CSED451: Computer Graphics (Fall 2025)

Assignment 3: 3D Drawing

Submission due (3-1): Sunday, November 2, 2025, 11:59PM
Submission due (3-2): Saturday, November 22, 2025, 11:59PM

Please note that many sections are the same as in the previous assignment(s). For your convenience, the sections that have been changed are marked with an asterisk (*).

1. Introduction*

In this assignment, you will extend the previous assignment by applying 3D graphic.

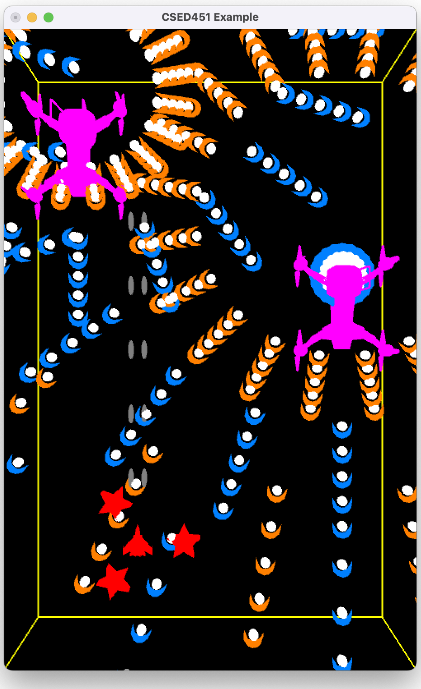

Figure 1. Bullet Hell shooter enhanced with 3D graphic.

For inspiration, you may refer to examples of Bullet Hell shooters with 3D graphic:

- Touhou: New World
- Nier Automata
- Homura Hime

2. Requirements*

Minimum Goals*

Assignment 3-1 and 3-2 must satisfy all the minimum requirements of Assignment 2 (and 1). In case of any conflicts, the requirements specified in Assignment 3-1 and 3-2 take precedence.

- Framework **(Important)**
  - In Assignment 3-1, you must use the OpenGL legacy pipeline as in the previous assignments.
  - In Assignment 3-2, you must use OpenGL Shading Language (GLSL) and must not use the legacy pipeline at all.
- All entities
  - All entities must be represented by 3D models.
    - You may use provided 3D models in `assets/*.obj` or you may create/obtain 3D models of your own.
    - Blender is useful for creating or visualizing 3D models.
  - All entities must be represented by a single scene graph. You are free to determine the type of the scene graph, the meaning of each node, and the relationships among them according to your design philosophy.

