3GC3 Fall 2023 - Assignment 3
Timmy and the Bucket are at Disco

In assignment 3, we will still use tinyobj to load mesh but add color and lighting to the scene.

Figure 1. Timmy and the bucket are at Disco

Task 1: Use Texture for Mesh Color

1.1 Load Obj mesh with texture coordinate information

In assignment 2, we talked about obj mesh format. In addition to vertex positions and normals, obj format includes texture coordinates as well, i.e. lines that start with `vt` define the texture coordinate list.

Similar to the way we load vertex positions and normals, we can load texture coordinate (uv) information to the buffer for OpenGL rendering.

```
for(auto id : shapes[0].mesh.indices)
  {
      int vid = id.vertex_index;
```

```
        int nid = id.normal_index;
        int tid = id.texcoord_index;

        //fill in vertex positions
        vbuffer.push_back(attrib.vertices[vid*3]);
        vbuffer.push_back(attrib.vertices[vid*3+1]);
        vbuffer.push_back(attrib.vertices[vid*3+2]);
        //normal
        nbuffer.push_back(attrib.normals[nid*3]);
        nbuffer.push_back(attrib.normals[nid*3+1]);
        nbuffer.push_back(attrib.normals[nid*3+2]);
        //tex coord
        tbuffer.push_back(attrib.texcoords[tid*2]);
        tbuffer.push_back(attrib.texcoords[tid*2+1]);
    }
```

1.2 Load Texture Image using stb_image.h

Along with obj meshes, in assignment 3, texture images are provided for these meshes, which are either in png, jpg or jpeg format. We will load these images in code for mesh coloring.

To load images, include the provided header file stb_image.h in your OpenGL project. Please add the following declarations in the cpp file that uses the stb_image.h.

```
#define STB_IMAGE_IMPLEMENTATION
#include "stb_image.h"
```

After the declaration, you can start using functions exposed in stb_image.h. To load any image, you can use the following code:

```
int width, height, nrChannels;
stbi_set_flip_vertically_on_load(true);
unsigned char *data = stbi_load(tex_path.c_str(), &width, &height, &nrChannels, 0);
```

It loads the image at path tex_path into an unsigned char array data. Extra information about the image is in width, height, nrChannels.

1.3 Use Texture Color for Rendering

We've loaded the texture coordinates into tbuffer in 1.1 and loaded the texture image into data in 1.2. Now we need to set up the rendering with the texture coord and texture image.

First you need to pass the texture coordinate tbuffer to the vertex shader, the same way as vertex positions and normals.

In the OpenGL **main** program, generate texture id, bind texture and set the texture parameters, and associate it with the texture image content at data.

```
unsigned int textureT;
glGenTextures(1, &textureT);
glBindTexture(GL_TEXTURE_2D, textureT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB,
GL_UNSIGNED_BYTE, data);//define the texture using image data
stbi_image_free(data);//don't forget to release the image data
```

In the **fragment shader**, add **uniform sampler2D** myTexture; and in the main function of the fragment shader, use FragColor = texture(myTexture, UV); to retrieve texel color at tex coord UV, and use it for the final FragColor. Note that UV here is the texture coordinate you passed to the shader from tbuffer.

The above highlighted code snippet is provided for example purposes. You are free to work on your own implementation. https://learnopengl.com/Getting-started/Textures provides very detailed and easy-to-follow explanations about how to use textures.

Table 1. Rendering Configurations

Background color	(0.3f, 0.4f, 0.5f, 1.0f)
Window Size	Window Width = 1024; Window Height = 768;
Model Matrix	Identity
View Matrix	glm::mat4 view = glm::lookAt(glm::vec3(50,100,200), glm::vec3(0,80,0), glm::vec3(0,1,0));
Projection Matrix	glm::mat4 proj = glm::perspective(glm::radians(60.0f), 4.0f / 3.0f, 0.1f, 1000.0f);
Others	Cull backface + Depth Test with GL_LESS

Please render the following objs with textures in the scene (make sure that texel color is used for the FragColor in your fragment shader).

- Timmy.obj as the mesh + Timmy.png as the texture, save the rendered result to timmy_tex.ppm
- Bucket.obj as the mesh + Bucket.jpg as the texture, save the rendered result to bucket_tex.ppm
- Floor.obj as the mesh + Floor.jpeg as the texture, save the rendered result to floor_tex.ppm
- Bucket on the floor in the scene, and save the rendered result to bucketOnFloor_tex.ppm

Figure 2. Result image from a different view direction.

