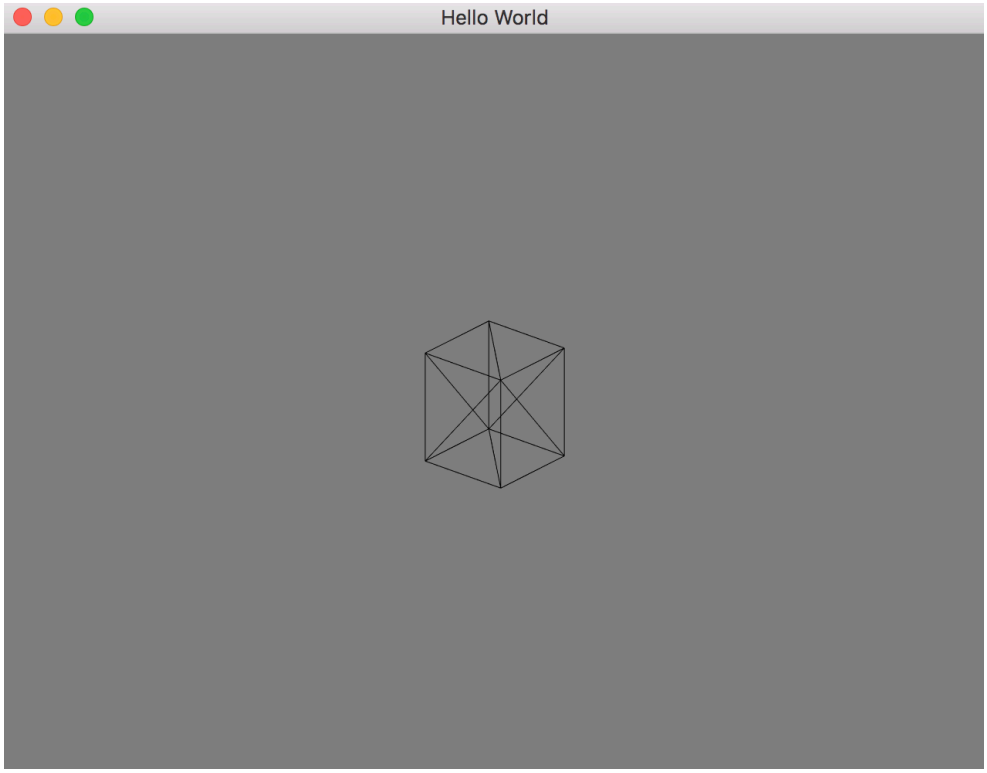
Yujun Sun

CSCI-GA 2270-001 Computer Graphics

Assignment 3: 3D Scene Editor

Report

1.1 Scene Editor



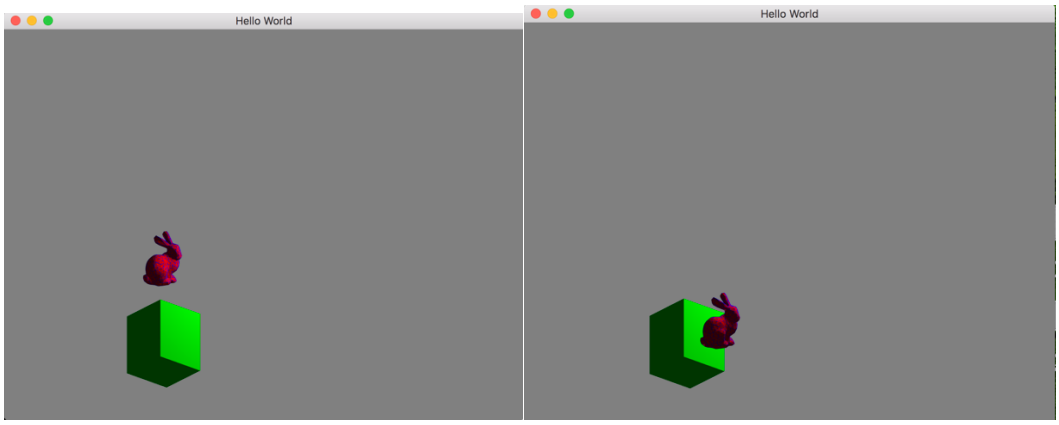
Construct a Model struct to store the information of each object: including the VBOs, vertices, normal and rotation, scaling, translation and model matrix, whether it is selected and its shading method.

