VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangam",Belgavi-590014

A CG Mini Project Report on

"ROCKET LAUNCH SIMULATION"

Submitted to

Visvesvaraya Technological University

In the partial fulfilment of requirements for the award of degree

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

SAKEENA IRAM	**4RA21CS074**
SAHANA SM	**4RA22CS073**

Under the Guidance of

Mr. SANJAY M, BE.,M.TECH

Assistant Professor, Dept. of CSE

Rajeev Institute of Technology,Hassan

RAJEEV INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

HASSAN-573201

2023-2024

RAJEEV INSTITUTE OF TECHNOLOGY
HASSAN

(Approved by AICTE ,New Delhi and Affiliated to VTU, Belagavi.)
Plot#1-D,Growth Centre, Industrial Area,B-M Bypass Road,Hassan–573201
Ph:(08172)-243180/83/84 Fax:(08172)-243183

Department of Computer Science and Engineering

CERTIFICATE

Certified that the **CG Mini Project** entitled **"ROCKET LAUNCH SIMULATION"** is carried out by **Ms. SAKEENA IRAM [4RA21CS074]** and **Ms. SAHANA S M [4RA21CS073]** respectively, a bonafide students of **Rajeev Institute Of Technology,** Hassan in partial fulfillment of the requirements for the **6th Semester** of **Bachelor Of Engineering** in **Computer Science And Engineering** of the **Visvesvaraya Technological University**, Belagavi during the year 2023-2024. The mini Project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said Degree.

Mr. SANJAY M	**Dr. ARJUN B C**
Assistant Professor	Head of the Department
Dept. of Computer Science & Engineering	Dept. of Computer Science & Engineering
RIT,HASSAN	RIT, HASSAN

Name of the examiners	Signature with date
1.	……
2. …….....	……

DECLARATION

We **SAKEENA IRAM , and SAHANA S M,** bearing USN **4RA21CS074**, **4RA21CS073,** students of 6th Semester B.E in **Computer Science and Engineering**, **Rajeev Institute of Technology, Hassan**, hereby declare that the work being presented in the dissertation entitled " **ROCKET LAUNCH SIMULATION** " has been carried out by us under the supervision of guide **Mr. Sanjay M**, Assistant Professor, Computer Science and Engineering, **Rajeev Institute of Technology, Hassan**, as partial fulfillment of requirement for the award of B.E Degree of **Bachelor of Engineering in Computer Science and Engineering** at **Visvesvaraya Technological University, Belagavi** is and authentic record of my own carried out by us during the academic year 2023-2024.

SAKEENA IRAM
(4RA21CS074)

SAHANA SM
(4RA21CS073)

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned our efforts with success.

We would like to profoundly thank our **Management of Rajeev Institute of Technology** & our President **Dr. Rachana Rajeev** for providing such a healthy environment.

We would like to express our sincere thanks to our principal **Dr. Mahesh P K**, Rajeev Institute of Technology for his encouragement that motivated us for successful completion of mini project work.

We wish to express our gratitude to **Dr. Arjun B C**, Head of the Department of Information Science& Engineering for providing a good working environment and for his constant support and encouragement.

It gives us great pleasure to express our gratitude to **Mr. Sanjay M** Assistant Professor, Departmentof Computer Science& Engineering for his expert guidance, initiative and encouragement that led us to complete this mini project.

We would also like to thank all our staffs of Information Science and Engineering department who have directly or indirectly helped us in the successful completion of this mini project and, we would like to thank our parents.

SAKEENA IRAM
(4RA21CS074)

SAHANA S M
(4RA22CS073)

ABSTRACT

In this project we are demonstrating about The rocket launching project is a 2D animation of a satellite launching process. In our project rocket will be static prior to launching. When a launch is performed through interactions, the countdown begins for the launch. The countdown is 10 to 0, once the countdown is finished rocket is launched. Once the rocket is launched, actual animation starts here. Rocket starts moving into the sky, after sometime it reaches the space. The next phase of the rocket launch is dismantling stage1, in this stage the first fuel supplier will be dismantled. Now rocket starts moving with the help of second fuel supplier. The next stage will be dismantling stage2, here the fuel supplier 2 will be dismantled. The next phase is forced dismantling stage, where the unwanted parts are expelled. After this phase satellite is launched.

CONTENTS

LIST OF FIGURES

LIST OF SNAPSHOTS

Chapter 1

INTRODUCTION TO COMPUTER GRAPGICS

Computer graphics are graphics created using computers and more generally, the representation and manipulation of image data by a computer. The development of computer graphics has made computers easier to interact with, and better for understanding and interpreting many types of data. Developments in computer graphics have a profound effect on many types of media and have revolutionized animation, movies and the video game industry.

1.1 Overview of Computer Graphics

The term computer graphics has been used in a broad sense to describe "almost everything on computers that is not text or sound. It is one of the most powerful and interesting facts of computer. There is a lot that you can do apart from drawing figures of various shapes.

Today, computers and computer-generated images touch many aspects of daily life. Computer image is found on television, in newspapers, for example in weather reports, in all kinds of medical investigation and surgical procedures. In the media such graphs are used to illustrate papers, reports, and other presentation material.

Many powerful tools have been developed to visualize data. Computer generated imagery can be categorized into several different types: 2D, 3D, 5D, 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, other specialized fields have been developed like information visualization, and scientific visualization more concerned with "the visualization of three dimensional phenomena" (architectural, m of volumes, surfaces, illumination sources, and so forth, perhaps with adynamic (time component). Medical, biological, etc.), where emphasis is on realistic renderingsof volumes, surfaces, illumination sources, and so forth, perhaps with a dynamic (time component).

A computer graphics system is a computer system with all the components of the general purpose computer system. There are five major elements in system: input devices, processor, memory, frame buffer, output devices.

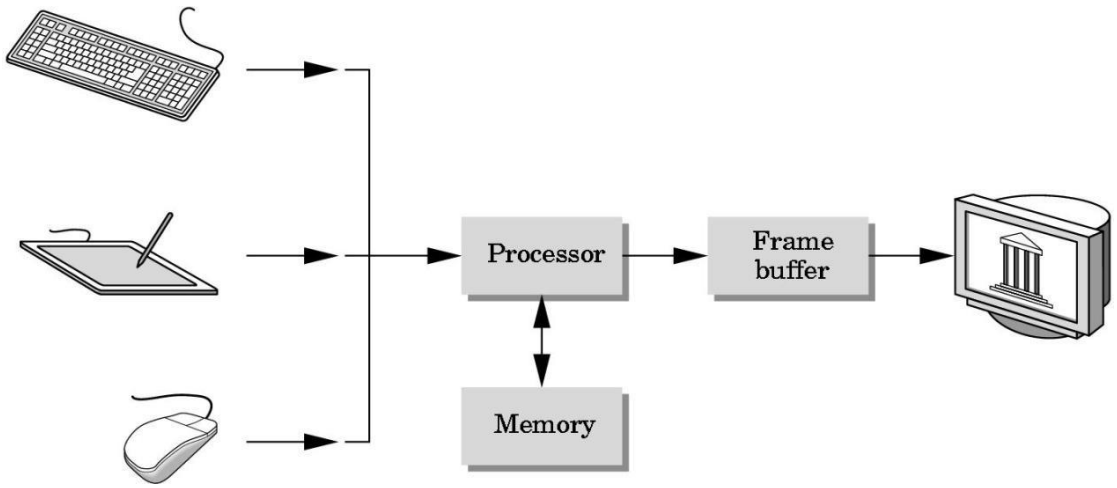


Fig 1.1: CG Model

1.2 History of Computer Graphics

Computer graphics was first created as a visualization tool for scientists and engineers in government and corporate research centers such as Bell Labs and Boeing in the 1950s. Later the tools would be developed at universities in the 6os and 70s at places such as Ohio State University, MIT, University of Utah Corn shell, North Carolina and the New York Institute of technology. The early breakthroughs that took place in academic centers continued at research centers such as the famous Xerox PARC in the 1970‟s. These efforts broke first into broadcast video graphics and then major motion pictures in the late 70‟s and early 1980‟s. Computer graphic research continues to day around the production companies. Companies such as George Luca‟s Industrial light and magic are constantly redefining the cutting edge of computer graphic technology in order to present the world with a new synthetic digital reality.

