**Computer Graphics (UCS505)**

# Project on
# Rocket Launching Using OpenGL.

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

**B.E. Third Year – COE 6**

**Submitted to:-**

**Dr.Raghav**

**Submitted by:-**

**Aman Aayush(102283049)**

**Gulika Agarwal(102103155)**

**Department of Computer Science & Engineering ,**

**Thapar Institute Of Engineering &Technology**

**Patiala,Punjab-147001**

## Table of Contents

## 1. Introduction:

In the quest for space exploration and scientific discovery, simulating rocket launches has become an integral part of aerospace engineering. The ability to visualize and analyze the dynamics of a rocket launch not only aids in the design and testing phases but also serves as a powerful educational tool. In this project, we delve into the realm of computer graphics to develop a simulation of a rocket launch.

Objective:-

The primary objective of this project is to create a realistic and immersive simulation of a rocket launch using computer graphics techniques. Through this simulation, we aim to provide insights into the various stages of a launch, including liftoff, ascent, stage separation, and payload deployment. Additionally, we seek to explore the intricacies of rocket physics, such as thrust, gravity, aerodynamics, and fuel consumption, to accurately model the behavior of the launch vehicle.

Use:-

Simulating rocket launches using computer graphics holds significant educational, scientific, and practical value. It enables students and enthusiasts to gain insights into the complexities of space exploration, fosters interest in STEM fields, and facilitates hands-on learning experiences. Moreover, it serves as a valuable tool for engineers and researchers in optimizing rocket designs, conducting virtual tests, and analyzing mission outcomes.

## 2. computer graphics Concepts

**OpenGL:** The code utilizes OpenGL, an open-source graphics library, for rendering 2D graphics.

1. **Coordinate Systems:** It uses 2D Cartesian coordinate systems to specify the positions of objects on the screen.
2. **Primitive Shapes:** The code draws primitive shapes such as polygons (for rocket, ground, houses), circles (for planets), and lines (for wires, road markings).
3. **Color:** It specifies colors using RGB values to create visually appealing graphics.
4. **Transformation:** The code uses translation to simulate motion, such as moving the rocket from the launch pad to the camera position.
5. **Animation:** Animation is achieved by repeatedly redrawing the scene with slight changes to create the illusion of motion (e.g., rocket launch, stars moving).
6. **Text Rendering:** It displays text on the screen to provide information and

instructions to the user.

7. **Event Handling:** It responds to user input events (keyboard input) to control the simulation flow (e.g., start, quit).
8. **Simulation:** The code simulates the launch of a rocket, including its movement, exhaust fumes, and interaction with the environment (e.g., ground, stars, Mars).
9. **Control Flow:** It manages the control flow of the program to transition between different states of the simulation (e.g., home screen, rocket launch, motion).

These concepts collectively contribute to creating an interactive and visually engaging rocket launching simulation.

## 3. User-defined functions

1. **drawstring***(int x, int y, char s):* Draws a string of characters at the specified coordinates (x, y) on the screen.
2. **semicircle(float radius, float u, float v):** Draws a semicircle with the specified radius and center coordinates (u, v).
3. **stars():** Draws a set of stars on the screen.
4. **stars1():** Draws another set of stars on the screen.
5. **static_rocket():** Renders the static scene of the rocket on the launch pad.
6. **rocket_to_cam_pos():** Simulates the transition of the rocket from the launch pad to the camera position.
7. **rocket_in_motion():** Simulates the motion of the rocket after launch.
8. **mars(float radius):** Draws the planet Mars with the specified radius.
9. **control():** Determines the state of the rocket launch and controls the sequence of events.
10. **page():** Designs the home screen with instructions for the simulation.

These functions handle various aspects of the simulation, including rendering objects, controlling motion, displaying text, and managing the overall flow of the program.

