NITTE EDUCATION TRUST | N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India
(ISO 9001:2015 Certified), Accredited with 'A' Grade by NAAC
☎: 08258 - 281039 – 281263, Fax: 08258 – 281265
Department of Computer Science and Engineering
B.E. CSE Program Accredited by NBA, New Delhi from 1-7-2018 to 30-6-2021

Report on Mini Project

"Rocket launch and Simulation"

Course Code: 18CS607

Course Name: Computer Graphics Lab

Semester: VI Section : C

Submitted To,

Mr. Puneeth RP

Asst Prof ,Dept of CSE,NMAMIT

Submitted By:

Name: Sachin Singh USN:4NM18CS142
Name: Sahana Kulal USN:4NM18CS144

Date of submission: **Signature of Course Instructor**

Rocket Launch Simulation

Abstract

Computer Graphics is a core technology in digital photography, film, video games, computer displays, and many specialized applications. Modelling, Geometric processing, rendering geometric transformation, visibility, simulation of light Animation of Lifelike characters, natural phenomena, their interactions and surrounding environments are some of the applications of Computer Graphics.

This project aims to implement a rocket launch simulation using OpenGL, which is an open-source cross platform interface for rendering 2D and 3D graphics. It will also allow user interactions using keyboard. Through this, we will be able to understand important OpenGL functions and GLUT libraries. This project also familiarizes us with implementing transformations like translation, rotation and scaling, properties of light, movement etc. on different objects.

Table of Contents:

CHAPTER 1

INTRODUCTION

Computer Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D pattern recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. 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. OpenGL, short for “Open Graphics Library” is an application programming interface (API) designed for rendering 2D and 3D graphics. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization and flight simulation. It is also used in video games, where it competes with direct 3D on Microsoft Windows Platforms It provides a common set of commands that can be used to manage graphics in different applications and on multiple platforms.

OpenGL serves for two main purposes,

- To hide the complexities of interfacing with different 3D accelerators, by presenting programmer with a single, uniform API.

- To hide the differing capabilities of hardware platforms, by requiring that all implementations support the full OpenGL feature set.

OpenGL has historically been influential on the development of 3D accelerator, promoting a base level of functionality that is now common in consumer level hardware.

The Project is mainly divided into two part i.e, in first part when the loading of rocket is not complete an error is displayed on the screen and the rocket doesn't take off and asks the user to load and launch the rocket into space. When the rocket has completely exhausted is fuel from adjoining tanks, then the extra weight is lost by disconnecting them. When the final capsule reached the orbiting location the last part is also disconnected and the satellite is placed on a planet. We have used input device keyboard to interact with the program.

This project is developed in C/C++ in Code Blocks.

CHAPTER 2

IMPLEMENTATION DETAILS

The project "Rocket Simulation" is meant as a source of recreation where one can sit in front of the computer and have the vision of a rocket launching in space. This package is developed to provide a visualization of the rocket launch in a simple 2D manner. It is aimed to create stars and a planet, where the satellite at the end lands on the planet.

This chapter documents a detailed description of the implementation of our project. We have incorporated several inbuilt OpenGL functions in this project.

a) SYNTAX AND EXPLAINATION:

The header files used are:

1. **#include <stdlib.h>:** This is C library function for standard input and output.
2. **#include<GL/glut.h>**: This header is included to read the glut.h, gl.h and glu.h.
3. **#include<math.h>**: This is a C library function for performing certain mathematical operations.

Init() – This functions initializes OpenGL and performs initial operations. The following functions are used in init() function:

- **glClearColor(. . .):** Whenever we wish to draw a new frame, the window must be cleared by using the 4-dimensional(RGBA)color system. The above function must be enabled to make the window on screen solid and white.

- **myinit()**: Here we initialize the color buffer, set the point size and set window co-ordinate values.

- **display():** This function creates and translates all the objects in a specified location in a particular order.

- **glutPostRedisplay():** It ensures that display will be drawn only once each time program goes through the event loop.

- **glutMainLoop():** This function whose execution will cause the program to begin an event processing loop. In the main function the following OpenGL functions are present:

- **glutInit(&argc, argv)** – It is used to initialize the GLUT library. 'argc' is a pointer to the program's unmodified argc variable from main. Upon return, the value pointed to by argc will be updated, because glutInit extracts any command line options intended for the GLUT library. 'argv' is the program's unmodified argv variable from main. Like argc, the data for argv will be updated because glutInit extracts any command line options understood by the GLUT library.

- **glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_ DEPTH)** – This specifies the type of display mode when creating a window.

- **glutInitWindowSize and glutInitWindowPosition** - defines the size and position (upper left corner) of the window on the screen. If omitted, the default window size is 300x300 and its placement is left up to the window manager.

- **glutCreateWindow(“ “)** – This creates a top-level window. The name will be provided to the window system as the window's name. The intent is that the window system will label the window with the name.

- **glutDisplayFunc( )** – This sets the display call back for the current window. When GLUT determines that the normal plane for the window needs to be redisplayed, the display call back for the window is called.

- **glutReshapeFunc( )** – This sets the reshape call back for the current window. The reshape call back is triggered when a window is reshaped. A reshape call back is also triggered immediately before a window's first display call back after a window is created or whenever an overlay for the window is established.

- **glutKeyboardFunc( )** – This sets the keyboard call back for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard call back. The key call back parameter is the generated ASCII character. The state of modifier keys such as Shift cannot be determined directly; their only effect will be on the returned ASCII data.

- **glutMainLoop ()** – This tells the program to enter the GLUT event processing loop. This just means the program sits and waits for things to happen, such as window refreshes, window resizes, mouse clicks, key presses, etc. glutMainLoop never exits, so it should always be the last line of the main program.

For the implementation of this project we use functions namely

i) **Static rocket()**
ii) **rocket_to_cam_pos()**
iii) **rocket_in_motion()**

The other functions used for detailing are:

i) **stars()**
ii) **mars()**

- **Static rocket()** is used to display the rocket without motion with the backround of buildings and a clear blue sky.

- **Rocket_to_cam_pos()** is used to show the rocket launching,the exhausts are displayed and the change of backround from blue to black.

- **Rocket_in_motion()** is used to display the splitting of the rocket in space.

- **Mars()** is used to display the planet onto the screen.

- **Stars()** is used to display the stars onto the screen.

And the function **control()** is used to control the timing of these various function using flag and count.

