HW1: OBJ Loader

Computer Graphics

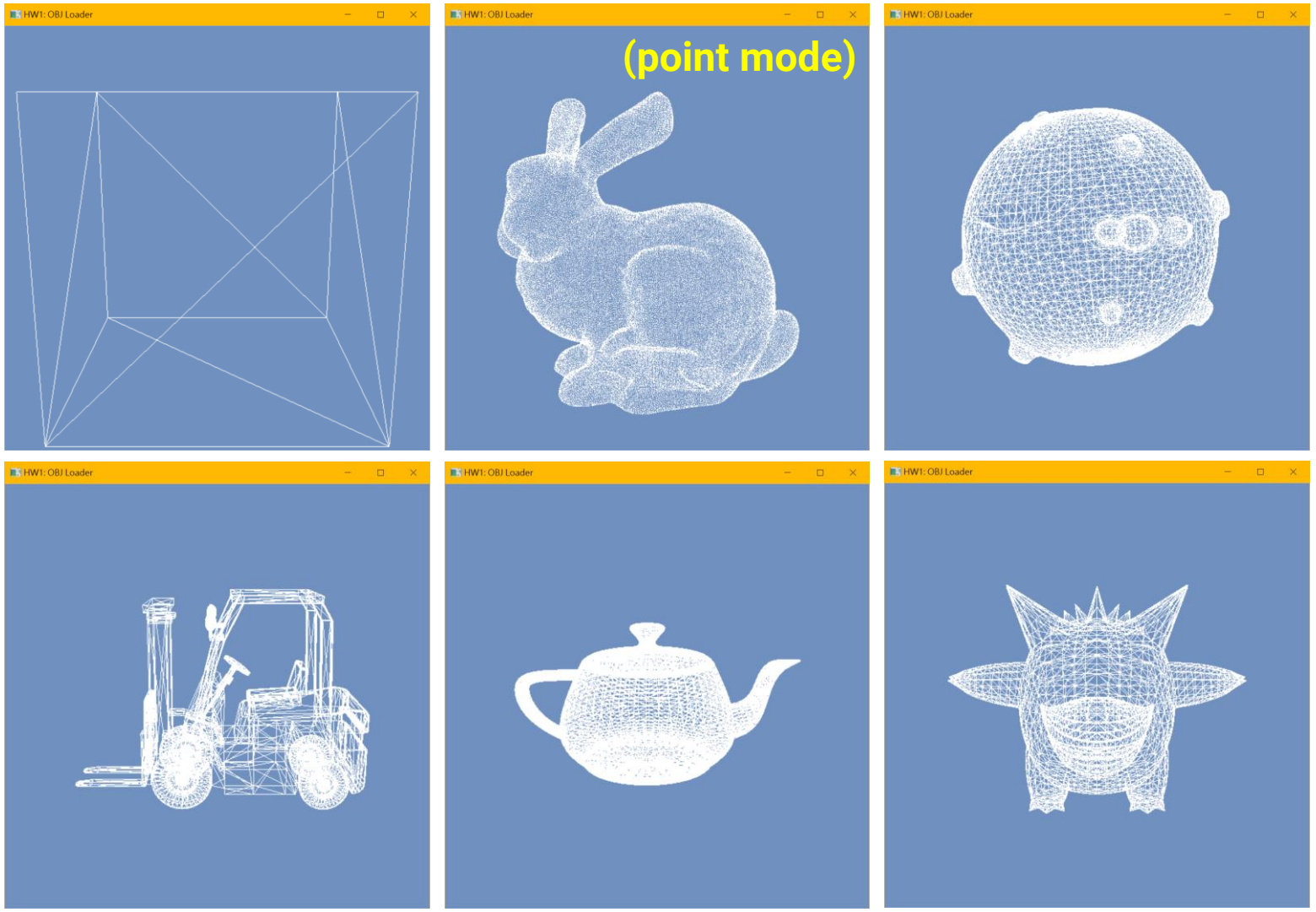
Yu-Ting Wu

HW Description

- **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

Reference Results



Grading Policy

- **Loading the model correctly (60%)** [Test Models]
  - Use **index buffer** (instead of using glDrawArrays)
  - **Subdivide** a polygon into **triangles** if it has more than three vertices
- **Model normalization (15%)**
  - Modify the positions of vertices such that the **center** of the model will be located at the **origin (0, 0, 0)** and the **maximal extent** of the object bound equal to **1**
- **Dynamic loading and deletion (10%)**
  - Deletion means **releasing memory**, **NOT** making it invisible
  - Can control with the keyboard (to load or delete by pressing keys), menu, or other GUI

Grading Policy (cont.)

