中山大学计算机学院

计算机图形学

本科生实验报告

（2023学年秋季学期）
课程名称：Computers Graphics

教学班级	专业(方向)	学号	姓名
计科二班	计算机科学与技术	21307185	张礼贤

Task 1

- **什么是OpenGL?**
  - OpenGL是用于渲染2D和3D图形的跨平台图形编程接口。它提供了一组函数和命令，允许开发者创建各种图形效果，从简单的图形元素到复杂的三维场景，包含了一系列可以操作图形、图像的函数。然而，OpenGL本身并不是一个API，它仅仅是一个由Khronos组织制定并维护的规范
- **计算机图形学**
  - 计算机图形学是研究如何使用计算机生成、处理和显示图像的学科。它涵盖了从简单的2D绘图到复杂的3D建模和渲染的各种技术。
- **OpenGL与计算机图形学的关系是什么**
  - OpenGL与计算机图形学的关系是，OpenGL提供了一种编程接口和工具，使开发者能够在计算机上生成和处理图形。通过使用OpenGL，开发者可以绘制基本的图形、创建复杂的3D场景、进行纹理映射、光照效果、阴影等等。
  - OpenGL在计算机图形学领域被用作一个重要的工具和框架，帮助开发者实现各种视觉效果和图形交互

Task 2

修改着色器使得三角形的顶点上下颠倒

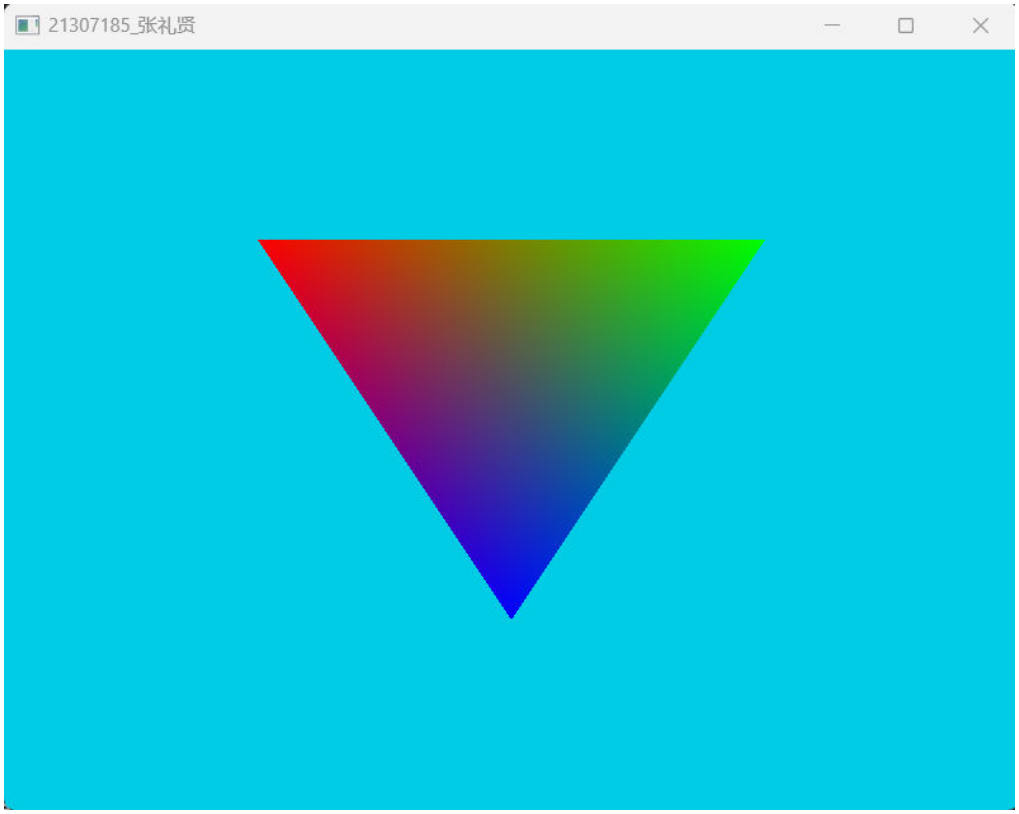
- 核心代码展示：

```
const char *vertexShaderSource = R"(
#version 330 core
layout (location = 0) in vec3 aPos;
layout (location = 1) in vec3 aColor;
out vec3 ourColor;
void main()
{
    gl_Position = vec4(aPos.x, -aPos.y, aPos.z, 1.0);
    ourColor = aColor;
}
)";
```

