Assignment 3-2 Report

Team Name: Endgame Duo
Team Members:

Name: shcha1220@gmail.com
- **Department:** CSE
- **Student ID:** 20200120
- **HEMOS ID:** carprefer

Name: SOONHO (snow kim)
- **Department:** CSE
- **Student ID:** 20200703
- **HEMOS ID:** rlatnsgh0708

Technical Details

Development Environment:

- **IDEs:** Visual Studio Code
- **Framework Versions:**
  - OpenGL: 4.2 (Compatibility Profile) Mesa 23.2.1
  - GLEW: 2.2.0-4
  - freeglut: 2.8.1-6
  - GLM: 1.0.1
- **Git:** https://github.com/csed451/graphics.git
- **Other Tools:**
  - GCC(g++), WSL(Ubuntu 22.04)

Implementation Details

- **Program Features Outline:**
  - **GLSL Renderer Backbone**
    - OpenGL Legacy 방식과 관련된 호출을 모두 제거하고, Shader Program과 전역 Renderer 클래스를 통해 uModel, uView, uProj, uColor 등을 관리하는 GLSL 파이프라인을 구축했다.
    - Mesh 클래스의 **load_from_obj()**에서는 OBJ를 읽자마자 VAO/VBO를 생성해 GPU에 올리고, **draw()** 는 glBindVertexArray 한 번만 호출하면 되도록 단순화했다.
    - GLEW 초기화 이후 **gRenderer.init()**를 호출한다. 이를 통해 게임 루프마다 begin_frame, end_frame 사이에서만 draw()를 실행하고 모든 객체가 셰이더 경로를 거치도록 구현했다.
  - **Camera & Mode Switching**
    - perspective와 ortho 함수 계산한 행렬을 **gRenderer.set_projection**에 즉시 반영한다.
    - 이를 통해 update_camera가 호출될 때마다 뷰 행렬을 uniform으로 전송한다.
    - W, C 키 이벤트는 기존과 동일하지만, 각 모드 전환 시 셰이더만 업데이트 하면 되기 때문에 Matrix Stack과 관련된 동작은 제거했다.
  - **Bounding Volume Visuals**
    - 경계 박스와 엔딩 씬에 사용되는 별들은 각각 전용 VAO를 사용해 GLSL 렌더링한다.
    - is_outside_window 로직은 그대로라 Player, Attack, Bullet이 범위를 벗어나면 즉시 되돌리거나 Pool에 반환된다.
  - **A wireframe rendering with hidden line removal**
    - hidden line을 가리기 위해서는 두 번에 걸쳐 draw를 수행해야 한다.
    - 첫번째 draw에서 DEPTH_TEST를 활성화하여 면을 그리되, color mask를 끄고 polygon mode를 채워 그려서 깊이 버퍼만 채워둔다.
    - 이후, 두번째 draw에서 DEPTH_TEST를 유지하되, depth mask를 끄고 선만 그리면 가려진 부분은 그려지지 않게 된다.

- **Detailed Description of Additional Requirements:**
  - **Animated entities**
    - Player 클래스 내부의 draw_shape()에서 입력 방향에 따라 rotate를 적용한 모델 행렬을 생성한다.
    - Enemy, Escort Plane, Orbit 등도 각각 스케일 및 회전을 모델 행렬에서 처리하여 셰이더 렌더링으로 반영되게 구현했다.

- **Rationale Behind Program's Design:**
  - **Scene Graph :**
    - **sceneRoot의 draw()나 update()**만 호출하면 전체 Entity를 GLSL 기반 파이프라인으로 한 번에 처리할 수 있도록 수정했다.

- **How Design was Implemented:**
  - **Mesh class 주요 변경 사항항:**
    - OBJ를 로딩하는 시점에 바로 VAO/VBO를 생성하여 GPU에 업로드한다.
    - draw 함수 안에서는 glBindVertexArray + glDrawArrays만 호출한다.
    - 각 Entity는 생성자에서 필요한 OBJ 경로를 load_mesh로 로딩하고, draw_shape()에서 추가적으로 필요한 glm 변환과 색상 변환 수치를 gRenderer.draw_mesh()의 인자자로 넘기는 방식을 사용했다.

End-User Guide

- **How to Run and Operate Your Program:**

  1. **컴파일 (Compilation):**
     이번엔 간단한 명령어 만으로 동작할 수 있도록 Makefile코드를 추가했다. 먼저 `src` 디렉토리로 이동한 후 터미널에 아래 명령어를 입력하여 게임을 실행한다.