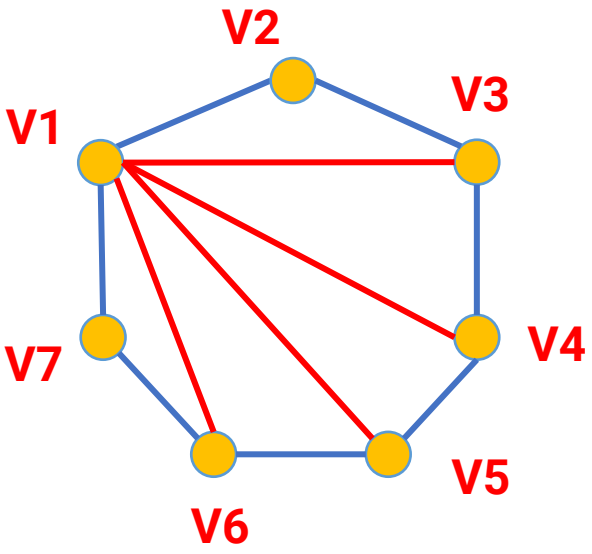
- **Code organization and coding style (5%)**
  - Variables naming
  - Comments
- **Report (5%)**
  - Introduce your implementation and put some screenshots
- **MISCs (5%)**
  - Load with UI (e.g., a menu or file dialog)

Polygon Subdivision

- OBJ files may have polygons rather than triangles
- 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

V1 V2 V3 V4 V5 V6 V7



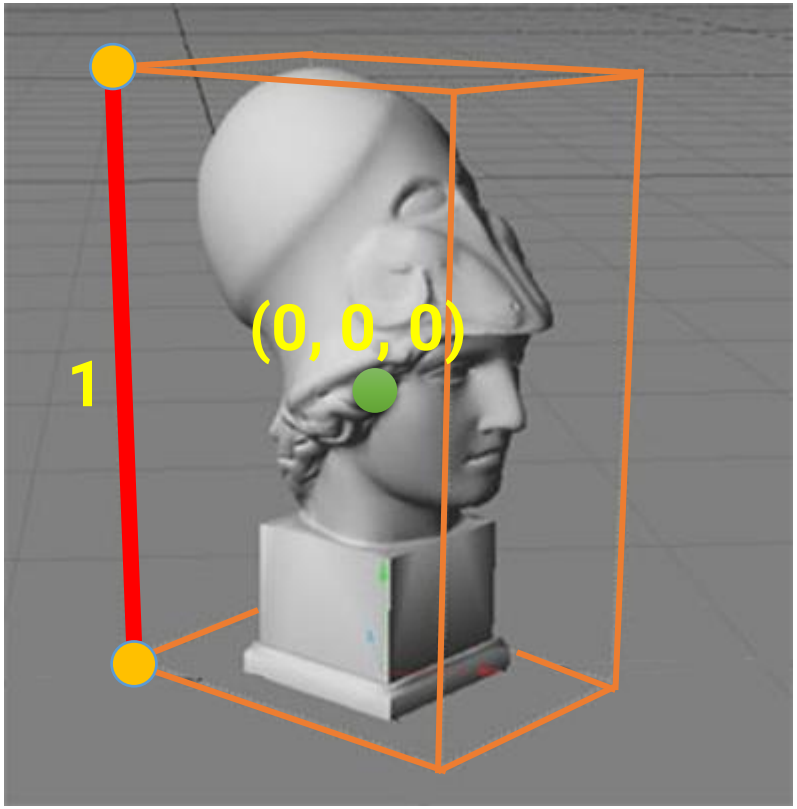
Triangles:
V1-V2-V3
V1-V3-V4
V1-V4-V5
V1-V5-V6
V1-V6-V7

You can check (or debug) your implementation using the file, "Polygons.obj"

Model Normalization

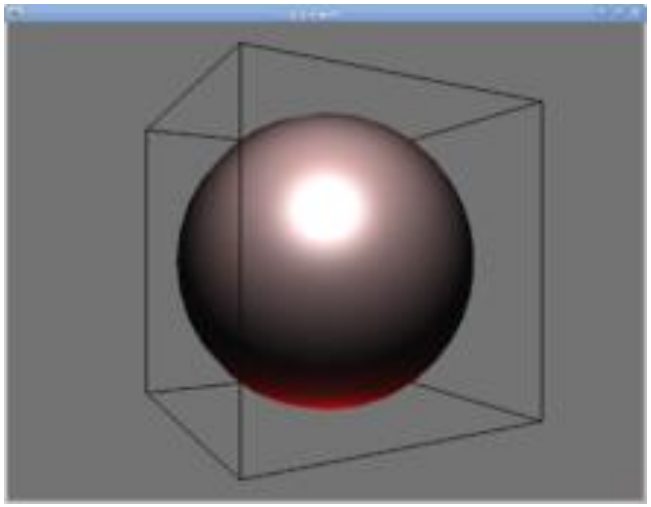
- For model normalization, find a way to map the **center** of the object to the **origin**, and the **longest axis of the model extent** to **1**

