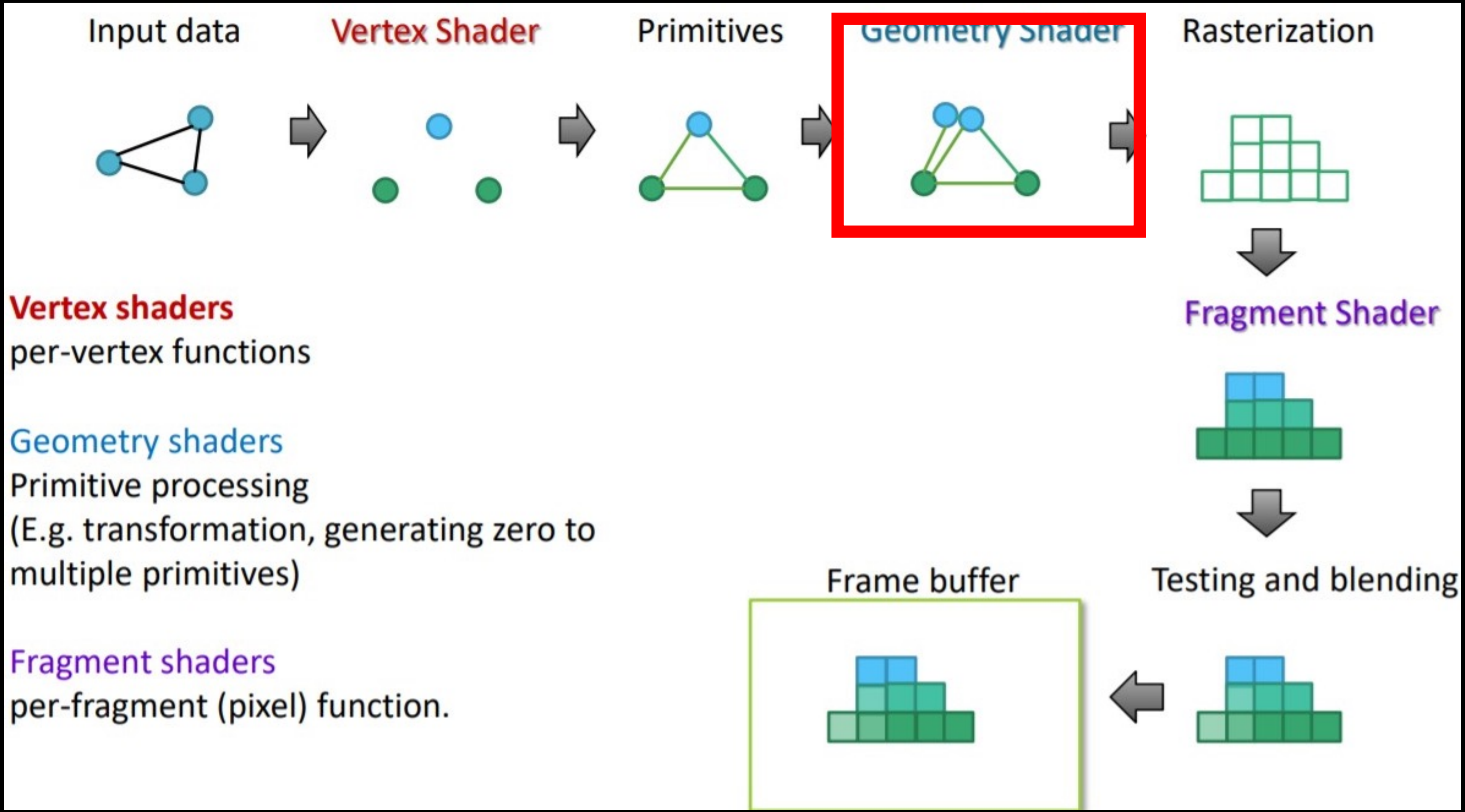
HW4

Geometry Shader

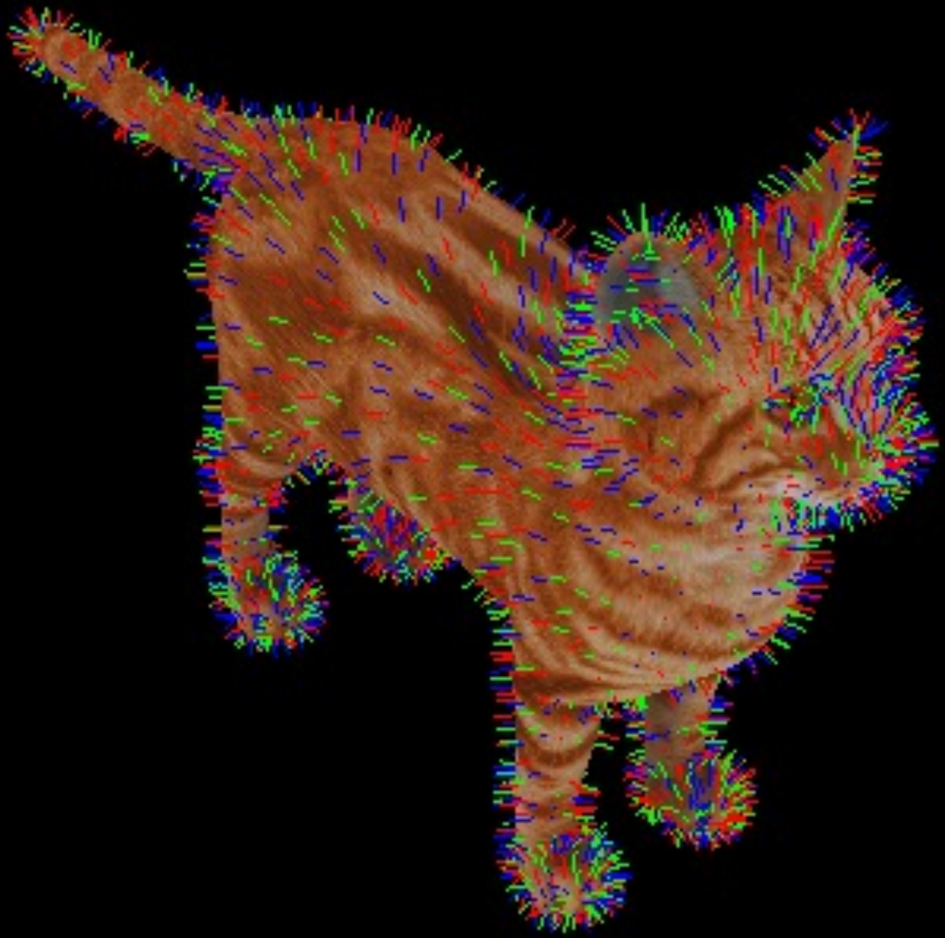


Vertex shaders
per-vertex functions

Geometry shaders
Primitive processing
(E.g. transformation, generating zero to multiple primitives)