Pseudo Code:

```
void control11()
{
   count12++;
   if(count12==250)
         flag11=1;

   else if (flag11 == 1 && (count12 == 600 || count12 == 601))
         rocket_to_cam_pos11();

   else if (flag11 == 1 && count12 >= 1000)
         rocket_in_motion11();
}

void static_rocket()
{
count12++;
if(count12==300)
flag11=1;
 if(flag11==0)
 {
        glClearColor(0.196078  ,0.6 ,0.8,1.0);
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
```

```
        glColor3f(0,0.5,0);
    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);//w1
                    glVertex2f(255,267.5);
                    glVertex2f(275.0,267.5);
                    glVertex2f(275.0,277.5);
                    glVertex2f(255.0,277.5);
                    glEnd();
glColor3f(1,1,1);
            glBegin(GL_POLYGON);//w2
                    glVertex2f(255,285.0);
                    glVertex2f(275.0,285);
                    glVertex2f(275.0,295);
                    glVertex2f(255.0,295);
                    glEnd();
glColor3f(1.0,0.8,0.0);
            glBegin(GL_POLYGON);//w3
                    glVertex2f(255,302.5);
                    glVertex2f(275.0,302.5);
                    glVertex2f(275.0,312.5);
                    glVertex2f(255.0,312.5);
                    glEnd();
glColor3f(0.0,0.0,0.2);
```

```
            glBegin(GL_POLYGON);//w4
                        glVertex2f(255,320.0);
                        glVertex2f(275.0,320.0);
                        glVertex2f(275.0,330.0);
                        glVertex2f(255.0,330.0);
                        glEnd();
glColor3f(1.0,0.8,0.0);
   glBegin(GL_POLYGON);//w5
                        glVertex2f(285,267.5);
                        glVertex2f(295.0,267.5);
                        glVertex2f(295.0,277.5);
                        glVertex2f(285.0,277.5);
                        glEnd();
glColor3f(1.0,0.8,0.0);
            glBegin(GL_POLYGON);//w6
                        glVertex2f(285,285.0);
                        glVertex2f(295.0,285);
                        glVertex2f(295.0,295);
                        glVertex2f(285.0,295);
                        glEnd();
glColor3f(1,1,1);
            glBegin(GL_POLYGON);//w7
                        glVertex2f(285,302.5);
                        glVertex2f(295.0,302.5);
                        glVertex2f(295.0,312.5);
                        glVertex2f(285.0,312.5);
                        glEnd();
glColor3f(0.0,0.0,0.2);
            glBegin(GL_POLYGON);//w8
                        glVertex2f(285,320.0);
                        glVertex2f(295.0,320.0);
                        glVertex2f(295.0,330.0);
                        glVertex2f(285.0,330.0);
                        glEnd();

glColor3f(1,0.5 ,0.3);
glBegin(GL_POLYGON);//house 2
   glVertex2f(310.0,250.0);
   glVertex2f(360.0,250.0);
   glVertex2f(360.0,300.0);
   glVertex2f(310.0,300.0);
   glEnd();

            glColor3f(0.7,0.7,0.7);
            glBegin(GL_POLYGON);//w1
                        glVertex2f(315,267.5);
                        glVertex2f(335.0,267.5);
                        glVertex2f(335.0,277.5);
                        glVertex2f(315.0,277.5);
                        glEnd();
glColor3f(1,1,1);
            glBegin(GL_POLYGON);//w2
```

```
                glVertex2f(315,285.0);
                glVertex2f(335.0,285);
                glVertex2f(335.0,295);
                glVertex2f(315.0,295);
                glEnd();
glColor3f(1.0,0.8,0.0);
        glBegin(GL_POLYGON);//w3
                glVertex2f(345,267.5);
                glVertex2f(355.0,267.5);
                glVertex2f(355.0,277.5);
                glVertex2f(345.0,277.5);
                glEnd();
glColor3f(0.0,0.0,0.2);
        glBegin(GL_POLYGON);//w4
                glVertex2f(345,285.0);
                glVertex2f(355.0,285);
                glVertex2f(355.0,295);
                glVertex2f(345.0,295);
                glEnd();



glColor3f(0.5,0.2 ,0.0);
                glBegin(GL_POLYGON);//house 3
        glVertex2f(190.0,250.0);
        glVertex2f(240.0,250.0);
        glVertex2f(240.0,375.0);
        glVertex2f(190.0,375.0);
        glEnd();
        glColor3f(0.7,0.7,0.7);
        glBegin(GL_POLYGON);//w1
                glVertex2f(195,267.5);
                glVertex2f(215.0,267.5);
                glVertex2f(215.0,277.5);
                glVertex2f(195.0,277.5);
                glEnd();
glColor3f(1.0,0.8,0.0);
        glBegin(GL_POLYGON);//w2
                glVertex2f(195,285.0);
                glVertex2f(215.0,285);
                glVertex2f(215.0,295);
                glVertex2f(195.0,295);
                glEnd();
glColor3f(0.0,0.0,0.2);
        glBegin(GL_POLYGON);//w3
                glVertex2f(195,302.5);
                glVertex2f(215.0,302.5);
                glVertex2f(215.0,312.5);
                glVertex2f(195.0,312.5);
                glEnd();
glColor3f(1,1,1);
        glBegin(GL_POLYGON);//w4
```

```
                glVertex2f(195,320.0);
                glVertex2f(215.0,320.0);
                glVertex2f(215.0,330.0);
                glVertex2f(195.0,330.0);
                glEnd();

   glBegin(GL_POLYGON);//w5
                glVertex2f(195,345.0);
                glVertex2f(215.0,345.0);
                glVertex2f(215.0,355.0);
                glVertex2f(195.0,355.0);
                glEnd();
glColor3f(1.0,0.8,0.0);
        glBegin(GL_POLYGON);//w6
                glVertex2f(225,267.5);
                glVertex2f(235.0,267.5);
                glVertex2f(235.0,277.5);
                glVertex2f(225.0,277.5);
                glEnd();

        glBegin(GL_POLYGON);//w7
                glVertex2f(225,285.0);
                glVertex2f(235.0,285);
                glVertex2f(235.0,295);
                glVertex2f(225.0,295);
                glEnd();
glColor3f(1,1,1);
        glBegin(GL_POLYGON);//w8
                glVertex2f(225,302.5);
                glVertex2f(235.0,302.5);
                glVertex2f(235.0,312.5);
                glVertex2f(225.0,312.5);
                glEnd();
glColor3f(1.0,0.8,0.0);
        glBegin(GL_POLYGON);//w9
                glVertex2f(225,320.0);