  - 在顶点着色器代码中，gl_Position 是一个特殊的内建变量，用于表示顶点的裁剪空间坐标。通常情况下，顶点着色器负责将顶点从模型空间（或局部坐标空间）变换到裁剪空间，以便进行透视投影和视口变换。

  - 在上述代码中，gl_Position 被设置为一个四维向量 vec4，其中 aPos 的 x 分量保持不变，y 分量取相反数，z 分量保持不变，w 分量为 1.0。这个操作可以实现反转或翻转三角形，从而达到实验效果。

- 实验结果展示：



Task 3

一、在箱子的角落里放置四个笑脸

- **核心代码展示：**

```
//main.cpp
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,
GL_REPEAT);

//4.2.texture.fs
#version 330 core
out vec4 FragColor;

in vec3 ourColor;
in vec2 TexCoord;

uniform sampler2D texture1;
uniform sampler2D texture2;

void main()
{
    FragColor = mix(texture(texture1, TexCoord),
texture(texture2, TexCoord / 2), 0.2);
}
```

- 在OpenGL中，纹理坐标通常是规范化的坐标，范围从0.0到1.0。当纹理坐标超出了这个范围时，就需要定义一种规则来决定超出范围的部分应该如何处理。
- 为了在箱子的角落里放置四个箱子，可以使用参数GL_REPEAT，这意味着当纹理坐标超出0.0到1.0的范围时，它们会被重复。具体来说，如果纹理坐标超出了1.0，它们将被"重复"，即1.1会变成0.1，1.2会变成0.2，以此类推。这就创建了一个平铺的效果，使纹理在超出1.0后继续重复。
- 而在片段着色器中，为了是笑脸重复而木箱不变化，可以将木箱的texcoord / 2，从而不会重复，形成四个笑脸在木箱的四个角。

- **实验结果展示：**

二、尝试其他几种不同的环绕方式

- GL_REPEAT(重复纹理图像)

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,
GL_REPEAT);
```

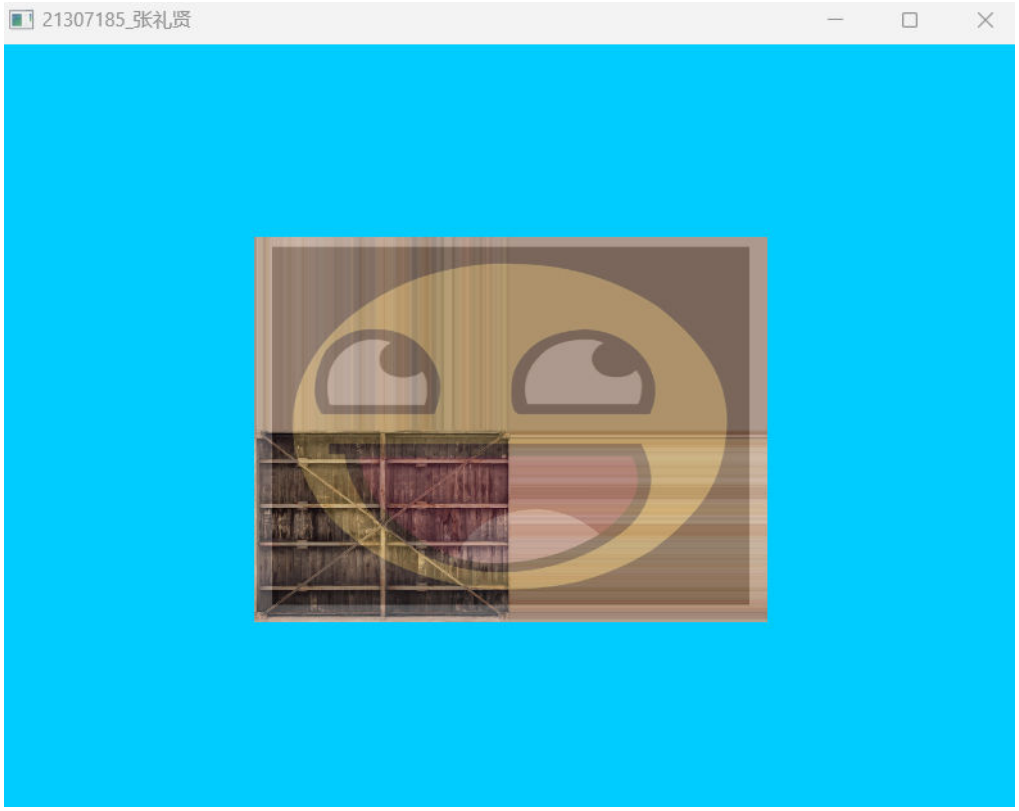
- GL_MIRRORED_REPEAT(镜像重复)

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
GL_MIRRORED_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,
GL_MIRRORED_REPEAT);
```