1.3 Applications of Computer Graphics

Nowadays Computer Graphics used in almost all the areas ranges from science, engineering, medicine, business, industry government, art, entertainment, education and training.

1.3.1 CG in the field of CAD

Computer Aided design methods are routinely used in the design of buildings, automobiles, aircraft, watercraft, spacecraft computers, textiles and many other applications.

1.3.2 CG in Presentation Graphics

Another major application area presentation graphics used to produce illustrations for reports or generate slides. Presentation graphics is commonly used to summarize financial, statistical, mathematical, scientific data for research reports and other types of reports.2D and 3D bar chart to illustrate some mathematical or statistical report.

1.3.3 CG in computer Art

CG methods are widely used in both fine art and commercial art applications. Artists use a variety of computer methods including special purpose hardware, artist‟s paintbrush program, other pain packages, desktop packages, mathematics packages, animation packages that provide facility for designing object motion.

1.3.4 Image Processing

Concerned with fundamentally different operations. In CG a computer is used to create a picture. Image processing on the other hand applies techniques to modify existing pictures such as photo scans and TV scans.

1.3.5 User Interface

It is a common for software packages to provide a graphical interface . A major component ofa graphical interface is a window manager that allows a user to display multiple window area. Interface also displays menus, icons for fast selection and processing

1.3.6 Education and Training

Computer generated models of physical, financial, economic system often acts as education aids. For some training application special system are designed. Ex: specialized system for simulator for practice sessions or training of ship captain, aircraft pilots and traffic control.

Chapter 2

INTRODUCTION TO OPENGL

OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. Since its introduction in 1992, OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms.

OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualizationfunctions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment.

OpenGL Available Everywhere: Supported on all UNIX® workstations, and shipped standard with every Windows 95/98/2000/NT and MacOS PC, no other graphics API operates on a wider range of hardware platforms and software environments.

OpenGL runs on every major operating system including Mac OS, OS/2, UNIX, Windows 95/98, Windows 2000, Windows NT, Linux, OPEN Step, and BeOS; it also works with every major windowing system, including Win32, MacOS, Presentation Manager, and X- Window System. OpenGL is callable from Ada, C, C++, Fortran, Python, Perl and Java and offers complete independence from network protocols and topologies.

2.1 Basic OpenGL Operation

Most of our application will be designed to access OpenGL directly through functions in three libraries. Functions in the main GL (or OpenGL in windows) library have names that begin with letters GL.

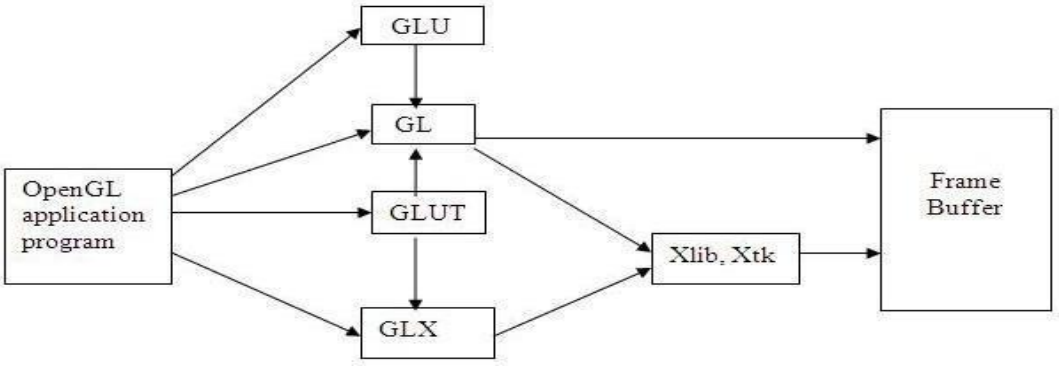


Fig 2.1: Library Organization

2.2 OpenGL Command

OpenGL commands use the prefix gl and initial capital letters for each word making up the command name. Similarly, OpenGL defined constants begin with GL_, use all capital letters and use underscores to separate words (like GL_COLOR_BUFFER_BIT). To create a "Bull's Eye" target game using OpenGL commands in Python, you'll need to use OpenGL's functions to handle drawing shapes and managing the game scene. Here‟s a detailed example ofhow to implement the game with OpenGL commands.

1. Initialization:

gluOrtho2D (0, 800, 0, 600): Sets up a 2D orthographic projection where the coordinates range from (0,0) to (800,600).

2. Target Drawing:

glBegin (GL_TRIANGLE_FAN): Begins drawing a filled circle. The fan is a set of triangles that fan out from a central point. - glVertex2f (x, y): Specifies the vertices of the circle. glColor3fv(color): Sets the color for the subsequent vertices.

3. Arrow Drawing:

•glBegin (GL_QUADS): Draws a rectangle (the arrow) using four vertices.

•glVertex2f (x, y): Specifies the vertices of the rectangle.

4. Clearing the Screen:

•glClear (GL_COLOR_BUFFER_BIT): Clears the screen buffer to prepare for the next frame. OpenGL Features Used.

• Primitive Shapes: Circles are created using triangle faas, and rectangles (arrows) are created using quads.

• Color Management: Colors are set using glColor3fv()

5. Shader Programming:

• glCreateShader(type): Creates a shader object. type can be GL_VERTEX_SHADER, GL_FRAGMENT_SHADER, etc.

• glShaderSource(shader, count, string, length): Specifies the source code for a shader object.

Chapter 3

SYSTEM DESIGN

3.1 Design Overview

The system design for a rocket launch simulation involves creating a comprehensive framework that replicates the stages of a rocket launch with high fidelity. This simulation system integrates multiple components such as a physics engine to accurately model the forces and trajectories, real-time data processing to handle the dynamic nature of the launch, and visual rendering to provide an immersive experience. The design also incorporates user interfaces for inputting launch parameters and monitoring progress. Scalability and modularity are key considerations, allowing for enhancements and extensions, such as adding new rocket models or incorporating more detailed environmental factors. This robust and flexible design ensures the simulation is both realistic and adaptable to various educational and training needs.