                glVertex2f(235.0,320.0);
                glVertex2f(235.0,330.0);
                glVertex2f(225.0,330.0);
                glEnd();
glColor3f(0.0,0.0,0.2);
glBegin(GL_POLYGON);//w10
                glVertex2f(225,345.0);
                glVertex2f(235.0,345.0);
                glVertex2f(235.0,355.0);
                glVertex2f(225.0,355.0);
                glEnd();

         glColor3f(0.2,0.7 ,0.3);
                glBegin(GL_POLYGON);//house 4
        glVertex2f(370.0,250.0);
        glVertex2f(420.0,250.0);
```

```
            glVertex2f(420.0,395.0);
            glVertex2f(370.0,395.0);
            glEnd();
            glColor3f(0.7,0.7,0.7);
            glBegin(GL_POLYGON);//w1
                  glVertex2f(375,267.5);
                  glVertex2f(395.0,267.5);
                  glVertex2f(395.0,277.5);
                  glVertex2f(375.0,277.5);
                  glEnd();
glColor3f(1.0,0.8,0.0);
            glBegin(GL_POLYGON);//w2
                  glVertex2f(375,285.0);
                  glVertex2f(395.0,285);
                  glVertex2f(395.0,295);
                  glVertex2f(375.0,295);
                  glEnd();
glColor3f(0.0,0.0,0.2);
            glBegin(GL_POLYGON);//w3
                  glVertex2f(375,302.5);
                  glVertex2f(395.0,302.5);
                  glVertex2f(395.0,312.5);
                  glVertex2f(375.0,312.5);
                  glEnd();
glColor3f(1.0,0.8,0.0);
            glBegin(GL_POLYGON);//w4
                  glVertex2f(375,320.0);
                  glVertex2f(395.0,320.0);
                  glVertex2f(395.0,330.0);
                  glVertex2f(375.0,330.0);
                  glEnd();

   glColor3f(1,1,1);
   glBegin(GL_POLYGON);//w5
                  glVertex2f(375,345.0);
                  glVertex2f(395.0,345.0);
                  glVertex2f(395.0,355.0);
                  glVertex2f(375.0,355.0);
                  glEnd();
glColor3f(0.0,0.0,0.2);
glBegin(GL_POLYGON);//w6
                  glVertex2f(375,370.0);
                  glVertex2f(395.0,370.0);
                  glVertex2f(395.0,380.0);
                  glVertex2f(375.0,380.0);
                  glEnd();
glColor3f(0.0,0.0,0.2);
            glBegin(GL_POLYGON);//w7
                  glVertex2f(405,267.5);
                  glVertex2f(415.0,267.5);
                  glVertex2f(415.0,277.5);
                  glVertex2f(405.0,277.5);
```

```
                glEnd();
glColor3f(1.0,0.8,0.0);
        glBegin(GL_POLYGON);//w8
                glVertex2f(405,285.0);
                glVertex2f(415.0,285);
                glVertex2f(415.0,295);
                glVertex2f(405.0,295);
                glEnd();
glColor3f(1,1,1);
        glBegin(GL_POLYGON);//w9
                glVertex2f(405,302.5);
                glVertex2f(415.0,302.5);
                glVertex2f(415.0,312.5);
                glVertex2f(405.0,312.5);
                glEnd();
glColor3f(0.0,0.0,0.2);
        glBegin(GL_POLYGON);//w10
                glVertex2f(405,320.0);
                glVertex2f(415.0,320.0);
                glVertex2f(415.0,330.0);
                glVertex2f(405.0,330.0);
                glEnd();
glColor3f(1.0,0.8,0.0);
glBegin(GL_POLYGON);//w11
                glVertex2f(405,345.0);
                glVertex2f(415.0,345.0);
                glVertex2f(415.0,355.0);
                glVertex2f(405.0,355.0);
                glEnd();

glColor3f(1,1,1);
glBegin(GL_POLYGON);//w11
                glVertex2f(405,370.0);
                glVertex2f(415.0,370.0);
                glVertex2f(415.0,380.0);
                glVertex2f(405.0,380.0);
                glEnd();


                 glColor3f(0.9,0.5 ,0.1);
glBegin(GL_POLYGON);//house 5
   glVertex2f(130.0,250.0);
   glVertex2f(180.0,250.0);
   glVertex2f(180.0,300.0);
   glVertex2f(130.0,300.0);
   glEnd();
   glColor3f(0.0,0.0,0.2);
   glBegin(GL_POLYGON);//w1
                glVertex2f(135,267.5);
                glVertex2f(155.0,267.5);
                glVertex2f(155.0,277.5);
                glVertex2f(135.0,277.5);
```

```
                glEnd();
glColor3f(1.0,0.8,0.0);
        glBegin(GL_POLYGON);//w2
                glVertex2f(135,285.0);
                glVertex2f(155.0,285);
                glVertex2f(155.0,295);
                glVertex2f(135.0,295);
                glEnd();
glColor3f(1,1,1);
        glBegin(GL_POLYGON);//w3
                glVertex2f(165,267.5);
                glVertex2f(175.0,267.5);
                glVertex2f(175.0,277.5);
                glVertex2f(165.0,277.5);
                glEnd();
glColor3f(1.0,0.8,0.0);
        glBegin(GL_POLYGON);//w4
                glVertex2f(165,285.0);
                glVertex2f(175.0,285);
                glVertex2f(175.0,295);
                glVertex2f(165.0,295);
                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();


glColor3f(1.0,1.0,0.5);
drawSUN(75,450,25);

        for(j=0;j<=1000000;j++)
                ;
        glutSwapBuffers();
        glutPostRedisplay();
        glFlush();
        Sleep(1);
}

}


void rocket_to_cam_pos11()
{
        count11++;
count13++;

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((p11%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((p11%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();
        p11=p11+1;
for(j=0;j<=1000000;j++)
        ;
glutSwapBuffers();
glutPostRedisplay();
```

```
	glFlush();
}

void rocket_in_motion11()
{
	count11++;


for(i=195;i<=200;i++)
{
   if(count11>=5)
	 {
			glClearColor(0.0 ,0.0 ,0.0,1.0);
	glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
	if(flag12==0)
	{
	stars11();
	flag12=1;
	}
	else
	{
		stars12();

		flag12=0;
	}

	 }
	 else
	 {
	glClearColor(0.196078  ,0.6 ,0.8,1.0);
	glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
	 }
	 if(count11>=100)
	 mars11(40.0);

	 if(count11<=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(count11>=150){
	static int k = i;
	glColor3f(1.0,0.0,0.0);//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(count11<=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(count11<=110){
glColor3f(0.556863 ,0.137255  ,0.419608);
```

