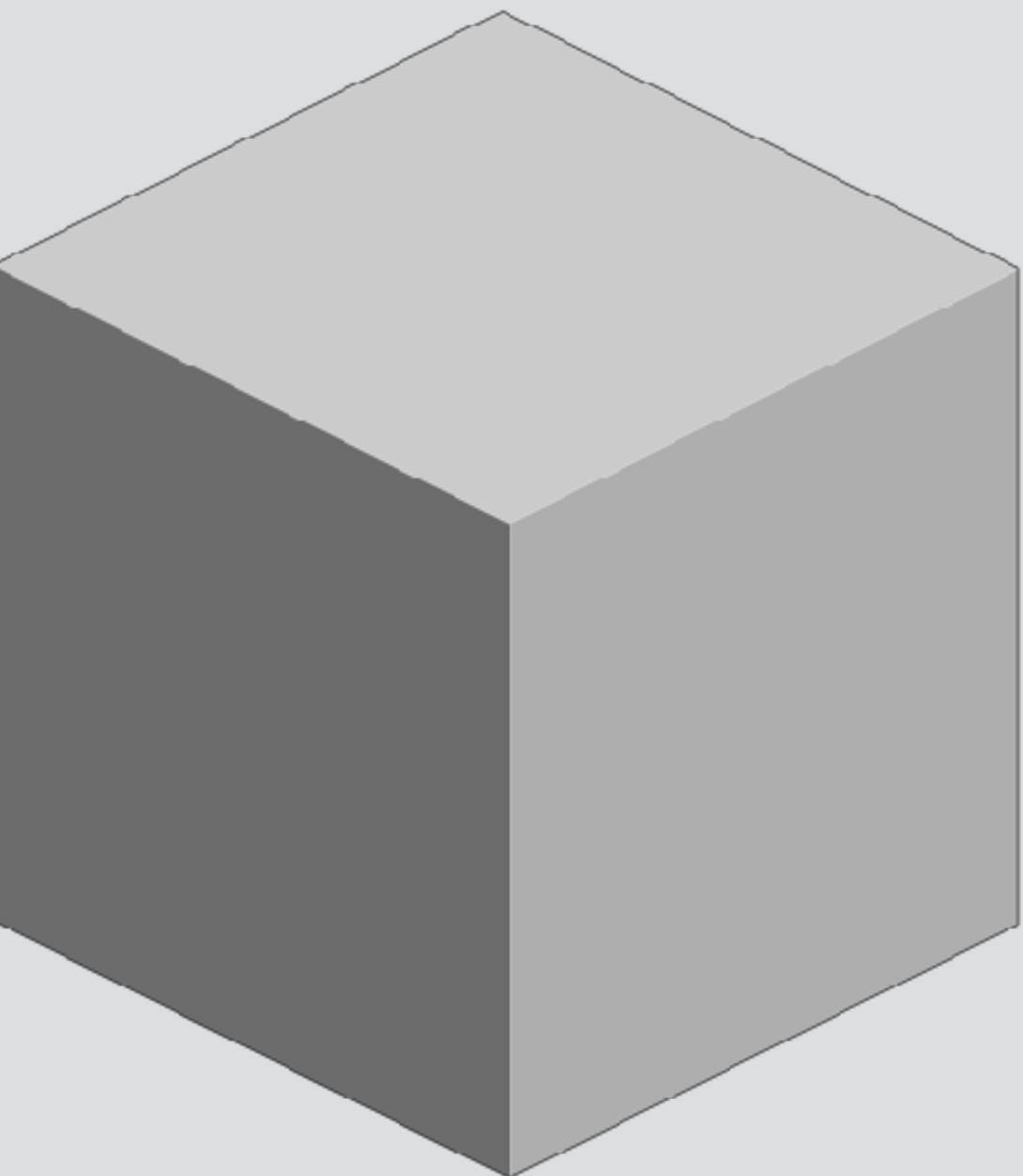


# 3D Graphics

## Part 2





Xboxworld.net



**14**  
AMMO

**59%**  
HEALTH

**2 3 1  
5 6 2**  
ARMS



**0%**  
ARMOR

**BULL 42 200  
SHEL 14 50  
ROCKT 0 50  
CELL 0 300**



+ 88

- 20

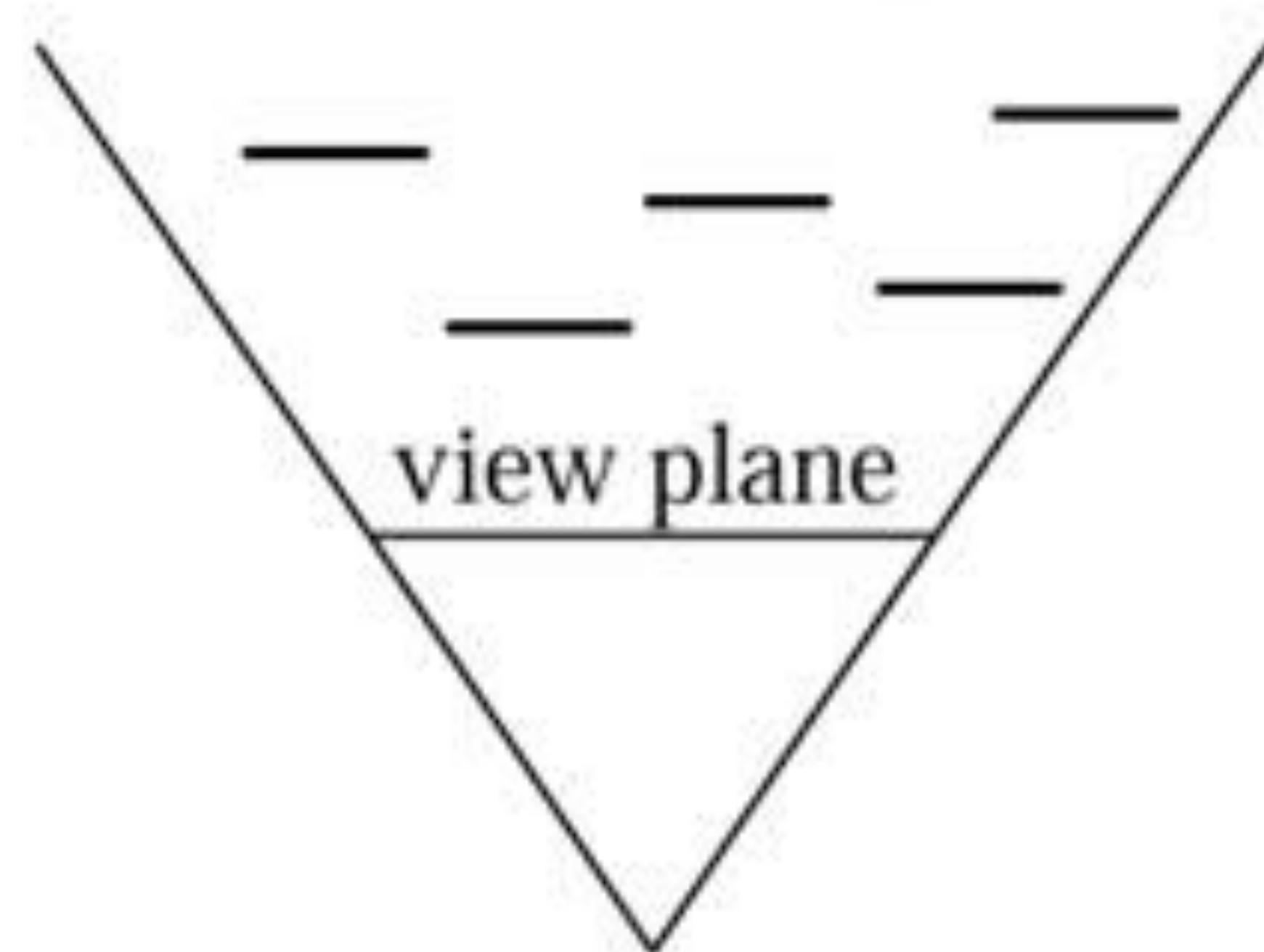
Billboards.