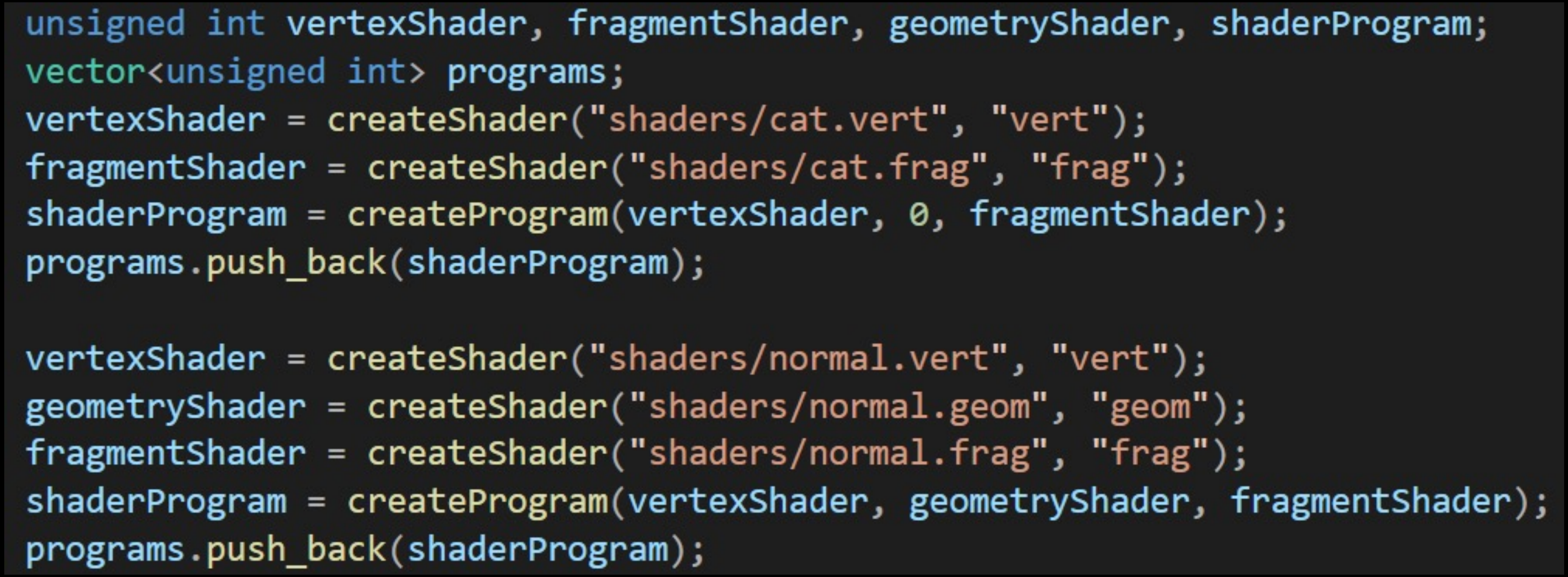
Fragment shaders
per-fragment (pixel) function.

Geometry Shader Example – normal visualizer



Geometry Shader

- GLuint createProgram(GLuint vert, GLuint geom, GLuint frag);
  If you don’t need the geometry shader, you can put “0” at geom



```
unsigned int vertexShader, fragmentShader, geometryShader, shaderProgram;
vector<unsigned int> programs;
vertexShader = createShader("shaders/cat.vert", "vert");
fragmentShader = createShader("shaders/cat.frag", "frag");
shaderProgram = createProgram(vertexShader, 0, fragmentShader);
programs.push_back(shaderProgram);

vertexShader = createShader("shaders/normal.vert", "vert");
geometryShader = createShader("shaders/normal.geom", "geom");
fragmentShader = createShader("shaders/normal.frag", "frag");
shaderProgram = createProgram(vertexShader, geometryShader, fragmentShader);
programs.push_back(shaderProgram);
```

Geometry Shader- declare the type of primitive input

- Declare the type of primitive input we're receiving from the vertex shader.
- Method ： Declaring a layout specifier in front of the ”in” keyword.

➢layout(primitive values) in;

primitive values	Rendering primitives(glDrawArrays)	Points per primitive
points	GL_POINTS	1
lines	GL_LINES or GL_LINE_STRIP	2
lines_adjacency	GL_LINES_ADJACENCY or GL_LINE_STRIP_ADJACENCY	4
Triangles	GL_TRIANGLES, GL_TRIANGLE_STRIP or GL_TRIANGLE_FAN	3
triangles_adjacency	GL_TRIANGLES_ADJACENCY or GL_TRIANGLE_STRIP_ADJACENCY	6

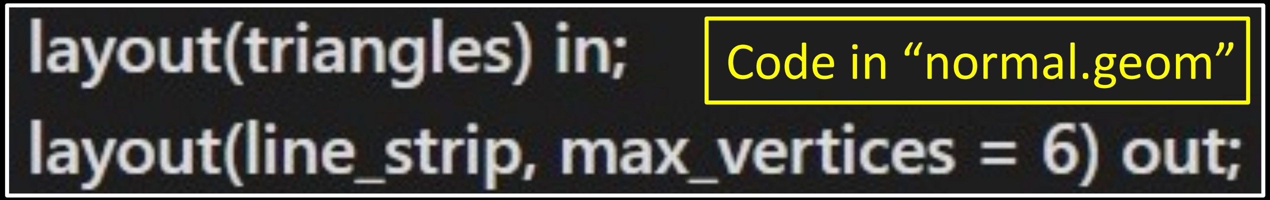
Geometry Shader- declare the type of primitive output

- We also need to specify a primitive type that the geometry shader will output.
- Method ： Declaring a layout specifier in front of the ”out” keyword.