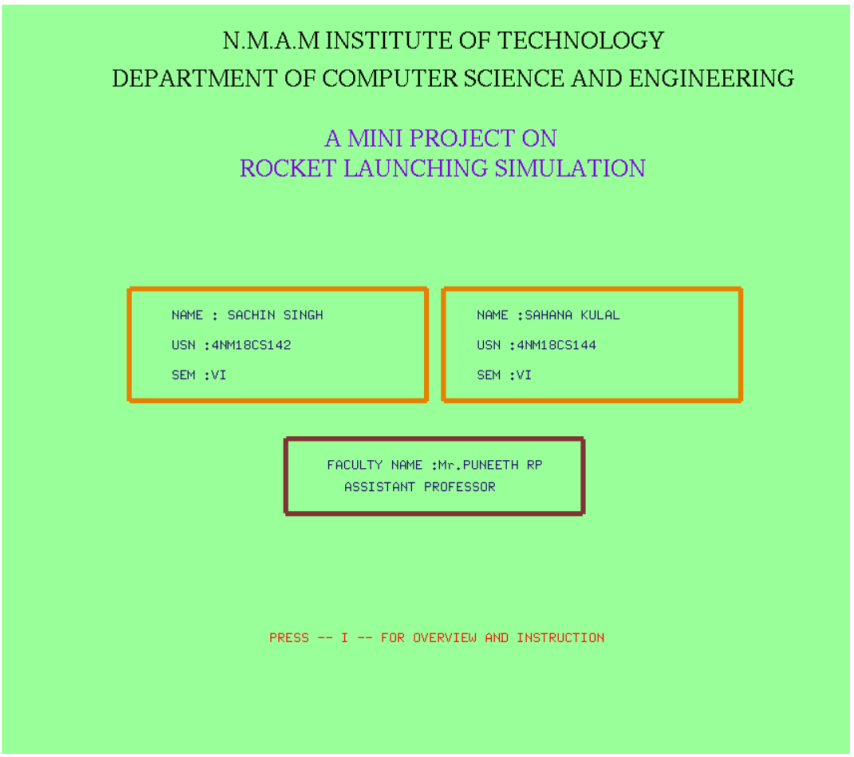
```
        glBegin(GL_POLYGON);//bottom_1_exhaust
    glVertex2f(237.5,20.0+i);
    glVertex2f(244.5,20.0+i);
    glVertex2f(241,0.0+i);
    glEnd();
        glBegin(GL_POLYGON);//bottom_2_exhaust
    glVertex2f(246.5,20.0+i);
    glVertex2f(253.5,20.0+i);
    glVertex2f(249.5,0.0+i);
    glEnd();
        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();
}
}
```