## 4. 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;

// fucntion to display the text content of the home screen
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();
}

//determines the state of rocket launch
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(1.37);
             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(1.0);
        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++;
```

```
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_BIT);
            }
            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){
```

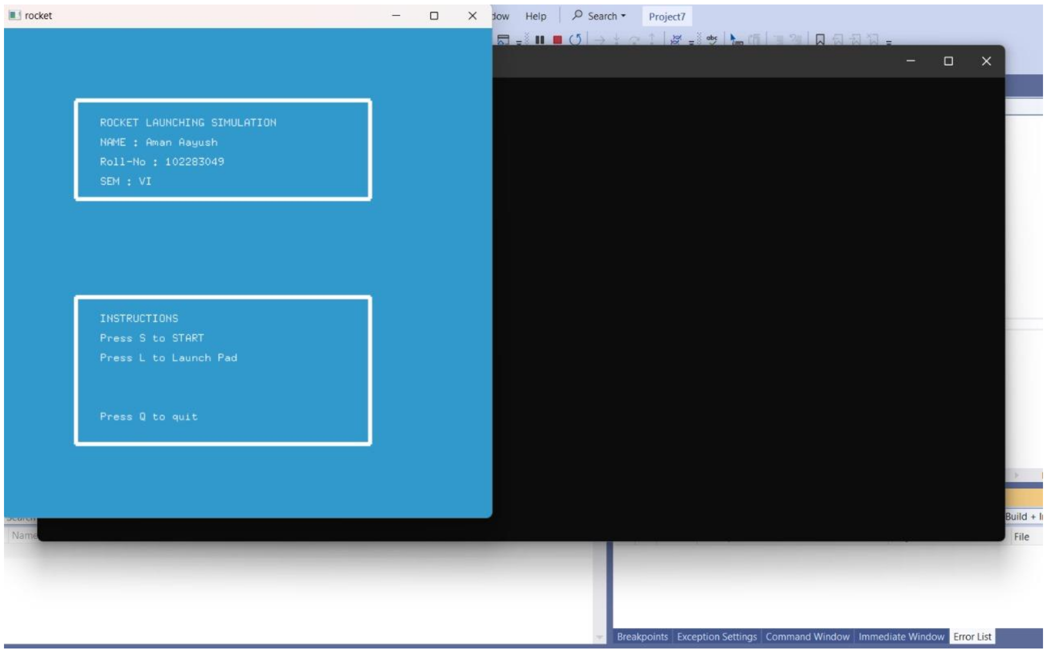
```
glColor3f(1.0,0.0,0.0);
glBegin(GL_POLYGON);//left_side_top
glVertex2f(237.5,120.0+i);
glVertex2f(217.5,95.0+i);
glVertex2f(237.5,95.0+i);
glEnd();
        glBegin(GL_POLYGON);//left_side_bottom
glVertex2f(237.5,20.0+i);
glVertex2f(217.5,20.0+i);
glVertex2f(237.5,70.0+i);
glEnd();
        glBegin(GL_POLYGON);//right_side_bottom
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<100;i++)
                        static_rocket();
                flag = 1;

        }

        if (key == 'L' || key == 'l')
        {
                for(int i=0;i<100;i++)
                        static_rocket();
        }

        if (key == 'Q' || key == 'q')
                exit(0);

}

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

        drawstring(100, 400, const_cast<char*>("ROCKET LAUNCHING SIMULATION"));
drawstring(100, 380, const_cast<char*>("NAME : DEBJYOTI GUHA"));
drawstring(100, 360, const_cast<char*>("USN : 1MJ16CS041"));
drawstring(100, 340, const_cast<char*>("SEM : VI"));
        glBegin(GL_LINE_LOOP);
        glVertex2d(75, 75);
        glVertex2d(375, 75);
        glVertex2d(375, 225);
        glVertex2d(75, 225);
        glEnd();

        drawstring(100, 200, const_cast<char*>("INSTRUCTIONS"));
drawstring(100, 180, const_cast<char*>("Press S to START"));
drawstring(100, 160, const_cast<char*>("Press L to Launch Pad"));
drawstring(100, 100, const_cast<char*>("Press Q to quit"));        glFlush();
}

//display all components
void display()
{
        if (flag == 0)
        {
                glClear(GL_COLOR_BUFFER_BIT);
                page();
                glutSwapBuffers();
        }
        else
                control();
        glFlush();
}