➢layout(primitive values, max_vertices) out;

primitive values ： points, line_strip, triangle_strip

max_vertices ： If you exceed this number, OpenGL won't draw the extra vertices.



```
layout(triangles) in;
layout(line_strip, max_vertices = 6) out;
```

Code in “normal.geom”

Geometry Shader- update attributes to geometry shader

- We can update some attributes(color, normal) from vertex shader to the geometry shader.
- Method： Using an interface block.

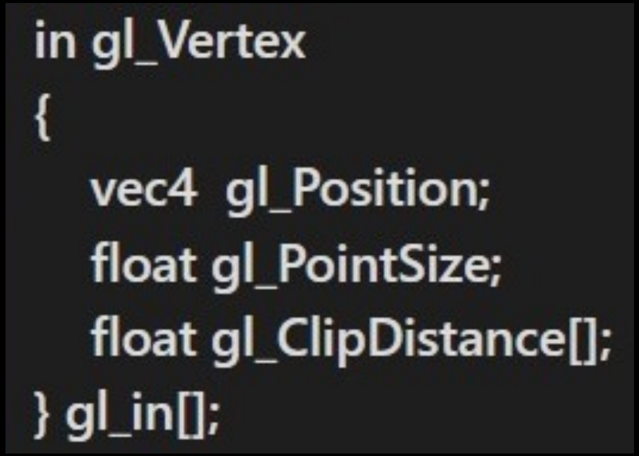
Code in vertex shader	Code in geometry shader

out VS_OUT {
vec3 normal;
//other attributes
} vs_out;
	in VS_OUT {
	vec3 normal;
	//other attributes
	} gs_in[];

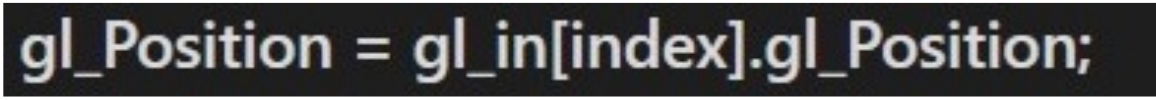
vs_out.normal	gs_in[index].normal (index： index for input vertices)

Geometry Shader- gl_in variable

- GLSL gives us a built-in variable called gl_in that internally (probably) looks something like this:



```
in gl_Vertex
{
    vec4  gl_Position;
    float gl_PointSize;
    float gl_ClipDistance[];
} gl_in[];
```

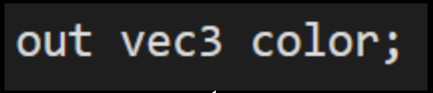


```
gl_Position = gl_in[index].gl_Position;
```

Code in “normal.geom”