CHAPTER 3

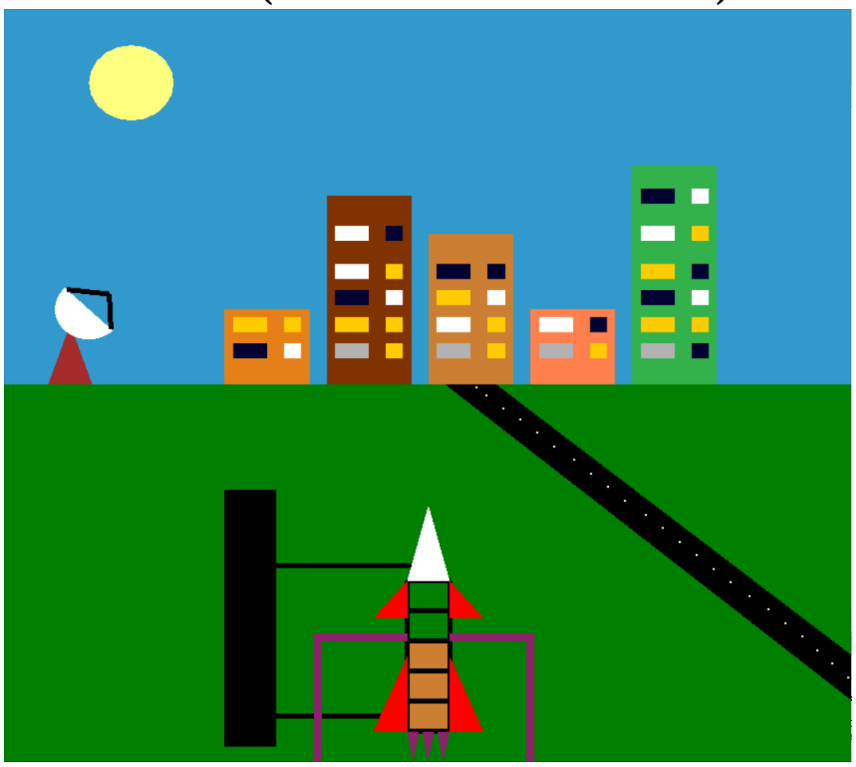
SCREENSHOTS

HOMESCREEN

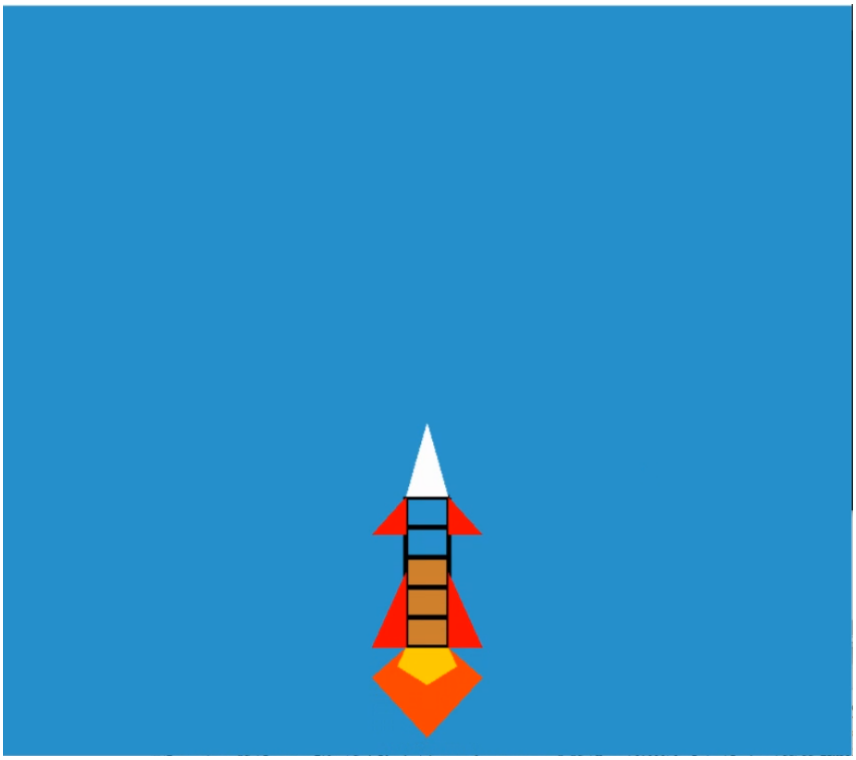


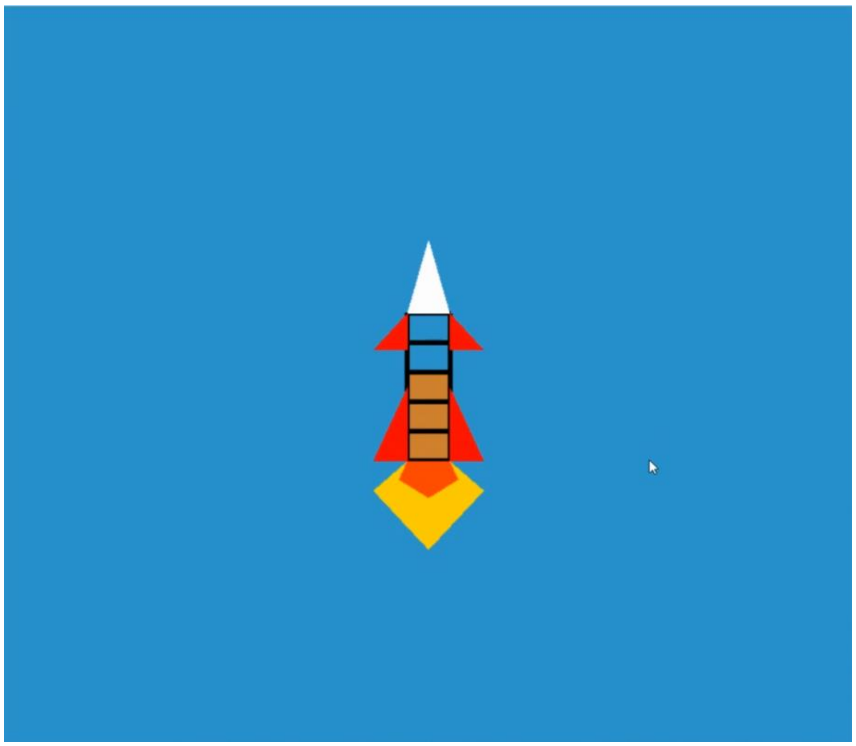
OVERVIEW AND INSTRUCTIONS

PART-1(INCOMPLETE LOADING)

