

Computer Graphics Project 2: Designing a Mesh-Based Scene

Handed out: April 21, 2026

Due: 23:59, May 17, 2026 (NO SCORE for late submissions!)

- LMS course home > Weekly Learning (주차학습) > Week 7> Project 2
- Compress your files into a zip file as in the following example. The zip file can be named whatever you want.

```
+ submission.zip
- main.py
- ...
- report.pdf
```

- Your program may consist of several python source files. But the main module should be in **main.py**. That is, your program should be executed with the following command:

```
python main.py
```

1. In this project, you will design and implement an interactive system that 1) loads OBJ meshes, 2) organizes them into a scene, and 3) demonstrates hierarchical modeling and animation

A. This project is not only about implementation, but about:

- i. **designing how mesh data is processed and rendered**
- ii. **constructing hierarchical relationships between objects**
- iii. **exploring how animation emerges from transformations**
- iv. **explaining and justifying your design choices**

B. Your implementation will extend your viewer from Project 1.

2. Program Requirements

A. Scene Viewer

- i. Load and render OBJ meshes
- ii. Use your camera and interaction design from Project 1.

1. You can extend or refine your interaction if needed.
- iii. In your **report**:
 1. You must clearly describe your interaction design in your report, including:
 - A. **how the user controls** the camera (e.g., mouse, keyboard, or both)
 - B. **how the camera changes** in response to each user input
 - Example:
 - Input: drag mouse
 - Behavior: camera rotates around the target point
 2. Even if your interaction design is unchanged from Project 1, you must still describe it for grading purposes.

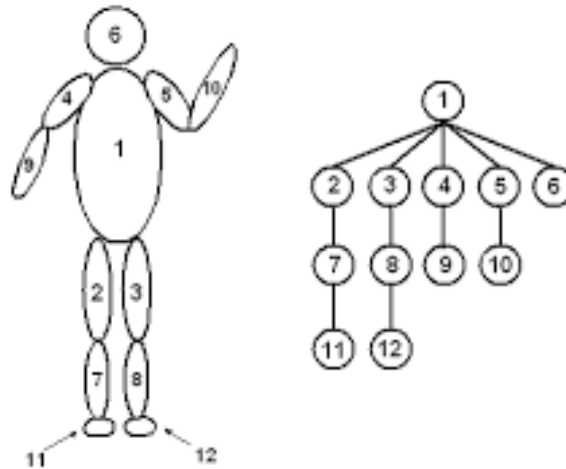
B. Mesh Processing Design

- i. You must design how your program converts parsed OBJ data into a renderable form.
- ii. Your program should correctly handle:
 1. **triangle meshes**
 2. **quad meshes**
 3. **meshes containing a mix of triangles and quads**
- iii. You may **choose your own approach to handle faces with more than three vertices** (e.g., triangulation or other methods).
- iv. Handling general n-gons is optional
- v. In your **report**:
 1. **explain how you process face data (triangles, quads, a mix of triangles and quads) and why you chose that approach**
 2. **explain how you organize data for rendering (e.g., vertex duplication, indexing, etc.) and why you chose that approach**

C. Hierarchical Modeling & Animation Design

- i. Your hierarchical model should consist of multiple meshes loaded from different OBJ files.
 - 1. Your hierarchical model must use **at least three different downloaded OBJ meshes**.
 - 2. Download cool free obj files from the Internet and use them. For example,
 - A. <https://free3d.com/>
 - B. <https://www.cgtrader.com/free-3d-models>
- ii. **You must include the OBJ files** used for your hierarchical model **in your submission**.
 - 1. Use **relative paths in your source code** so that the program runs correctly on the TA's computer.
 - 2. The "relative paths" in the source code should not be platform-dependent. **Use `os.path.join()` for platform-independent path joining.**
- iii. The model should be **automatically animated without any mouse or keyboard inputs**.
- iv. Your system should:
 - 1. have a clear **hierarchical structure with at least 3 levels** in the hierarchy.
 - 2. demonstrate **animation over time**
 - 3. **parent and child parts should be visually distinguishable**
 - 4. **all child parts must move relative to their parent parts**
- v. For example:
 - 1. articulated structures (robot arm, creature, etc.)
 - 2. grouped motion (moving a parent affects children)
 - 3. You are encouraged to refer to past outstanding project submissions for inspiration:
 - A. <https://cgrhyu.github.io/courses/2023-spring-cg.html>
- vi. In your **report**:
 - A. **illustrate the hierarchical structure** of your model