Bounding box:
a cube formed by
(MinX, MinY, MinZ) and
(MaxX, MaxY, MaxZ)



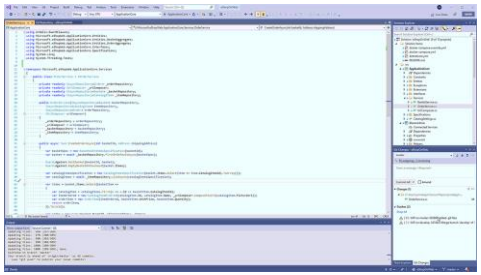
Model Normalization (cont.)

- Find the minimal **bounding box** of the 3D model
- Find the **center** of the 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**



Skeleton Code

- Please download the skeleton code from 數位學苑3.0
  - As mentioned in the first class, we will use **Microsoft Visual Studio Community** for the IDE



- All the libraries you need are located in the "**Library**" folder
- **Files**
  - **CG_HW1.cpp** (main program, GLUT callback functions)
  - **TriangleMesh.h**
  - **TriangleMesh.cpp**

  (C++ class for managing a 3D triangle mesh)

Skeleton Code (cont.)

```
// VertexPTN Declarations.
struct VertexPTN
{
    VertexPTN() {
        position = glm::vec3(0.0f, 0.0f, 0.0f);
        normal = glm::vec3(0.0f, 1.0f, 0.0f);
        texcoord = glm::vec2(0.0f, 0.0f);
    }
    VertexPTN(glm::vec3 p, glm::vec3 n, glm::vec2 uv) {
        position = p;
        normal = n;
        texcoord = uv;
    }
    glm::vec3 position;
    glm::vec3 normal;
    glm::vec2 texcoord;
};
```

Parse the OBJ file and fill geometry data into the container in TriangleMesh

```
std::vector<VertexPTN> vertices;
std::vector<unsigned int> vertexIndices;
```

Skeleton Code (cont.)

- **At least** add your implementation in the following classes or functions
  - **LoadFromFile(…)** in **TriangleMesh.cpp**
  - **CreateBuffers()** in **TriangleMesh.cpp**
  - **SetupScene(…)** in **CG_HW1.cpp**
  - **RenderSceneCB()** in **CG_HW1.cpp**
  - **ReleaseResources()** in **CG_HW1.cpp**
  - Update **numVertices**, **numTriangles**, and **objCenter** correctly
- Feel free to add other variables or functions if needed
- There are some codes related to building and applying **transformation** on vertices, **Please DO NOT TOUCH them**

STL Vector

- Vectors are **sequence containers** representing **arrays** that **can change in size**
- Use **contiguous** storage locations for the elements, which means that their elements can be accessed using offsets on regular pointers to its elements, and just as efficiently as in arrays
- To use, **#include <vector>**
- For more detailed documentation, please refer to https://learn.microsoft.com/zh-tw/cpp/standard-library/vector-class?view=msvc-170

STL Vector (cont.)

```
vector<char> testVec = { 'H', 'E', 'L', 'L', 'O' };
testVec.push_back('W');
testVec.push_back('O');
testVec.push_back('R');
testVec.push_back('L');
testVec.push_back('D');

for (unsigned int i = 0 ; i < testVec.size() ; ++i)
    cout << testVec[i] << " ";
cout << endl;
```