3.2 System Architecture:

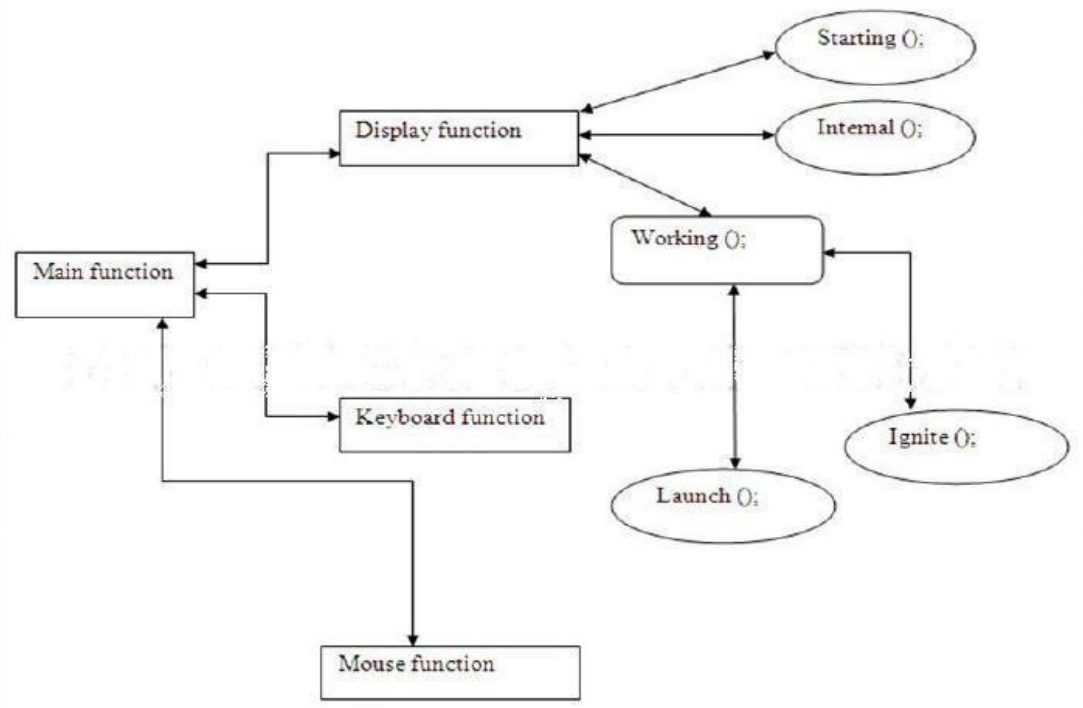


Fig 3.1: System Architecture

The system architecture for a rocket launch simulation consists of several interconnected components: the User Interface (UI) provides controls for configuring the simulation and displays real-time data; the Control Systems manage user inputs and coordinate the simulation's various subsystems; the Simulation Engine performs core calculations for rocket physics, trajectory, and mission logic; the Rendering Engine generates the visual representation of the simulation including the rocket, launch pad, and animations; Data Management handles the storage, analysis, and logging of simulation data and results; the Scenario Manager sets up and switches between different simulation scenarios and conditions; and the Feedback System offers real-time updates and performance metrics. These components work together to create an interactive and realistic simulation environment for analyzing rocket launches and training purposes.

3.3 Advantages

- Cost-Effectiveness
- Safety
- Realistic Training
- Experimentation and testing
- Environmental impact
- Educational Tool

3.4 Disadvantages

- Limited Realism
- High initial-cost
- Complexity
- Dependence on Technology
- Resource Intensive

Chapter 8

REQUIREMENT SPECIFICATION

4.1 Hardware Requirements

The standard output device is assumed to be a **Color Monitor**. It is quite essential for any graphics package to have this, as provision of color options to the user is a must. The **mouse**, the main input device, has to be functional i.e. used to move the car left or right in the game. A **keyboard** is used for controlling and inputting data in the form of characters, numbers i.e. to enter the player name in Need For Speed 2012, etc. Apart from these hardware requirements there should be sufficient hard disk space and primary memory available for proper working of the package to execute the program. Pentium III or higher processor, 16MB or more RAM. A functional display card.

Minimum Requirements Expected are cursor movement, creating objects like lines, squares, rectangles, polygons, etc. Transformations on objects/selected area should be possible. Filling of area with the specified color should be possible.

4.2 Software Requirements

The editor has been implemented on the OpenGL platform and mainly requires an appropriate version of eclipse compiler to be installed and functional in Ubuntu. Thought it is implemented in Open GL, it is pretty much performed and independent with the restriction that there is support for the execution of C and C++ files. Text Modes is recommended.

Operating System: Windows 7+ or Ubuntu

Language Used In Coding: C/C++

Application used: Code Blocks

Graphics: OpenGL

Chapter 5

IMPLIMENTATION

5.1 Functions Used

Void glColor3f(float red, float green, float blue);

This function is used to mention the color in which the pixel should appear. The number 3 specifies the number of arguments that the function would take. „f" gives the type that is float. The arguments are in the order RGB(Red, Green, Blue). The color of the pixel can be specified as the combination of these 3 primary colors.

Void glClearColor(int red, int green, int blue, int alpha);

This function is used to clear the color of the screen. The 4 values that are passed as arguments for this function are (RED, GREEN, BLUE, ALPHA) where the red green and blue components are taken to set the background color and alpha is a value that specifies depth of the window. It is used for 3D images.

Void glFlush();

Different GL implementations buffer commands in several different locations, including network buffers and the graphics accelerator itself. **glFlush** empties all of these buffers, causing all issued commands to be executed as quickly as they are accepted by the actual rendering engine. Though this execution may not be completed in any particular time period, it does complete in finite time.

void glutInit(int *argcp, char **argv);

glutInit will initialize the GLUT library and negotiate a session with the window system. During this process, glutInit may cause the termination of the GLUT program with an error message to the user if GLUT cannot be properly initialized. Examples of this situation include the failure to connect to the window system, the lack of window system support for OpenGL, and invalid command line options.glutInit also processes command line options, but the specific options parse are window system dependent.

glOrtho ( ) ;

Syntax: void glOrtho (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far);

The function defines an orthographic viewing volume with all parameters measured from the centre of the projection plane.

void glutMainLoop(void);

glutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never stop.

5.2 User Defined Function

Void display(void):

In this function first we should print the instructions that we would be displayed on the pop-up window.

Void displayloop():

Here we are display starting stage ,intermediate & final stage of Rocket.

Void myinit():

It sets background color and orthographic window.

Int main(int argc,charargv):**

Here we call all the function we defined previously in the program and this function creates a output window.

void control():

Manages the state transitions for the rocket's animation.

void stars():

Draws a static background of stars.

void static_rocket():

Draws the scene with the rocket in a static position.

void rocket_in_motion():

Animates the rocket's flight into space.

5.3 Flow Chart

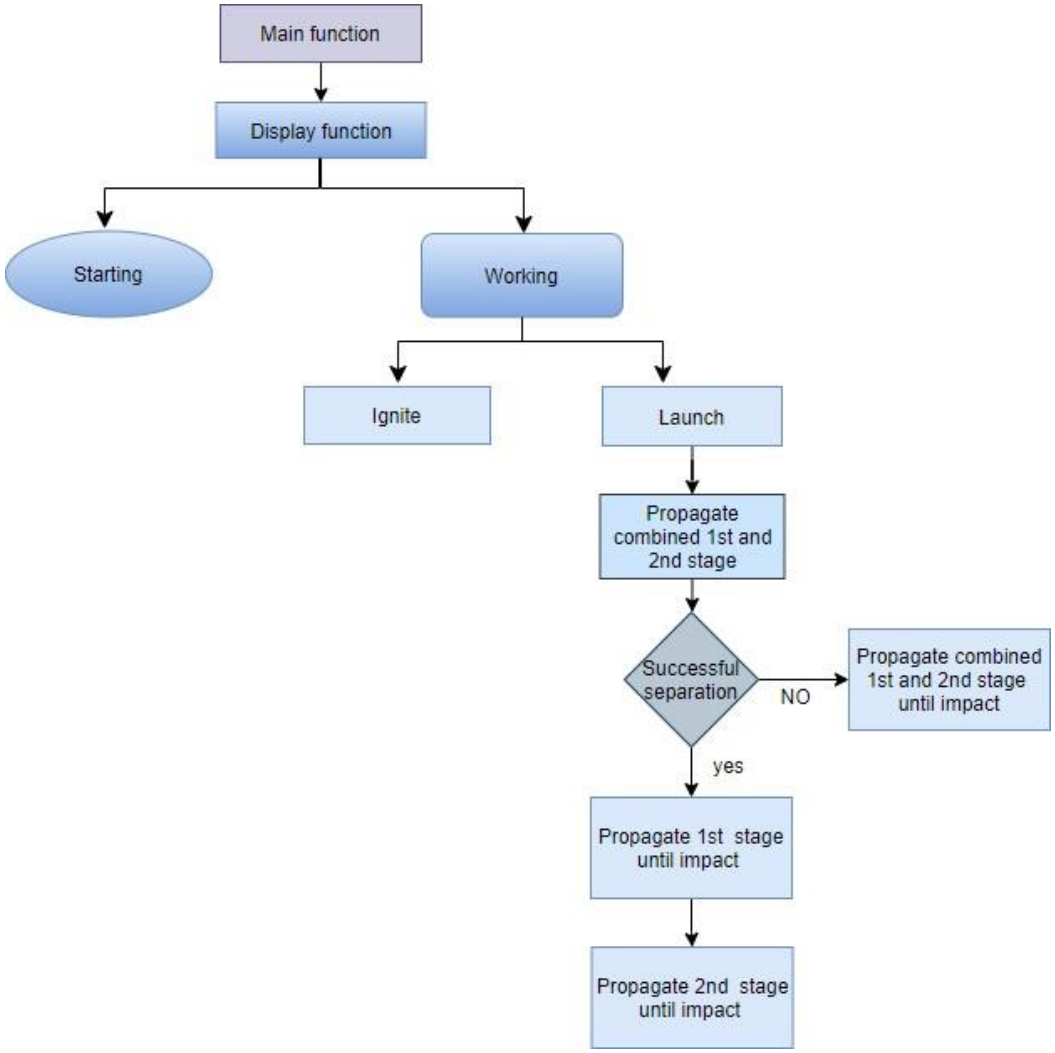


Fig 5.3 : Flow Chart Diagram

1. Main Function:

Control the overall flow of the rocket launch simulation.