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

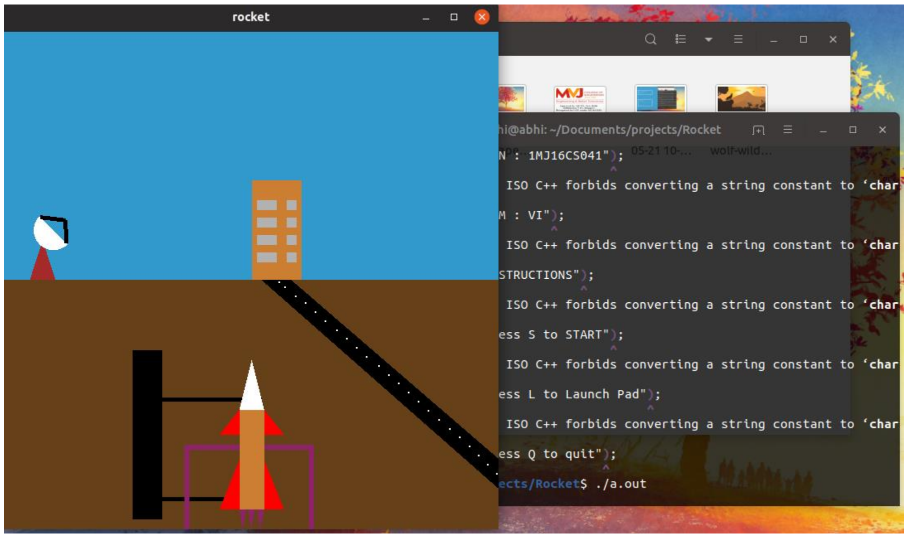
```
            glPointSize(1.0);
            gluOrtho2D(0.0,499.0,0.0,499.0);
}


int main(int argc,char*argv[])
{
            glutInit(&argc,argv);
            glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
            glutInitWindowSize(600,600);
            glutCreateWindow("rocket");
            myinit();
            glutKeyboardFunc(keyboard);
            glutDisplayFunc(display);
            glutIdleFunc(display);



  glutMainLoop();
  return 0;
}
```