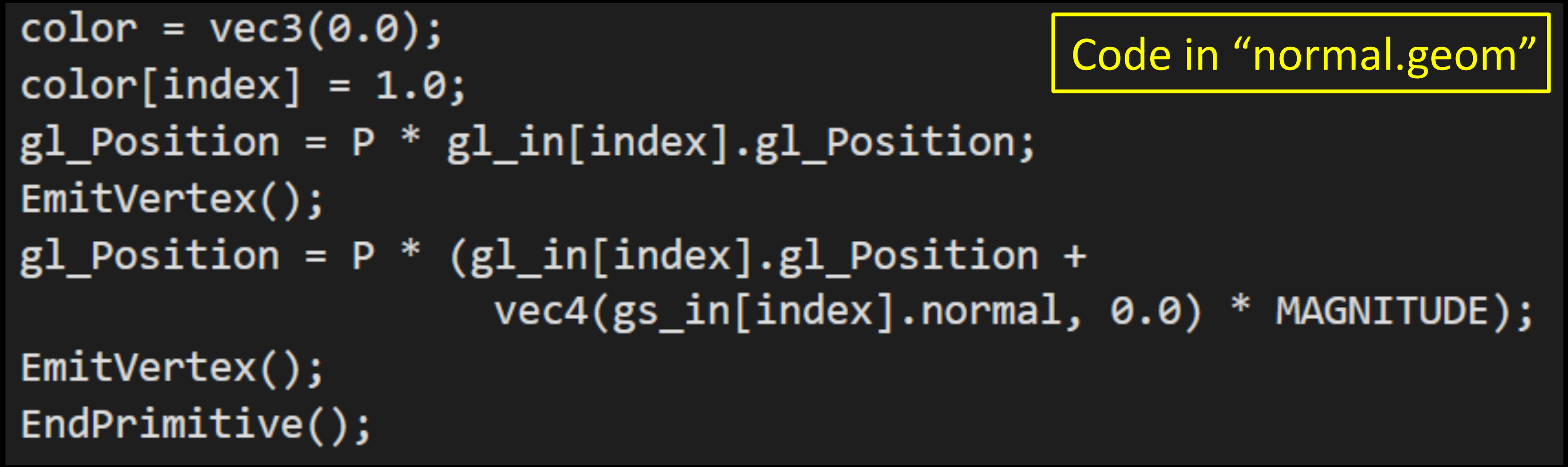
Geometry Shader- EmitVertex /EndPrimitive function

- Each time we call EmitVertex(), the vector currently set to gl_Position is added to the output primitive.
- Whenever EndPrimitive() is called, all emitted vertices for this primitive are combined into the specified output render primitive.



```
out vec3 color;
```

Data passed to fragment shader



```
color = vec3(0.0);
color[index] = 1.0;
gl_Position = P * gl_in[index].gl_Position;
EmitVertex();
gl_Position = P * (gl_in[index].gl_Position +
                   vec4(gs_in[index].normal, 0.0) * MAGNITUDE);
EmitVertex();
EndPrimitive();
```

Code in “normal.geom”

Reference： https://learnopengl.com/Advanced-OpenGL/Geometry-Shader

Load Model

- In obj file：(about face information)

  ```
  f vertex position/texture coordinate/normal
  f 1/1/1 473/2/2 1370/3/3                (3 vertice/primitive)
  f 1/1/1 473/2/2 1370/3/3 479/4/4        (4 vertice/primitive)
  f 1//1 473//2 1370//3                   (no texture coordinate)
  ```

- In Object.cpp file, the format of the face information must be f 1/2/3 or f 1//3. (f 1/3 cannot be read.)

  You can modify Object.cpp or write another code to read obj file.

- In geometry shader, you cannot render the object with glDrawArrays(GL_QUADS).

  You can put the quad into two triangles with another code by yourself.

HW4 - Animation with Three Types of Shaders

Homework 4- Goal

1. Make a 30~60 seconds video.
   Play the animation and introduce the features of the video and technique you have used.
2. Theme : Animation with Three Types of Shaders
3. Must include：
   (1) At least an object
   (2) Geometry shader
   (3) Generate at least one extra vertex

* You can refer to the examples on the Internet, but you must mention it in the introduction part of the video and cite the original source.

Homework 4- Recording tools

1. Screen recording :
   OBS : https://obsproject.com/

2. Introduce your video :
   (1) PowerPoint
   (2) Other video editing tools

Homework 4- Score

1. Creativity/ Richness/technical difficulty (40%)
2. Your code is executable (30%)
3. Votes from classmates (30%)
   (We will provide a Google sheet and let you choose 5 best videos)

*Requirements for geometry shader：
(1)You should do a different effect from the example code we provided, or your score will be zero.
(2) Developing a simple function with Geometry shader can meet the basic requirement.

Homework 4- Upload Format and Rules

1. Upload your video to Youtube.
2. Please hand in your video link and the whole project to e3 platform.
   File name: studentId_hw4.zip
   *If your uploading format doesn’t match our requirement, there will be penalty to your score. (-5%)
3. DeadLine: 2023/ 1 / 6 23: 59:59
4. If you submit your homework late, the score will be 0.
5. Use geometry shader to do this homework, otherwise you’ll get zero points.