American International University-Bangladesh (AIUB)

Faculty of Science and Technology

Department of Computer Science

CSC 3224: Computer Graphics

Project Report

Project name: Project Space

Section: L Group: 04

Submitted To

NOBORANJAN DEY

Submitted by:

NAME	ID
ABIDUR RAHMAN	23-50413-1
MD. AMIR HOSSAEN	23-50433-1
MD. KAMRUL HASAN	23-50738-1
MD. ASICKUZZAMAN	23-50985-1

Date of Submission : 15 September, 2025

Project Title: Project Space

Feature/ Component:

- Rocket Launch ,stops, and reverses
- Atmosphere Transition
- Celestial Objects
- Space Journey
- Moon Surface
- Rocket Landing
- Exploration Scene
- Drifting clouds
- Falling meteors
- Rotating Earth
- Moving satellite
- Astronauts raising flag of Bangladesh and banner of AIUB

Tools and Technologies:

- OpenGL
- GLUT
- C++
- Graphics Primitives (Lines, Polygons, Circles, Text and so on)
- Geogebra

Work distribution:

ABIDUR RAHMAN	23-50413-1	• **Scenario 3, 5: Rocket launch into space,Moon landing** Cloud, outer space, stars,satellite,rocket
MD. AMIR HOSSAEN	23-50433-3	• **Scenario 1:Introduction** Stars, moon, rocket, texts
MD. KAMRUL HASAN	23-50738-1	• **Scenerio 2: Space Journey** AIUB(Annex and D-building, rocket launce pad, roads, busses, clous,sun, pond, tree • **Scenerio 4: Outer space** Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, and Neptune, Rocket, satellite, mateore drop,stars
MD. ASICKUZZAMAN	23-50985-1	• **Scenerio 3: Rocket Landing** Satellite, stars • **Scenerio 5: Moon landing** Astronauts raising flag of Bangladesh and banner of AIUB

Introduction:

Our project focuses on the implementation of a computer graphics–based simulation that illustrates the key stages of a space exploration mission, specifically a rocket launch, space travel, and moon landing sequence. The system employs fundamental graphics techniques such as 2D and 3D transformations, scaling, translation, and rotation to animate the rocket's motion, while texture mapping, shading, and lighting effects are used to enhance the visual realism of the lunar surface and surrounding space environment. The background star field and moving celestial objects provide a sense of depth and immersion, making the simulation more engaging.

The project was implemented with the objective of demonstrating the practical application of core computer graphics concepts—rasterization, transformations, animation, scene rendering, and lighting—in simulating real-world scientific phenomena. By integrating these techniques, we aimed to create not just a technical demonstration but also an interactive and visually appealing representation of one of humanity's most iconic achievements: the journey from Earth to the Moon.

The significance of this project lies in its dual role as both an educational and inspirational tool. On one hand, it provides students and learners with a hands-on example of how graphics concepts are applied in practice, thereby strengthening their technical understanding. On the other hand, it highlights the historical and scientific importance of space exploration, sparking curiosity and appreciation for scientific advancement.

The target population for this project includes computer science and engineering students studying computer graphics, educators seeking interactive teaching tools, and space enthusiasts who wish to explore the concepts of rocket launches and lunar missions in a simplified yet visually rich manner. Through this project, we demonstrate how the fusion of creativity and technical knowledge can be used to simulate complex real-world events for educational, scientific, and entertainment purposes.

Tools Used in the Project

1. **glTranslatef(x,y,z)**
   This function moves (translates) an object from one position to another in 3D space. The parameters (x, y, z) specify how far the object is shifted along the X, Y, and Z axes. In your project, it is used to animate objects like stars, meteors, and the rocket by changing their positions smoothly.
2. **glScalef(x,y,z)**
   This function scales (resizes) an object along the X, Y, and Z directions. A value greater than 1 enlarges the object, while a value less than 1 shrinks it. You used it to adjust the relative size of stars, meteors, and other shapes to fit the scene properly.
3. **glColor3f(r,g,b)**
   This sets the current drawing color using RGB values (between 0.0 and 1.0). Every shape drawn afterward takes this color until another color is specified. You used it to color the background, stars, and meteor effects.

4. **glBegin()/glEnd()**
   These mark the start and end of drawing a primitive shape (like GL_POLYGON, GL_QUADS, GL_POINTS). Inside them, you define vertices (glVertex2f / glVertex3f) that form the shape.
5. **glVertex2f(x,y)/glVertex3f(x,y,z)**
   Defines the coordinates of a point (vertex) in 2D or 3D space. Multiple vertices combine to form polygons, stars, or other complex shapes in your simulation.
6. **glutBitmapCharacter(font,char)**
   This function renders a character (text) at the current raster position using a given bitmap font. You used it in renderBitmapString() to display text messages in the scene.
7. **glutPostRedisplay()**
   Marks the current window as needing to be redisplayed. It ensures animations update continuously by forcing the display() function to redraw the scene.
8. **glutTimerFunc(milliseconds,function,value)**
   Registers a timer callback function that executes after a specified interval (in ms). You used it for animations like the meteor movement and object bouncing.
9. **`glPushMatrix()` / `glPopMatrix()`**