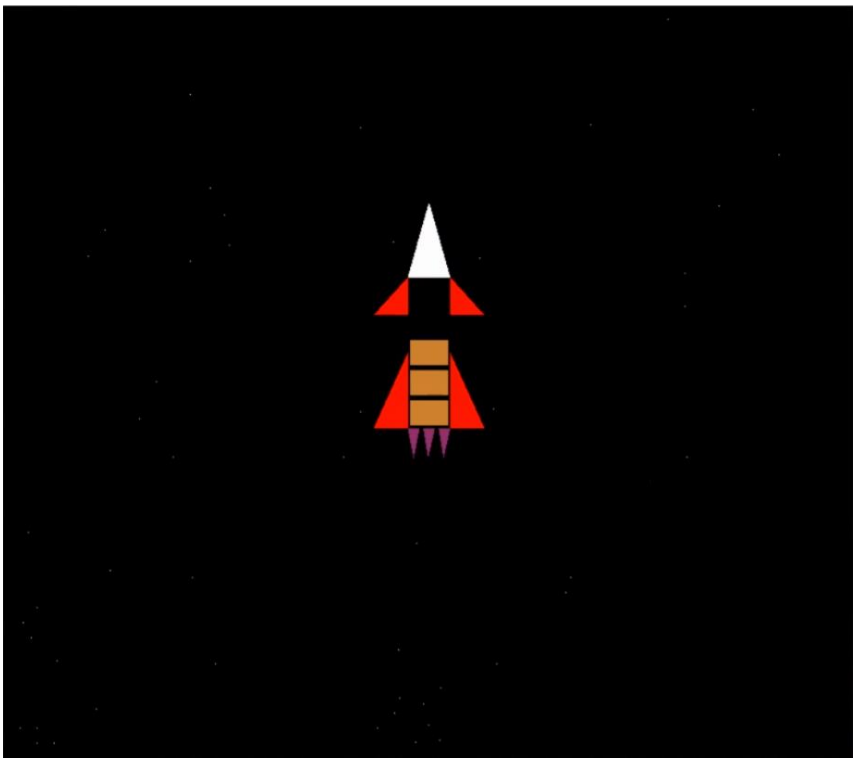


LAUNCH OF ROCKET WITH INCOMPLETE LOADING

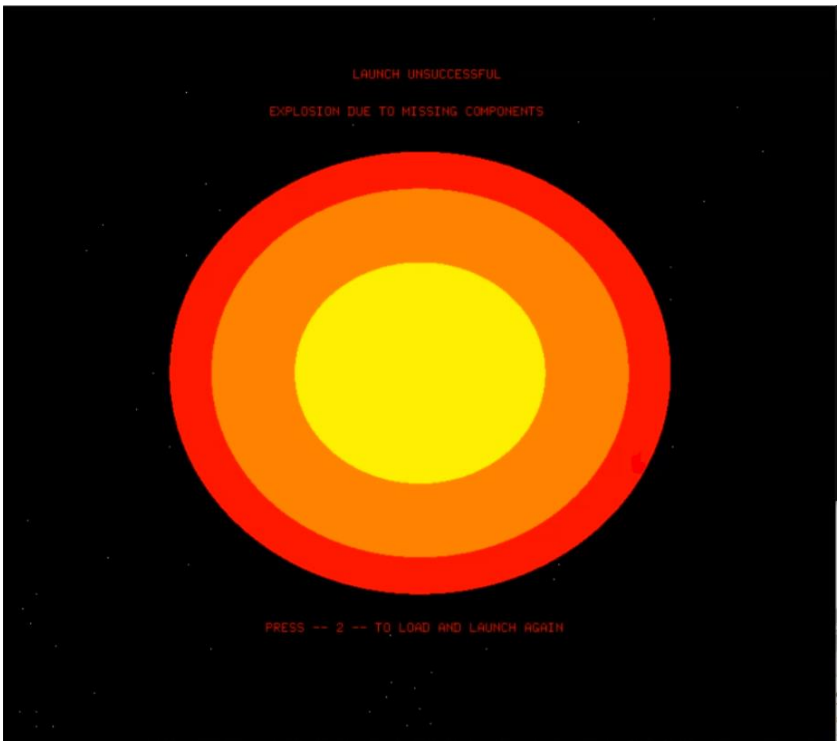




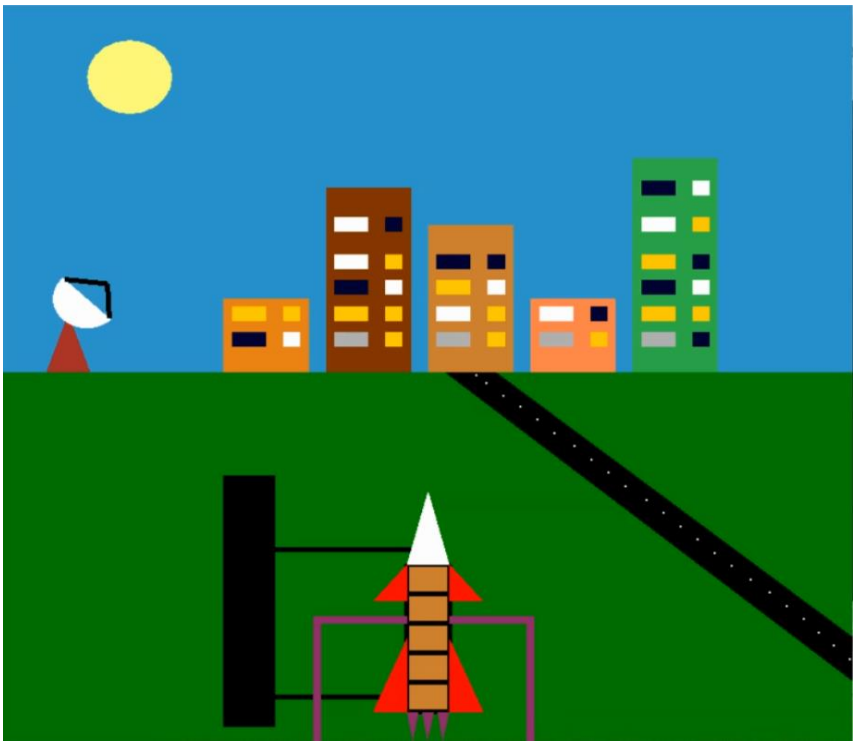
ENTERING INTO SPACE



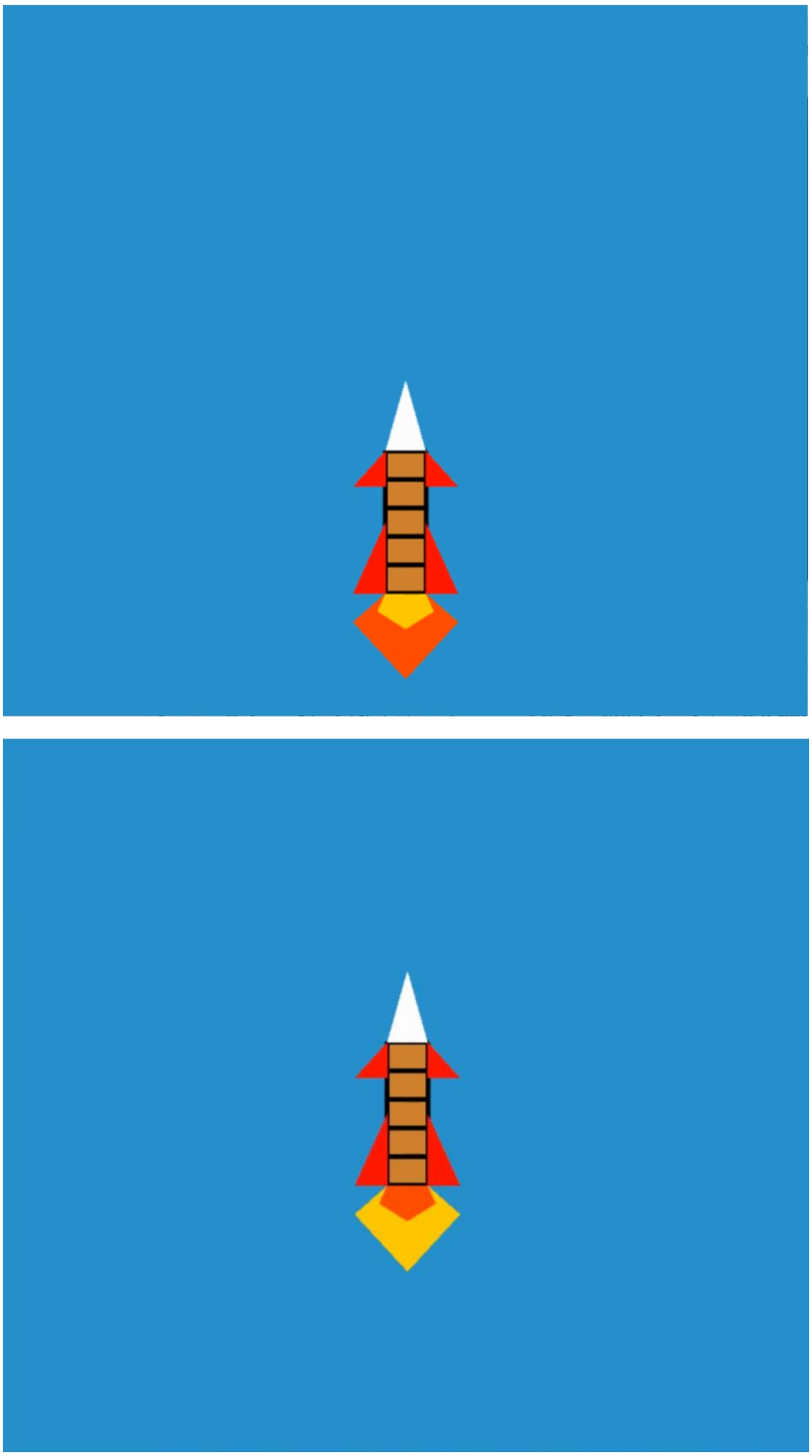
EXPLOSION DUE TO INCOMPLETE LOADING



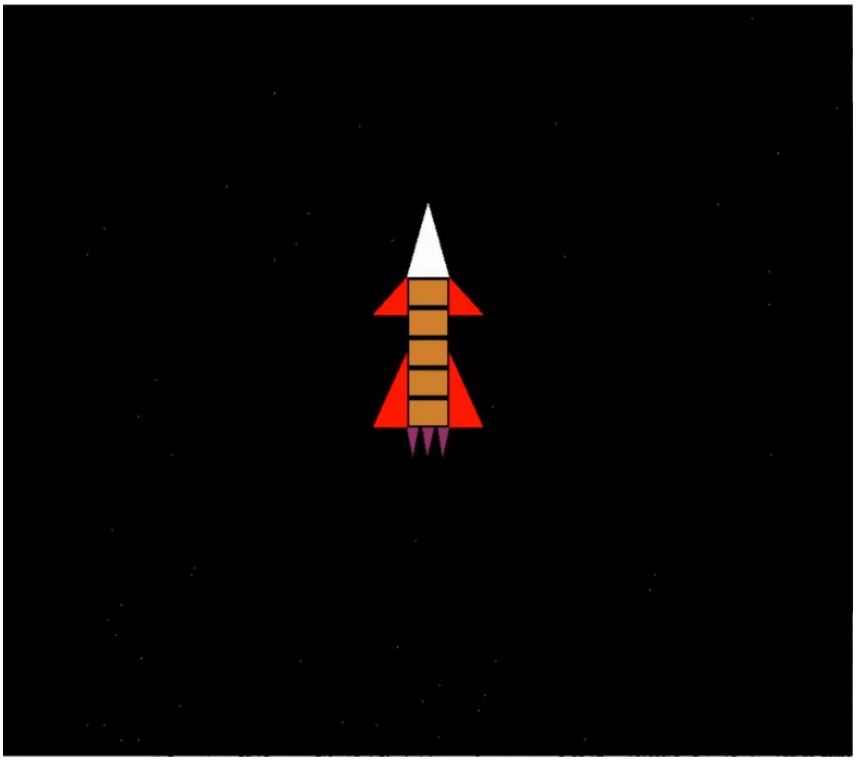
PART-1(COMPLETE LOADING)