- Initialize Simulation: Set up initial parameters for the simulation, including rocket specifications, environmental conditions, and mission objectives.

2. Display Function:

Show the current status of the rocket and simulation parameters.

- Show Current State: Display real-time data such as altitude, speed, and fuel levels. provide information on the rocket‟s current status.

3. Starting:

Begin the rocket launch sequence.

- Begin Launch Sequence **:**Perform final checks and prepare for ignition .Ensure all systems are ready for launch.

4. Working:

Execute core operations necessary for the launch.

- Execute Core Operations **:**Manage preparations and configurations before the engines start. Prepare rocket engines and configure flight parameters.

5. Ignite and Launch:

Start the rocket engines and initiate the launch.

- Start Rocket Engines **:**Ignite the engines to begin the rocket‟s ascent. Start the propellant flow and initiate the launch sequence.

6. Propagate:

Manage the rocket‟s ascent and flight dynamics.

- Handle Ascent Dynamics: Track the rocket‟s flight trajectory and adjust as needed. Update the rocket‟s position, speed, and trajectory

7. Successful Separation?:

Determine if the first stage has separated successfully from the rocket.

- Check Separation: Verify that Stage 1 has detached and is no longer part of the rocket. Confirm that the separation process was successful.

8. If No:

Abort the mission and handle failure.End the simulation due to the failure of Stage 1 separation.

9. If Successful: Propagate Stage 1 Until Impact

Continue tracking the first stage of the rocket until it impacts the ground.

- Continue Flight Dynamics for Stage 1: Manage the descent and impact of Stage 1.Track the rocket‟s trajectory and prepare for Stage 1‟s impact.

10. Separate Stage 2:

Initiate the ignition and operations of Stage 2.

- Ignite Stage 2 Engines: Start the engines for Stage 2 after Stage 1 has separated .Begin Stage 2‟s flight operations.

11. Propagate Stage 2:

Manage the flight dynamics for Stage 2.

- Continue Flight Dynamics for Stage 2**:** Track Stage 2‟s trajectory and manage its operations. Ensure Stage 2 progresses toward its mission goals.

12. Post-Launch Operations:

Finalize the simulation and analyze the results.

- Data Collection and Analysis: Collect and analyze data from the mission .Review the results of Stage 2 and finalize the mission.

.