# Billboards

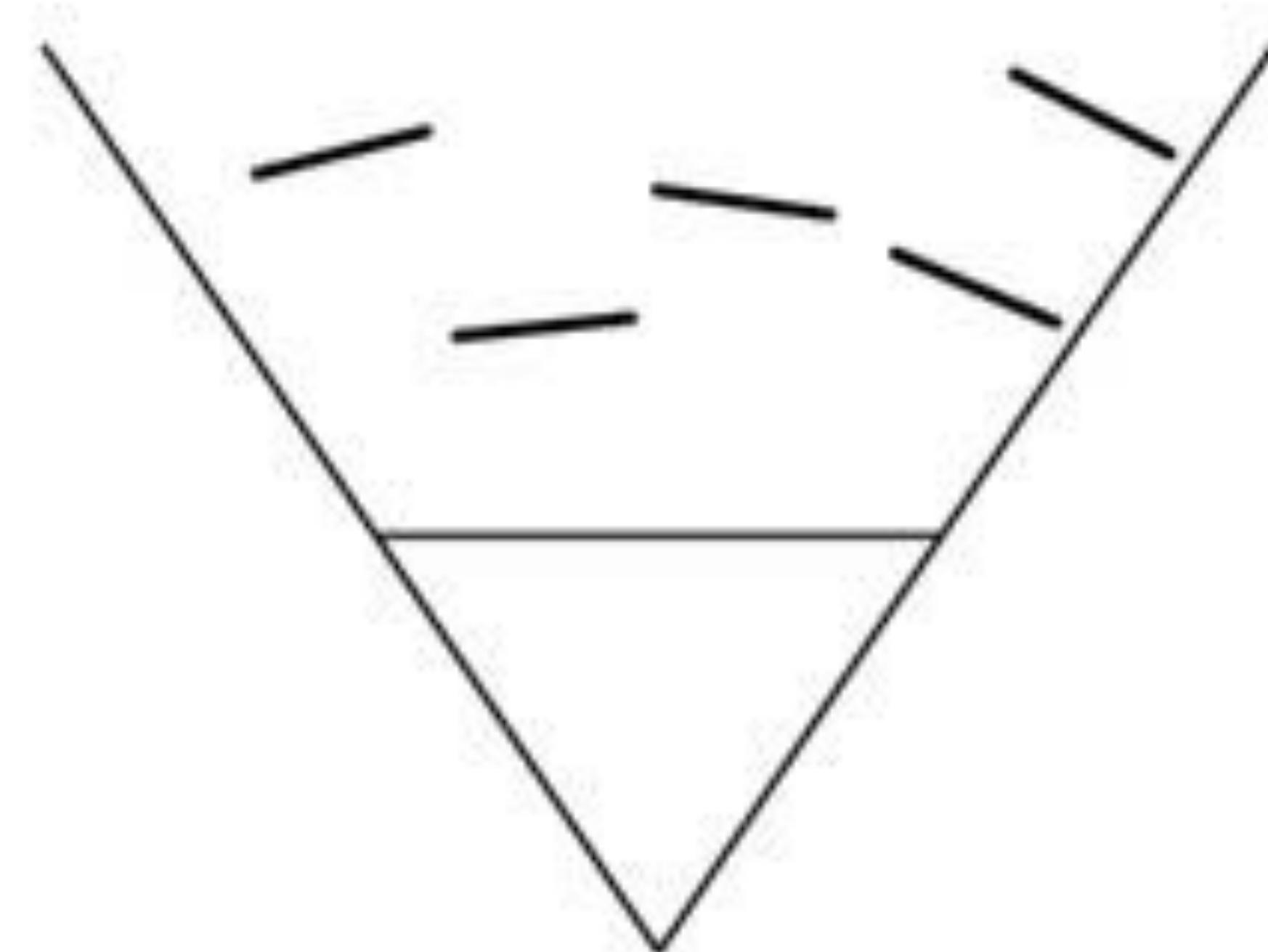
=

Sprites that always face the camera.

view plane aligned

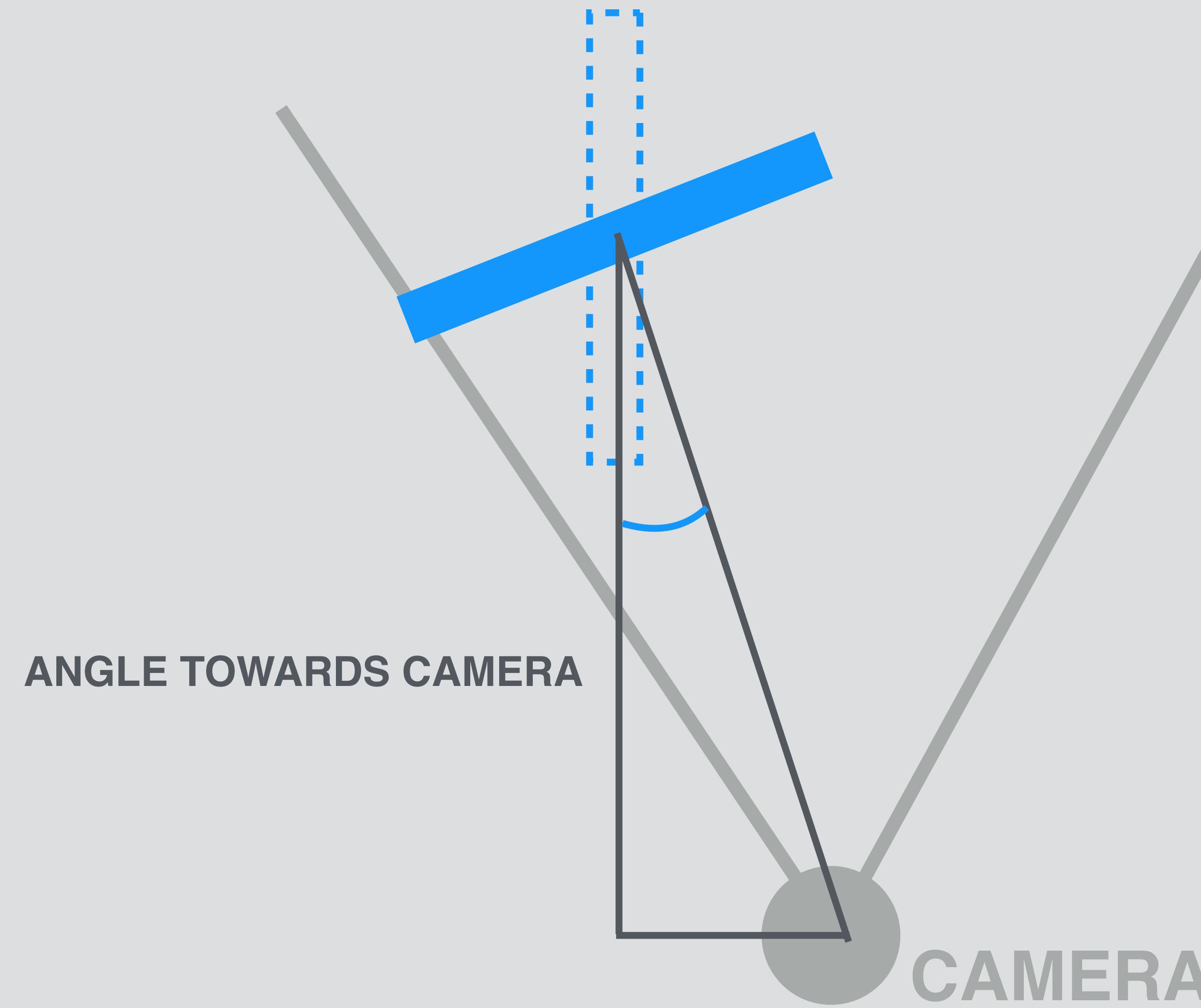


viewpoint oriented



# Method #1 - Rotate all billboards towards camera.

## Viewpoint oriented

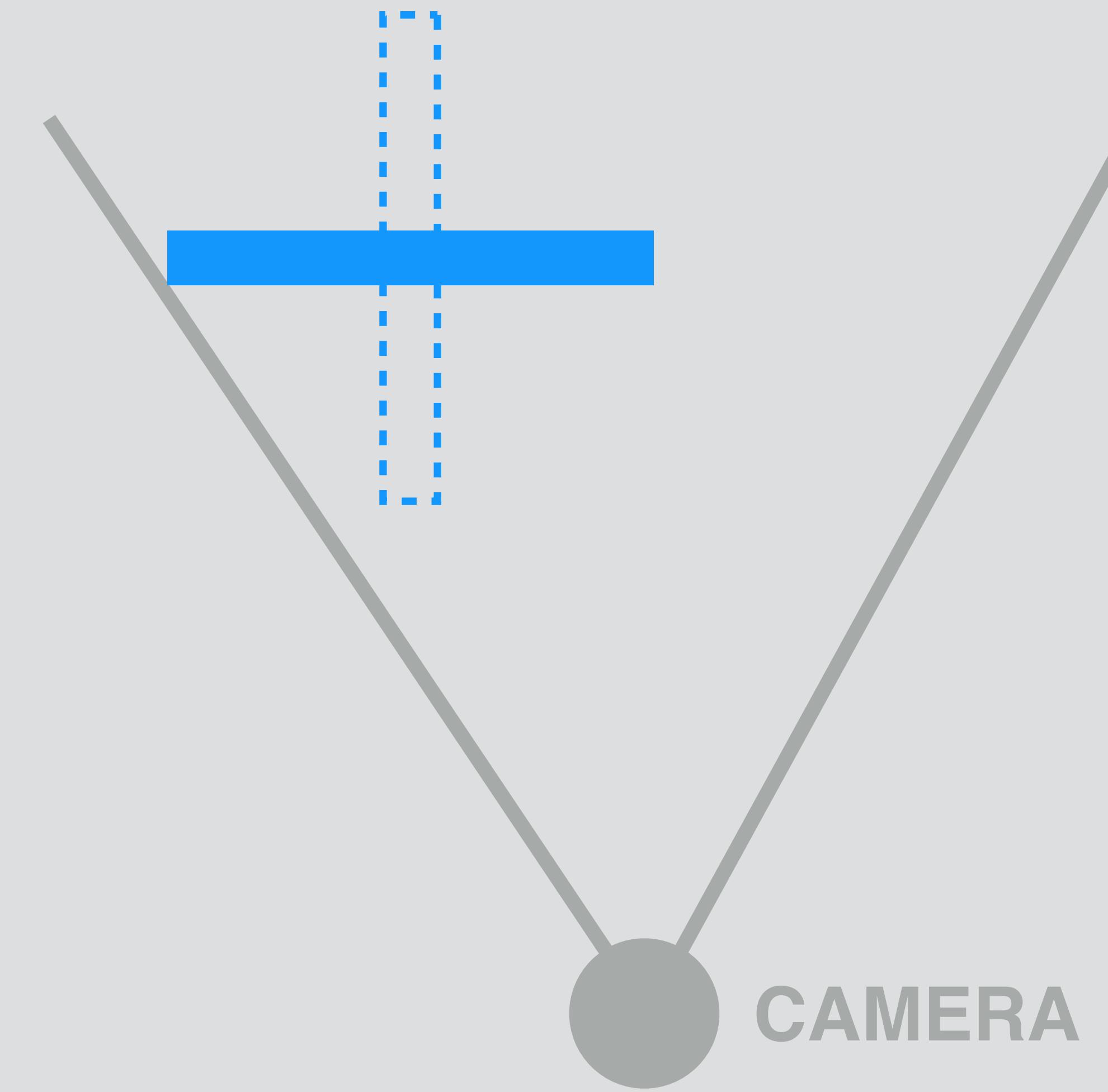


# Method #1 - Rotate all billboards towards camera. Viewpoint oriented

```
for(int i=0; i < entities.size(); i++) {  
  
    if(entities[i]->billboard) {  
  
        float directionX = entities[i]->position.x - camera->position.x;  
        float directionZ = entities[i]->position.z - camera->position.z;  
  
        entities[i]->rotation.y = atan2f(directionX, directionZ);  
    }  
  
    entities[i]->Render();  
}
```

Method #2 - Remove all rotation from the modelview matrix.

