HW1: OBJ Loader

Introduction to Computer Graphics

Yu-Ting Wu

HW Description

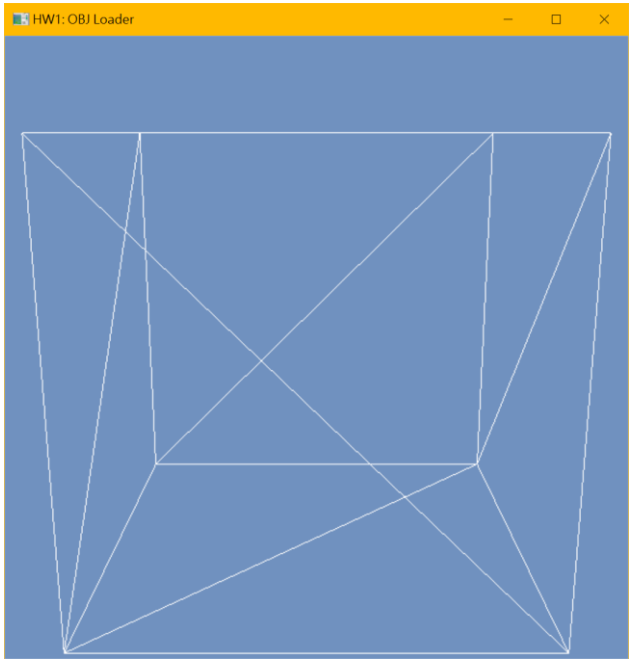
- **Web Link:**
  - https://kevincosner.github.io/courses/ICG2022/hw1.html

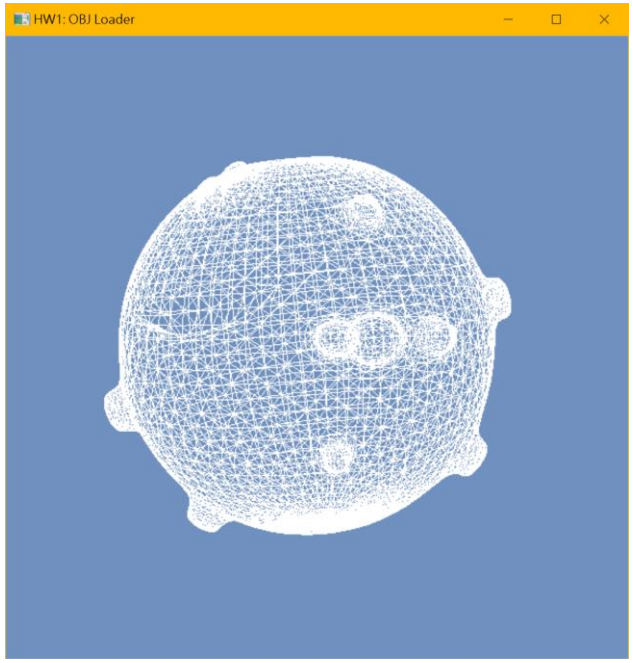
- **Major Task**
  - Implement a program to load the geometry data described in a *Wavefront Object File (*.obj)* and render the model on the screen
- **Minor Task**
  - Resize the model by normalizing its geometry data
  - Load and delete models dynamically

Grading Policy

- **Loading the model correctly (60%)** [Test Models]
- **Model normalization (10%)**
  - Make the center of the model located at the origin (0, 0, 0)
  - Make the maximal extent of the object bound equal to 1
- **Dynamic loading and deletion (10%)**
  - Control with the keyboard. E.g., press ‘o’ to load a model from the command line and press ‘d’ to delete the model
- **Code organization (10%)**
- **Report (5%)**
  - Introduce your implementation and put some screenshots
- **Bonus (5%)**
  - Load with UI, such as a menu or file dialog

Reference Results





Submission

- **Deadline: Oct. 16, 2022 (PM 11:59)**

- **Submission rule**
  - Will be announced later by TA

- **Late policy**
  - One day 90%
  - Two days 80%
  - Three days 70%
  - Four days 60%
  - Five days+ 50%