- i. an example level-4 structure:



B. explain **how transformations are applied and how animation is defined**

C. **explain why you designed** the hierarchy and animation in that way

D. Lighting Design

- i. Render all objects using **Phong illumination** and **Phong shading**.
- ii. Choose lighting parameters (light colors, light position, material colors, material shininess, ...) as you want.
- iii. In your **report**,
 1. explain **your choice of lighting parameters**
 2. **how they affect the visual appearance and readability** of your scene.
 3. **explain why your choices improve the visibility and understanding** of the scene

3. Video Requirements

- A. A **hyperlink to a video** demonstrating your program, uploaded to an online video streaming service (e.g., YouTube, Vimeo).
 - i. The video is intended to demonstrate functionality, not to provide an explanation.
 - ii. The video must clearly demonstrate:
 - rendering of meshes
 - hierarchical structure

- animation of the model
- iii. Explanations or annotations are not required.
- B. **The uploaded video MUST be publicly accessible.**
- C. The video should be **no longer than 20 seconds**.

4. Report Requirements

- A. The report should include:
 - i. **Scene Viewer report requirement (2-A-iii)**
 - ii. **Mesh Processing Design report requirement (2-B-vi)**
 - iii. **Hierarchical Modeling & Animation Design report requirement (2-C-vi)**
 - iv. **Lighting Design report requirement (2-D-iii)**
 - v. **Video link (4-A)**
- B. The report should be concise (**approximately 2–5 pages**). Clarity is more important than length.
- C. You may write the report in **either English or Korean**. There is **no advantage or disadvantage based on the language you choose**.

5. Sample obj files

- A. Sample OBJ files are provided **only for testing and debugging. Do not use them for your final hierarchical model.**
 - i. cube-tri.obj: A cube with triangles only
 - ii. cube-tri-quad.obj: A cube with triangles and quads

6. What to submit: A zip file including

- A. **Python source files (.py)**
 - i. Your program may consist of several python source files. But the main module should be in **main.py**.

B. A report (.pdf)

C. Note:

- The video must be accessible via a hyperlink included in the report.
- If the video link is missing or not accessible, it may be treated as missing and result in a penalty.

7. Runtime Requirements

- A. You must implement all requirements in a single program. This project DOES NOT require each requirement to be a separate program.
- B. The window size doesn't need to be (800, 800). Use the larger window that is enough to see the details of the viewer.
- C. **Your program must use OpenGL 3.3 Core Profile**, meaning that ...

i. Your python code must include:

```
glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3)
glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3)
glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE)
```

ii. Your shader code must start with:

```
#version 330 core
```

- D. **Your program should be able to run on Python 3.10 with only NumPy, PyOpenGL, glfw, PyGLM, Pillow installed. Do not use any other additional python modules.**
- E. Only **glfw** is allowed for event processing and window & OpenGL context management. **Do not use glut functions for this purpose.**
- F. **If your program does not meet these requirements, it may not run on the TA's computer, which may result in a significant penalty.**

8. Evaluation will be primarily based on:

- A. your mesh processing design and justification (as required in 2-B-vi)
- B. your hierarchical modeling and animation design and justification (as required in 2-C-vi)

- C. your lighting design and justification (as required in 2-D-iii)
- D. how clearly your video demonstrates your program
- E. Note:
 - The video and report are essential components of this project.
 - Insufficient or missing video/report may result in significant penalties.
 - Correct implementation alone is not sufficient for a high score.
 - Strong emphasis is placed on how clearly you explain your design choices.

9. Additional information

- A. obj file format reference: https://en.wikipedia.org/wiki/Wavefront_obj_file
- B. Python provides powerful string methods helpful for parsing an obj file. Among them, `split()` will be most useful.