**Viewplane aligned**



# Method #2 - Remove all rotation from the modelview matrix. Viewplane aligned

```
attribute vec4 position;
attribute vec2 texCoord;

uniform mat4 modelMatrix;
uniform mat4 viewMatrix;
uniform mat4 projectionMatrix;

varying vec2 texCoordVar;
void main()
{
    mat4 modelViewMatrix = viewMatrix * modelMatrix;

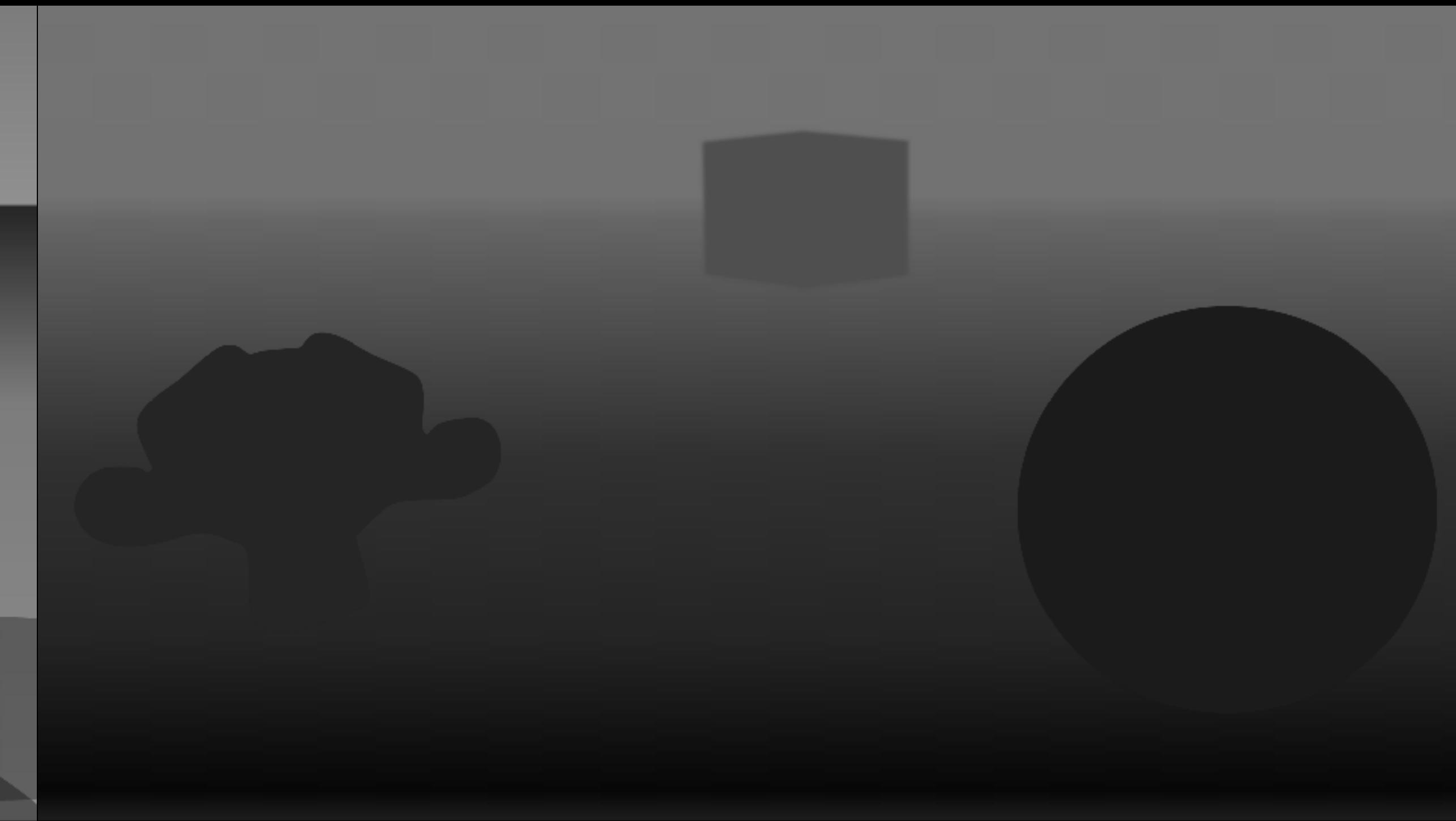
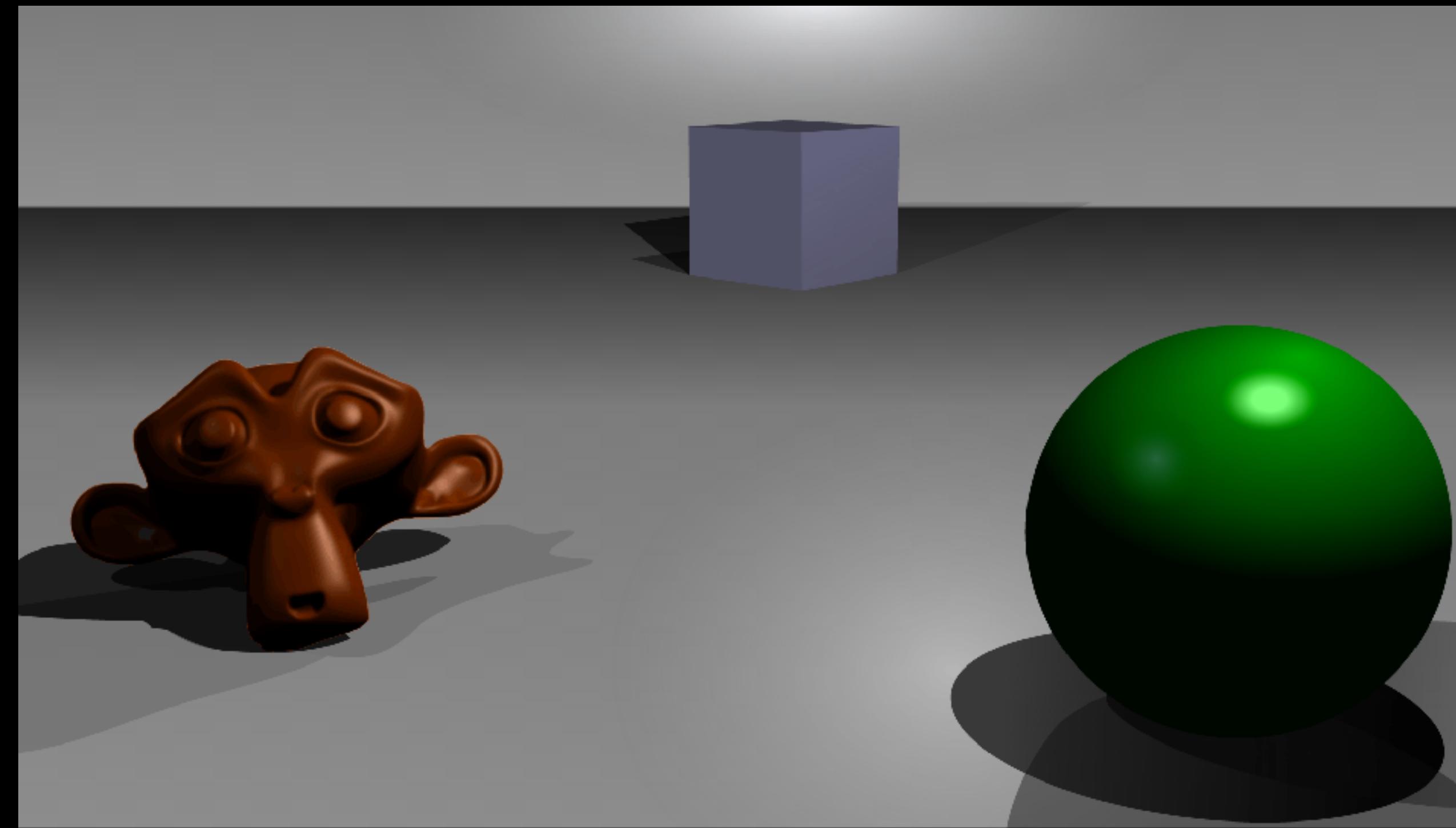
    modelViewMatrix[0][0] = 1.0;
    modelViewMatrix[0][1] = 0.0;
    modelViewMatrix[0][2] = 0.0;

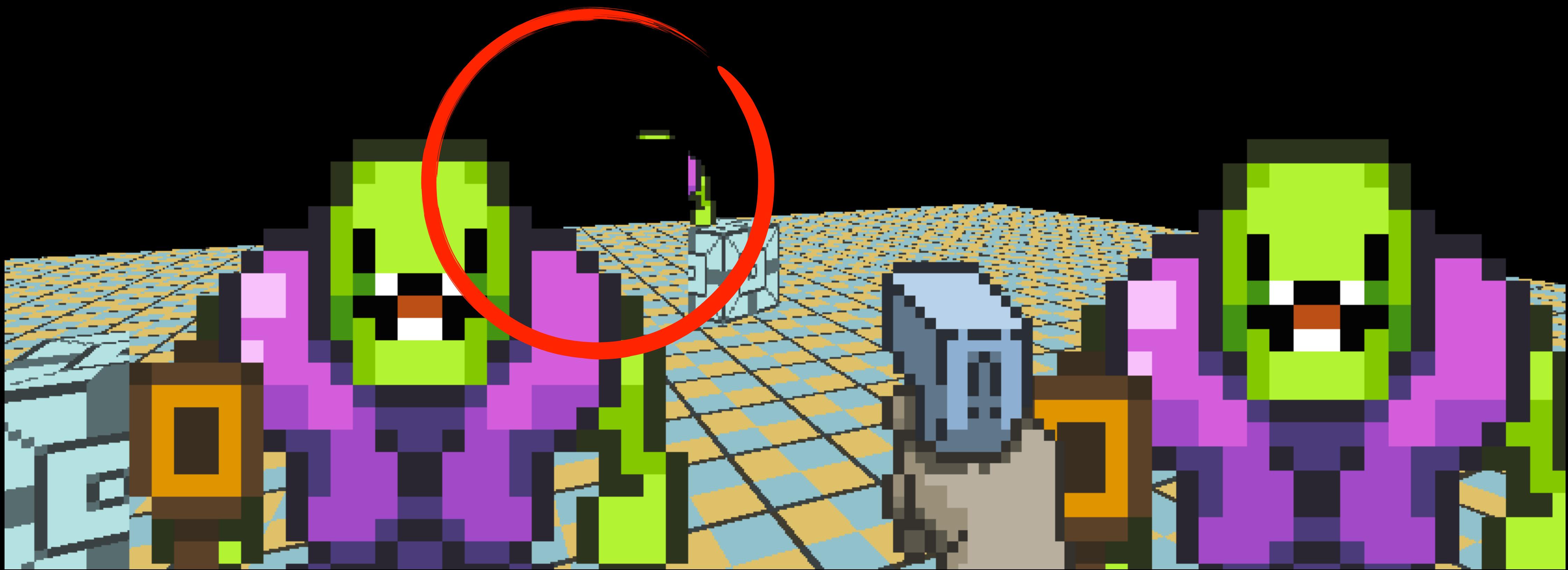
    modelViewMatrix[1][0] = 0.0;
    modelViewMatrix[1][1] = 1.0;
    modelViewMatrix[1][2] = 0.0;

    modelViewMatrix[2][0] = 0.0;
    modelViewMatrix[2][1] = 0.0;
    modelViewMatrix[2][2] = 1.0;

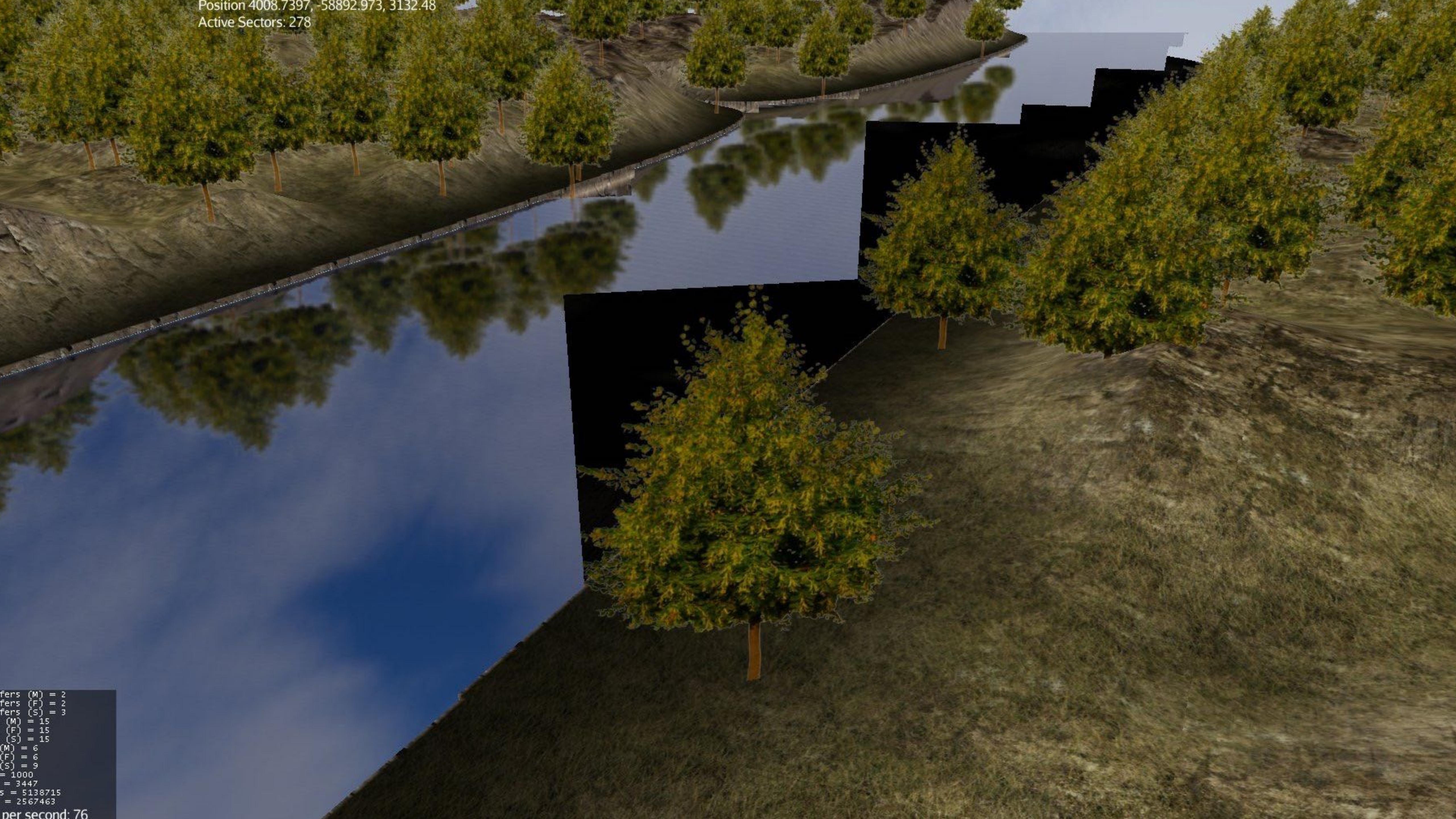
    vec4 p = modelViewMatrix * position;
    texCoordVar = texCoord;
    gl_Position = projectionMatrix * p;
}
```