Using load_off_file to load bumpy_cube.off and bunny.off, compute their normal in the method.

1.2 Object Control

Yujun Sun

CSCI-GA 2270-001 Computer Graphics



Using similar method as raytracing on the mouse to judge if the closer object is selected, compare the distance. When an object is selected, its color will turn red. Since the keyboard press is limited, I make use of the mods to allow pressing both Ctrl and a key to realize rotation of a single object.

Operation on the selected object can be realized by keypress shown below:

W: Translation z+.
Ctrl W: Rotate down
S: Translation z-
Ctrl S: Rotate UP
A: Translation z-
Ctrl A: Rotate UP
D: Translation x-
Ctrl D: Rotate right
Z: Translation y+
Ctrl Z: Rotate clockwise
C: Translation y-
Ctrl C: Rotate counterclockwise
Q: Scale up
E: Scale down
X: Deletion
4: Switch to wireframe mode for selected object
5 Switch to flat shading mode for selected object
6 Switch to phong shading mode for selected object
7 Activate orthogonal projection
8 Activate perspective projection

For flat shading, compute the normal of each face and push the normal to three vertices of the face.

For phong shading, compute the average of the normal of neighboring faces and take the average of it to be the normal of the vertex.

1.3 Camera control

Yujun Sun

CSCI-GA 2270-001 Computer Graphics

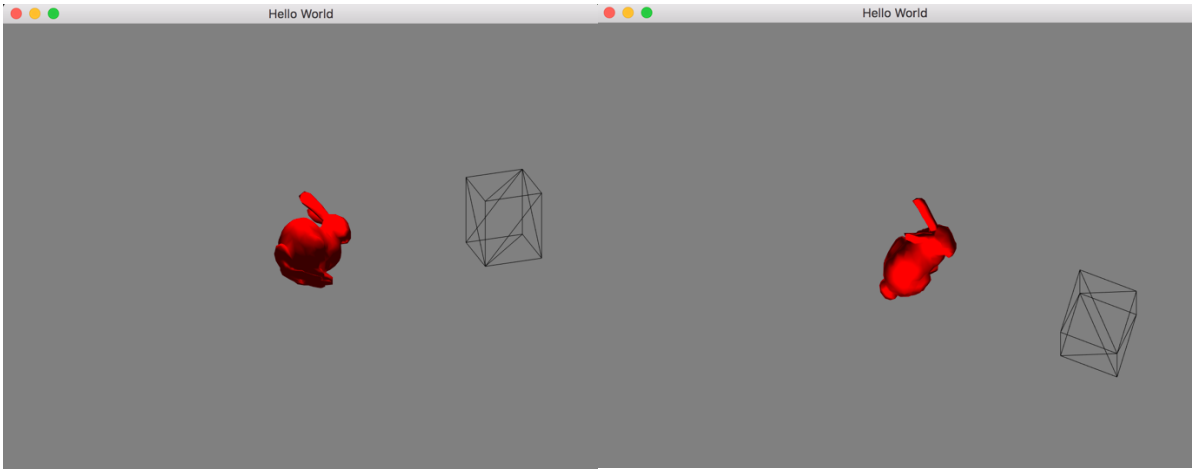
Edit the framebuffer_size_callback to have the objects in the window not distorted after resize the window.

```
    int w = height * aspect;
    int left = (width - w) / 2;
    glViewport(left, 0, w, height);
```

Camera position is controlled by rho, theta and phi.

```
   Eigen::Vector3f cameraPos(
        camera_rho * sin(camera_theta) * cos(camera_phi),
        camera_rho * sin(camera_theta) * sin(camera_phi),
        camera_rho * cos(camera_theta)
        );
```

Using the lookAt algorithm to control the view.



Construct 2 matrixes for orthogonal projection and perspective projection. Using the bool perspectiveActive to control which one is to be used in the program.