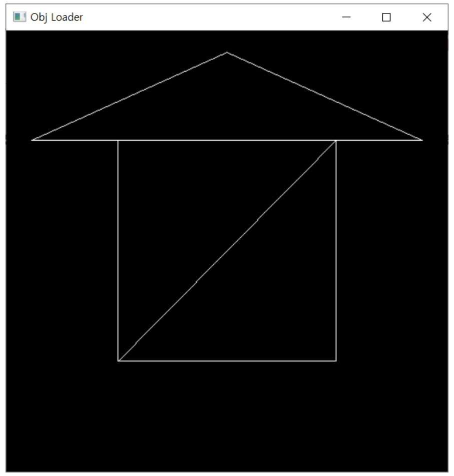
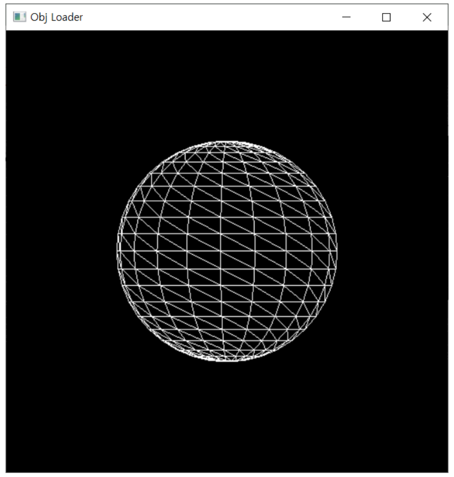
Homework 5: Simple OBJ loader

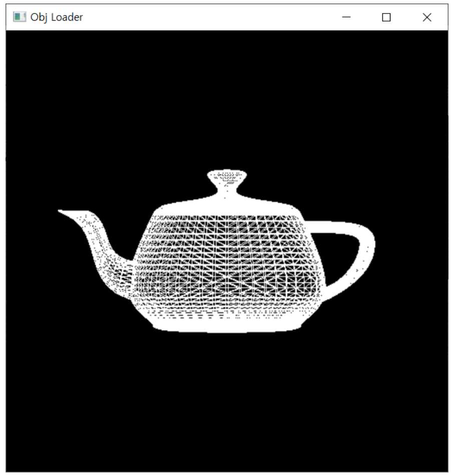
- Goal
  - To load simple OBJ files and render them as white wireframes
- Requirements
  - Able to read three obj files, **house.obj, sphere.obj**, and **teapot.obj**, to obtain their respective vertex positions and topological face information
  - Constantly rotate your models about the Y-axis.
  - Interactively render the selected model as a wireframe
    - Key ‘1’: draw the house.
    - Key ‘2’: draw the sphere.
    - Key ‘3’: draw the teapot.



house.obj



sphere.obj



teapot.obj

Implementation guideline

- Make a loader function as follows:

```
bool load_obj(GLvec& vertices, std::vector<GLuint>& faces, const char* filepath)
```

  - Here, “vertices” and “faces” are the vectors to store vertex positions and vertex indices of each face, respectively. “filepath” is the path to a target OBJ file.
  - You can assume that every face in each input OBJ file is a triangle such that the face is always composed of only three vertices.
  - The function may ignore lines starting with “#”, “vt”, “n”, “g” while carefully reading the lines starting with “v” and “f”.
  - The function also may ignore indices other then vertex indices for each face, for example,

    f **5**/8/9 **6**/13/10 **8**/14/11

    where you may only take into account the first numbers, 5, 6, 8.

- Example

```
v -0.5 -0.5 0       ← Bottom left
v  0.5 -0.5 0       ← Bottom right
v  0.5  0.5 0       ← Top right
v -0.5  0.5 0       ← Top left
v -0.9 0.5 0        ← Left side
v  0.9 0.5 0        ← Right side
v  0.0 0.9 0        ← Top of room

g MySquare
f 1 2 3             ← Base triangle 1
f 1 3 4             ← Base triangle 2
g MyTriangleRoof
f 5 6 7             ← Triangle
```

vertices =

-0.5	-0.5	0	0.5	-0.5	0
0.5	0.5	0	-0.5	0.5	0
-0.9	0.5	0	0.9	0.5	0
0.0	0.9	0			

faces =

0	1	2	0	2	3
4	5	6			

Global variable definition

```cpp
const GLuint num_of_models = 3;

const char* obj_filepath[num_of_models] = {
        "house.obj",
        "sphere.obj",
        "teapot.obj"
};

GLvec vertices[num_of_models];
std::vector<GLuint> faces[num_of_models];


GLuint vao[num_of_models];
GLuint vbo[num_of_models][2];
```

Program initialization

```cpp
void init()
{
    srand(clock());
    program = build_program();
    for (int i = 0; i < num_of_models; ++i) {
        load_obj(vertices[i], faces[i], obj_filepath[i]);

        glGenVertexArrays(1, &vao[i]);
        glBindVertexArray(vao[i]);
        glGenBuffers(2, vbo[i]);
        bind_buffer(vbo[i][0], vertices[i], program, "vPosition", 3);
        bind_buffer(vbo[i][1], faces[i], program);
    }

    glEnable(GL_DEPTH_TEST);
    glDepthFunc(GL_LESS);

    glLineWidth(1.0f);
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);

    glEnable(GL_CULL_FACE);
    glCullFace(GL_BACK);
}
```

Load OBJ files.

Make element array buffers to store vertex indices of each face.

Set polygon drawing mode to line drawing (GL_LINE) with thickness of 1.0.
(default mode: GL_FILL)

Enable back-face culling.

- Implementation of **bind_buffer(…)** for element array buffers

```
void bind_buffer(GLint buffer, std::vector<GLuint>& vec, int program)
{
      glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, buffer);
      glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(GLuint) * vec.size(), vec.data(), GL_STATIC_DRAW);
}
```

- How to render with element array buffers:

```
GLuint active_vao = 0;

...

void display()
{
      glBindVertexArray(vao[active_vao]);

      ...
      glDrawElements(GL_TRIANGLES, faces[active_vao].size(), GL_UNSIGNED_INT, (void*)0);
      glFlush();

      glutPostRedisplay();
}
```

You need to change its value according to user's keyboard input.

Make a draw call using glDrawElements(…)

- What to submit:
  - A **zip file** that compresses the following files:
    - **Project source files** except libraries.
      - Clean your project before compression by selecting **Build → Clean Solution** in the main menu.
    - Three OBJ files: **house.obj, sphere.obj**, **teapot.obj**
    - Screen capture images for each model: **house.png**, **sphere.png**, **teapot.png**
  - File name format
    - **hw5_000000.zip**, where 000000 must be replaced by your own student ID.
- Due date: **to be announced**