# Blending and the Z-Buffer





Position 4008.7397, -58892.973, 3132.48  
Active Sectors: 278



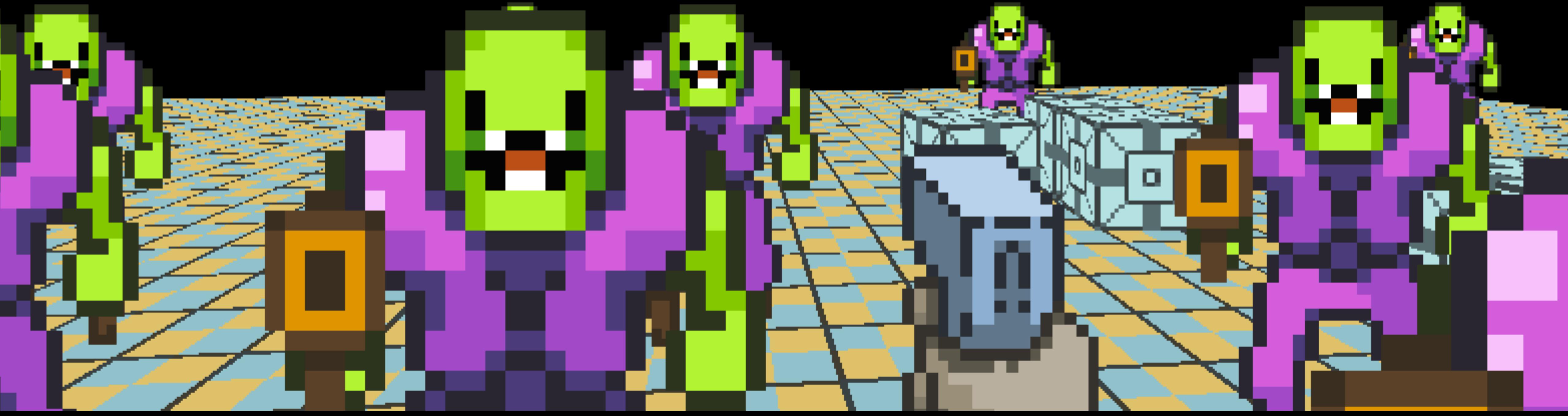
fers (M) = 2  
fers (F) = 2  
fers (S) = 3  
(M) = 15  
(F) = 15  
(S) = 15  
(M) = 6  
(F) = 6  
(S) = 9  
= 1000  
= 3447  
s = 5138715  
= 2567463  
per second: 76