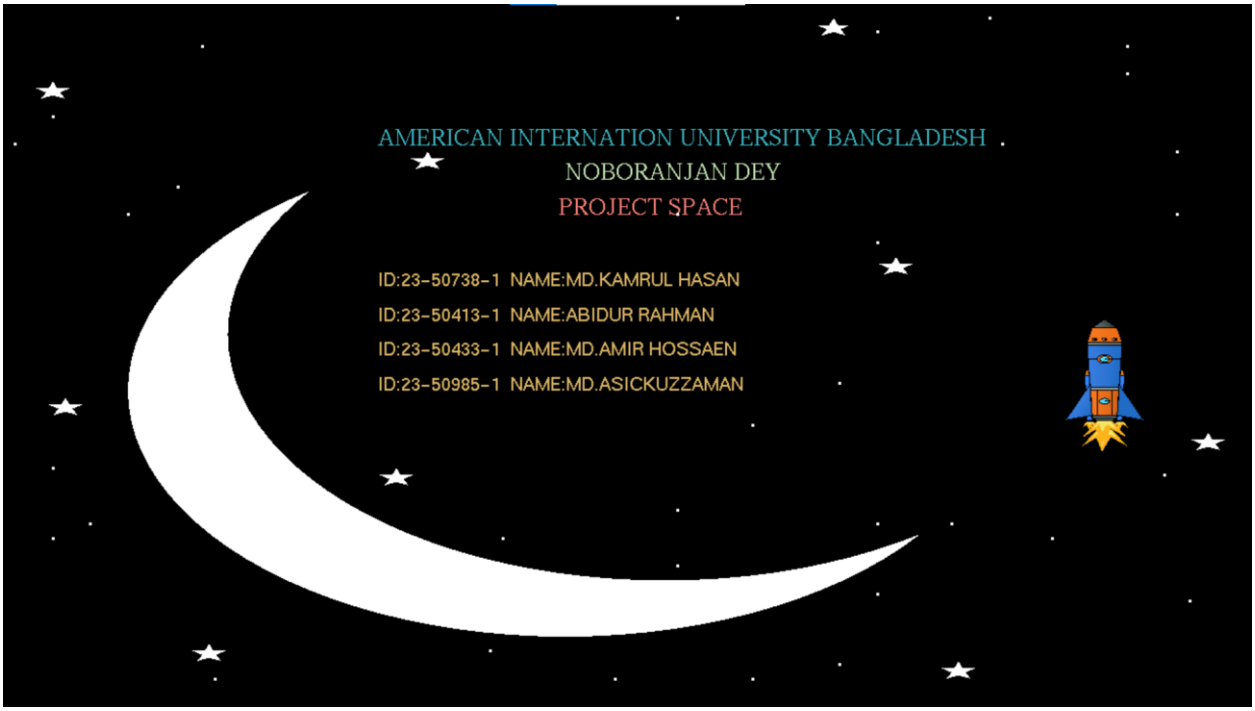
Save and restore the current transformation state. Used so that cloud translations and rocket scaling don't affect other objects in the scene.

10. **glPushMatrix() / glPopMatrix()**
    These functions save and restore the current transformation state. They are used so that scaling or translating clouds and rocket parts does not affect other objects in the scene.
11. **cos(angle) / sin(angle)**
    Trigonometric functions used inside loops to calculate circular vertex positions.
    Combined with **GL_TRIANGLE_FAN**, they create circular cloud layers and rocket windows efficiently.
12. **GL_QUADS**
    Draws rectangular shapes such as the rocket's main body and side wings.
13. **GL_TRIANGLES**
    Draws pointed shapes like the rocket's nose cone and angular wing tips.
14. **GL_POLYGON**
    Creates irregular shapes, for example, flames beneath the rocket boosters.
15. **GL_POINTS**
    Adds small points within clouds to give a textured, detailed effect.
16. **GL_LINES + glLineWidth()**
    Draws borders and outlines around the rocket's body, wings, and windows to make the structure sharper and more defined.
17. **GL_LINE_LOOP**
    Used to draw borders/outlines for rocket windows (top, bottom, mini).
    Circles generated with cos(angle)/sin(angle) inside loops.
18. **GL_POLYGON**
    Draws rocket flames with irregular, pointed shapes.
    Multiple colors layered for realistic flame effect.

