CPT205 CourseWork2 Report

Zichen Qiu
ID: 2252705
CPT205-2425-S1-Computer Graphics

I. Background

THE task involved creating a sophisticated 3D scene blending static and interactive elements, mirroring real-world scenarios. Leveraging freeglut, a key library for OpenGL, was crucial for functions like window management and input processing.

By using freeglut and OpenGL, the assignment explored various graphics techniques such as geometry creation, transformations, and lighting effects. This hands-on project tested and enhanced understanding of computer graphics concepts through practical application.

The report details the design and implementation of a dynamic 3D scene, showcasing creativity, technical skills, and adherence to programming best practices.

II. Design and Features

The original intention of this theme is to make a super simplified version of ***SimplePlanes***[1], simulating the flight of an airplane through simple graphics.

The following are the implementation methods of some elements in the picture:

A. Cuboid, Cylinder & Cone

Considering that the main body of the aircraft can mostly be simplified into a cylinder, and the wings are mostly rectangular blocks, and the subsequent programs can be reused. I first constructed the following two functions.

```
void drawCylinder(float down_radius, float
    top_radius, float height, float centerX,
    float centerY, float centerZ, float
    xAngle, float yAngle, float zAngle, float
     red, float green, float blue, GLuint
    Texture)
```

```
void drawCube(float X_length, float Y_length
    , float Z_length, float centerX, float
    centerY, float centerZ, float xAngle,
    float yAngle, float zAngle, float red,
    float green, float blue, GLuint Texture)
```

In the subsequent graphics generation, you can specify, for example, the size on the three axes, the coordinates of the center point, the direction of rotation along the three axes, the reflected color (if there is no texture), and the texture image.

Also, to reduce the amount of code and enhance code readability, some for loops are used.

B. The Plane

Among the numerous aircraft in history, I chose the ***Mikoyan-Gurevich MiG-19***[2] as a reference. It has several advantages. First, the main body of the aircraft is almost a cylinder, and second, important parts such as the wings are almost cuboids. In addition, its shape is classic enough.

The aircraft is made of riveted aluminum skin, with flames shooting out of the tail and a dashboard at the front of the cockpit that indicates the current forward speed. The red line on the dashboard represents the stall speed, the yellow line represents the impending stall warning, and the green line represents the current speed. The wheels of an airplane also spin after they touch the ground. The length of the flame emitted by the engine is determined by the forward flight speed.

The aircraft can be operated with the keyboard and the perspective can be switched. For specific operations, please refer to the following text or the console where the program is running.

In general, the aircraft uses technologies such as Geometry Creation, Hierarchical Modeling, Transformations, Viewing and Projection, Lighting and Materials, Texture Mapping, Animation and Interactions.