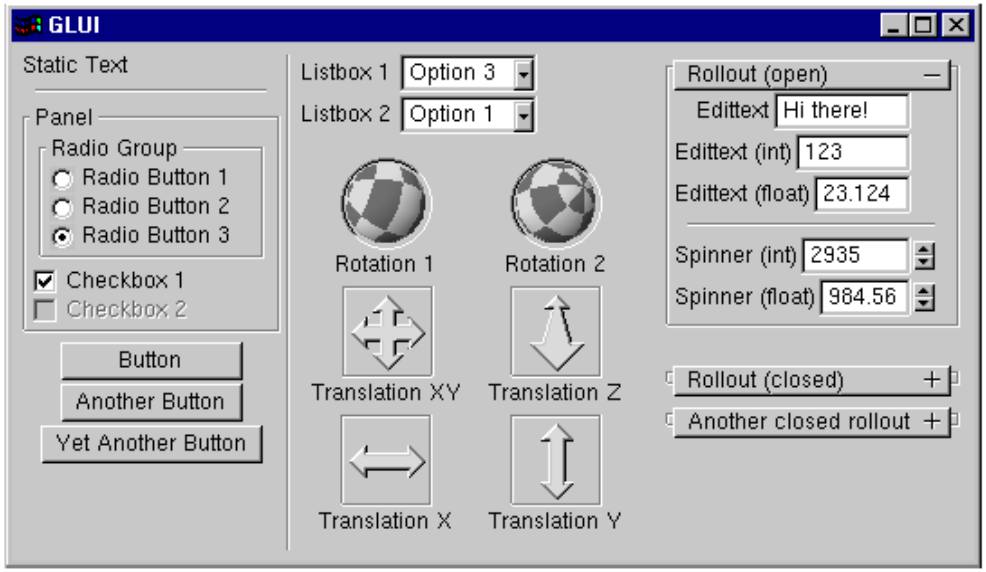
dynamically insert data at the end

HELLOWORLD

Other Resources

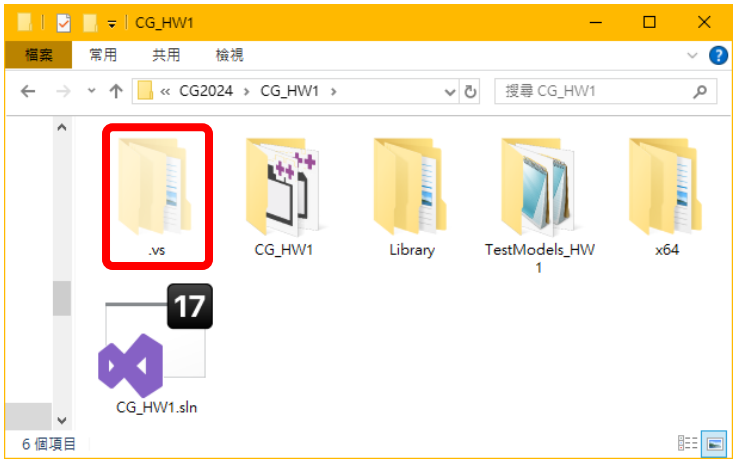
- **Using pop-up menu in FreeGLUT**
  - https://www.lighthouse3d.com/tutorials/glut-tutorial/popup-menus/

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



Submission

- **Deadline: Oct. 20, 2024 (PM 11:59)**
- **Submission rule**
  - Package your **VisualStudio Community project** and the **report** to a file **named "學號_姓名_HW1.zip"**
  - Upload the zip file to "數位學苑3.0"
  - You can reduce the file size by deleting the hidden **.vs folder**
- **Late policy**
  - One day 90%
  - Two days 80%
  - Three days 70%
  - Four days 60%
  - Five days+ 50%



Hints

- Review the **OBJ Model format** in Slides: Geometry Representation
- Recall C++ File I/O
- Review how to **create** and **fill vertex** and **index buffer**
  - Upload **vertex data** stored in **vector<VertexPTN> vertices** into the **vertex buffer**
  - Upload **index** stored in **vector<unsigned int> vertexIndices** into the **index buffer**
- Review how to **render** with **vertex** and **index buffer**

Hints (cont.)

- **Parse OBJ file**

TexCube.obj - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
# Blender v2.76 (sub 0) OBJ File: ''
# www.blender.org
```
comments

```
mtllib TexCube.mtl
```
specify material file

```
v 1.0 -1.0 -1.0
v 1.0 -1.0 1.0
v -1.0 -1.0 1.0
v -1.0 -1.0 -1.0
v 1.0 1.0 -1.0
v 1.0 1.0 1.0
v -1.0 1.0 1.0
v -1.0 1.0 -1.0
```
vertex position declaration

```
vt 0.0 0.0
vt 0.0 1.0
vt 1.0 0.0
vt 1.0 1.0
```
vertex texture coordinate declaration

```
vn 0.0 -1.0 0.0
vn 0.0 1.0 0.0
vn 1.0 0.0 0.0
vn -0.0 0.0 1.0
vn -1.0 -0.0 -0.0
vn 0.0 0.0 -1.0
```
vertex normal declaration

```
vector<glm::vec3> positions;
vector<glm::vec3> normals;
vector<glm::vec2> uvs;

// Create new vertices if needed.
VertexPTN newVertex;
// Fill in vertex attributes.
vertices.push_back(newVertex);

// Determining indices.
vertexIndices.push_back(...);
```
vertex index

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

Hints (cont.)

- Start with the **simplest** model: Triangles.obj
  - Ensure the file is parsed correctly
  - Ensure the vertex data and index data are filled correctly
- Do the most important item first
  - You can **postpone** the **polygon subdivision**, **normalization**, **dynamic loading**, and **UI** until you can correctly render a cube on the screen

Pitfalls

- For the face declaration in an OBJ file
  - The indices of position, normal, and texture coordinate start with **1**
  - 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  8/2/2  7/1/2  6/3/2
f  5/4/2  8/2/2  6/3/2
f  2/4/1  3/2/1  4/1/1
f  1/3/1  2/4/1  4/1/1
```


END OF PRESENTATION
ANY QUESTIONS?