CHAPTER 06

SOURCE CODE

```
#include<GL/glut.h>
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
#include<string.h>

const float DEG2RAD = 3.14159/180; void stars();
int p;
void stars1();
void static_rocket();
void rocket_to_cam_pos();
void rocket_in_motion();
void mars(float radius);

float i,j,count=0,count1=0,count3=0,flag=0,flag1=0,t=0,f=0,flag3=0;

void drawstring(int x, int y, char *s)
{
        char *c;
        glRasterPos2i(x, y);
        for (c = s; *c != '\0'; *c++)
                glutBitmapCharacter(GLUT_BITMAP_8_BY_13, *c);
}

void semicircle(float radius,float u,float v)
{
        glColor3f(1.0 ,1.0 ,1.0);
glBegin(GL_POLYGON);

for (int i=135; i<=315; i++)
  {
        float degInRad = i*DEG2RAD;
        glVertex2f(u+cos(degInRad)*radius,v+(sin(degInRad))*radius);//100,100
        specifies centre of the circle
  }
 glEnd();
}
void control()
{
        count1++;
        if(count1==250)
```

```
flag=1;
```

```
        else if (flag == 1 && (count1 == 600 || count1 == 601))
                rocket_to_cam_pos();

        else if (flag == 1 && count1 >= 1000)

                rocket_in_motion();
}

Void stars()
{
        glColor3f(1.0,1.0,1.0);
        glPointSize(2.37);
        glBegin(GL_POINTS);
        glVertex2i(10,20);
        glVertex2i(20,100);
        glVertex2i(30,10);
        glVertex2i(15,150);
        glVertex2i(17,80);
        glVertex2i(200,200);
        glVertex2i(55,33);
        glVertex2i(400,300);
        glVertex2i(330,110);
        glVertex2i(125,63);
        glVertex2i(63,125);
        glVertex2i(20,10);
        glVertex2i(110,330);
        glVertex2i(440,430);
        glVertex2i(32,65);
        glVertex2i(110,440);
        glVertex2i(210,230);
        glVertex2i(390,490);
        glVertex2i(12,90);
        glVertex2i(400,322);
        glVertex2i(420,366);
        glVertex2i(455,400);
        glVertex2i(20,20);
        glVertex2i(111,120);
        glVertex2i(401,200);
        glVertex2i(230,30);
        glVertex2i(220,20);
        glVertex2i(122,378);
        glVertex2i(133,340);
        glVertex2i(345,420);
        glVertex2i(130,360);
        glVertex2i(333,120);
        glVertex2i(250,22);
```

```
        glVertex2i(242,11);
        glVertex2i(280,332);
        glVertex2i(233,40);
        glVertex2i(210,418);
        glVertex2i(256,12);
        glVertex2i(288,232);
        glVertex2i(247,36);
        glVertex2i(229,342);
        glVertex2i(257,47);
        glVertex2i(290,63);
        glVertex2i(232,72);
        glVertex2i(243,143);
        glVertex2i(100,200);
        glVertex2i(90,250);
        glVertex2i(80,225);
        glVertex2i(50,333);
        glVertex2i(60,350);
        glVertex2i(243,143);
        glVertex2i(243,143);
        glEnd();
}

void stars1()
{
       int l;
        glColor3f(1.0,1.0,1.0);
        glPointSize(1.25);
        glBegin(GL_POINTS);
        glVertex2i(50,20);
        glVertex2i(70,100);
        glVertex2i(80,10);
        glVertex2i(65,150);
        glVertex2i(67,80);
        glVertex2i(105,33);
        glVertex2i(450,300);
        glVertex2i(380,110);
        glVertex2i(175,63);
        glVertex2i(113,125);
        glVertex2i(70,10);
        glVertex2i(160,330);
        glVertex2i(490,430);
        glVertex2i(82,65);
        glVertex2i(160,440);
        glVertex2i(440,490);
```

```
        glVertex2i(62,90);
        glVertex2i(450,322);
        glVertex2i(420,366);
        glVertex2i(455,400);
        glVertex2i(60,20);
        glVertex2i(111,120);
        glVertex2i(451,200);
        glVertex2i(280,30);
        glVertex2i(220,20);
        glVertex2i(132,378);
        glVertex2i(173,340);
        glVertex2i(325,420);
        glVertex2i(180,360);
        glVertex2i(383,120);
        glVertex2i(200,22);
        glVertex2i(342,11);
        glVertex2i(330,332);
        glVertex2i(283,40);
        glVertex2i(210,418);
        glVertex2i(256,12);
        glVertex2i(288,232);
        glVertex2i(247,36);
        glVertex2i(229,342);
        glVertex2i(257,47);
        glVertex2i(290,63);
        glVertex2i(232,72);
        glVertex2i(243,143);
        glVertex2i(100,200);
        glVertex2i(90,250);
        glVertex2i(80,225);
        glVertex2i(50,333);
        glVertex2i(60,350);
        glVertex2i(243,143);
        glVertex2i(243,143);
        glEnd();
        for(l=0;l<=10000;l++)
}
void static_rocket()
{
count1++;
//if(count1==150)
//flag=1;
if(flag==0)
{
```

```
glClearColor(0.196078 ,0.6 ,0.8,1.0);
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

glColor3f(0.4,0.25,0.1);
        glBegin(GL_POLYGON);//green ground
        glVertex2f(0.0,0.0);
        glVertex2f(0.0,250.0);
        glVertex2f(270.0,250.0);
        glVertex2f(500.0,50.0);
        glVertex2f(500.0,0.0);
        glEnd();
        glBegin(GL_POLYGON);//green ground
        glVertex2f(280.0,250.0);
        glVertex2f(500.0,250.0);
        glVertex2f(500.0,60.0);
        glEnd();
        glColor3f(0.0,0.0,0.0);
                glBegin(GL_POLYGON);//road
        glVertex2f(260.0,250.0);
        glVertex2f(290.0,250.0);
        glVertex2f(500.0,70.0);
        glVertex2f(500.0,40.0);
        glEnd();
        glColor3f(0.0,0.0,0.0);

        glColor3f(0.8,0.498039 ,0.196078);
                glBegin(GL_POLYGON);//house 1
        glVertex2f(250.0,250.0);
        glVertex2f(300.0,250.0);
        glVertex2f(300.0,350.0);
        glVertex2f(250.0,350.0);
        glEnd();
        glColor3f(0.7,0.7,0.7);
        glBegin(GL_POLYGON);//HOUSE A
                glVertex2f(255,267.5);
                glVertex2f(275.0,267.5);
                glVertex2f(275.0,277.5);
                glVertex2f(255.0,277.5);
                glEnd();
        glBegin(GL_POLYGON);//HOUSE B
                glVertex2f(255,285.0);
                glVertex2f(275.0,285);
                glVertex2f(275.0,295);
                glVertex2f(255.0,295);
                glEnd();
```

```
glBegin(GL_POLYGON);//HOUSE C
        glVertex2f(255,302.5);
        glVertex2f(275.0,302.5);
        glVertex2f(275.0,312.5);
        glVertex2f(255.0,312.5);
        glEnd();

glBegin(GL_POLYGON);//HOUSE D
         glVertex2f(255,320.0);
         glVertex2f(275.0,320.0);
         glVertex2f(275.0,330.0);
         glVertex2f(255.0,330.0);
         glEnd();

glBegin(GL_POLYGON);//HOUSE E

        glVertex2f(285,267.5);

        glVertex2f(295.0,267.5);

        glVertex2f(295.0,277.5);

        glVertex2f(285.0,277.5);

         glEnd();
glBegin(GL_POLYGON);//HOUSE F
        glVertex2f(285,285.0);
        glVertex2f(295.0,285);
        glVertex2f(295.0,295);
        glVertex2f(285.0,295);
        glEnd();
glBegin(GL_POLYGON);//HOUSE G
        glVertex2f(285,302.5);
        glVertex2f(295.0,302.5);
        glVertex2f(295.0,312.5);
        glVertex2f(285.0,312.5);
        glEnd();
glBegin(GL_POLYGON);//HOUSE H
        glVertex2f(285,320.0);
        glVertex2f(295.0,320.0);
        glVertex2f(295.0,330.0);
        glVertex2f(285.0,330.0);
        glEnd();
        glColor3f(0.647059 ,0.164706 ,0.164706);
        glBegin(GL_POLYGON);//solid cone
        glVertex2f(26,250);
        glVertex2f(52,250);
```

```
                    glVertex2f(39,290);
                    glEnd();
                    semicircle(20.0,50,300);

        glColor3f(0.0,0.0 ,0.0);
                     glBegin(GL_LINES);//wires
                     glVertex2f(37,313);
                     glVertex2f(62,310);
                     glVertex2f(63,287);
                     glVertex2f(62,310);
                     glEnd();
              glColor3f(1.0,1.0,1.0);

              glEnd();

              glPointSize(2.0);

        glColor3f(1.0,1.0 ,1.0);
glBegin(GL_POINTS);//road paint
                    glVertex2f(497,56);
                    glVertex2f(488,65);
                    glVertex2f(479,74);
                    glVertex2f(470,83);
                    glVertex2f(460,92);
                    glVertex2f(450,101);
                    glVertex2f(439,110);
                    glVertex2f(428,119);
                    glVertex2f(418,128);
                    glVertex2f(408,137);
                    glVertex2f(398,146);
                    glVertex2f(388,155);
                    glVertex2f(378,164);
                    glVertex2f(366,173);
                    glVertex2f(356,182);
                    glVertex2f(346,191);
                    glVertex2f(336,200);
                    glVertex2f(324,209);
                    glVertex2f(314,218);
                    glVertex2f(304,227);
                    glVertex2f(294,234);
                    glVertex2f(284,243);
                    glVertex2f(278,248);
                    glEnd();
```

```
glColor3f(0.0,0.0,0.0);//stand object
glBegin(GL_POLYGON);
glVertex2f(130,10.0);
glVertex2f(160,10.0);
glVertex2f(160,180.0);
glVertex2f(130,180.0);
glEnd();
glBegin(GL_LINES);
glVertex2f(130,30.0);
glVertex2f(262,30.0);

glVertex2f(130,130.0);
glVertex2f(260,130.0);
glEnd();

glColor3f(0.8,0.498039 ,0.196078);
glBegin(GL_POLYGON);//core
        glVertex2f(237.5,20.0);
        glVertex2f(262.5,20.0);
        glVertex2f(262.5,120.0);
        glVertex2f(237.5,120.0);
glEnd();

glColor3f(1.0,1.0,1.0);//bonnet
glBegin(GL_POLYGON);//front
glVertex2f(237.5,120.0);
glVertex2f(262.5,120.0);
glVertex2f(250,170.0);
glEnd();
glColor3f(1.0,0.0,0.0);
glBegin(GL_POLYGON);//left_side_top
glVertex2f(237.5,120.0);
glVertex2f(217.5,95.0);
glVertex2f(237.5,95.0);
glEnd();
        glBegin(GL_POLYGON);//left_side_bottom
glVertex2f(237.5,20.0);
glVertex2f(217.5,20.0);
glVertex2f(237.5,70.0);
glEnd();
         glBegin(GL_POLYGON);//right_side_bottom
glVertex2f(262.5,20.0);
glVertex2f(282.5,20.0);
glVertex2f(262.5,70.0);
glEnd();
         glBegin(GL_POLYGON);//right_side_top
glVertex2f(262.5,120.0);
glVertex2f(262.5,95.0);
glVertex2f(282.5,95.0);
glEnd();
glColor3f(0.556863 ,0.137255 ,0.419608);
        glBegin(GL_POLYGON);//bottom_1_exhaust
```

```
        glVertex2f(237.5,20.0);
        glVertex2f(244.5,20.0);
        glVertex2f(241,0.0);
        glEnd();
                glBegin(GL_POLYGON);//bottom_2_exhaust
        glVertex2f(246.5,20.0);
        glVertex2f(253.5,20.0);
        glVertex2f(249.5,0.0);
        glEnd();
                glBegin(GL_POLYGON);//bottom_3_exhaust
        glVertex2f(262.5,20.0);
        glVertex2f(255.5,20.0);
        glVertex2f(258.5,0.0);
        glEnd();
        glBegin(GL_POLYGON);//left_stand_holder
        glVertex2f(182.5,85.0);
        glVertex2f(182.5,0.0);