- GL_CLAMP_TO_EDGE(重复纹理坐标边缘)

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,
GL_CLAMP_TO_EDGE);
```

由于笑脸的边缘颜色不明显，这里选用了木箱作为边缘拉伸



- GL_CLAMP_TO_BORDER(指定超出坐标部分的颜色)

```
float borderColor[] = { 0.2f, 0.2f, 0.8f, 1.0f };
glTexParameterfv(GL_TEXTURE_2D,
GL_TEXTURE_BORDER_COLOR, borderColor);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,
GL_REPEAT);
```

Task 4

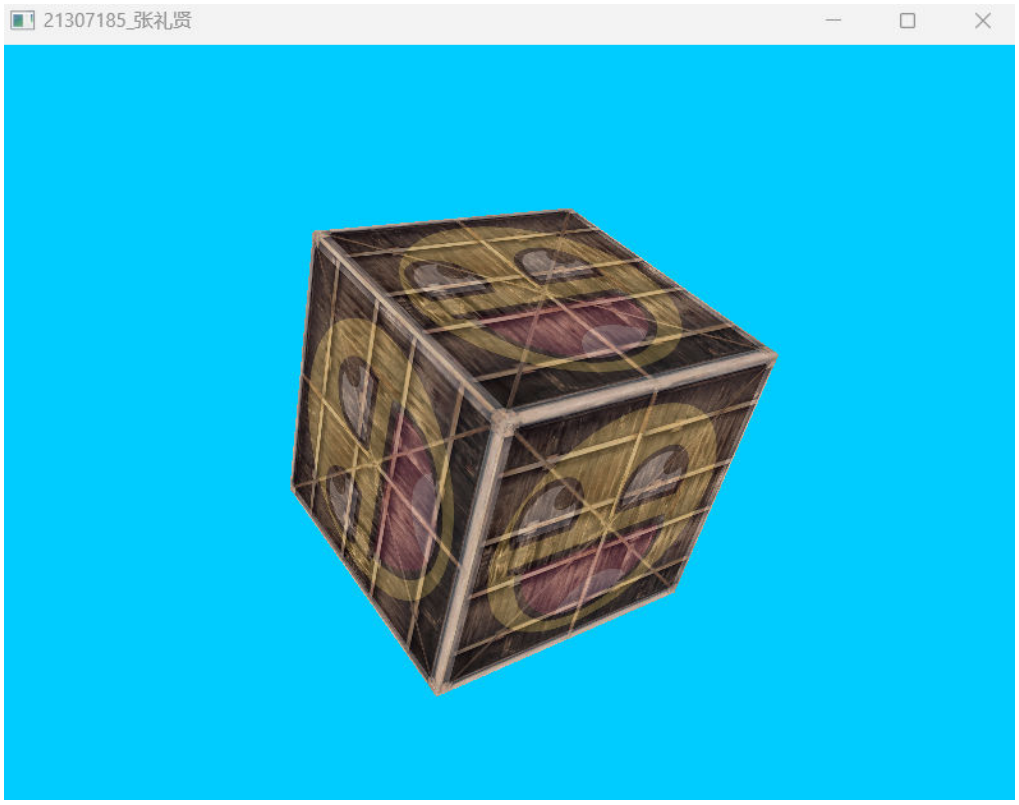
对GLM的projection函数中的FoV和aspect-ratio参数进行实验,并说明他们是如何影响透视平截头体的

- **参数说明:**

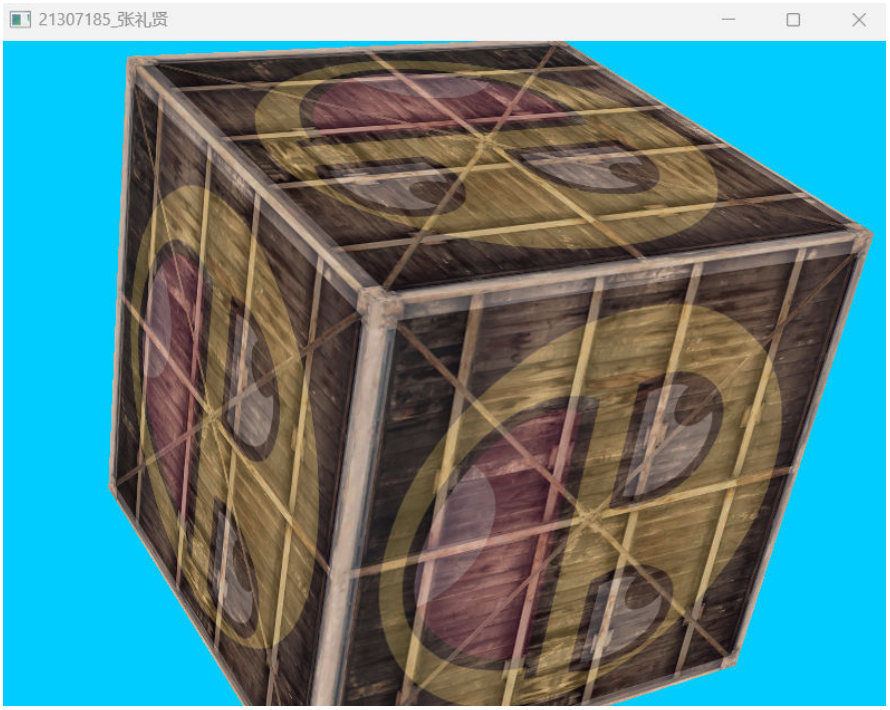
  1. **FoV（Field of View）**：视场角，它表示可见区域的张开程度，通常以角度为单位。视场角决定了你可以在一个单一帧内看到多少内容。较大的视场角将使你能够看到更多的东西，但可能导致透视效果更强烈。较小的视场角将产生更强的变焦效果，通常用于模拟远距离的摄像头或望远镜。

  2. **aspect-ratio**：纵横比，通常表示为屏幕的宽度与高度之比。它定义了你的视图空间的宽高比例，以确保图形在不同屏幕上能够正确呈现。通常情况下，你会将它设置为窗口的宽度除以窗口的高度，以适应当前窗口的纵横比。