Task 2: Shading with Point Light

We will add a simple point light to the scene. In the fragment shader, use the Phong model to compute the FragColor (ignore the specular part), which is the following simplified formula:

ambient light color * obj color + diffuse light color * obj color * max(0, n·l)

You may pass the light position, color information to your fragment shader for color computation e.g. **uniform vec3** lightPos; or hard code some parameters.

Configuration 1: point light position (300, -10, -100), diffuse color (1, 1, 1) attenuated by distance with kc = 1, kl =0.35*1e-4, kq = 0.44*1e-4, and ambient color (0.2, 0.2, 0.2) with no attenuation. Render only Timmy in the scene, and save the rendered image to timmy_point1.ppm. Render only the bucket in the scene, and save the rendered image to bucket_point1.ppm.

Configuration 2: point light position (-50, 0, 300), diffuse color (1, 1, 1) attenuated by distance with kc = 1, kl =0.007*1e-4, kq = 0.0002*1e-4, and ambient color (0.2, 0.2, 0.2) with no attenuation. Render only Timmy in the scene, and save the rendered image to timmy_point2.ppm. Render only the bucket in the scene, and save the rendered image to bucket_point2.ppm.

Now let's move the light source position in the while loop, set theta = 0 initially before the while loop in the main function for rendering. Inside the loop, increase theta by 0.05f each time.

```
while (!glfwWindowShouldClose(window))
{
        //….
        glm::vec3 lightPos(sin(theta)* 150, 100, cos(theta)*150);
        theta += 0.05f;
        // set lightPos for rendering
}
```

Light diffuse color (1, 1, 1), attenuate with coefficient kc = 1, kl =0.007*1e-4, kq = 0.0002*1e-4, ambient color (0.2, 0.2, 0.2) with no attenuation. Render only Timmy in the scene, the running program will show the moving light shed on the mesh. Capture a video of the rendering window that lasts at least 8 sec, and save the video to timmy_point_dyn.*. Render only the bucket in the scene, and capture a video of the rendering window that lasts at least 8 sec, save the video to bucket_point_dyn.*. * here indicates the video format, such as mp4, mov, and etc.

Figure 3. Lighted example with everything in the scene with different light configuration and view angle.

Task 3: Shading with Spotlights

Remove the point light in Task2 from the scene. Add three spotlights in the scene to simulate Disco lighting effect. Use the same Phong model formula for color computation, and thus you can largely reuse the code in Task2, by just summing over contributions from the three lights.

.

Table 2. Spot light configuration

	Diffuse Color	Attenuation	Ambient Color	Position	Spot Direction	Cut-off Angle
Spotlight R	(1, 0, 0)	kc = 1, kl =0.35*1e-4, kq = 0.44*1e-4	(0.2, 0.2, 0.2) no attenuation Note: each spotlight has its ambient contribution	(0, 200, 0)	(50, -200, -50)	M_PI/6.0f
Spotlight G	(0, 1, 0)				(-50, -200, -50)	
Spotlight B	(0, 0, 1)				(0, -200, 50)	

Put Timmy, the bucket and the floor all into the scene. Render the scene, capture a ppm image and save it to “at_disco.ppm”. Figure 1. shows the example result in a different view direction.

Now let’s spin the spotlights around the positive y-axis counterclockwise by theta each time in the while loop in the main function.

```
while (!glfwWindowShouldClose(window))
{
        //….
```

```
        //rotate spotDirR, spotDirG, spotDirB around y-axis by theta
        theta += 0.05f;
        // set spotDir for rendering
}
```

Put Timmy, the bucket and the floor all into the scene. Capture a video of the rendering window for at least 8 seconds, and save it to “at_disco_dyn.*”

Submission Checklist

Submit your **code** together with all the captured ppm images and videos.

3 points - “timmy_tex.ppm”
3 points - “bucket_tex.ppm”
3 points - “floor_tex.ppm”
5 points - “bucketOnFloor_tex.ppm”

5 points - “timmy_point1.ppm”
5 points - “bucket_point1.ppm”
5 points - “timmy_point2.ppm”
5 points - “bucket_point2.ppm”
7 points - “timmy_point_dyn.*”
7 points - “bucket_point_dyn.*”

15 points - “at_disco.ppm”
20 points - “at_disco_dyn.*”

17 points - source code

Submission deadline is Nov. 30th, 2023.

Acknowledgement:

The asset of Timmy is from https://www.mixamo.com
The assets of bucket and floor are from https://polyhaven.com