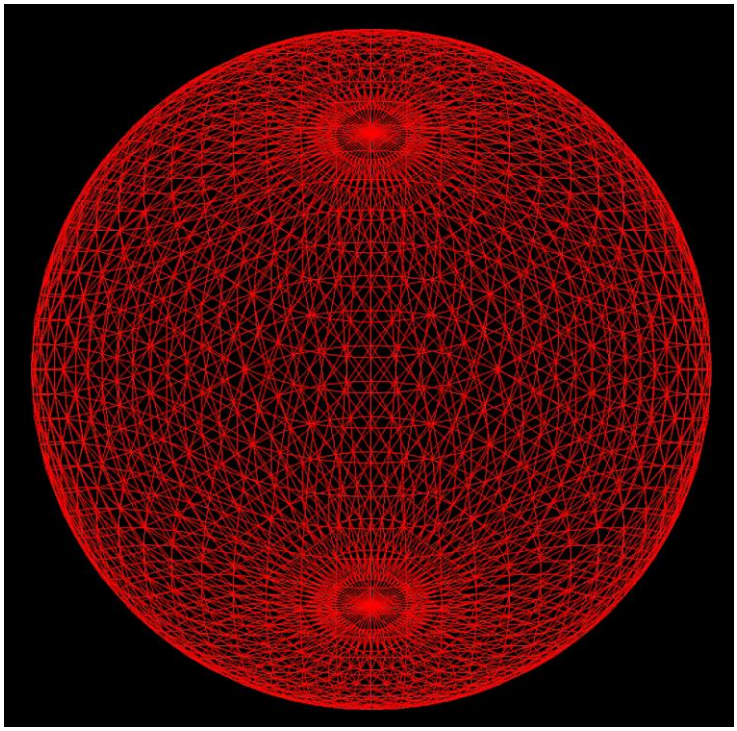
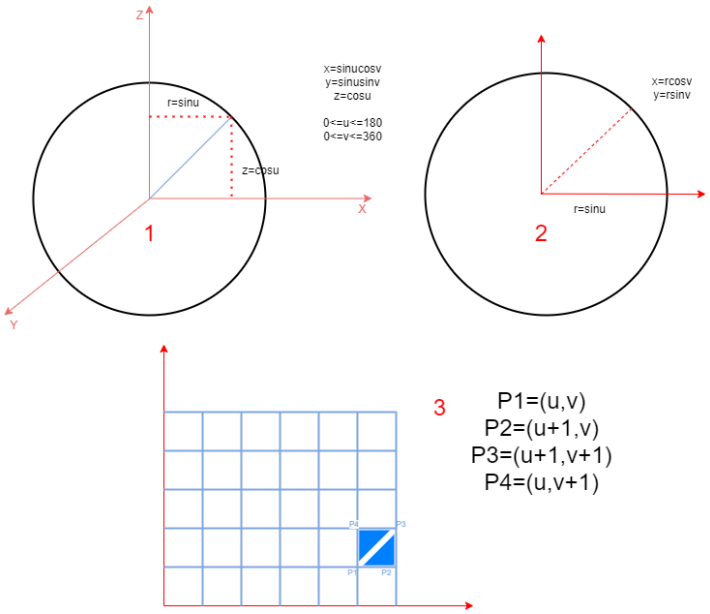
Realization of a panoramic video layer with OpenGL

This part of the project understanding requires OpenGL foundation. There is lots of great lessons on google. For example, https://learnopengl.com/

1.How to realize a sphere in OpenGL?

As we know, the basic graphics component in OpenGL is triangle. If we want to realize a sphere in OpenGL, we must use triangles to assemble. We can use lots of triangles to make up the surface as the below picture shown.





Step 1: Calculate the z coordinate

To draw a spherical surface, first cut into multiple sections according to the z direction, each section is a circle; that is, the part cut by 1 in the above figure is the circle shown in 2.
The truncated circle has a height, which is the z-coordinate;

The height of the circle is actually the cone angle formed by the circle and the center of the circle that determines the truncated height of the circle. This can be easily calculated

Step 2: the radius of the circle of the contour section r

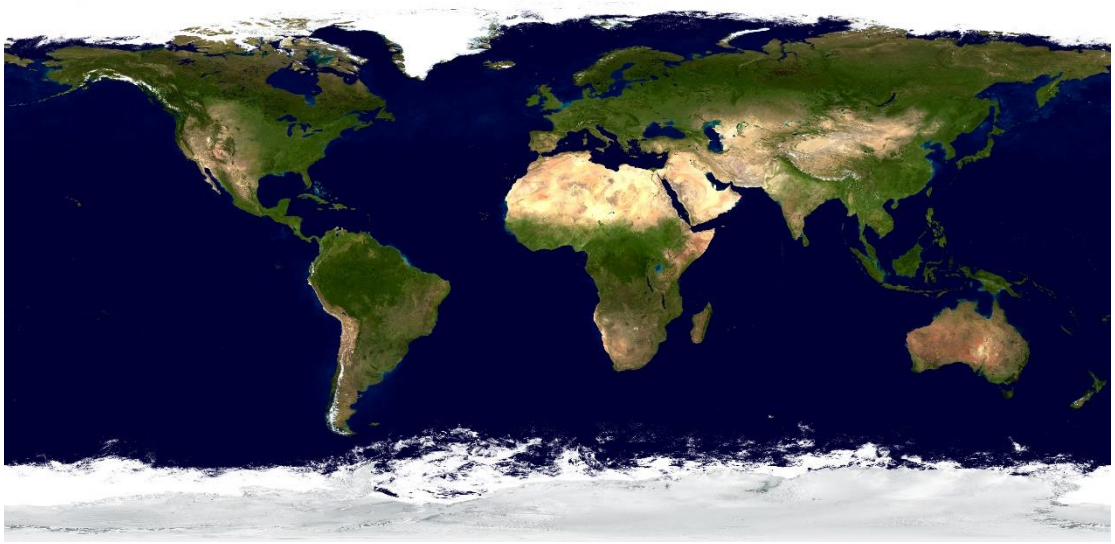
The radius of this circle is also easy to calculate