        glVertex2f(187.5,0.0);
        glVertex2f(187.5,80.0);
        glVertex2f(237.5,80.0);
        glVertex2f(237.5,85.0);
        glVertex2f(182.5,85.0);
        glEnd();
        glBegin(GL_POLYGON);
glVertex2f(312.5,85.0);//right_stand_holder
        glVertex2f(312.5,0.0);
        glVertex2f(307.5,0.0);
        glVertex2f(307.5,80.0);
        glVertex2f(262.5,80.0);
        glVertex2f(262.5,85.0);
        glVertex2f(312.5,85.0);
        glEnd();

        for(j=0;j<=1000000;j++)
                        ;
        glutSwapBuffers();
        glutPostRedisplay();
        glFlush();
}
}
void rocket_to_cam_pos()
{
       count++;
count3++;

for(i=0;i<=200;i++)
{
        glClearColor(0.196078 ,0.6 ,0.8,1.0);
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

        glColor3f(0.8,0.498039 ,0.196078);
```

```
glBegin(GL_POLYGON);//core
        glVertex2f(237.5,20.0+i);
        glVertex2f(262.5,20.0+i);
        glVertex2f(262.5,120.0+i);
        glVertex2f(237.5,120.0+i);
glEnd();

glColor3f(1.0,1.0,1.0);//bonnet
glBegin(GL_POLYGON);//front
glVertex2f(237.5,120.0+i);
glVertex2f(262.5,120.0+i);
glVertex2f(250,170.0+i);
glEnd();
glColor3f(1.0,0.0,0.0);
glBegin(GL_POLYGON);//left_side_top
glVertex2f(237.5,120.0+i);
glVertex2f(217.5,95.0+i);
glVertex2f(237.5,95.0+i);
glEnd();
        glBegin(GL_POLYGON);//left_side_bottom
glVertex2f(237.5,20.0+i);
glVertex2f(217.5,20.0+i);
glVertex2f(237.5,70.0+i);
glEnd();
 glBegin(GL_POLYGON);//right_side_bottom
glVertex2f(262.5,20.0+i);
glVertex2f(282.5,20.0+i);
glVertex2f(262.5,70.0+i);
glEnd();
        glBegin(GL_POLYGON);//right_side_top
glVertex2f(262.5,120.0+i);
glVertex2f(262.5,95.0+i);
glVertex2f(282.5,95.0+i);
glEnd();
glColor3f(0.556863 ,0.137255 ,0.419608);
        glBegin(GL_POLYGON);//bottom_1_exhaust
glVertex2f(237.5,20.0+i);
glVertex2f(244.5,20.0+i);
glVertex2f(241,0.0+i);
glEnd();
        glBegin(GL_POLYGON);//bottom_2_exhaust
glVertex2f(246.5,20.0+i);
glVertex2f(253.5,20.0+i);
glVertex2f(249.5,0.0+i);
glEnd();
        glBegin(GL_POLYGON);//bottom_3_exhaust
glVertex2f(262.5,20.0+i);
glVertex2f(255.5,20.0+i);
glVertex2f(258.5,0.0+i);
glEnd();
```

```
if((p%2)==0)
    glColor3f(1.0,0.25,0.0);
        else
        glColor3f(1.0,0.816,0.0);

                glBegin(GL_POLYGON);//outer fume
         glVertex2f(237.5,20+i);
         glVertex2f(234.16,16.66+i);
         glVertex2f(230.82,13.32+i);
         glVertex2f(227.48,9.98+i);
         glVertex2f(224.14,6.64+i);
         glVertex2f(220.8,3.3+i);
         glVertex2f(217.5,0+i);
         glVertex2f(221.56,-5+i);
         glVertex2f(225.62,-10+i);
         glVertex2f(229.68,-15+i);
         glVertex2f(233.74,-20+i);
         glVertex2f(237.8,-25+i);
         glVertex2f(241.86,-30+i);
         glVertex2f(245.92,-35+i);
         glVertex2f(250,-40+i);
         glVertex2f(254.06,-35+i);
         glVertex2f(258.12,-30+i);
         glVertex2f(262.18,-25+i);
         glVertex2f(266.24,-20+i);
         glVertex2f(270.3,-15+i);
         glVertex2f(274.36,-10+i);
         glVertex2f(278.42,-5+i);
         glVertex2f(282.5,0+i);
         glVertex2f(278.5,4+i);
         glVertex2f(274.5,8+i);
         glVertex2f(270.5,12+i);
         glVertex2f(266.5,16+i);
         glVertex2f(262.5,20+i);//28 points
         glEnd();
                 if((p%2)==0)
         glColor3f(1.0,0.816,0.0);
         else
                 glColor3f(1.0,0.25,0.0);

         glBegin(GL_POLYGON);//inner fume
         glVertex2f(237.5,20+i);
         glVertex2f(236.5,17.5+i);
         glVertex2f(235.5,15+i);
         glVertex2f(234.5,12.5+i);
         glVertex2f(233.5,10+i);
         glVertex2f(232.5,7.5+i);
         glVertex2f(236,5+i);
         glVertex2f(239.5,2.5+i);
         glVertex2f(243,0+i);
         glVertex2f(246.5,-2.5+i);
```

```
                glVertex2f(250,-5+i);
                glVertex2f(253.5,-2.5+i);
                glVertex2f(257,0+i);
                glVertex2f(260.5,2.5+i);
                glVertex2f(264,5+i);
                glVertex2f(267.5,7.5+i);
                glVertex2f(266.5,10+i);
                glVertex2f(265.5,12.5+i);
                glVertex2f(264.5,15+i);
                glVertex2f(263.5,17.5+i);
                glVertex2f(262.5,20+i);//21 points

                glEnd();
                p=p+1
                    ;
          for(j=0;j<=1000000;j++)
                    ;
          glutSwapBuffers();
          glutPostRedisplay();
          glFlush();
}
}
void rocket_in_motion()
{
        count+=0.25;
for(i=195;i<=200;i++)
{
      if(count>=5)
         {
                          glClearColor(0.0 ,0.0 ,0.0,1.0);
         glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
         if(flag1==0)
         {
         stars();
         flag1=1;
         }
         else
         {
                 stars1();
                 flag1=0;
          }
          }
          else
           {
          glClearColor(0.196078 ,0.6 ,0.8,1.0);
         glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BI
         T);
           }
         if(count>=100)
         mars(20.0);
```

```
if(count<=130){
glColor3f(0.8,0.498039,0.196078);
glBegin(GL_POLYGON);//core
        glVertex2f(237.5,20.0+i);
        glVertex2f(262.5,20.0+i);
        glVertex2f(262.5,120.0+i);
        glVertex2f(237.5,120.0+i);
 glEnd();
 }
 if(count>=150)
 { static int k =
 i;
 glColor3f(1.0,1.0,1.0);//satellite
 glBegin(GL_POLYGON);//core
        glVertex2f(237.5,150.0+k);
        glVertex2f(252.5,150.0+k);
        glVertex2f(252.5,120.0+k);
        glVertex2f(237.5,120.0+k);
 glEnd();

 glBegin(GL_POLYGON);//side-panels
        glVertex2f(237.5,140.0+k);
        glVertex2f(230,140.0+k);
        glVertex2f(230,130.0+k);
        glVertex2f(237.5,130.0+k);

        glVertex2f(262.5,140.0+k);
        glVertex2f(227.5,140.0+k);
        glVertex2f(227.5,130.0+k);
        glVertex2f(262.5,130.0+k);
 glEnd();
 }
 else{
 glColor3f(1.0,1.0,1.0);//bonnet
 glBegin(GL_POLYGON);//front
 glVertex2f(237.5,120.0+i);
 glVertex2f(262.5,120.0+i);
 glVertex2f(250,170.0+i);
 glEnd();
  }
 if(count<=120){
 glColor3f(1.0,0.0,0.0);
 glBegin(GL_POLYGON);//left_side_top
 glVertex2f(237.5,120.0+i);
 glVertex2f(217.5,95.0+i);
 glVertex2f(237.5,95.0+i);
 glEnd();
          glBegin(GL_POLYGON);//left_side_bottom
 glVertex2f(237.5,20.0+i);
 glVertex2f(217.5,20.0+i);
 glVertex2f(237.5,70.0+i);
```

```
            glEnd();
                    glBegin(GL_POLYGON);//right_side_bottom
            glVertex2f(262.5,20.0+i);
            glVertex2f(282.5,20.0+i);
            glVertex2f(262.5,70.0+i);
            glEnd();
                    glBegin(GL_POLYGON);//right_side_top

            glVertex2f(262.5,120.0+i);
            glVertex2f(262.5,95.0+i);
            glVertex2f(282.5,95.0+i);

            glEnd();
            }
            if(count<=110){
            glColor3f(0.556863 ,0.137255 ,0.419608);
                    glBegin(GL_POLYGON);//bottom_1_exhaust
            glVertex2f(237.5,20.0+i);
            glVertex2f(244.5,20.0+i);
            glVertex2f(241,0.0+i);
            glEnd();
                    glBegin(GL_POLYGON);//bottom_2_exhaust
            glVertex2f(246.5,20.0+i);
            glVertex2f(253.5,20.0+i
            );
            glVertex2f(249.5,0.0+i);
            glEnd();
                    glBegin(GL_POLYGON);//bottom_3_exhaust
            glVertex2f(262.5,20.0+i);
            glVertex2f(255.5,20.0+i);
            glVertex2f(258.5,0.0+i);
            glEnd();
            }

            for(j=0;j<=1000000;j++)
                    ;
            glutSwapBuffers()
            ;
            glutPostRedisplay(
            ); glFlush();
}
}

void mars(float radius)
{
           glBegin(GL_POLYGON);
      for (int i=0; i<=359; i++)
            {
              float degInRad = i*DEG2RAD;
              glVertex2f(300+f+cos(degInRad)*radius,500-t+(sin(degInRad))
```

```
        *radius);//100,100 specifies centre of the circle
          }
          glEnd();
          t=t+0.1;
          f=f+0.1;
          }

        //keys that trigger manual Lanch
           void keyboard(unsigned char key, int x, int y)
{
         if (key == 'S' || key == 's'){
                        for(int i=0;i<300;i++)
                                static_rocket();
                        flag = 1;

          }

          if (key == 'L' || key == 'l')
          {
                        for(int i=0;i<700;i++)
                                static_rocket();
          }
                if (key == 'Q' || key == 'q')
                        exit(0);

        }
        //design of homescreen
        void page()
        {
                glColor3f(1, 1, 1);
                glLineWidth(5);
                glBegin(GL_LINE_LOO
                P); glVertex2d(75, 425);
                glVertex2d(375, 425);
                glVertex2d(375, 325);
                glVertex2d(75, 325);
                glEnd();

                drawstring(100, 400, "ROCKET LAUNCHING SIMULATION");
                drawstring(100, 380, "NAME : SAKEENA IRAM, SAHANA S M
                "); drawstring(100, 360, "USN : 4RA21CS074, 4RA21CS073");
                drawstring(100, 340, "SEM : VI");

                glBegin(GL_LINE_LOOP);
                glVertex2d(375, 225);
                glVertex2d(75, 75);
                glVertex2d(375, 75);
                glVertex2d(75, 225);
                glEnd();
```

```
                drawstring(100, 200, "INSTRUCTIONS");
                drawstring(100, 180, "Press S to START");
                drawstring(100, 160, "Press L to Launch
                Pad"); drawstring(100, 100, "Press Q to
                quit"); glFlush();
}
//display all components
void display()
{


        if (flag == 0)
        {
          glClear(GL_COLOR_BUFFER_BI
          T); page();
           glutSwapBuffers();
        }
        }
        else
                control();
        glFlush();
  }

  void myinit()
  {
            //int i;
            glClearColor(0.196078 ,0.6 ,0.8,1.0);


            glPointSize(1.0);
            gluOrtho2D(0.0,499.0,0.0,499.0);
  }
int main(int argc,char*argv[])
{
            glutInit(&argc,argv);
            glutInitDisplayMode(GLUT_DOUBLE|GLU
            T_RGB); glutInitWindowSize(600,600);
            glutCreateWindow("rocket");
            myinit();
            glutKeyboardFunc(keyboard);
            glutDisplayFunc(display);
            glutIdleFunc(display);


glutMainLoop();
return 0;
}
```