Skeleton Code

- Please download the skeleton code from the course website or 數位學苑3.0
- **At least** add your implementation in the following classes or functions
  - *LoadFromFile(…)* in *trianglemesh.cpp*
  - *SetupScene()* in *ICG2022_HW1.cpp*
  - *RenderSceneCB()* in *ICG2022_HW1.cpp*
  - *ReleaseResources()* in *ICG2022_HW1.cpp*
  - Update *numVertices*, *numTriangles*, *objCenter*, and *objExtent* correctly
- Feel free to add other variables or functions if needed

Useful Materials

- **OBJ Model format**

cube.obj - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
# Unit-volume cube with the same texture coordinates on each face.
#
# Created by Morgan McGuire and released into the Public Domain on
# July 16, 2011.
#
# http://graphics.cs.williams.edu/data
```
comments

```
mtllib default.mtl
```
specify material file

```
v -0.5 0.5 -0.5
v -0.5 0.5 0.5
v 0.5 0.5 0.5
v 0.5 0.5 -0.5
v -0.5 -0.5 -0.5
v -0.5 -0.5 0.5
v 0.5 -0.5 0.5
v 0.5 -0.5 -0.5
```
vertex position declaration

```
vt 0 1
vt 0 0
vt 1 0
vt 1 1
```
vertex texture coordinate declaration

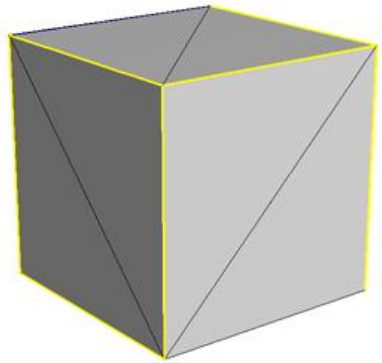
```
vn 0 1 0
vn -1 0 0
vn 1 0 0
vn 0 0 -1
vn 0 0 1
vn 0 -1 0
```
vertex normal declaration

第 39 列，第 29 行 160% Unix (LF) UTF-8

f P/T/N P/T/N P/T/N

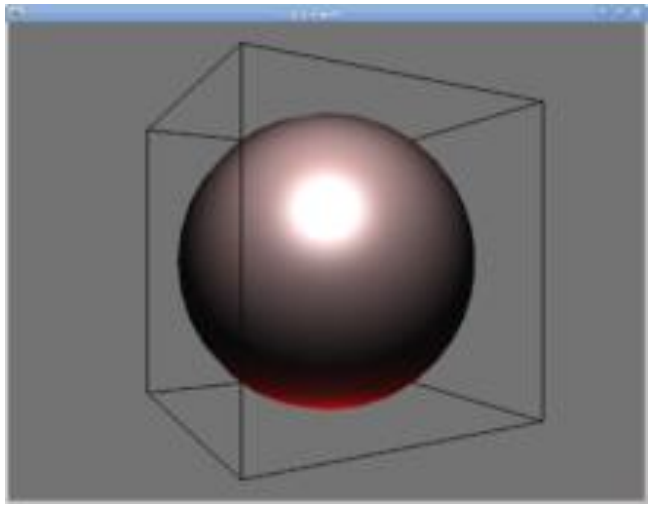
```
g cube
usemtl default

