Lab 8: Fake Shadow

The purpose of this lab is for you to learn how to draw fake shadows based on location of a light source, shapes of objects, and the plane that the shadow will be on. **IMPORTANT**: For this lab, you must already have a working implementation of the `look_at()` function. If your `look_at()` function does not work yet, you can use rotation about x-axis for a small positive degree to see shadows.

Shader Programs

For this project, you are going to use a very simple shader programs that incorporate a model view matrix as shown below:

`vshader.glsl`

```
#version 120

attribute vec4 vPosition;
attribute vec4 vColor;
varying vec4 color;

uniform mat4 model_view_matrix;
uniform vec4 light_position;
uniform int is_shadow;

int main()
{
    if(is_shadow == 0)
    {
        gl_Position = model_view_matrix * vPosition;
        color = vColor;
    }
    else
    {
        float x = ...;
        float z = ...;
        gl_Position = model_view_matrix * vec4(x, 0, z, 1);
        color = vec4(0, 0, 0, 1);
    }
}
```

Note that inside the `vshader.glsl`, you can access each element of a vector using `.x`, `.y`, `.z`, and `.w`. For example, to access the `x` element of a vertex position, use `vPosition.x`. Similarly, to access the `y` element of the `light_position`, use `light_position.y`.

`fshader.glsl`

```
#version 120
```

```
varying vec4 color;

int main()
{
    gl_FragColor = color;
}
```

Again, the `model_view` matrix in the vertex shader should be the result of your `look_at()` function. If you want to, you can also incorporate projection matrix (either `ortho()` or `frustum()` function).

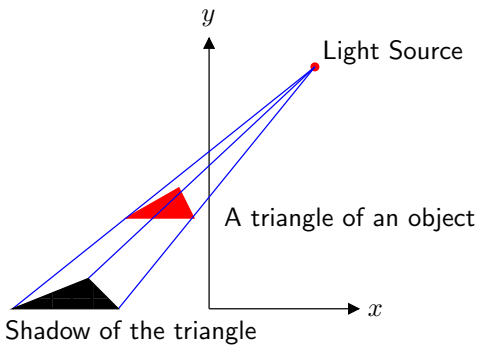
Building the World

For simplicity, create two objects in your work, a sphere and a cube with the following specifications:

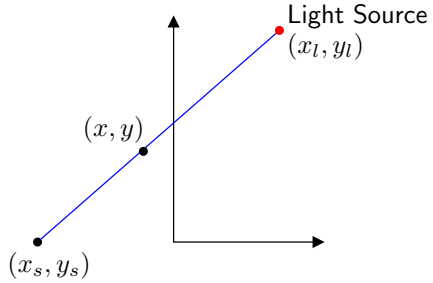
- **Sphere**: radius of the sphere should be 0.5 and the center of the sphere should be at $(-0.5, 0.25, 0.0)$. Your cube can have any colors that you like. Note that this sphere is sitting on the plane $y = 0$.
- **Cube**: size of this cube should be $0.5 \times 0.5 \times 0.5$ and the center of the cube should be at $(0.5, 0.25, 0.0)$. For the cube, make sure each side has different color. Similarly to the sphere, this cube is sitting on the plane $y = 0$.

Create Fake Shadows

In our case, a shadow of an object will be a flat object with black color that lies on the plane $y = 0$. We are going to create a fake shadow based on projection as shown below:



The location of a vertex of a shadow (x_s, y_s) causes by the light source located at (x_l, y_l) and the location of a vertex of an object (x, y) can be easily calculate. Consider the projection below:



From the above figure, the ratio $\frac{y_l - y}{x_l - x}$ and the ratio $\frac{y_l - y_x}{x_l - x_s}$ are the same. We can use the fact to solve x_s:

$$\frac{y_l - y}{x_l - x} = \frac{y_l - y_s}{x_l - x_s}$$
$$x_l - x_s = (y_l - y_s)\frac{x_l - x}{y_l - y}$$
$$x_s = x_l - (y_l - y_s)\frac{x_l - x}{y_l - y}$$

Since we are projecting to the play $y = 0$, $y_s = 0$ which makes the above equation becomes:

$$x_s = x_l - y_l\frac{x_l - x}{y_l - y}$$

The z coordinate can be calculated using the same formula:

$$z_s = z_l - y_l\frac{z_l - z}{y_l - y}$$

Thus, if we have a light source at (x_l, y_l, z_l), the projection of a point (x, y, z) onto the plane $y = 0$ will be located at

$$(x_l - y_l\frac{x_l - x}{y_l - y}, 0, z_l - y_l\frac{z_l - z}{y_l - y})$$

Note that we need to project all vertices of an object onto the plane $y = 0$. This can be done in the vertex shader. Thus, you need to draw twice, one for the object, and the other for shadow. Your display function should look like the following:

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

glUniform1i(is_shadow_location, 0);
glDrawArrays(GL_TRIANGLES, 0, num_vertices);

glUniform1i(is_shadow_location, 1);
glDrawArrays(GL_TRIANGLES, 0, num_vertices);

glutSwapBuffers();
```

Do not forget to use `glGetUniformLocation` for all uniform variables in the vertex shader and send your model view matrix as well as the light position into the graphic pipeline. You can either send those every time the `display()` function is called or just once inside the `init()` function. Note that the vshader program shown above is not completed. You need to put your calculation for `x` and `z`.

Requirements

When your program runs, it should as a user for the following information:

- an eye point (eye_x, eye_y, eye_z),
- an at point (at_x, at_y, at_z), and

- a location of the light source (x_l, y_l, z_l)

Simply use `scanf()` function to read in three floating pint at a time. For example

```
float eye_x, eye_y, eye_z;

printf("Please enter an eye point: ");
scanf("(%f,%f,%f)", &eye_x, &eye_y, &eye_z);
```

What a user needs to do is to enter three floating-point numbers as shown below:

```
Please enter an eye point: (0.0, 0.1, 0.1)
Please enter an at point: (0.0, 0.0, 0.0)
Please enter an light location: (0.0, 5.0, 0.0)
```

After you receive all information, use the eye point and at point to generate the model view matrix using your `look_at()` function, use the location of the light source to generate shadow vertices. This process must be done before you transfer data into the graphics pipeline.

HINT

- Do not forget the if a light source very close to an object, the shadow will be huge. So, put your light source a little further.
- Since we did not incorporate the projection matrix, you can only see ± 1 (OpenGL canonical view volume). Make sure you eye point is not too far from the objects. Otherwise, you will not be able to see anything.

Due Date and Submission

For the due date, please check the lab in the CourseWeb. Submit your program (zip all files) to the CourseWeb under this lab by the due date. You also need to run this lab in front of the TA to verify that it works correctly. **No late submission will be accepted**.