# Discarding fragments.

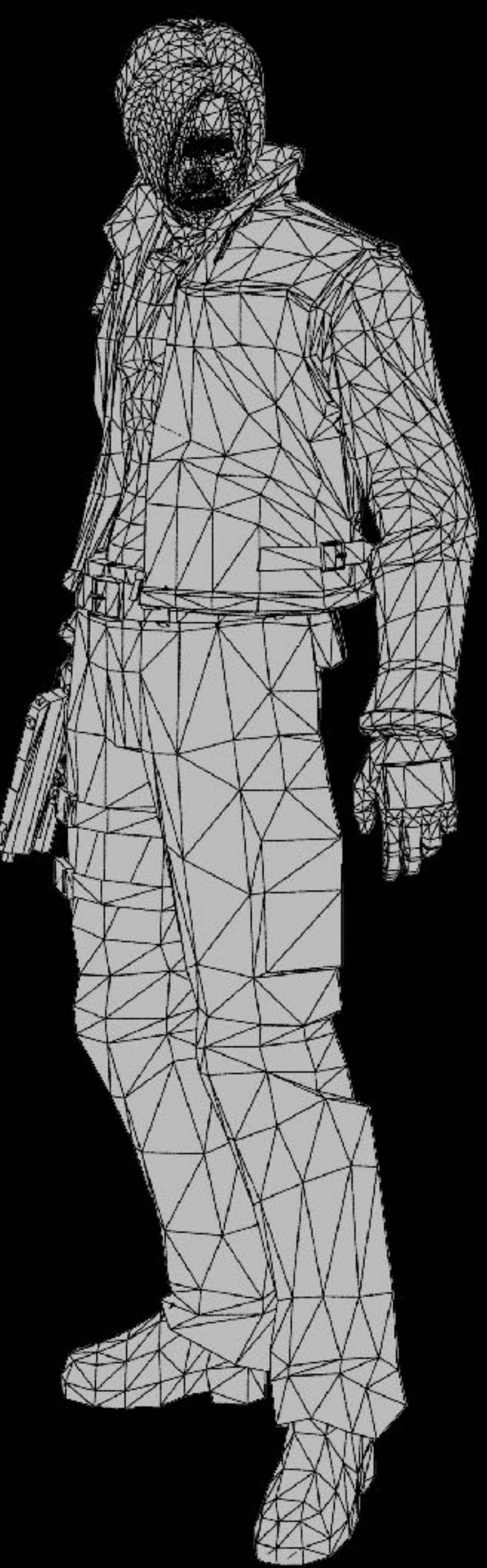
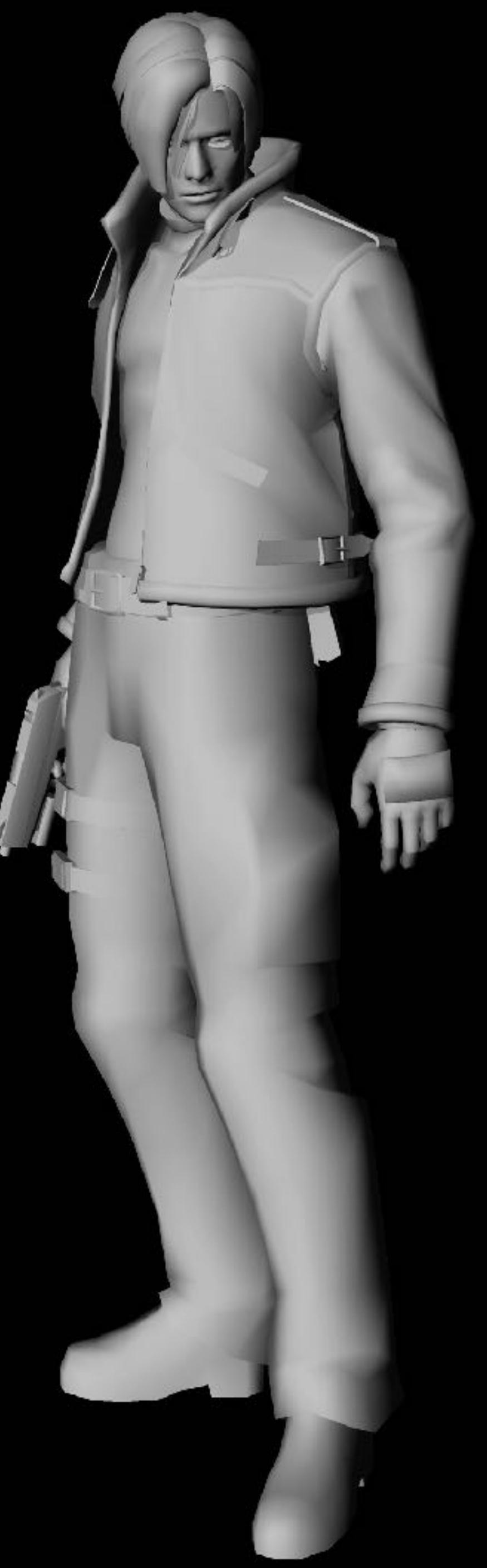
# Discard fragment if alpha value is 0.

```
uniform sampler2D diffuse;
varying vec2 texCoordVar;

void main()
{
    gl_FragColor = texture2D(diffuse, texCoordVar);
    if(gl_FragColor.a == 0.0) {
        discard;
    }
}
```