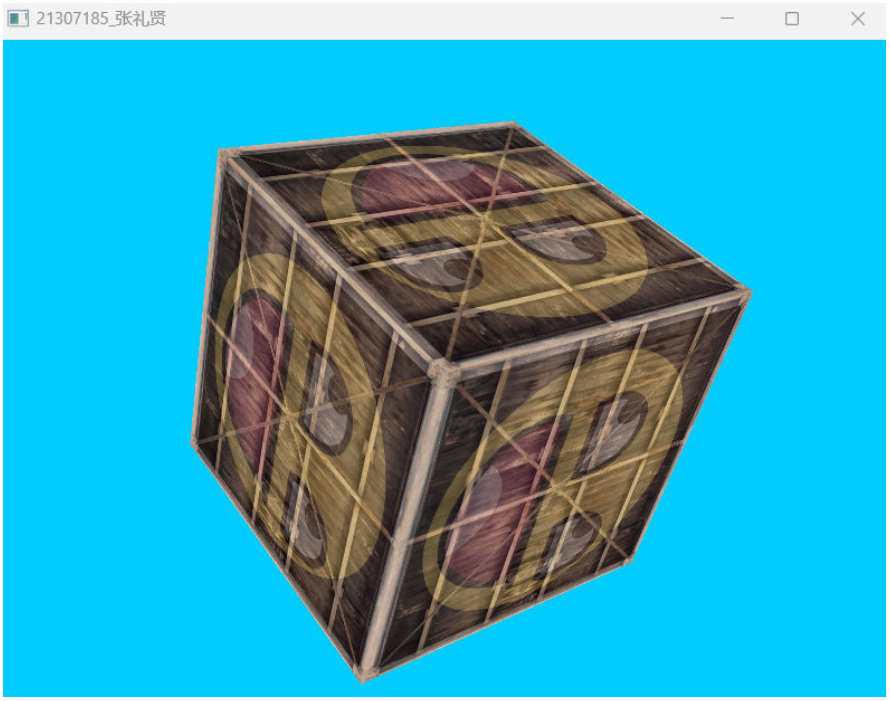
- **实验结果展示：** 基准图片：



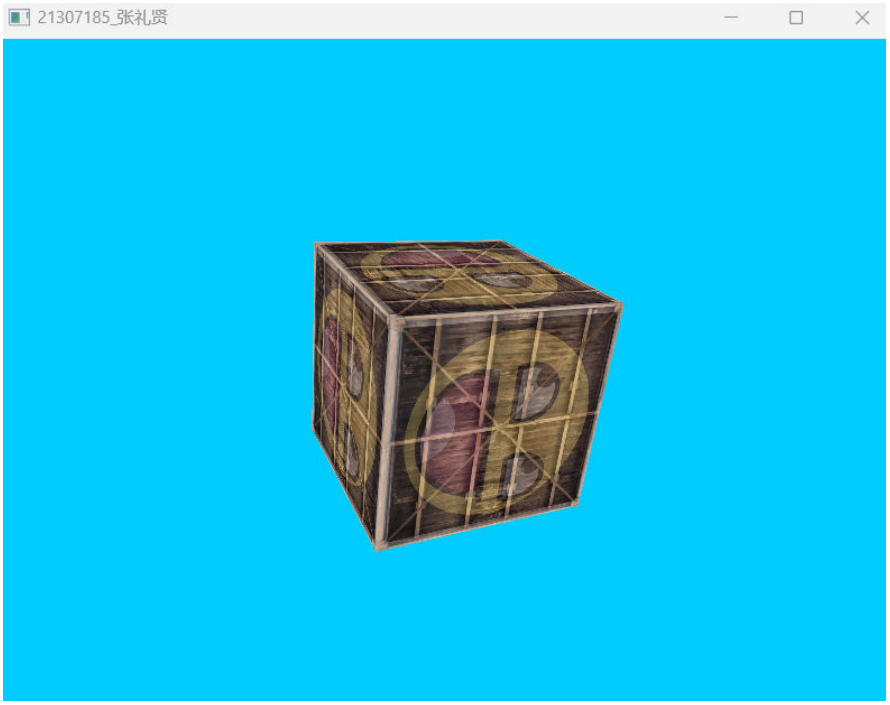
1. 改变 Fov(以 45 为基准)：
    - Fov = radians(25.0f)



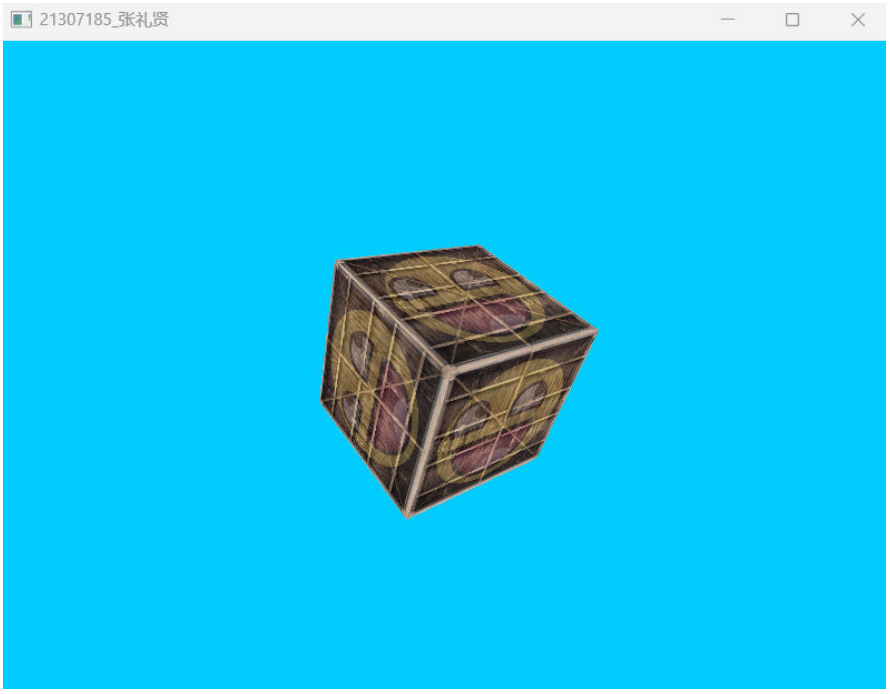
    - Fov = radians(35.0f)



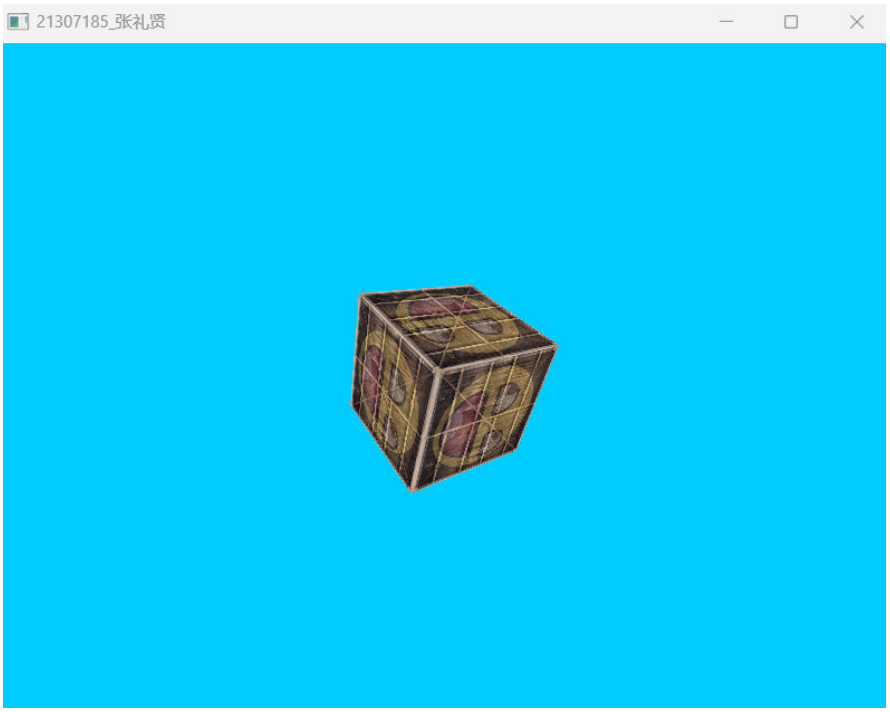
- Fov = radians(55.0f)



- Fov = radians(65.0f)



- Fov = radians(80.f)



2. 改变 aspect ratio(以4/3为基准)

- aspect ratio = (16 : 9)

- aspect ratio = (5 : 4)

- aspect ratio = (1 : 1)

- **实验结果分析：**

  1. 改变 FoV：

     - 增大 FoV：会扩大你能够看到的区域，使其更广阔。这会导致一种广角效果，让场景中的对象看起来较小，但能够看到更多周围的环境。
     - 减小 FoV：会缩小你的视野，使其更加聚焦。这会导致一种望远镜效果，让场景中的对象看起来较大，但只能看到很小的一部分。

  2. 改变 aspect ratio：

     - 增大 aspect ratio：会拉伸水平方向，使物体在水平方向上看起来更加宽广，而在垂直方向上看起来更加收缩。
     - 减小 aspect ratio：会拉伸垂直方向，使物体在垂直方向上看起来更加宽广，而在水平方向上看起来更加收缩。