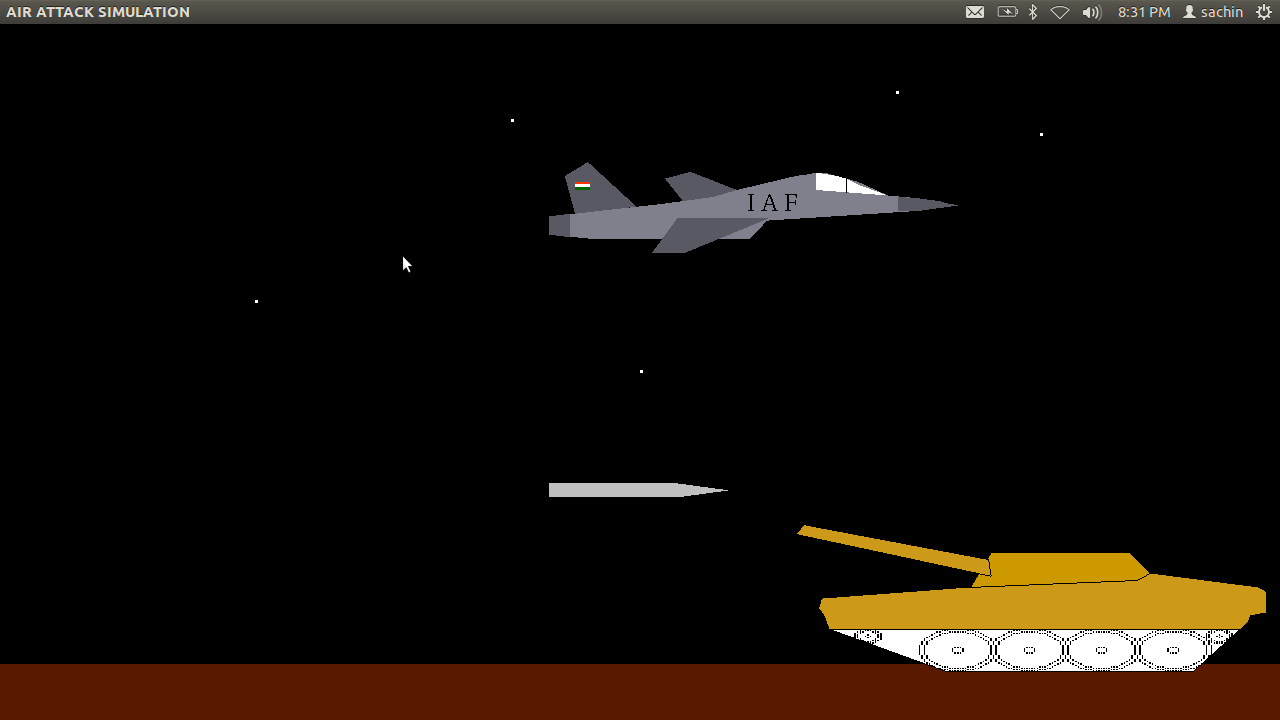
**Fig 6.2** Night Background with options



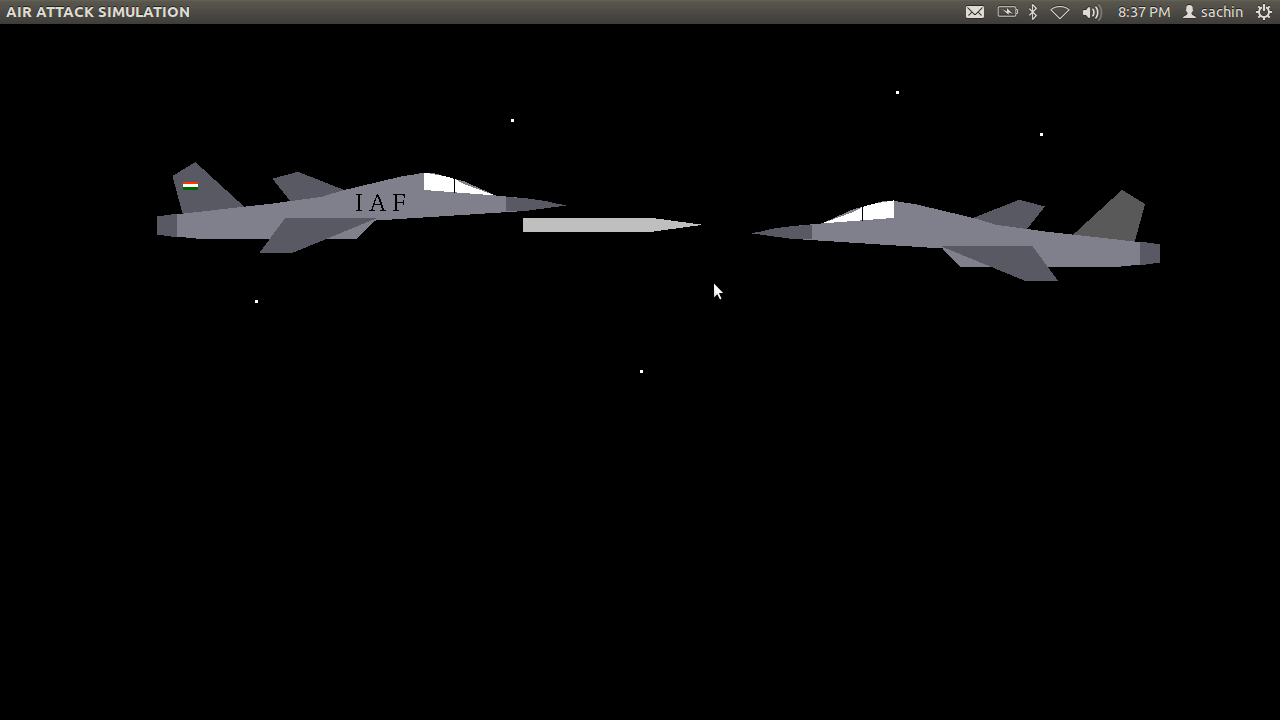
**Fig 6.3** Building is set as Target and blast!!!



**Fig 6.4** Other Options



**Fig 6.5** Tank is selected as Target



**Fig 6.6** Enemy Jet is selected as Target



**CONCLUSION**

An attempt has been made to develop an OpenGL graphics package which meets all the necessary requirements that were set out. It is user friendly and provides an easy interaction for the user. The user can very easily use this tool to draw or manipulate a drawing. The interfaces are mouse and keyboard driven and the user can select a function by clicking on an option representing that function or by pressing keys in the keyboard. I finally conclude that this graphics package satisfies all requirements and provides good entertainment.

**FUTURE ENHANCEMENTS**

We can modify the project by adding some of the features using openGL in future. The project can further be implemented by adding materials, the texture effect and animation. Modification can be done to change the manner of execution, more interaction could be given by mouse interface and keyboard interface. The project can further be implemented by other modifications if required. This project has been developed using selected primitives that are easy to implement. It can be enhance using more complex functions that are available in the GLUT library

**REFERENCE**

[1] Donald Hearn and Pauline Baker: Computer Graphics- OpenGL Version, 3rd Edition, Pearson Education, 2004.

[2] F.S. Hill Jr. Computer Graphics Using OpenGL, 3rd Edition, Phi, 2009

[3] James D Foley, Andryes Van Dam, Steven K Feiner, John F Hughes, Computer Graphics, Pearson Education 1997.

[4] http://vtucs.com/computer-graphics-projects/Co

[5] [http://cglabprojects.blogspot.in/2014/03/-Air Attack](http://cglabprojects.blogspot.in/2014/03/-Air%20Attack) Simulation.html