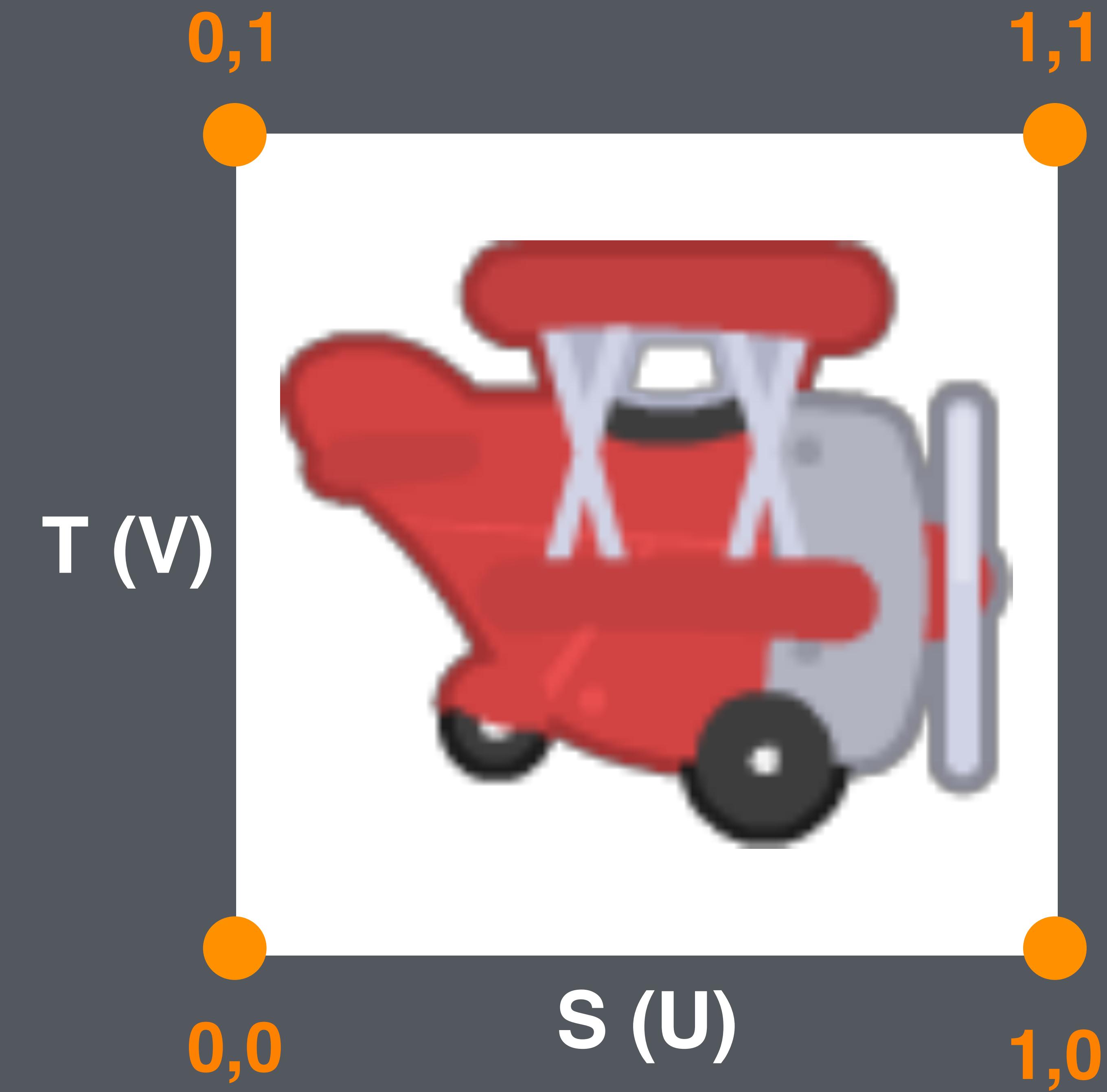


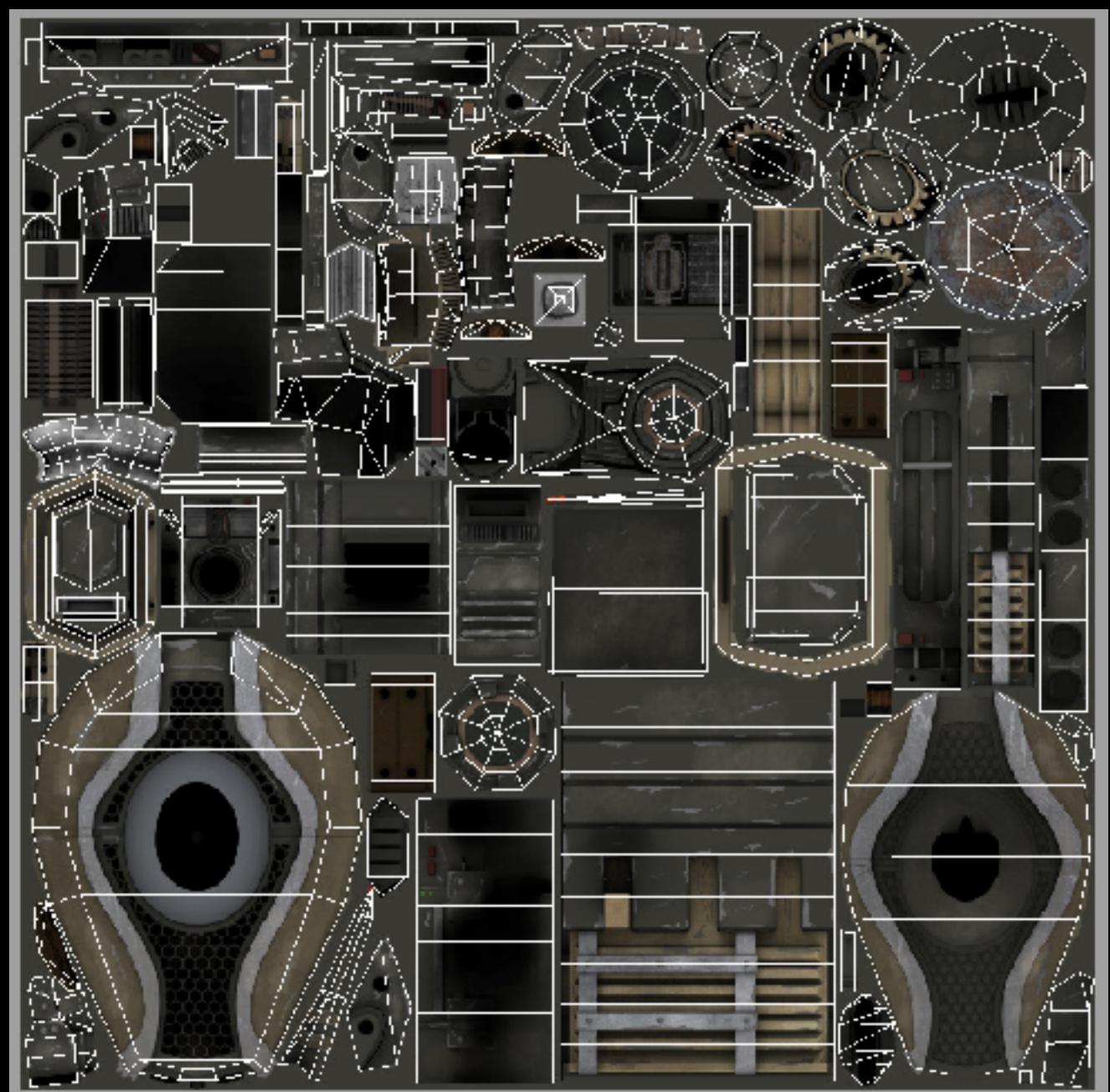
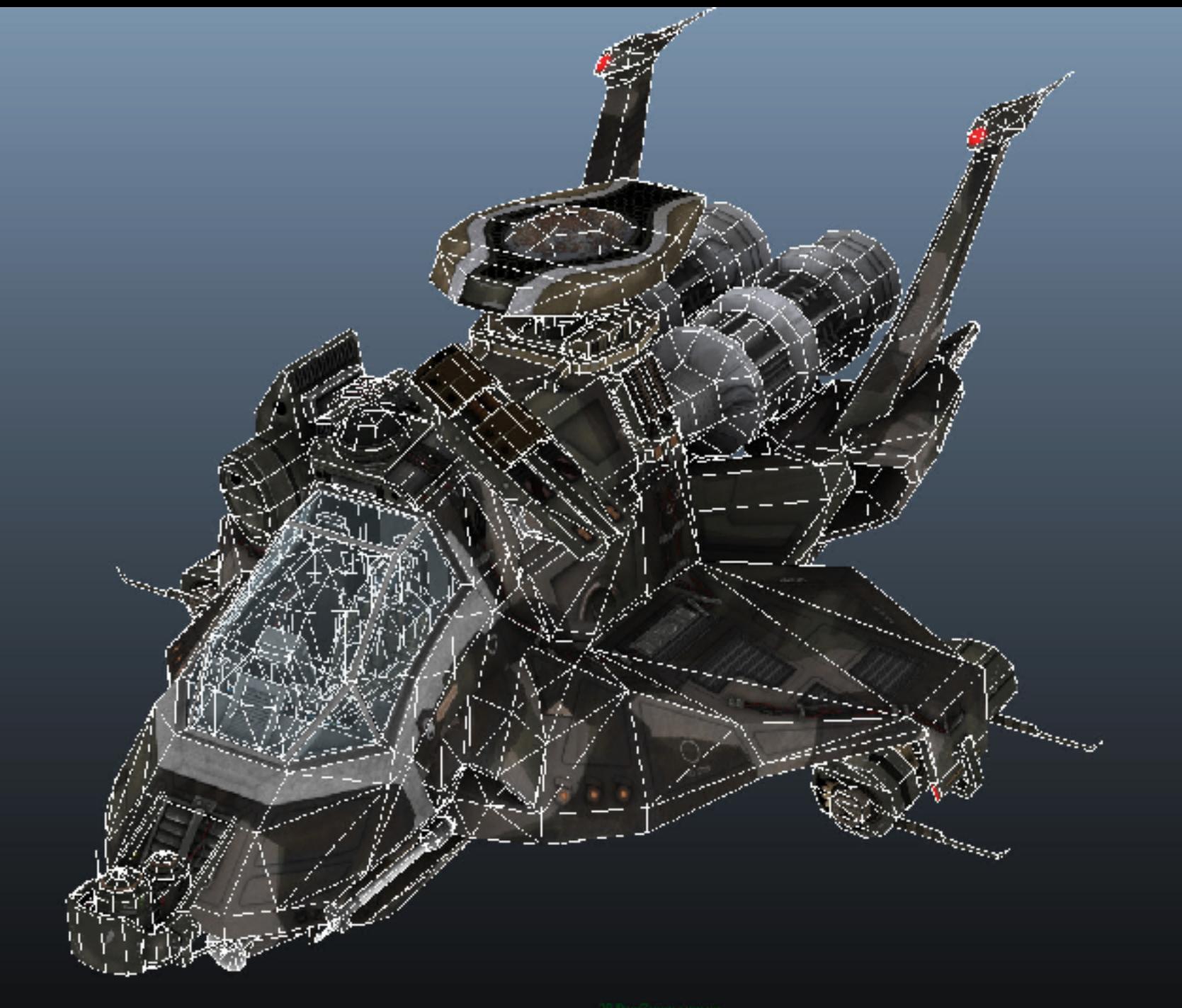
# Working with 3D meshes.



Position coordinates.

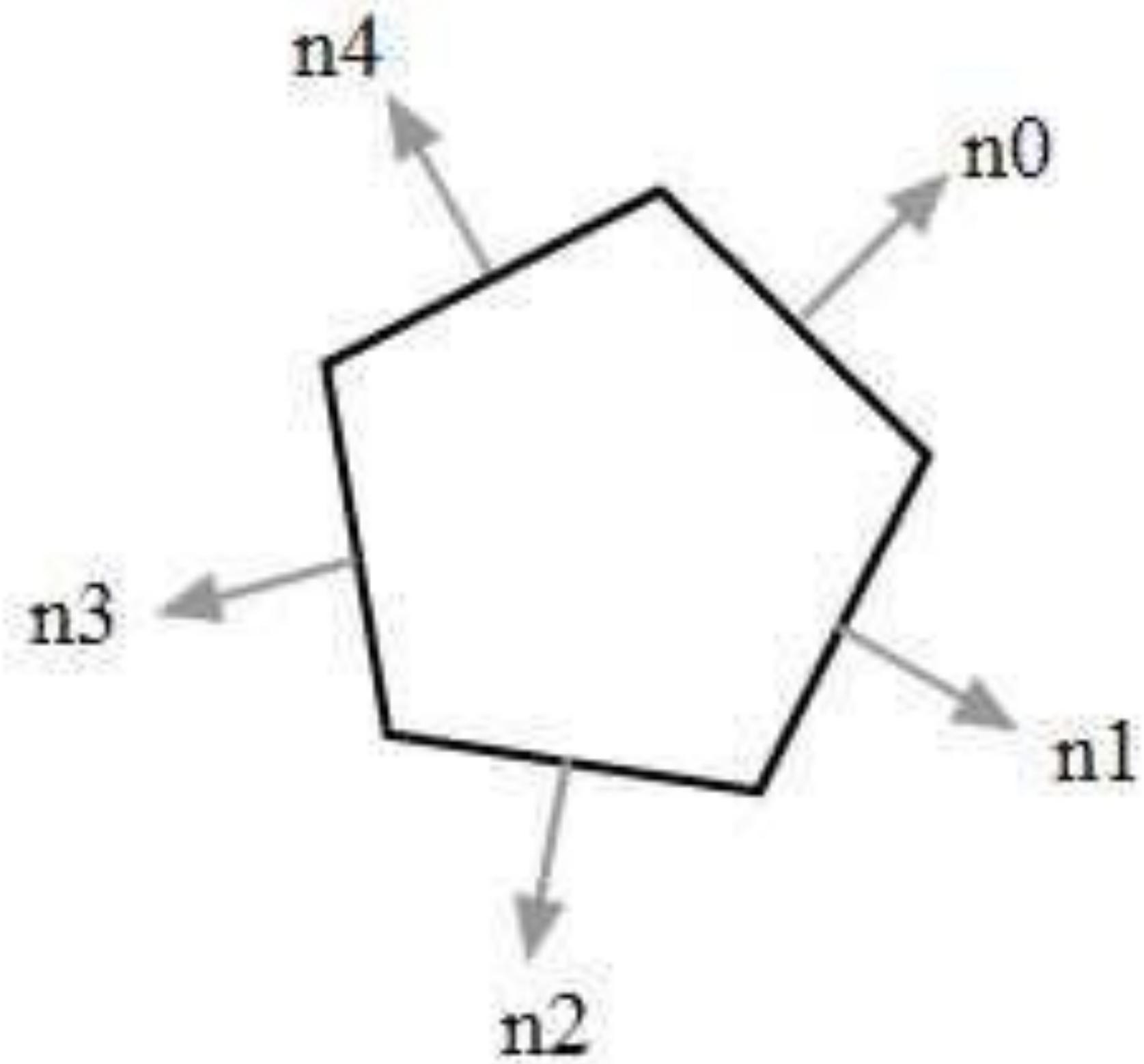
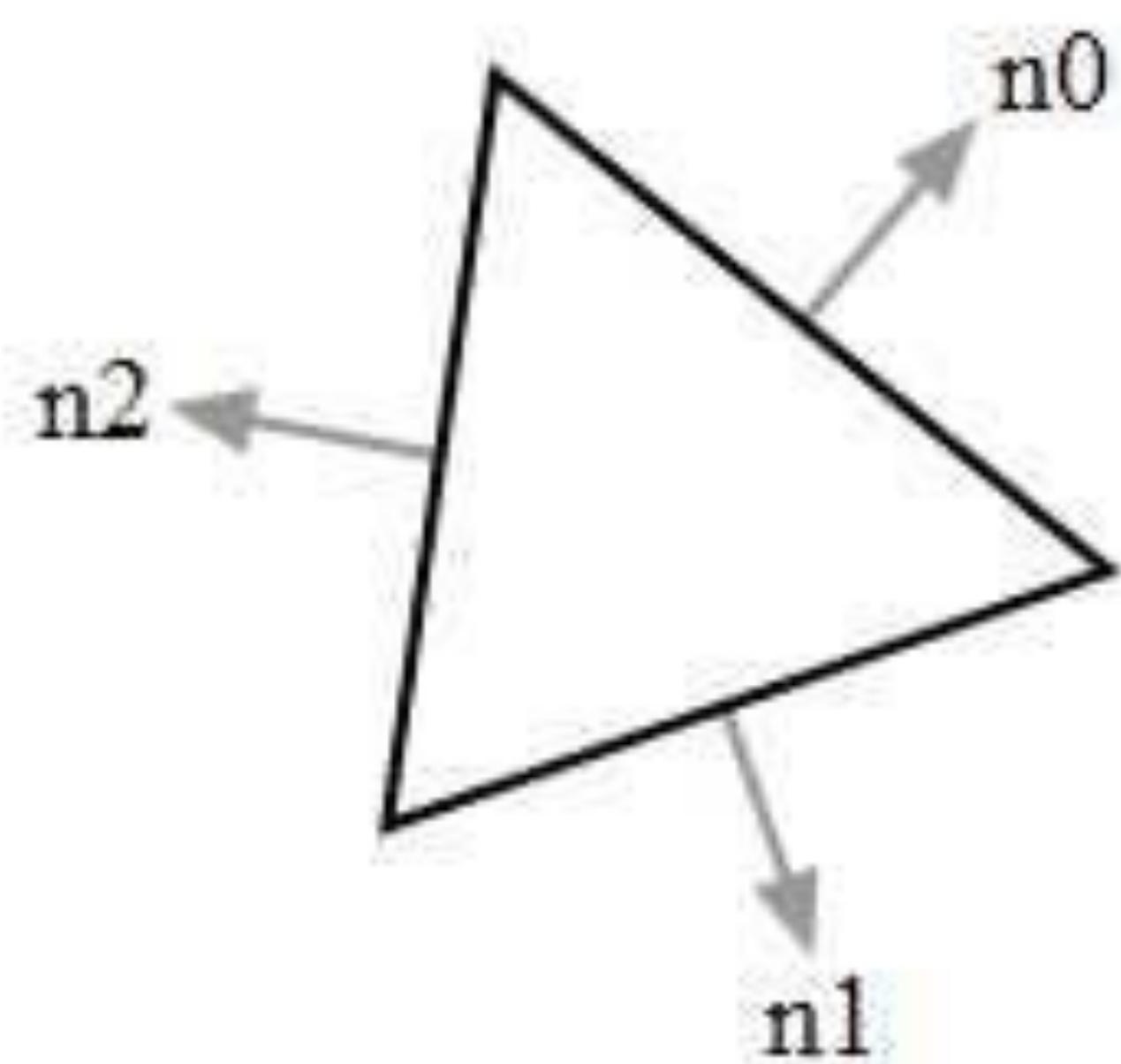
# Texture coordinates.

