Assignment 3: Planets in the Box

Department : 소프트웨어학과
Student Id : 2020312914
Name : Minho Baek

Requirements

Create and locate Cornell Box

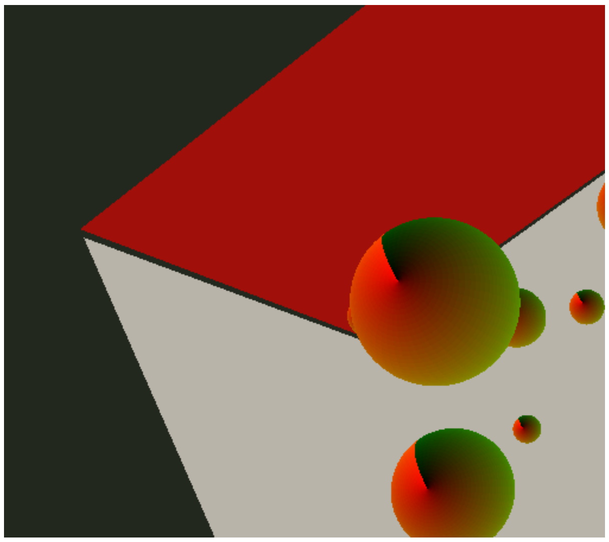
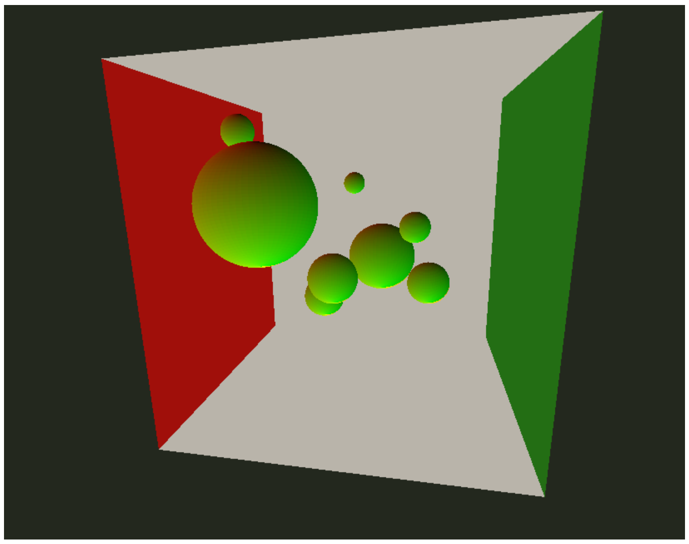
Similar to the implementation of `create_circles`, I constructed the Cornell Box using a function named `create_walls`, which returns a vector of individual wall objects. Each wall is defined with precise transformation data. For example, the back wall is defined as:
(note that the normal vector is included for consistency, but since lighting is not implemented, it has no visual effect)

```
p = {
    {
        {vec3(549.6f, 0.0f, 559.2f), vec3(0,0,-1), vec2(1,0)},
        {vec3(0.0f, 0.0f, 559.2f), vec3(0,0,-1), vec2(0,0)},
        {vec3(0.0f, 548.8f, 559.2f), vec3(0,0,-1), vec2(0,1)},
        {vec3(556.0f,548.8f, 559.2f), vec3(0,0,-1), vec2(1,1)}
    },
    vec4(0.725f, 0.710f, 0.680f, 1)
};
planes.emplace_back(p); // back
```

In total, six walls were created, and each wall is rendered using two triangles, resulting in exactly 12 triangles as specified. The vertex indexing for the triangles follows the standard quad-to-triangle conversion: `std::vector<uint> indices = { 0, 1, 2, 0, 2, 3 };` - a method consistent with the earlier sphere implementation. One notable difference is that all six walls were combined into a single VAO, and the rendering process maintained wall order by using the index divided by 6. Additionally, to ensure the interior of the box is visible, `GL_BACK` culling was applied.

While the official Cornell Box data specifies precise floating-point coordinates such as 549.6 and 556.0, I rounded them to simpler integer values like 275 and 550 to simplify collision handling. This choice reduces computational complexity and makes collision detection more robust.

Wall colors were also carefully assigned in accordance with the specification (e.g., red for the left wall, green for the right wall, and light gray for the other surfaces), ensuring both visual accuracy and compliance with the assignment requirements.

Create 9 planets

To approximate the planetary system, I initialized nine planets with the following radius values: `float radius[] = { 90.0f, 18.0f, 21.0f, 27.0f, 24.0f, 60.0f, 45.0f, 39.0f, 33.0f };`

These represent the Sun, Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, and Neptune, respectively. Each planet was also assigned a random speed within the range of 100 to 200, and a random movement direction determined by randomized `theta` and `phi` values in spherical coordinates.

After initialization, the rendering and transformation process followed the same pipeline as implemented in **Assignment 1**.

Physics on planets

The physical properties of the spheres follow the same model as in **Assignment 1**. During initialization, the `initCollision()` function was used to ensure that no spheres intersect at their initial positions. The main difference in this assignment is the inclusion of the Z-axis, enabling full 3D motion.

Object collisions were handled using separate velocity and direction variables for each sphere. By applying standard physical equations, 3D collisions were implemented appropriately based on these properties.

Camera Zooming and Panning Implementation

Zooming

Zooming is activated by vertical mouse movement while holding either the right mouse button or shift + left button. The zooming functionality moves the camera's `eye` position along the viewing direction vector (`n`), defined as the normalized vector from `at` to `eye` :
`c.eye = c.eye + (c.eye - c.at) * -p1.y;`

This approach ensures a physically correct zoom behavior without altering the field of view (`fovy`) or projection parameters, as required.

Panning

Panning is triggered by using the middle mouse button or ctrl + left button.
The implementation computes orthogonal `u` and `v` basis vectors spanning the `uv` plane, which is perpendicular to the viewing direction:

```
    vec3 n = (cam0.eye - cam0.at).normalize();
    vec3 u = (cam0.up.cross(n)).normalize();
    vec3 v = (n.cross(u)).normalize();
```

Both `eye` and `at` are translated along these `u` and `v` axes to preserve view orientation and maintain visual consistency.

Result

"Pasted image 20250517225126.png" 을 찾지 못했습니다.

"Pasted image 20250517225150.png" 을 찾지 못했습니다.