Step 3: Calculate x and y on the circle

When the radius of the circle is determined, x and y are easy to calculate.

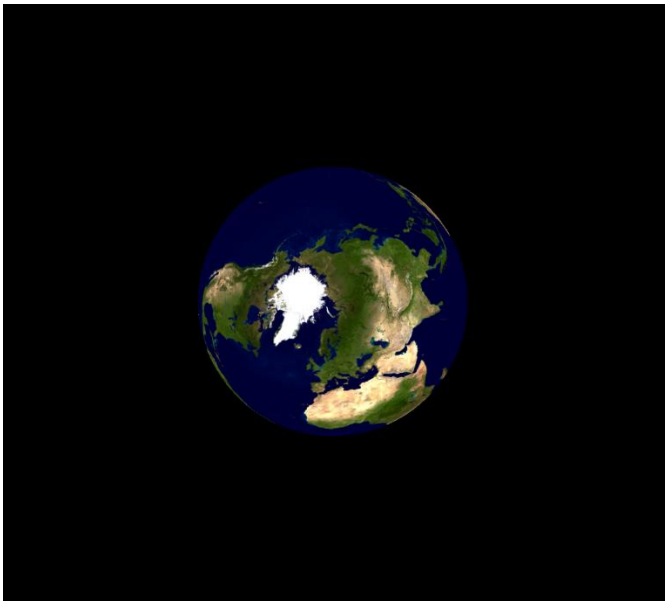
2. Algorithm test

We need use the earth picture to render a 3D earth in OpenGL. The realization ideas is using algorithm to generate vertex and texture vertex data of sphere. Then post the vertex data to shader.



3.Test result

Due to MVP Matrix, we can control the sphere rotating along xyz any axis



4.Move the demo test to panoramic video layer

We can render a 3D earth before, so we can render a video now. The panoramic video layer needs panoramic video source. This is a panoramic video source and we need render it on sphere model.

5. Render result

These sreenshot is shoted from test video.In this test video , I am moving the camera view, so I can see different areas in this panoramic video source.

(1)

(2)

(3)

(4)

6. Core Algorithm

```
void Sphere::init(int r,int precision){

    numVertices = (precision + 1) * (precision + 1);
    numIndices = precision * precision * 6;
    for (int i = 0; i < numVertices; i++) { vertices.push_back(glm::vec3()); }
    for (int i = 0; i < numVertices; i++) { texCoords.push_back(glm::vec2()); }
    for (int i = 0; i < numVertices; i++) { normals.push_back(glm::vec3()); }
    for (int i = 0; i < numVertices; i++) { tangents.push_back(glm::vec3()); }
    for (int i = 0; i < numIndices; i++) { indices.push_back(0); }

    // calculate triangle vertices
    for (int i = 0; i <= precision; i++) {
        float lon = toRadians(i*(180.0f/precision));
        for (int j = 0; j <= precision; j++) {
            float lat = toRadians(j*(360.0f/precision));
            float x = r*sin(lon)*cos(lat);
            float y = r*sin(lon)*sin(lat);
            float z = r*cos(lon);

            vertices[i*(precision + 1) + j] = glm::vec3(x, y, z);
            texCoords[i*(precision + 1) + j] = glm::vec2(((float)j / precision), ((float)i / precision));
            normals[i*(precision + 1) + j] = glm::vec3(x, y, z);

            // calculate tangent vector
            if (((x == 0) && (y == 1) && (z == 0)) || ((x == 0) && (y == -1) && (z == 0))) {
                tangents[i*(precision + 1) + j] = glm::vec3(0.0f, 0.0f, -1.0f);
            }
            else {
                tangents[i*(precision + 1) + j] = glm::cross(glm::vec3(0.0f, 1.0f, 0.0f), glm::vec3(x, y, z));
            }
        }
    }
    // calculate triangle indices
    for (int i = 0; i<precision; i++) {
        for (int j = 0; j<precision; j++) {
            indices[6 * (i*precision + j) + 0] = i*(precision + 1) + j;
            indices[6 * (i*precision + j) + 1] = i*(precision + 1) + j + 1;
            indices[6 * (i*precision + j) + 2] = (i + 1)*(precision + 1) + j;
            indices[6 * (i*precision + j) + 3] = i*(precision + 1) + j + 1;
            indices[6 * (i*precision + j) + 4] = (i + 1)*(precision + 1) + j + 1;
            indices[6 * (i*precision + j) + 5] = (i + 1)*(precision + 1) + j;
        }
    }
}
```