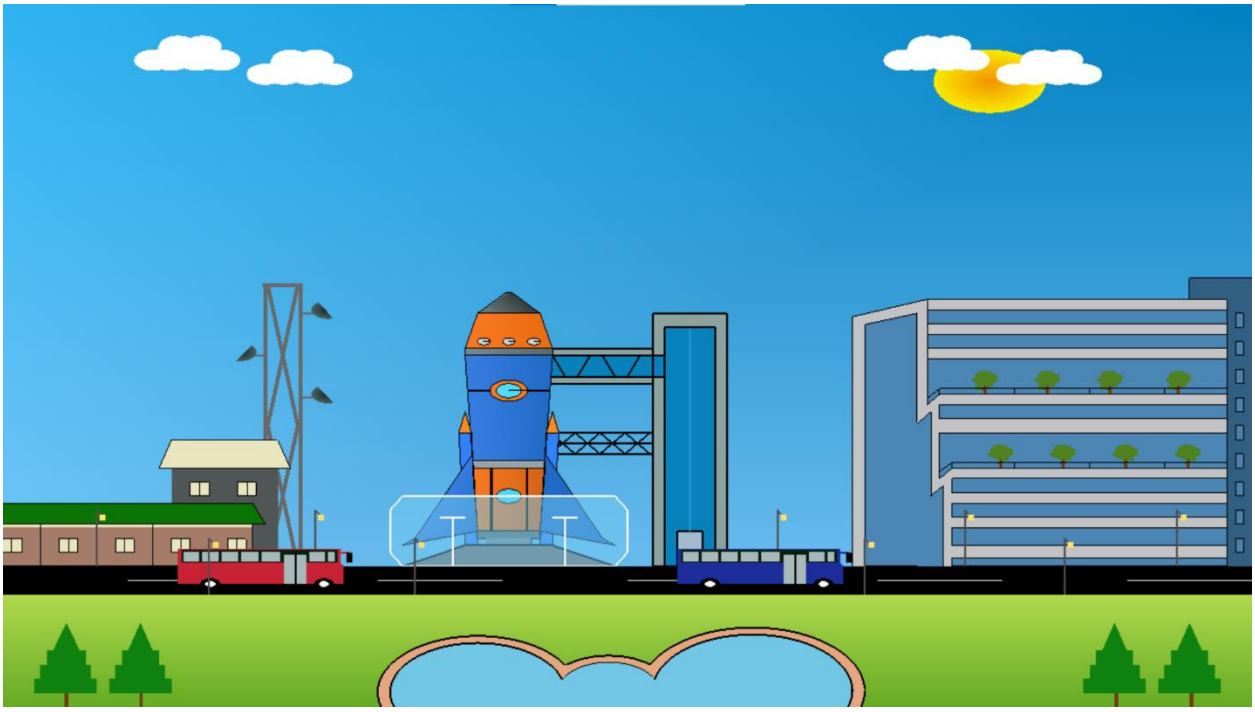
19. **GL_TRIANGLE_FAN** (for clouds)
    Creates filled circular cloud layers.
    Multiple circles of different radii and positions overlap to give fluffy clouds.
20. **GL_TRIANGLES / GL_QUADS** (additional shapes)
    Small triangles for parts of the rocket structure.
    Quads for base or rectangular elements in the scene.
21. **GL_POINTS + glPointSize()**
    Adds small details like cloud textures or highlights.
22. **Vertex placement**
    Carefully adjusted x, y coordinates and radii for each circle/shape to arrange clouds, rocket windows, and flames properly.
23. **glEnable(GL_BLEND);**
    Enables blending in OpenGL.
    Blending allows you to mix the colors of overlapping objects based on their alpha value (transparency).
    Commonly used for:
    Semi-transparent clouds
    Smoke or flame effects
    Glass or water surfaces
24. **glDepthMask(GL_FALSE);**
    Temporarily disables writing to the depth buffer.
    Depth buffer usually keeps track of how far each pixel is from the camera to handle occlusion.
    When drawing transparent objects, you often don't want them to block objects behind them, so you disable depth writes while blending.
25. **glEnable(GL_BLEND);**
    Turns blending on.
    Without this, the alpha values in your colors are ignored.
26. **glDepthMask(GL_FALSE);**
    Prevents transparent objects from writing to the depth buffer.
    This avoids issues where a semi-transparent object might incorrectly block objects behind it.
27. **glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);**
    Defines how colors are mixed based on alpha.
    GL_SRC_ALPHA: Uses the alpha of the object you're drawing.
    GL_ONE_MINUS_SRC_ALPHA: Uses the inverse alpha of what's already in the framebuffer.