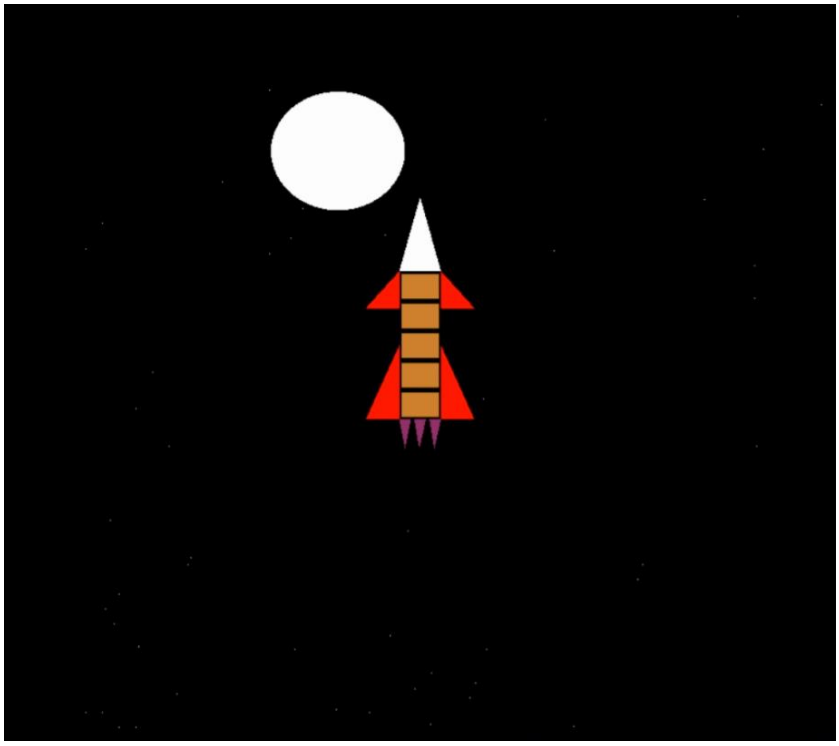
LAUNCH OF ROCKET WITH INCOMPLETE LOADING