## 5. Output/Screenshots

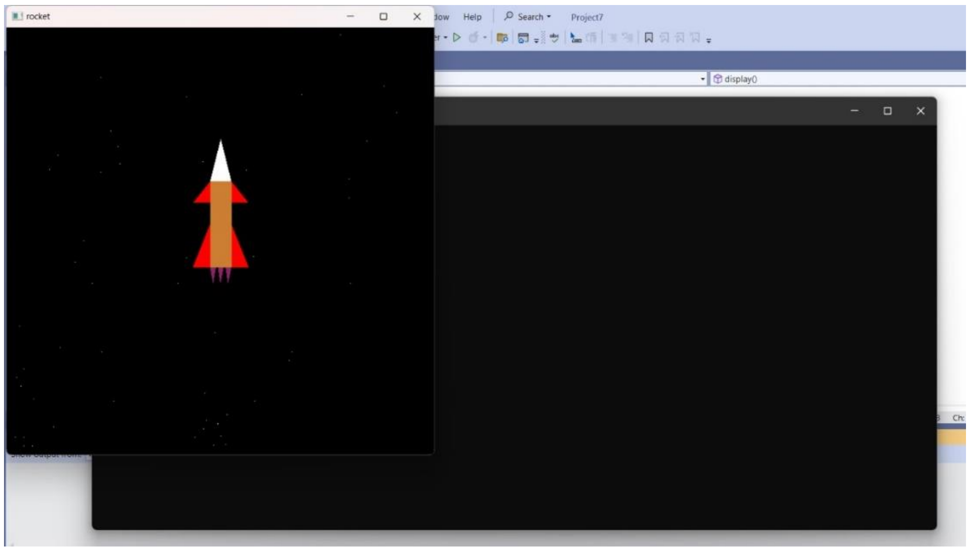


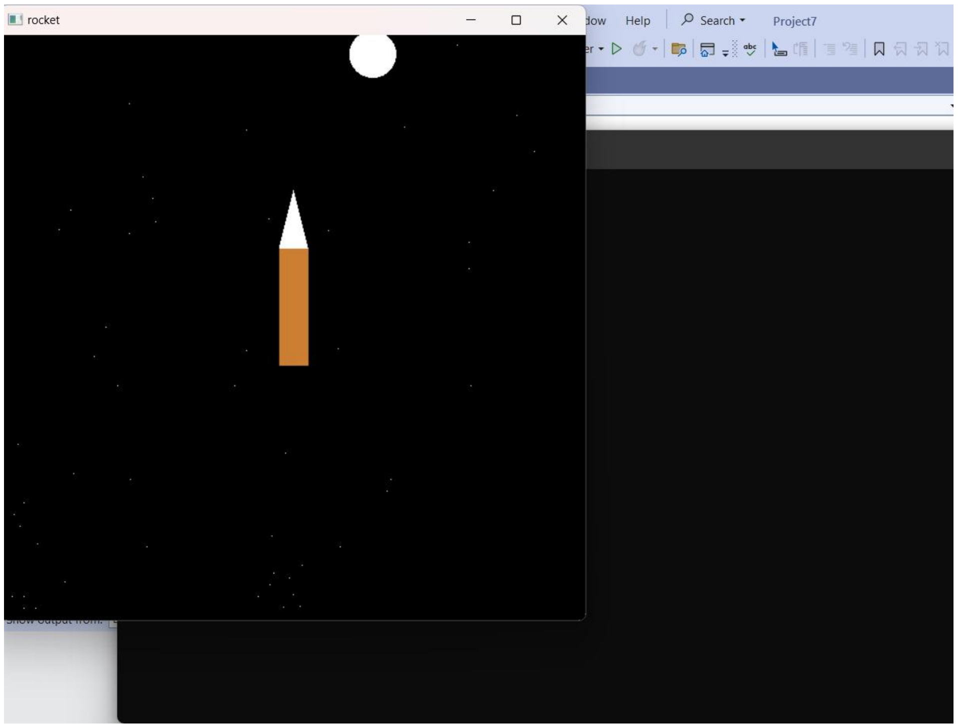
```
ROCKET LAUNCHING SIMULATION
NAME : Aman Aayush
Roll-No : 102283049
SEM : VI
```

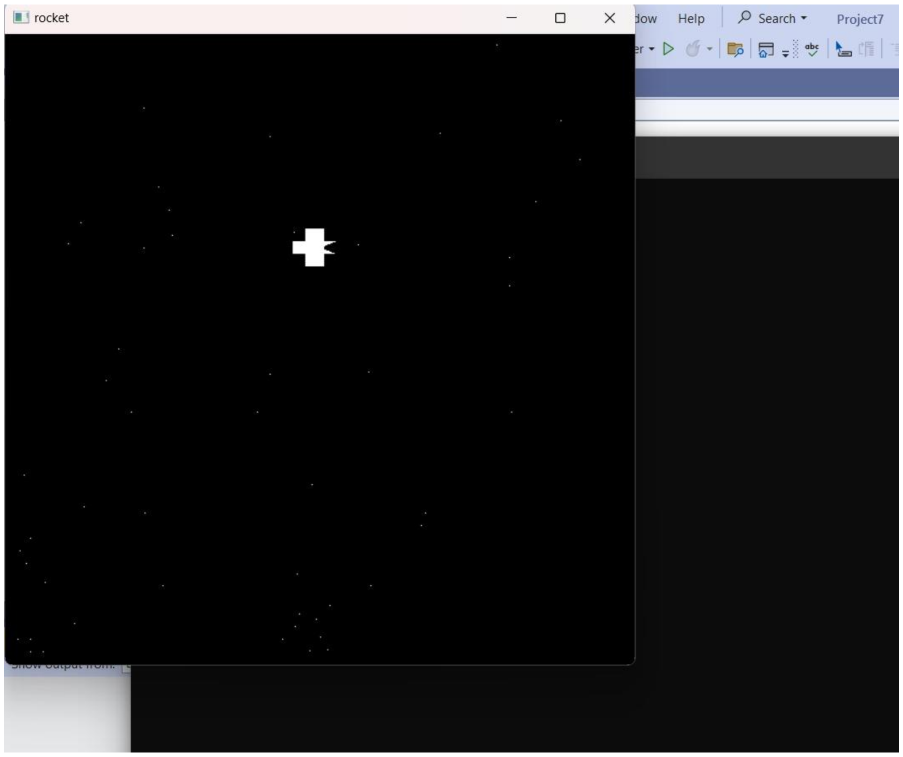
```
INSTRUCTIONS
Press S to START
Press L to Launch Pad


Press Q to quit
```



```
hi@abhi: ~/Documents/projects/Rocket
N : 1MJ16CS041");
                  ^
 ISO C++ forbids converting a string constant to ‘char
M : VI");
      ^
 ISO C++ forbids converting a string constant to ‘char
STRUCTIONS");
             ^
 ISO C++ forbids converting a string constant to ‘char
ess S to START");
                ^
 ISO C++ forbids converting a string constant to ‘char
ess L to Launch Pad");
                     ^
 ISO C++ forbids converting a string constant to ‘char
ess Q to quit");
               ^
ects/Rocket$ ./a.out
```


rocket
Help
Search
Project7
display()


rocket
Help
Search
Project7


rocket
dow Help Search Project7