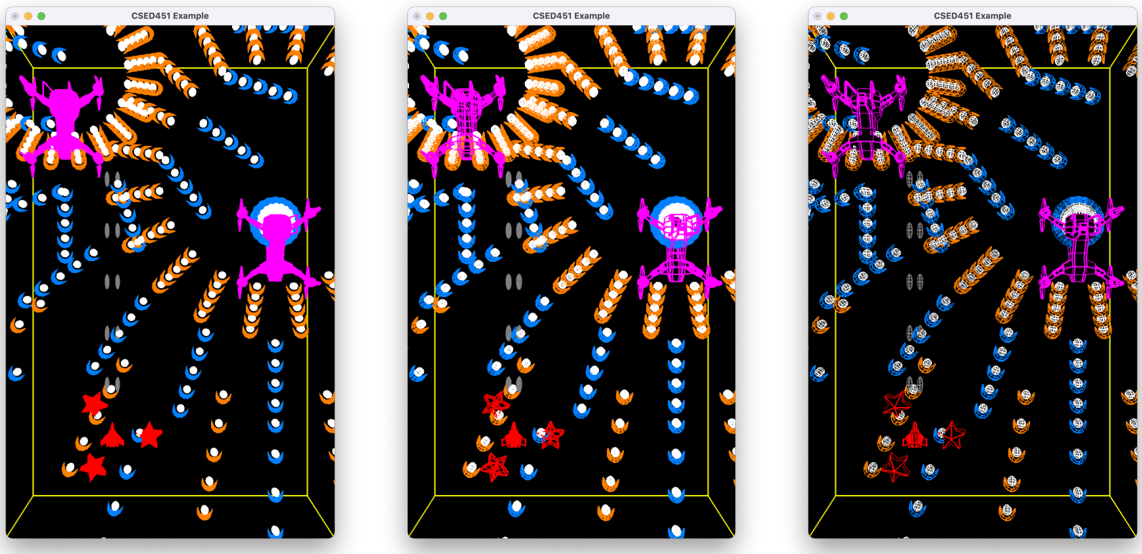

Figure 2. An example of three types of graphic styles. An opaque polygon style (left), a wireframe style (middle), and a wireframe style with hidden line removal (right)

- Graphic Style (see Figure 2)
    - In Assignment 3-1, the program must support two types of graphic styles.
        - The first one is an opaque polygon style,
        - and the second one is a wireframe style.
    - In Assignment 3-2, the program must support an additional style:
        - a wireframe rendering with hidden line removal.
    - The graphic styles must be controllable inside the program. For example, press `w` key to change between the graphic styles.

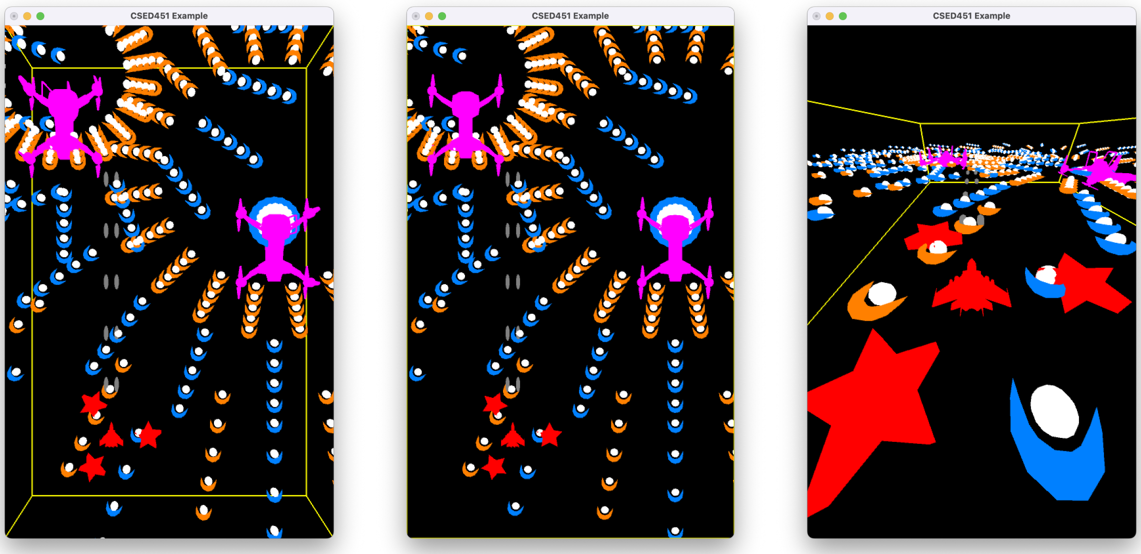

Figure 3. An example of three types of views. A top view with perspective projection (left), a top view with orthographic projection (middle), and a third-person view with perspective projection (right)

- Camera and Rendering (see Figure 3)
    - The camera must support three types of views.
        - The first one is a top view with perspective projection,
        - the second one is a top view with orthographic projection,
        - and the third one is a third-person view with perspective projection, positioned close to the player.
    - The views must be controllable inside the program. For example, press `c` key to change between the views.
- Player
    - As in the previous assignments, the player cannot move outside the in-game bounding box, and the bounding box must be visualized.
- Bullet
    - A bullet disappears once it leaves the in-game bounding box.