     ```
     >> make (or make all)
     ```

     위 명령어를 입력하면 `src` 디렉토리 내에 `main`이라는 실행 파일이 생성된다.

  2. **실행 (Execution):**
     생성한 파일을 실행하면 게임이 시작된다. make run을 입력하면 빌드 및 실행까지 한 번에 할 수 있다.
     make 명령어를 통해 이미 실행파일을 생성했다면, 간단히 ./main을 통해서도 실행가능하다.

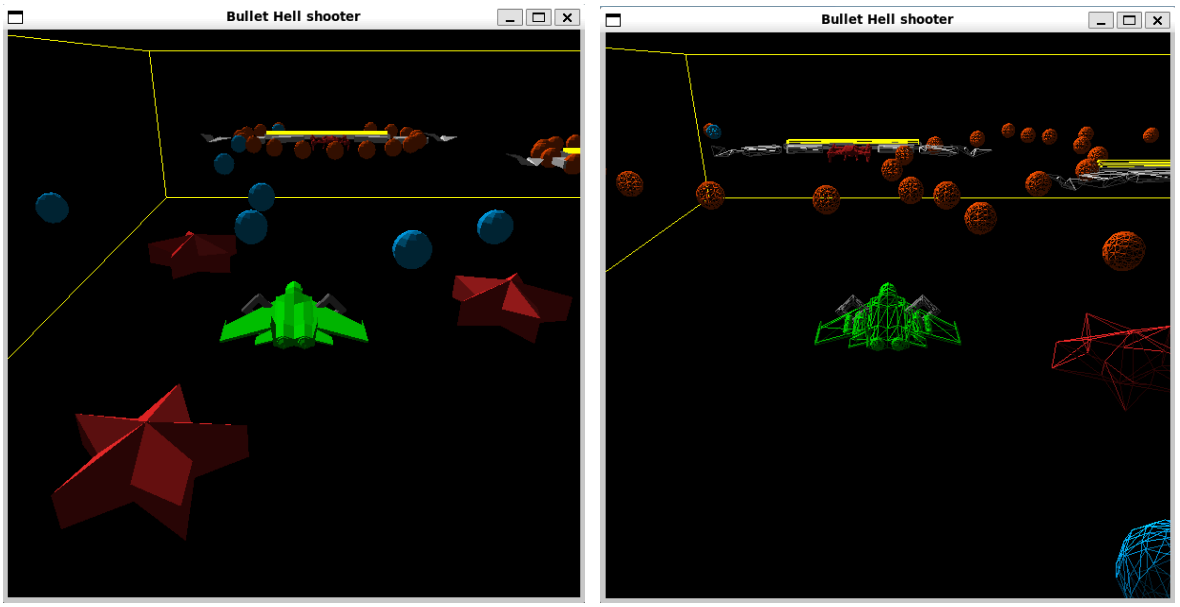
     ```
     >> make run (or ./main)
     ```

  3. **실행파일 정리:**
     ```
     >> make clean (/build 폴더와 main 실행파일 삭제)
     ```

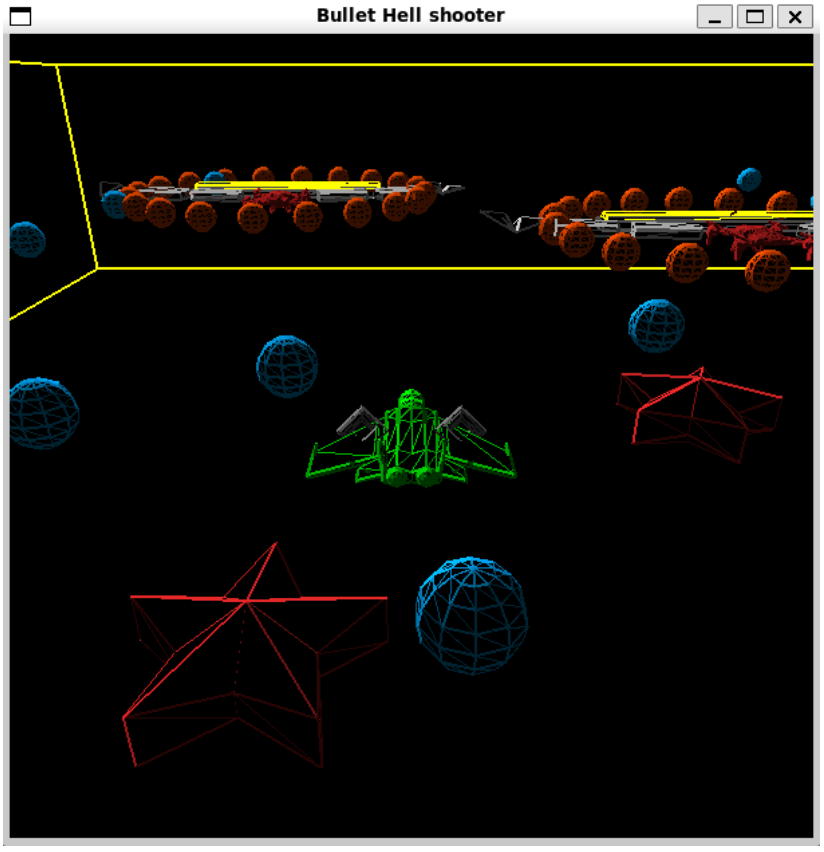
  4. **조작법 (Controls):**
     - **이동:**
       - ↑, ↓, ←, →: 방향키를 사용하여 Player 이동
     - **공격:**
       - `Space Bar`: 누르고 있는 동안 총알 발사
     - **게임 종료 시:**
       - `R` 또는 `r`: 게임 재시작
       - `Q`, `q`, 또는 `ESC`: 게임 종료