f -8/-4/-6 -7/-3/-6 -6/-2/-6
f -8/-4/-6 -6/-2/-6 -5/-1/-6
f -8/-4/-5 -4/-3/-5 -3/-2/-5
f -8/-4/-5 -3/-2/-5 -7/-1/-5
f -6/-4/-4 -2/-3/-4 -1/-2/-4
f -6/-4/-4 -1/-2/-4 -5/-1/-4
f -5/-4/-3 -1/-3/-3 -4/-2/-3
f -5/-4/-3 -4/-2/-3 -8/-1/-3
f -7/-4/-2 -3/-3/-2 -2/-2/-2
f -7/-4/-2 -2/-2/-2 -6/-1/-2
f -3/-4/-1 -4/-3/-1 -1/-2/-1
f -3/-4/-1 -1/-2/-1 -2/-1/-1
```
face data
(adjacency, submesh)

第 20 列，第 7 行 160% Unix (LF) UTF-8



Useful Materials

- **Model normalization**
  - Find the center of a 3D model
  - Find the minimal **bounding box** of a 3D model
  - Find the maximal extent axis of the bounding box
  - Find a mapping to make the model located at the origin and its maximal extent axis equal to 1



Pitfalls

- For the face declaration in an OBJ file
  - The indices of position, normal, and texture coordinate might appear as negative numbers
  - The indices of position, normal, and texture coordinate start with **1**
  - A face might be declared as a polygon (>= 3 vertices), you should **split them into triangles** if needed
  - Some OBJ files downloaded from the Internet might have no normals or texture coordinates
    - But I will avoid using this kind of files

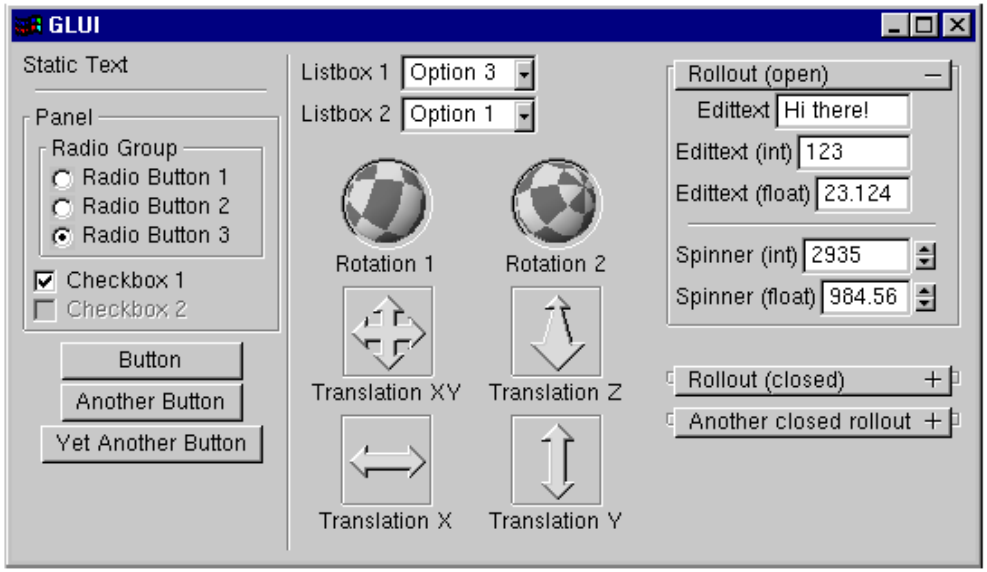
f P/T/N P/T/N P/T/N

```
f -8/-4/-6 -7/-3/-6 -6/-2/-6
f -8/-4/-6 -6/-2/-6 -5/-1/-6
f -8/-4/-5 -4/-3/-5 -3/-2/-5
```

Other Resources

- **Using pop-up menu in FreeGLUT**
  - https://www.lighthouse3d.com/tutorials/glut-tutorial/popup-menus/

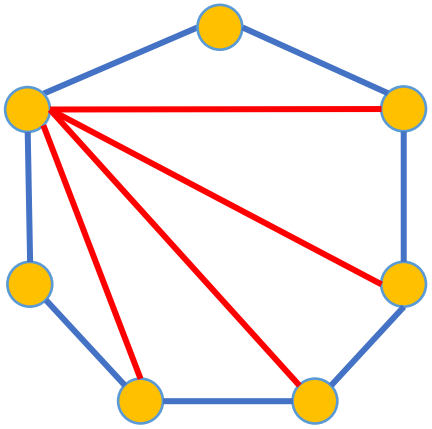
- **Building the Simplest GUI in FreeGLUT**
  - **GLUI:** https://github.com/libglui/glui



Update (2022/10/03)

- It is **fine** to use your program structure (rather than using the skeleton code); **however**, you should **set the camera to the same configuration as the skeleton code**, and **describe your program structure in your report**
- It is **essential** to subdivide a polygon into triangles if it has more than three vertices

```
f 4738/4739/4538 3420/1238/1237 4680/1237/1236 2608/1389/1388 4679/1388/1387 2607/1403/1402 4678/1402/1401
```



Update (2022/10/03)

- For model normalization, find an equation to map the center of the object to the origin, and the longest axis of the model extent to 1