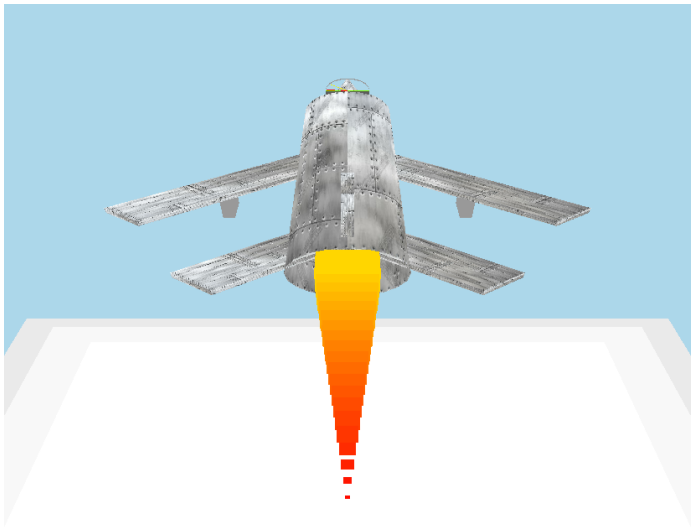

Fig. 1. The plane viewed from behind

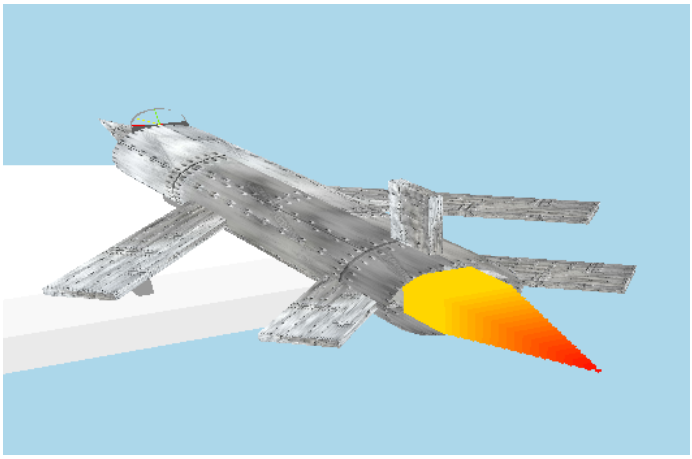

Fig. 2. The plane of another view

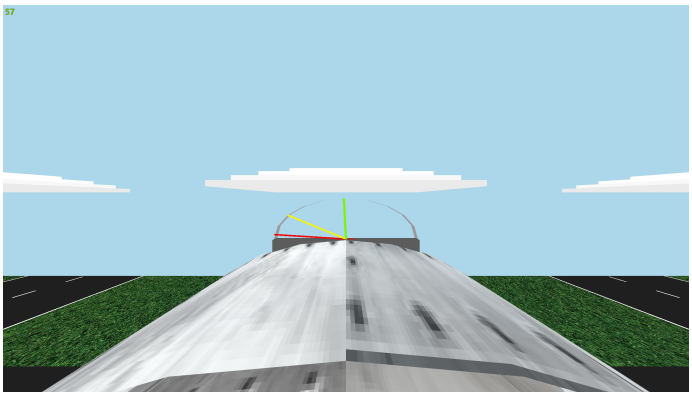

Fig. 3. Cockpit view

C. Surfaces such as Runways and Lawns

On the ground, all objects in the picture are made up of rectangular blocks, the difference is that some are colored and some are textured.

The game attempts to restore a small airport with three runways. There are some planes parked on the runways on both sides. The distance between the runways is also limited, and it will fail if you cross it.

On the ground outside the airport, a large green rectangle is directly drawn to simulate the ground. This will not only make it difficult for players to tell whether they have crossed the airport, but also save computer performance while reminding them that it is grass.

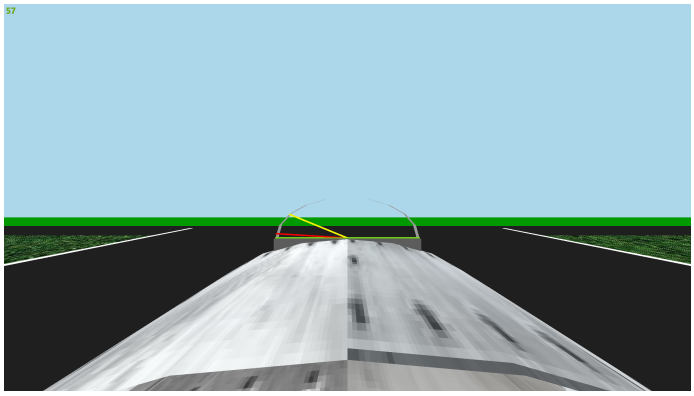

Fig. 4. End of the Runway

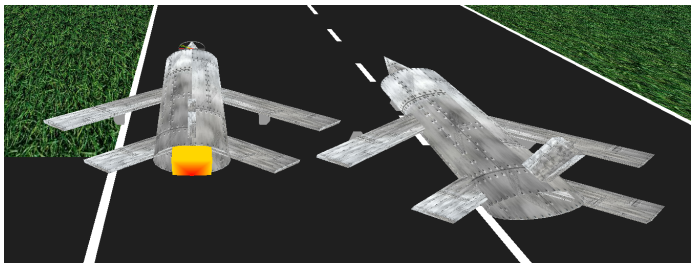

Fig. 5. The difference between aircraft at control and aircraft on the runway

D. Sky and Clouds

As shown in the previous picture, the sky is directly realized by setting ***glClearColor()***, and the clouds are realized by drawing rectangles of different colors.

III. 3D OPENGL GRAPHICS TECHNIQUES

A. Creation of geometry

The program generates a large number of 3D graphics, including but not limited to cubes, cylinders and cones.

B. Transformations

Different types of geometric transformation are applied, including scaling, translation, rotation and so on. For example, small objects need to be moved/adjusted to fit in the overall effects, and in this assignment, these are mostly realized by translation and scaling, wrapped by ***glPushMatrix()*** and ***glPopMatrix()***.

C. Hierarchical modeling

This program includes a number of geometry primitives, and these symbols are instanced using an instance transformation. Then, individual objects are grouped into a hierarchy that is represented by a tree structure.