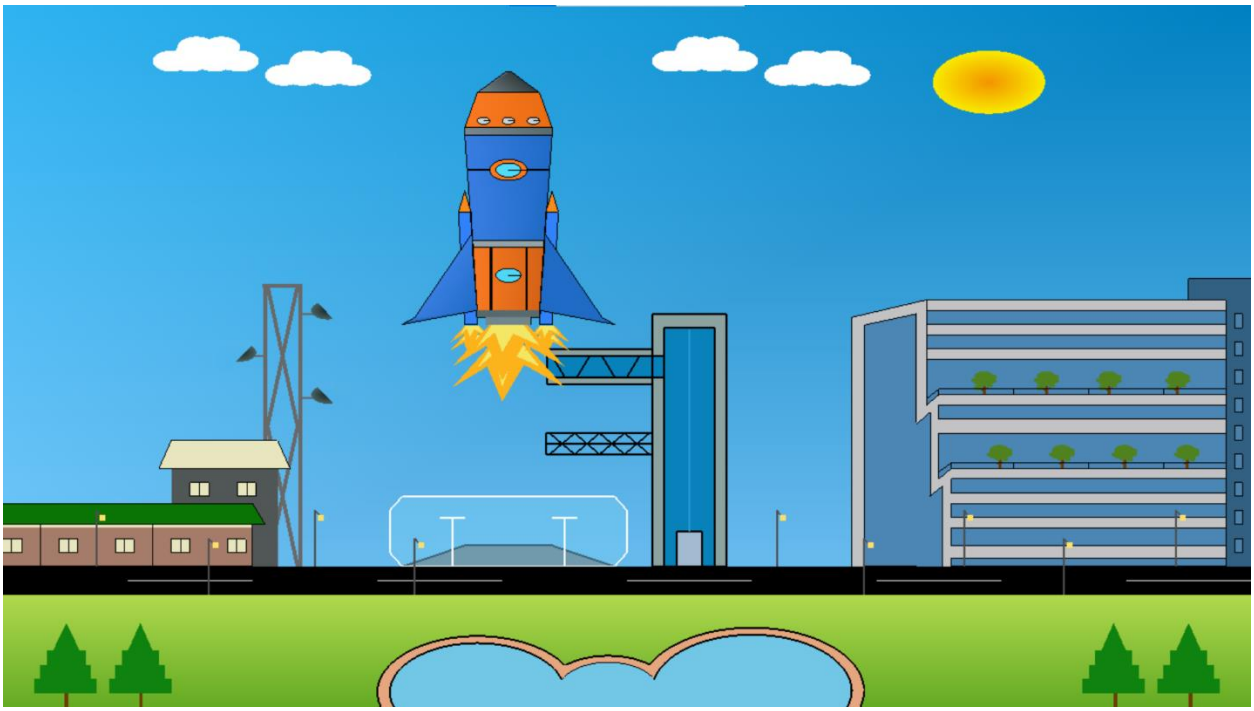
Screenshots:

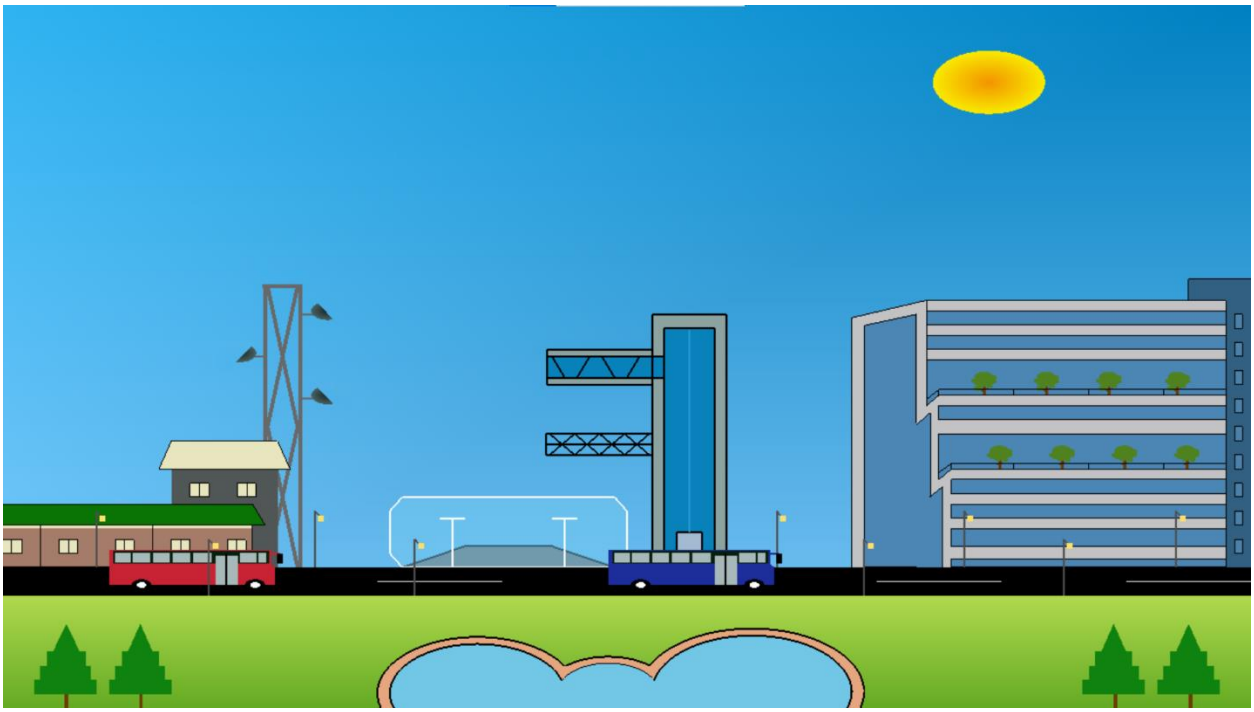
Scenerio 1:



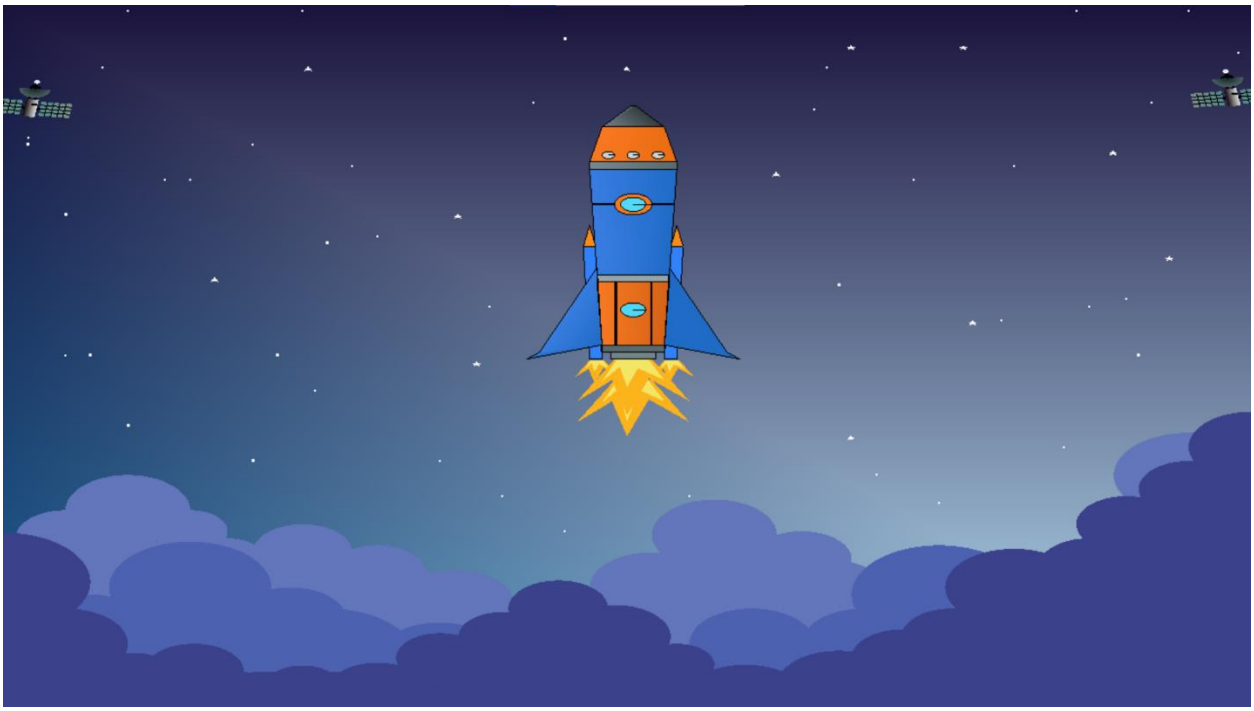
Scenerio 2:



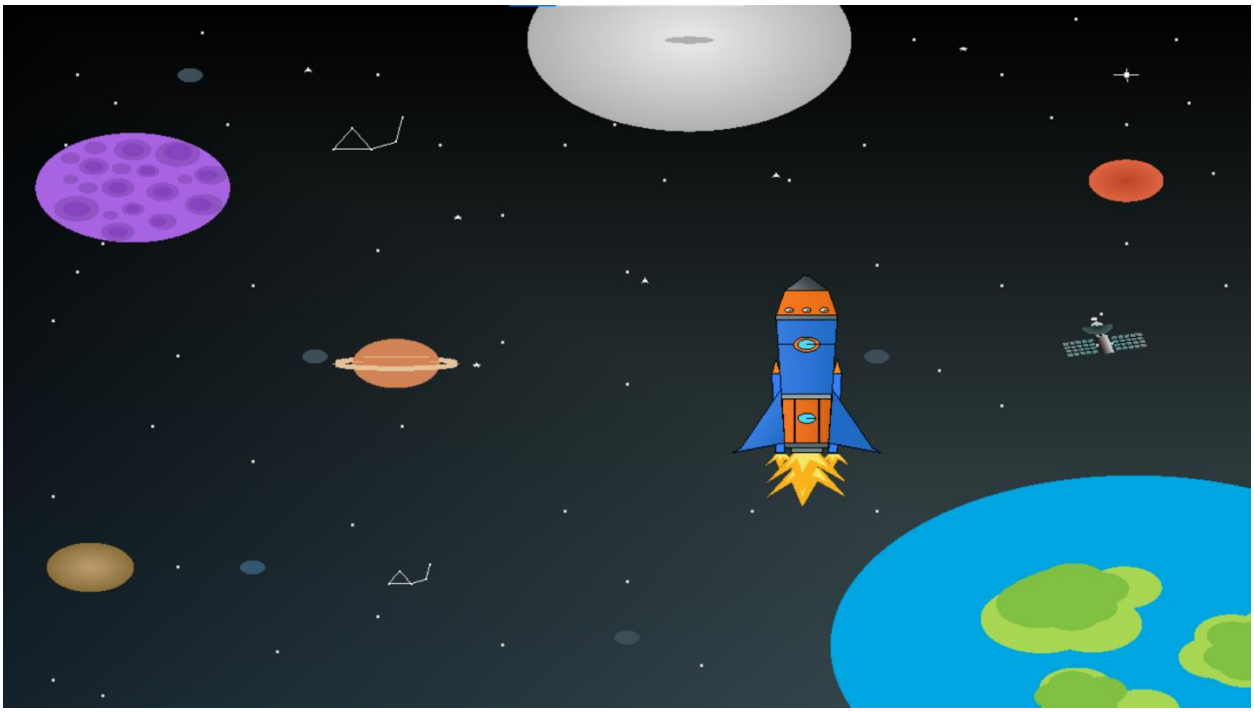


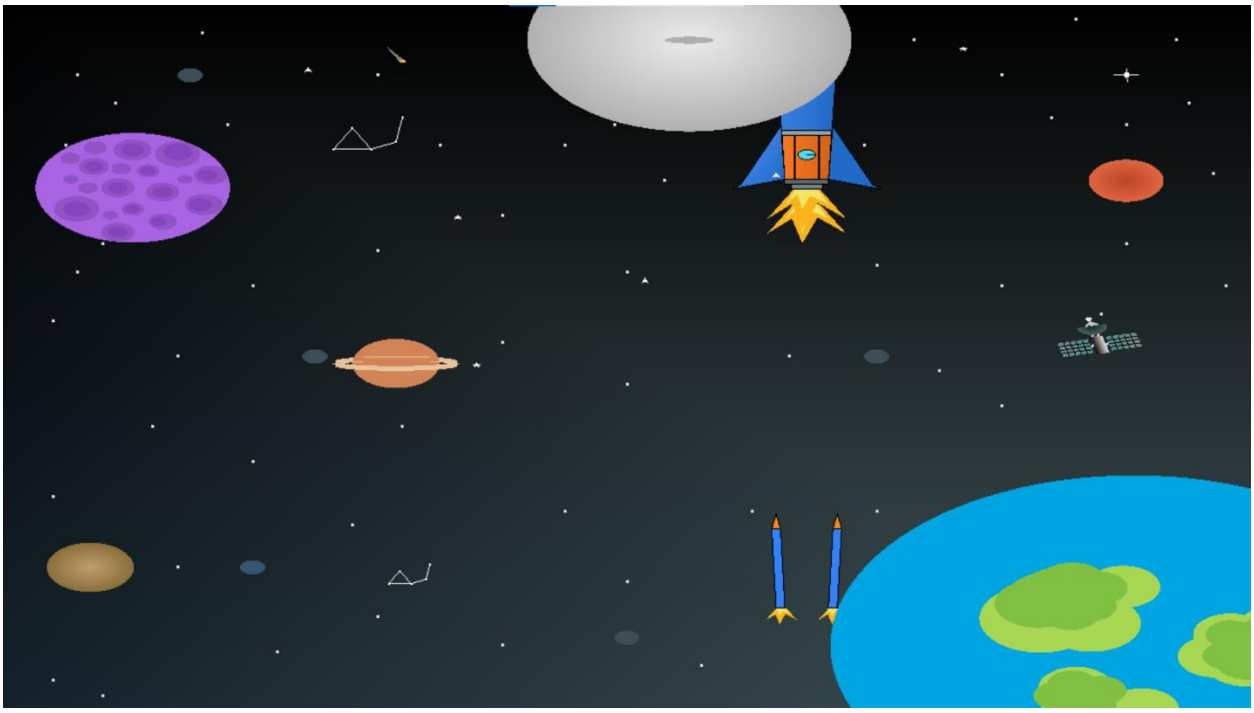


Scenerio 3:

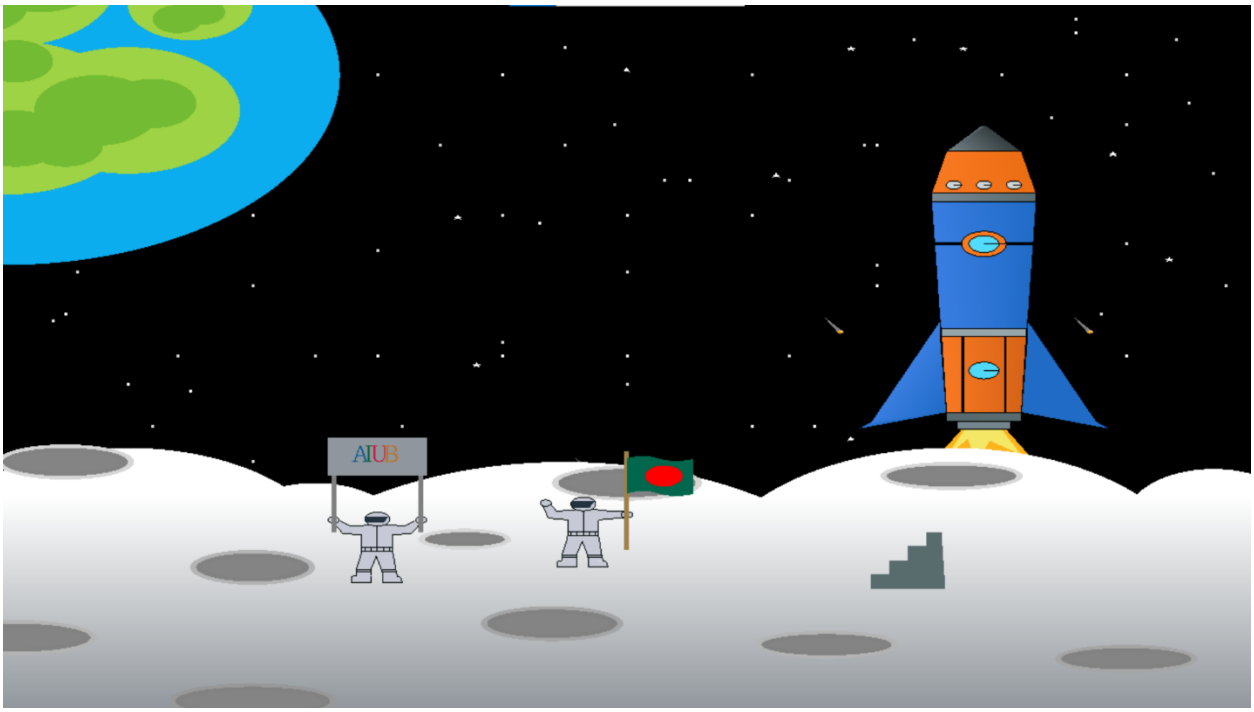


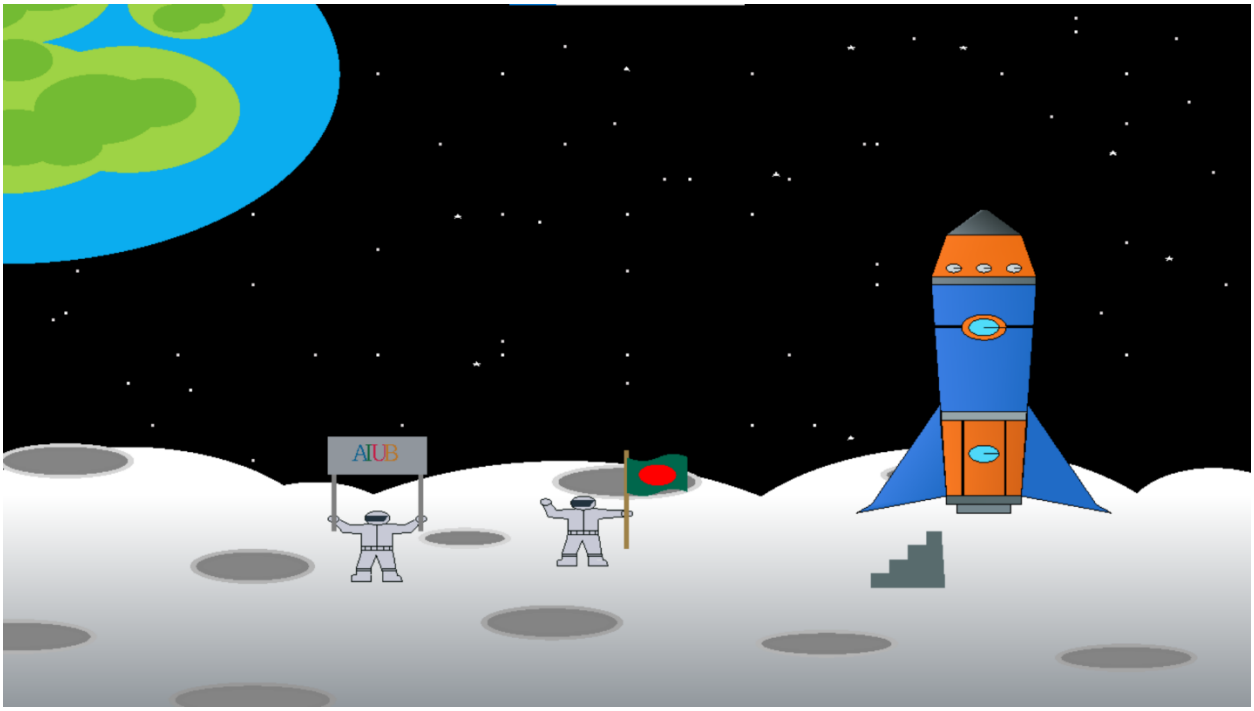
Scenerio 4:





Scenerio 5:




AIUB

Project Details:

Scenario Flow:

Scene 1 : Animated stars and circles with project title.
Scene 2 : Daytime environment with sun, clouds, roads, buses, trees, buildings, and a rocket on the launchpad.
Scene 3 : Space environment with rockets, clouds, stars, and meteors. Dual rocket launches with animation timing.
Scene 4 : Dark space with stars, planets (Earth, Mars, Venus), satellites, meteors, and rockets traveling.
Scene 5 : Astronauts planting flags (Bangladesh & AIUB), stairs setup, and rocket landing on the moon surface.

Scenario Description: The simulation follows a continuous, multi-scene journey structured as follows.

Scene 1: Rocket Launch Pad (Earth)
The rocket is on the launchpad, ready for takeoff. The environment includes a detailed base station and sky background. Animated ignition with fire and smoke simulates engine thrust, and the rocket begins its upward motion.

Control key:

j: Rocket fast forward

l: Rocket slow forward

k: Rocket stop

;: Rocket backward

Scene 2: Earth's Atmosphere Exit
The rocket ascends, gradually leaving Earth's atmosphere. The background transitions from bright blue sky to the dark vacuum of space. Stars appear, and Earth shrinks below, emphasizing altitude and distance.

Control key:

v: Start clouds + bus movement

b: Stop clouds + bus

n: Rocket launch upward

m: Stop rocket

Scene 3: Space Journey
The rocket travels through space. Celestial objects such as stars, planets, and asteroids form the background. Animation techniques create a dynamic sense of forward motion.

Control key:

t: Rocket forward/up

u: Rocket backward/down

y: Rocket stop

Meteors auto-start after ~25s

Scene 4: Moon Approach
As the rocket nears the moon, its surface becomes more visible with craters, mountains, and shadows for realism. The rocket begins a controlled descent with smooth landing animation.

Control key:

w: Both rockets upward

s: First rocket downward

a: Second rocket downward

d: Stop both rockets

Meteors auto-start after ~30s

Scene 5: Moon Landing
The rocket touches down on the lunar surface. Dust effects highlight the landing impact, emphasizing precision and scale.

Control key:

↑ Up Arrow: Increase rocket speed

↓ Down Arrow: Stop rocket

← Left Arrow: Move stairs & astronauts

→ Right Arrow: Stop stairs & astronauts

Meteors auto-start after ~40s

Project Impact:

For Job Purposes

- **Computer Graphics and Visualization:** Skills in rendering realistic scenarios are valuable in industries like gaming, simulation, and architecture.
- **Programming:** Expertise in OpenGL and graphics programming with C++ is highly applicable in technical roles.
- **Animation and Object Manipulation:** Proficiency in transformations (translation, scaling, rotation, reflection) can be applied in animation or 3D modeling.
- **UI/UX Design:** Experience designing visually interactive interfaces for applications.
- **Real-time Simulation:** Developing systems for vehicular simulations, AR/VR applications, or urban planning.
- **Automation of Visual Effects:** Automating the reuse of assets (e.g., trees, streetlights) saves time in large projects.

For Higher Studies Purposes

- **Advanced Computer Graphics:** Research in photorealistic rendering, scene optimization, or GPU programming.
- **Simulation and Modeling:** Extending to urban infrastructure simulations or autonomous vehicle environments.
- **Machine Learning in Graphics:** Enhancing rendering with AI, e.g., neural rendering for efficient computations.
- **Human-Computer Interaction (HCI):** Studying user interactions with visualized road or space scenarios.
- **Environmental Modeling:** Researching accurate depictions of weather, terrain, and natural elements in simulations.
- **Virtual Reality (VR) and Augmented Reality (AR):** Applying visualizations to immersive experiences for training or education.

Conclusion

This project successfully demonstrates the journey of a rocket from Earth to the Moon using computer graphics techniques. By dividing the simulation into multiple scenes — rocket launch, atmosphere exit, space journey, moon approach, landing, and surface exploration — each stage of the mission was visualized in a structured and interactive way.
The implementation applied core graphics concepts such as 2D/3D transformations, animation, texture mapping, and lighting effects to create realistic motion and environments. Each scene highlighted a different aspect of computer graphics, from scaling and translation in the launch sequence to shading and texturing on the lunar surface.
Through this project, we gained a deeper understanding of how computer graphics can simulate real-world scenarios. Overall, the project not only fulfilled its objectives but also provided valuable learning in combining creativity with technical implementation to bring a real-world concept — the Moon landing — into a visualized, animated experience.

Reference :

Github link: https://github.com/kamrul-hasan007/Computer-Graphics/tree/main

Attachment:

Graph plotting files