Computer Graphics Homework 2 Report

Created	@November 10, 2022 9:51 AM
Last edited time	@November 10, 2022 5:02 PM
Related to 記事 (Column)	
Tags	作業
URL	

Implementation

Load Model from Obj File

In Obj File, there are total four type of data to deal.

- `v` vertex position data
- `vn` normal direction data
- `vt` texture coordinate data
- `f` Indicate an face in the object by the format of `<index of v>/<index of vt>/<index of vn>`

To convert the obj file data to `Class Model;` , When reading Obj file, I store `v`, `vn`, `vt` to to three `vector<glm::vec3>` , and access them by index provided by `f` to load model.

Basic Texture Shader

I refer to `example.cpp` to prepare `VBO` object to pass `position`, `normal`, `texcoord` data to shader. Because OpenGL is a state machine, when we want to set something like buffer, we need to bind it first, instructions following will apply to current binding buffer.

```
GLuint VBO[3]; // position, normal, texCoord
glGenBuffers(3, VBO);
glBindBuffer(GL_ARRAY_BUFFER, VBO[0]);
glBufferData(GL_ARRAY_BUFFER, sizeof(float) * model->positions.size(), model->positions.data(), GL_STATIC_DRAW);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), (void*)0);
glEnableVertexAttribArray(0);
```

`glBufferDate()` move data to buffer, `glVertexAttribPointer()` describes the data format of the buffer. The first paremeter `0` in `glVertexAttribPointer(0, ...)` and `glEnableVertexAttribArray(0)` match `0` in `layout(location = 0) in vec3 position;` in `basic.vert`, which mean the buffer data will be use as position.

The orginal code given by TA has implemented texture loading function, I just active and bind texture in `doMainLoop()` by the following code.

```
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, model->textures[ctx->objects[i]->textureIndex]);
```

To pass `texcoord` to fragment shader from vertex shader, use `TexCoord = texCoord;` .

In fragment shader `basic.frag`, I use `texture(ourTexture, TexCoord)` to get texture color.

Create Plane Model with Texture

This part is trivial, refer to example code and check the structure inside `Model` and `Object` class, assign correct attribute to them.

```
void loadModels(){
  ...
  m = new Model();
  float w = 8.192/2;
  float h = 5.12/2;
  float positions[] = {w, 0, h, w, 0, -h, -w, 0, -h, -w, 0, h};
  float normals[] = {0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0};
  float texcoords[] = {2, 2, 2, 0, 0, 0, 0, 2};
  for (auto& it : positions) m->positions.push_back(it);
  for (auto& it : normals) m->normals.push_back(it);
  for (auto& it : texcoords) m->texcoords.push_back(it);
  m->textures.push_back(createTexture("../assets/models/Wood_maps/AT_Wood.jpg"));
  m->numVertex = 4;
  m->drawMode = GL_QUADS;
  ctx.models.push_back(m);
}

void setupObjects(){
  ...
  ctx.objects.push_back(new Object(2, glm::translate(glm::identity<glm::mat4>(), glm::vec3(4.096, 0, 2.56))));
}
```

Light Shader

Pass `light` `Material` `viewPos` `ModelNormalMatrix` to shader by folloing function.

```
GLint loc = glGetUniformLocation(programId, "{var name}");
glUniform{n}{f/i}(loc, {value});
```

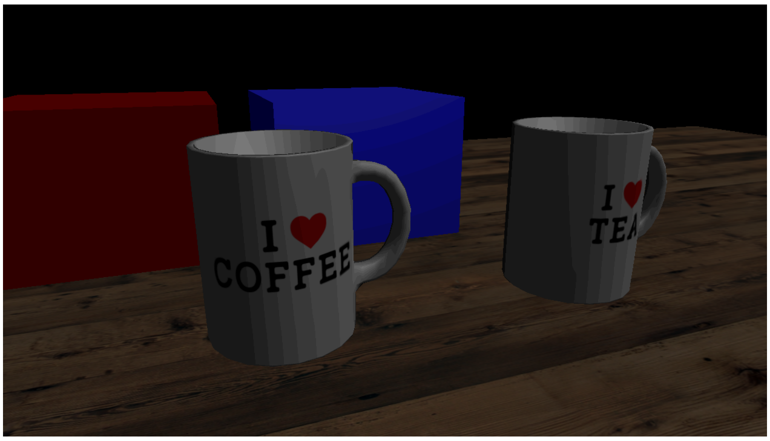
`ModelNormalMatrix` can be got by $(Model^{-1})^T$

```
const float* modelNormalMatPtr = glm::value_ptr(glm::transpose(glm::inverse(modelMatrix)));
```

In shader, calculate color based on TA's comment in code. However, I didn't notice that I should reverse light direction to make them point out from fragment, this mistake take me a lot of time to find out.

Bouns Toon Style Shader

Presss **C** to active toon style with light shader on.



Ladder Mapping

Just make the value of color to be discrete.

```
int x = int(color.x * 255) / 8; x *= 8;
int y = int(color.y * 255) / 8; y *= 8;
int z = int(color.z * 255) / 8; z *= 8;
color = vec4(float(x)/255, float(y)/255, float(z)/255, 1.0);
```

Silhouette

If the dot product of `normal` and `camera.front` is close to zero, which mean the face is perpendicular to camera, the face is sihoutte.

```
float edge = dot(normalize(Normal), normalize(-cameraDir));
if(edge <= 0.05){
    color *= 0.5;
}
```

Reference:

- 卡通渲染（上） https://zhuanlan.zhihu.com/p/25595069
- 卡通渲染（下） https://zhuanlan.zhihu.com/p/25939794