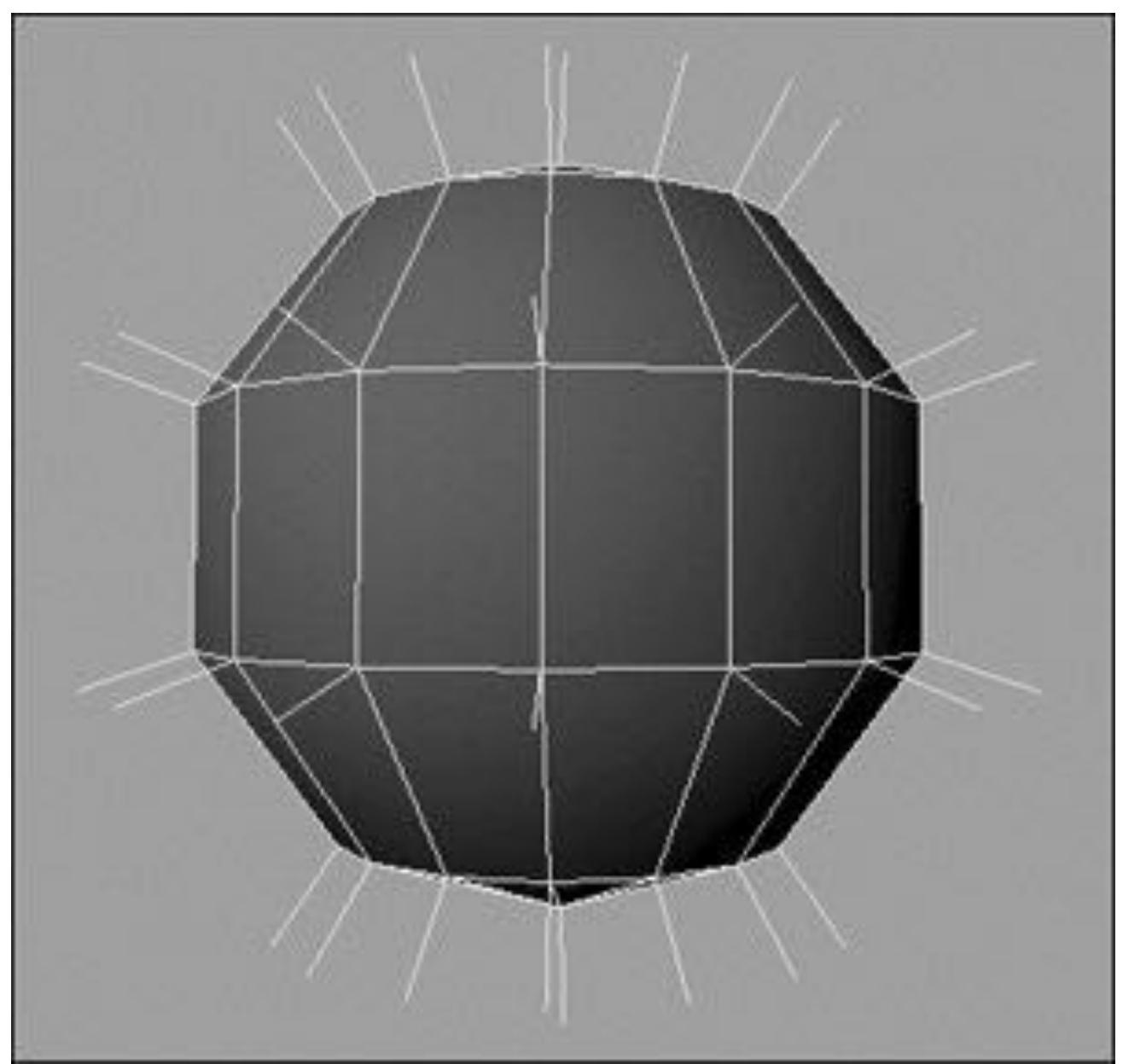
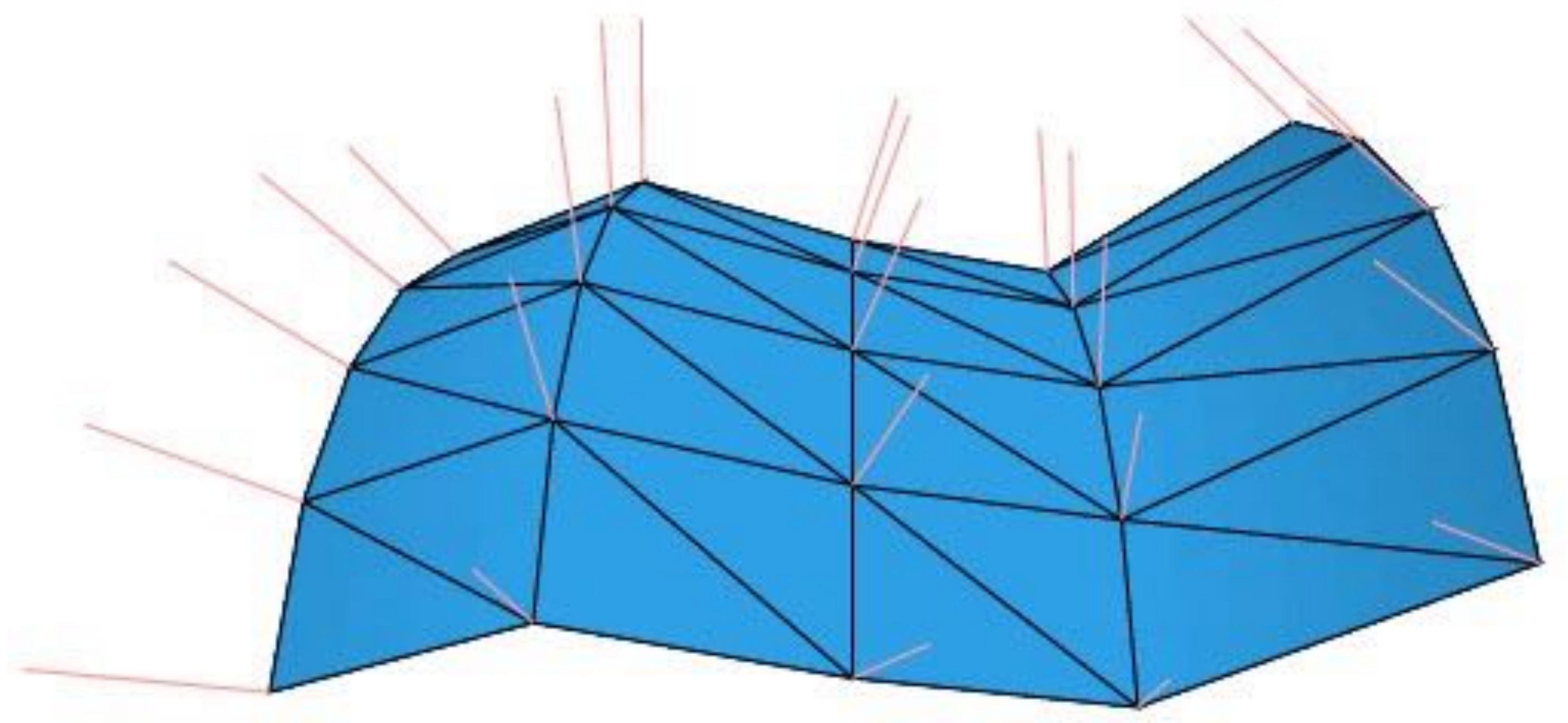




Copyright © 2012, Dylan Dunbar

# Vertex normals.





```
class Mesh {  
public:  
  
    void Render(ShaderProgram &program);  
    void loadOBJ(const char *fileName);  
  
    std::vector<float> positions;  
    std::vector<float> texcoords;  
    std::vector<float> normals;  
};
```

```
void Mesh::Render(ShaderProgram &program) {  
  
    glVertexAttribPointer(program.positionAttribute, 3, GL_FLOAT, false, 0, positions.data());  
    glEnableVertexAttribArray(program.positionAttribute);  
  
    glVertexAttribPointer(program.texCoordAttribute, 2, GL_FLOAT, false, 0, texcoords.data());  
    glEnableVertexAttribArray(program.texCoordAttribute);  
  
    glDrawArrays(GL_TRIANGLES, 0, positions.size()/3);  
  
    glDisableVertexAttribArray(program.positionAttribute);  
    glDisableVertexAttribArray(program.texCoordAttribute);  
}
```