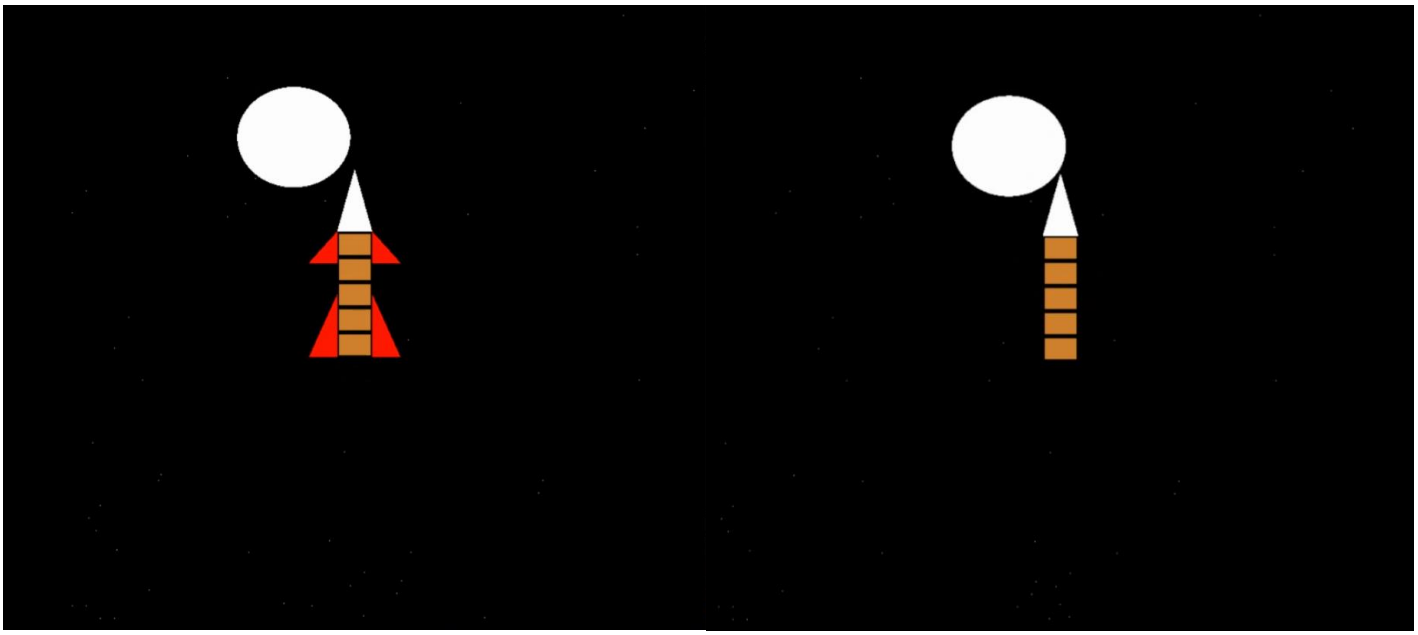
ENTERING INTO SPACE

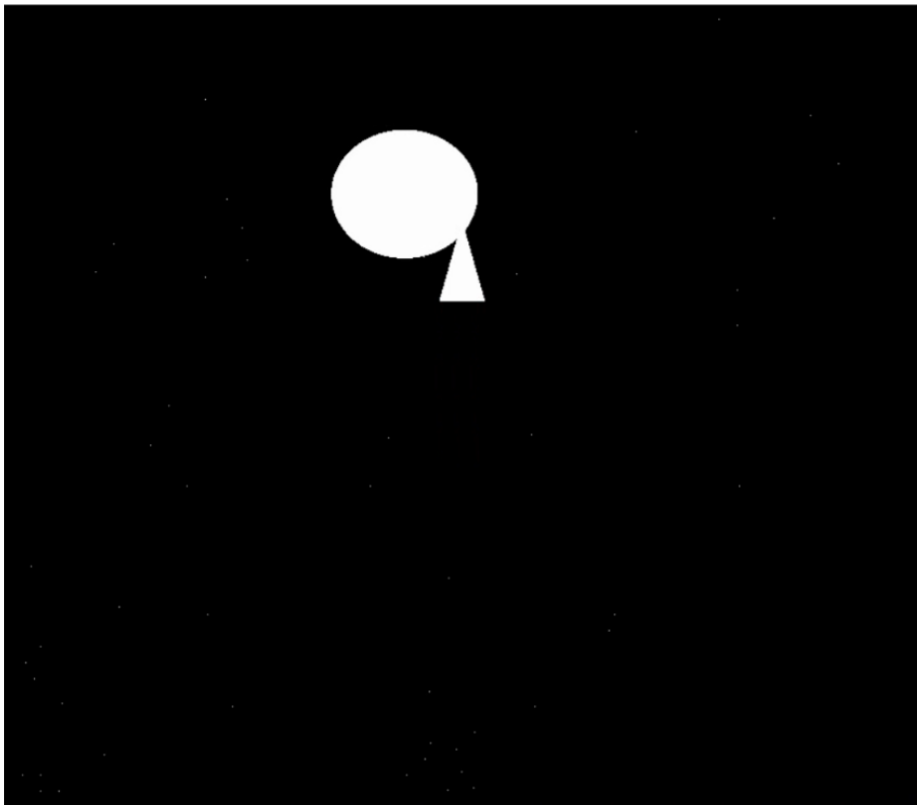


REACHING MARS

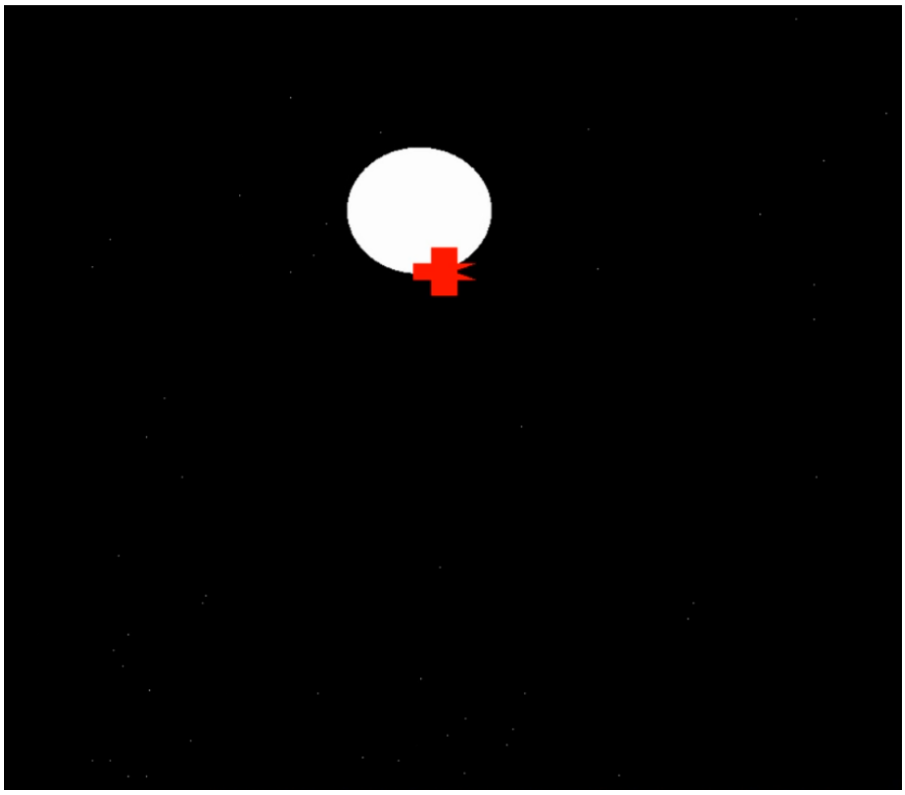


SPLITTING OF ROCKET INTO STAGES

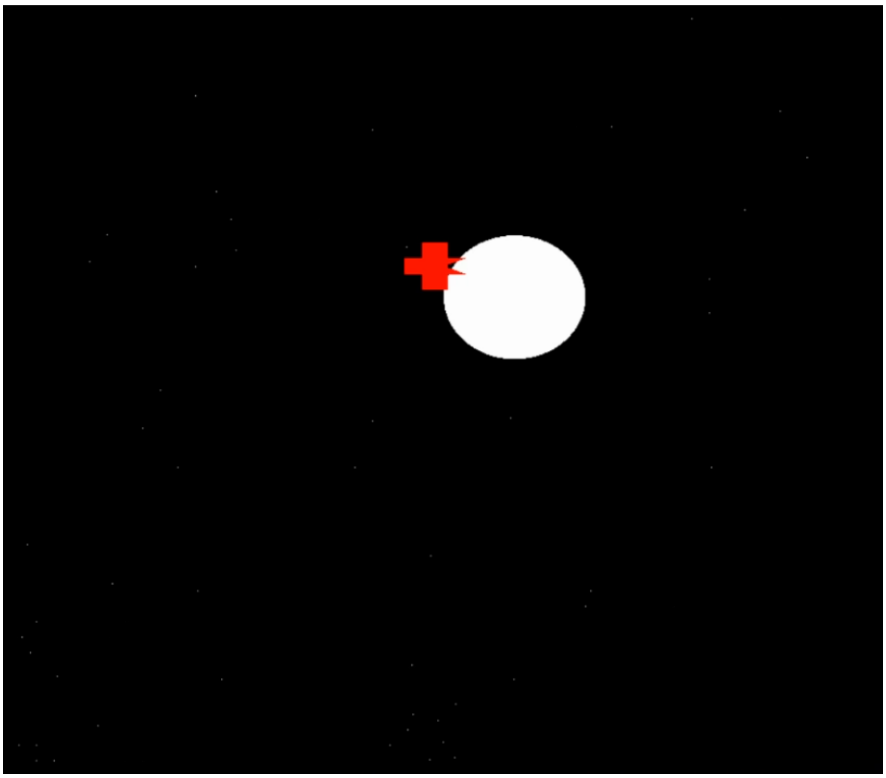




SATELITE REVOLVING AROUND PLANET



LANDING OF SATELITE ON PLANET



CHAPTER 4

CONCLUSION

The conclusion here – is that the application of this project i.e where can this project be used.

This project would be helpful in developing game front ends. Another major use is that since rocket launching is very sophisticated, we can use it as a pre launch preparation for scientists to analyze the different problems that can occur during an actual launch.

CHAPTER 5

REFERENCES

1. GitHub
2. YouTube
3. Stack Overflow