Chapter 07

TESTING AND RESULT

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

This game allows the gamer to overtake obstacles, and accomplish 5 levels of intense racing. It tells the user what will happen if they collide with an obstacle i.e. setting an example in real life. The levels automatically increment unless the vehicle has not crashed. 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.

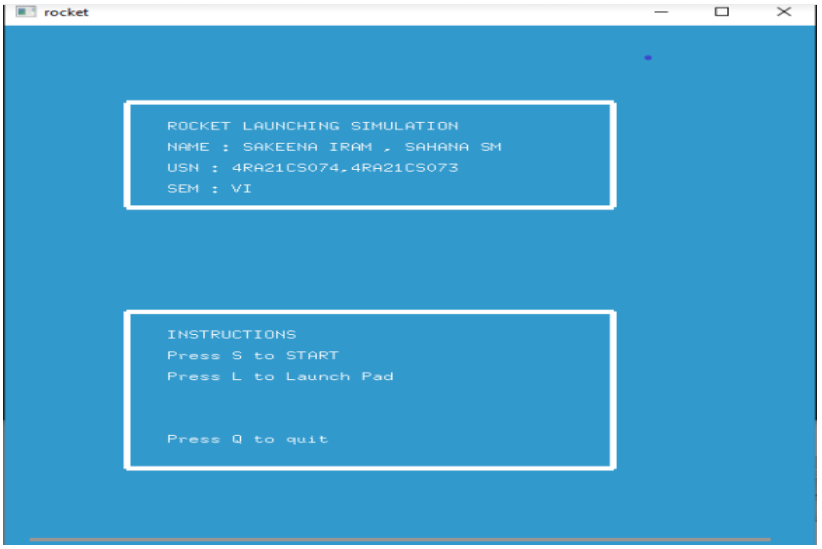
SYSTEM TESING

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 requirement.