# Loading meshes from OBJ files.

# Basic OBJ file structure

Indexes are **1-based**, V is **inverse** in Texture Coordinates.

```
# comment

# vertex
v -0.436826 -0.208037 1.060532

# texture coordinate
vt 0.859375 0.500000

# vertex normal
vn 0.707 0.000 0.707

# indices
# can reference vertex
# or vertex / texture coordinate
# or vertex / texture coordinate / normal
f 21/10/3 16/11/5 1/7/2
```

# Using tinyOBJLoader

```
#define TINYOBJLOADER_IMPLEMENTATION
#include "tiny_obj_loader.h"
```

```
void Mesh::loadOBJ(const char *fileName) {
    tinyobj::attrib_t attrib;
    std::vector<tinyobj::shape_t> shapes;
    std::vector<tinyobj::material_t> materials;
    std::string err;

    bool ret = tinyobj::LoadObj(&attrib, &shapes, &materials, &err, fileName, NULL, true);

    if(ret) {
        for(int i=0; i < shapes.size(); i++) {
            for(int j=0; j < shapes[i].mesh.indices.size(); j++) {

                unsigned int vertexOffset = shapes[i].mesh.indices[j].vertex_index * 3;
                unsigned int normalOffset = shapes[i].mesh.indices[j].normal_index * 3;
                unsigned int texOffset = shapes[i].mesh.indices[j].texcoord_index * 2;

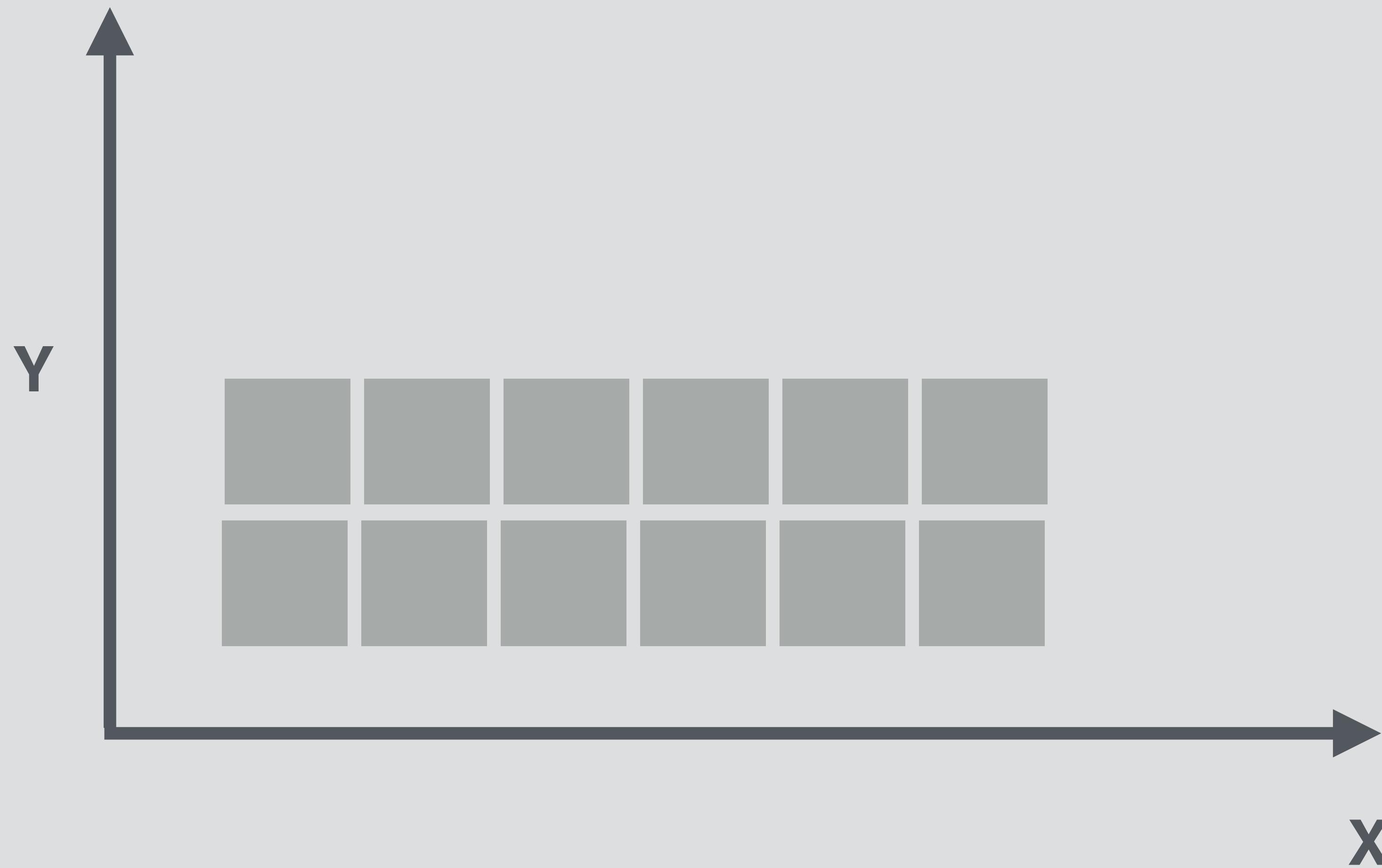
                positions.push_back(attrib.vertices[vertexOffset]);
                positions.push_back(attrib.vertices[vertexOffset+1]);
                positions.push_back(attrib.vertices[vertexOffset+2]);

                normals.push_back(attrib.normals[normalOffset]);
                normals.push_back(attrib.normals[normalOffset+1]);
                normals.push_back(attrib.normals[normalOffset+2]);

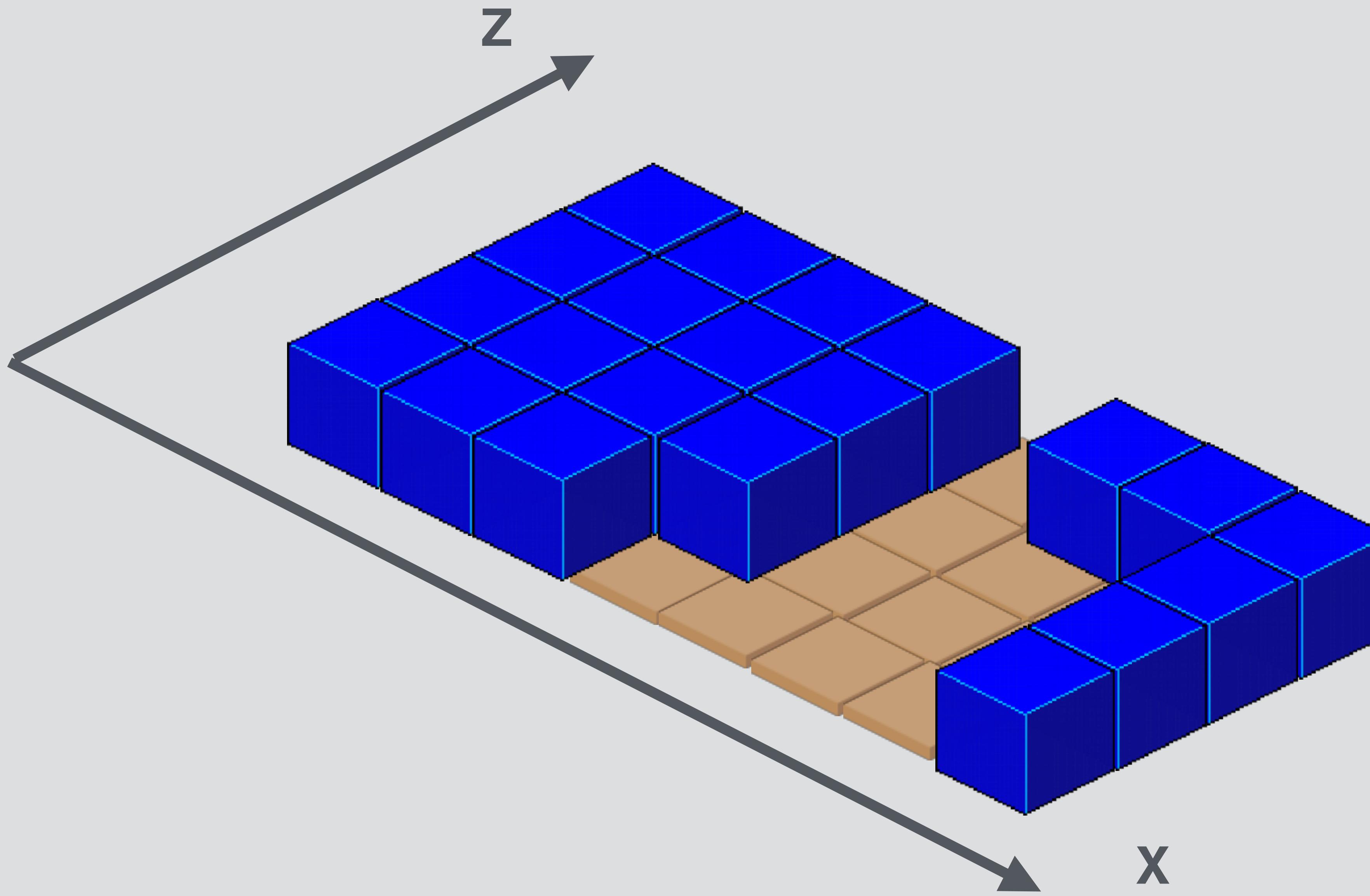
                texcoords.push_back(attrib.texcoords[texOffset]);
                texcoords.push_back(1.0f - attrib.texcoords[texOffset+1]);
            }
        }
    } else {
        std::cout << err << std::endl;
        assert(false);
    }
}
```

# Loading a 2D tilemap level in 3D

# In 2D



# In 3D





minehorse.com