For example, when designing the plane, the Direct Acyclic Graph (DAG) could be represented like this:

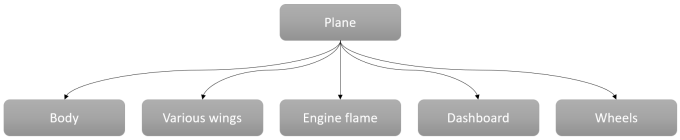

Fig. 6. DAG of a plane

For the whole background, it can be expressed as follow:

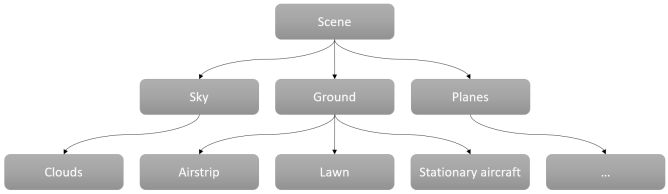

Fig. 7. DAG of the Scene

D. Viewing and Projection

The program uses perspective projection, which looks more realistic since this is how the human eye and cameras form images. In addition, the program allows the user to change the viewing angle through keyboard interaction, which will be mentioned in the next section.

E. Lighting and Materials

After observing most airports, I found that there are basically no bright point light sources at the airport, and the light intensity is basically the same at all locations on the runway. Therefore, this program does not use point light sources, but uses ambient lighting to simulate a real airport.

Considering that objects such as airports and airplanes are basically completely opaque, all materials are set to be opaque. As for the reflection of light, it will be slightly different on different objects, but basically the same.

F. Texture mapping

This program uses three textures, which are used to paste on the plane and the grass in the airport. As for the airstrip, I think it is better to use a cuboid directly, because it not only reduces the pressure on computer performance, but also is clearer than a texture.

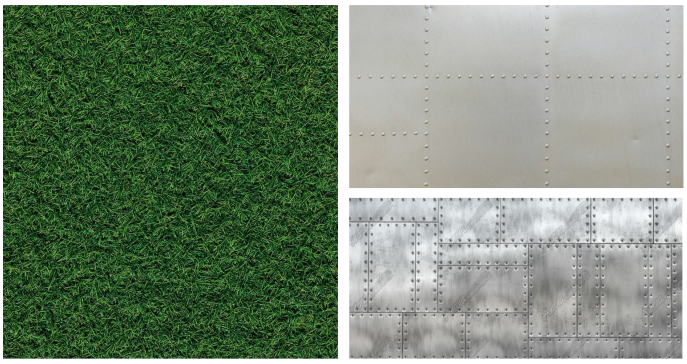

Fig. 8. Textures used

G. Animation and Interactions

After the game starts, the following words will be displayed in the terminal to explain the function of each button.

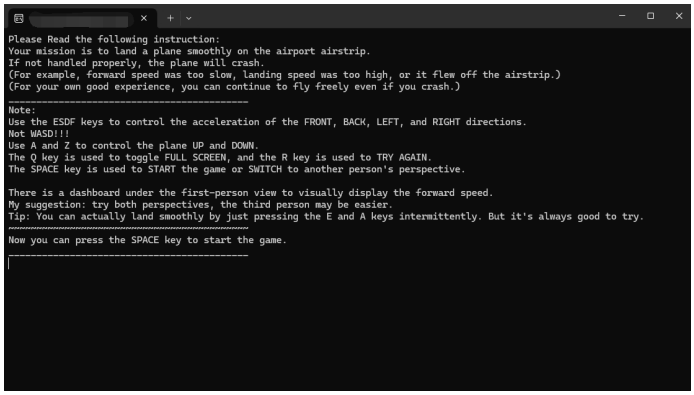

Fig. 9. Game instructions on the terminal

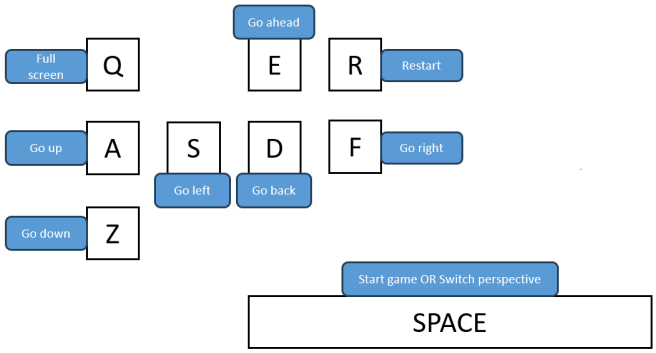

Fig. 10. Key operation instructions

REFERENCES

[1] *SimplePlanes*, https://www.simpleplanes.com/, [Online; accessed 8-December-2024].

[2] *Mikoyan-Gurevich MiG-19*, https://en.wikipedia.org/wiki/Mikoyan-Gurevich_MiG-19, [Online; accessed 8-December-2024].