Additional Goals*

To receive full credit, you must establish at least two additional goals beyond the minimum requirements above. These goals should be sufficiently challenging. Examples include:

- Animated entities. For example: the player is animated in response to its movement, that is, the player slightly leans left as it moves left.
- Destructible enemy body parts

Note that credit will be given **only for graphics-related features**. Little or no credit will be given for mere game features (e.g., pause menus, gamepad support, multi-stage boss fights, etc.). You may choose additional goals from the examples above, but remember that more generous credit will be given for original goals you propose yourself.

Non-goals*

- You do not need to implement texture mapping, lighting, or shading, as these techniques do not count toward credit.

3. Submission*

Each team must submit one report along with the program's source code **and asset files such as 3D model files**.

Report

Your report can be any document file format. However, please do **not** use `*.hwp`. Your report should be brief and concise, and include the following:

- Basic information:
    - The name of the team.
    - Each team member's name, department, student ID, and HEMOS ID.
- Technical details:
    - The development environment used (e.g., IDEs, framework versions, etc.).
    - Any additional technical background necessary to understand the source code.
- Implementation details:
    - An outline of your program's features.
    - Detailed description of your additional requirements.

    - The rationale behind your program's design.
    - How you implemented your design.
    - Any additional background necessary to understand the design and implementation.
    - Do **not** include your program's source code in the report.
- End-user guide:
    - How to run and operate your program.
    - Screenshots of each feature you described.
- Discussions/Conclusions:
    - The obstacles you encountered during development and how you resolved them.
    - Idea on how you can improve your program further.
    - What you learned or concluded from this assignment.
- References:
    - **Clearly state references** for any work not created by your team. (e.g., something from tech blogs, Stack Overflow posts, etc.)
    - **If any part of your assignment is found to be someone else's work without proper references, it will be considered cheating.**
- **AI-assisted coding references**:
    - If you used any AI-assisted coding tools in writing your program (e.g., OpenAI Codex, Claude Code, Google Gemini, Cursor, Amazon Kiro, GitHub Copilot, etc.), clearly state which tools you used, how you used them, and where in your program they were applied.
    - You should also mention AI tools you consulted, even if their outputs were ultimately not included in your final program.
    - Provide an estimate of what percentage of your assignment was completed with AI assistance, and explain your reasoning. *This percentage will not affect your credit at all.*

Program

- Include a brief `README.md` file that describes your source code files.
- Your source code is expected to meet the technical requirements specified in the provided setup document.

Assets*

- Include all assets required by your program.

- Do **not** load assets using absolute paths in your program. When compiling your program, the TA will assume that the directory structure is preserved as submitted.

4. Scoring Criteria

Basic Rules

- **Minimum requirements** (40%)
  - The program must run properly and meet the minimum requirements.
- **Additional requirements** (20% + *10% extra credit*)
  - You must define and implement additional requirements on your own.
  - Two valid additional requirements → +20% credit.
  - Three or more valid additional requirements → up to +30% total (20% base + 10% extra).
  - You may propose more than the minimum if you are unsure whether certain requirements will qualify for full credit.
- **Program design and implementation** (30%)
  - The program must be designed and implemented properly to meet the requirements using OpenGL.
- **Report structure and `README.md`** (10%)
  - The report and `README.md` must be clear and concise.
  - The structure and formatting of the report will also be evaluated.

Deduction

- If you miss the submission deadline, your score will be reduced by 10%.
- For every additional 24 hours late, your score will be reduced by another 10%p.
- For example,
  - Submitting 8 hours late: -10%
  - Submitting 25 hours late: -20%
  - Submitting 216 hours (9 days) late: no credit will be given.
- Note: After the submission deadline, you may only submit via the TA's email (`yoonha.hwang@postech.ac.kr`).

If you have any further questions, please post your question on Q&A board on PLMS.