Chapter 08

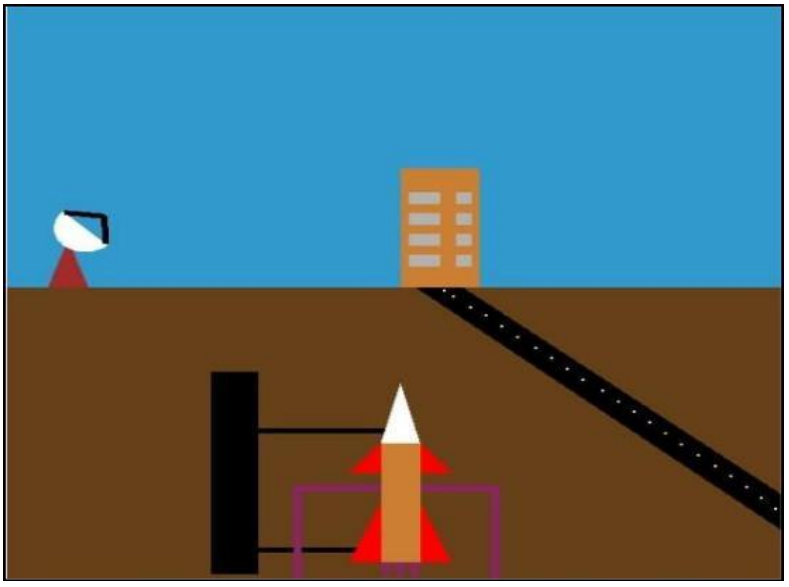
SNAPSHOTS

7.1 Control Instruction



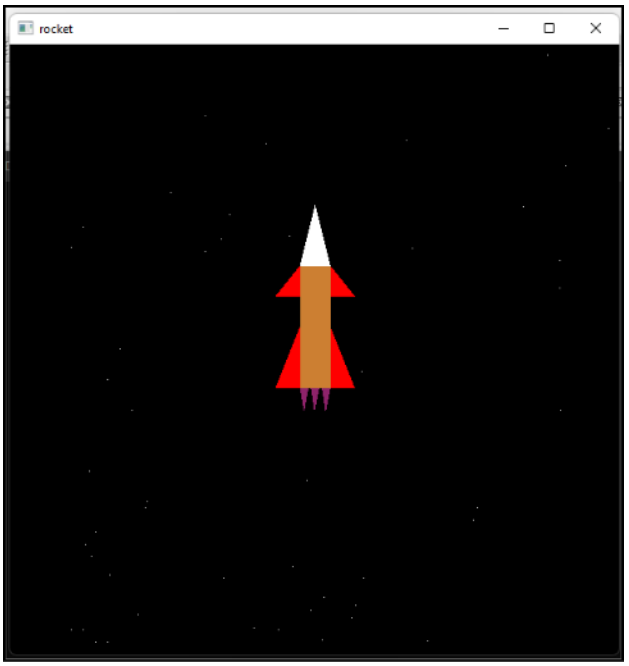
SNAPSHOT 7.1: Control Instruction

7.2 Launch pad



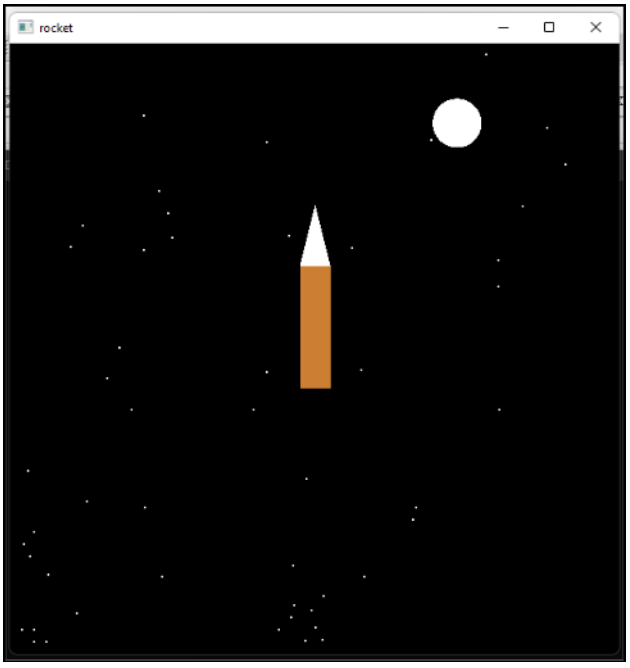
SNAPSHOT 7.2: Launch Pad

7.3 Stage 1



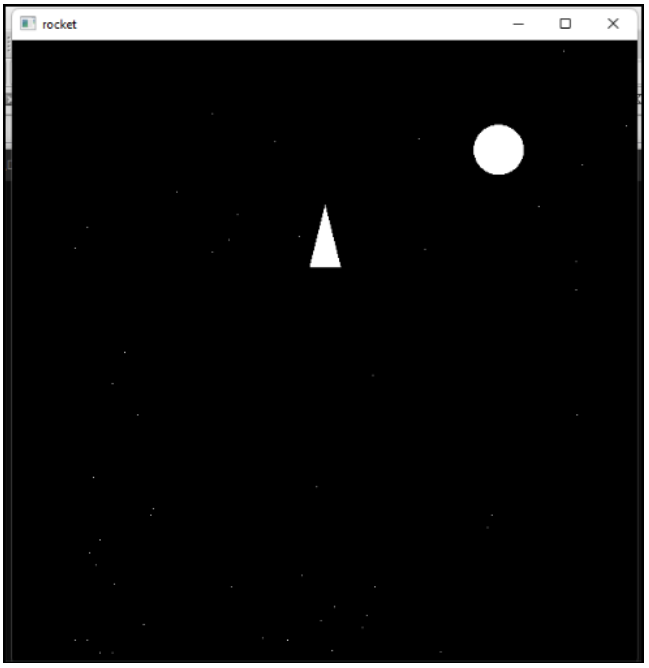
SNAPSHOT 7.3: Stage 1

7.4 Stage 2



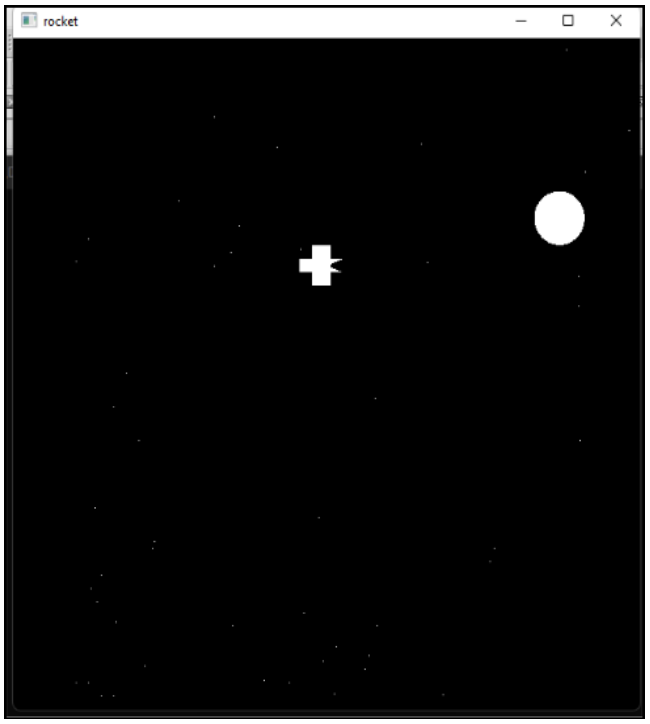
SNAPSHOT 7.4: Stage 2

7.5 Stage 3



SNAPSHOT 7.5:Stage 3

7.6 Satellite



SNAPSHOT 7.6: Satellite

Chapter 09

FUTURE SCOPE AND ENHANCEMENT

- **Educational Tools**: Rocket launch simulations can serve as advanced educational tools in schools and universities, helping students grasp complex principles of physics and engineering through interactive learning.
- **Training Programs**: Aerospace companies and space agencies can use these simulations for training engineers and astronauts, allowing them to practice and perfect launch sequences and troubleshooting in a risk-free environment.
- **Entertainment and Media**: High-quality simulations can enhance movies, video games,and virtual reality experiences, offering realistic depictions of space missions.
- **Public Outreach**: Space agencies can use simulations to engage and educate the public about space missions, increasing awareness and interest in space exploration.
- **Research and Development**: Simulations can aid in the R&D process, allowing scientists and engineers to model and test new rocket designs and launch procedures virtually before physical prototypes are constructed.
- **Commercial Space Ventures**: As commercial space travel becomes more prominent, simulations will be crucial for planning missions, training commercial astronauts, and optimizing launch procedures.
- **Advanced Visualization**: Incorporating AI and machine learning can enhance the predictive capabilities of rocket launch simulations, leading to more accurate and detailed visualizations.
- **Improved Physics Engines**: Enhance the accuracy of physics engines to better simulate forces, trajectories, and environmental interactions involved in rocket launches, aerodynamics, gravity, and atmospheric conditions.

CONCLUSION

In conclusion, the rocket launch simulation project represents a culmination of rigorous research, meticulous design, and advanced computational modeling aimed at replicating the complexities of real-world space missions. By integrating cutting-edge physics simulations, detailed environmental factors, and precise engineering parameters, the project not only enhances our understanding of rocketry but also serves as a robust educational tool and a valuable resource for aerospace professionals. With its ability to predict and analyze launch scenarios with accuracy and reliability, this simulation project stands as a testament to the ingenuity and collaborative effort required to push the boundaries of space exploration and technology development. Overall, the computer graphics rocket launch simulator project showcases the potential of blending technology and education, providing an engaging platform for users to explore and learn about the fascinating world of rocket launches. It serves as a valuable tool for students, space enthusiasts, and anyone interested in experiencing the thrill of space exploration from the comfort of their own computer.

REFERENCES

Books referred during this project work:

- Interactive Computer Graphics A top –down approach with OpenGL – Edward Angel, 5th Edition, Addition-Wesley 2008.
- Rocket Propulsion Elements" by George P. Sutton and Oscar Biblarz This book is a comprehensive guide to the principles of rocket propulsion. It covers the fundamental physics and engineering of rockets, making it a valuable resource for understanding the basics needed for simulation.
- Computer Graphics Using OpenGL - F.S. Hill. Jr. 2nd Edition, Pearson Education, 2001.

WEBSITES referred:

http://github.com
http://javapoint.com/xampp
http:// javapoint.com/opengl
http:// javapoint.com/html
www.google.com
www.openGL.com
www.w3schools.com
www.google.co.in
www.htmlguddies.com