  Assn3_2의 README를 참고하여, 직접 git clone하고 컴파일 및 실행하는 방법도 있다.

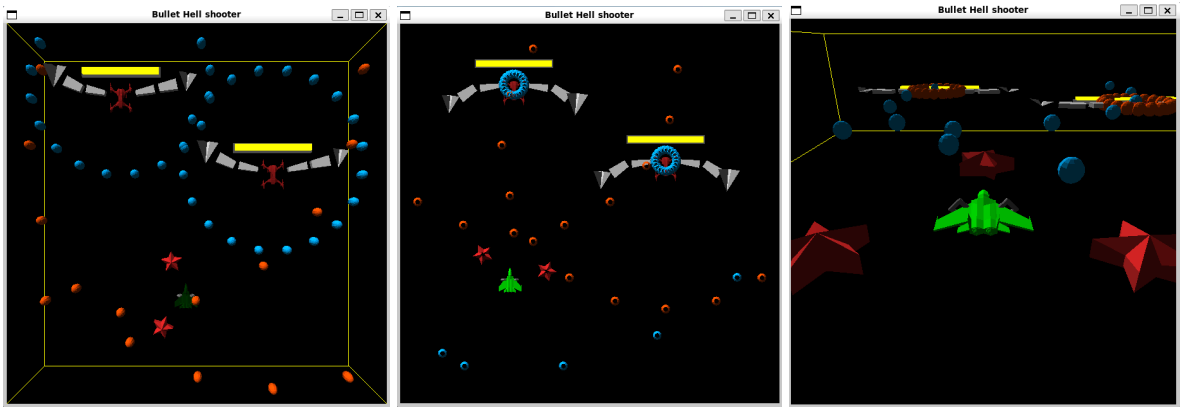
- **Screenshots of Each Feature:**
  1. **opaque polygon & wireframe**



  2. **wireframe with hidden line removal**

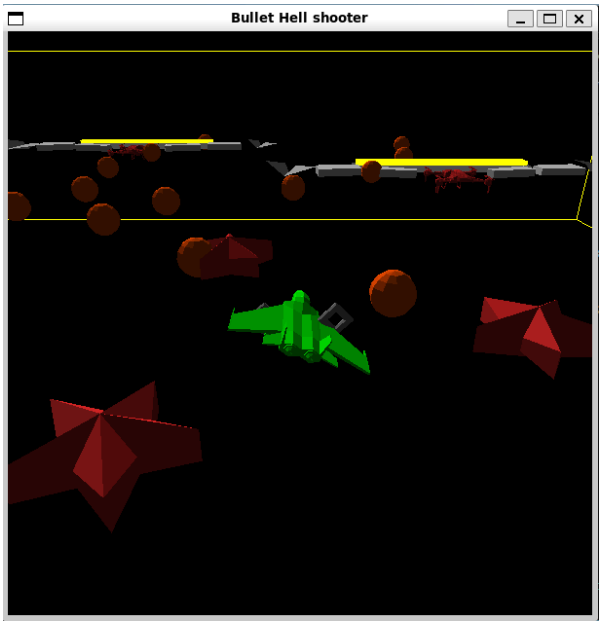


3. **top-view, orthographic view, third-person view**



4. **Destructible enemy body parts**

5. **animated entities(오른쪽 이동중)**



Discussions/Conclusions

- **Obstacles Encountered and Resolutions:**
  - 한 번의 draw만으로 hidden line removal을 구현할 수 없어, 면과 선으로 나누어 2번 그려 해결했다.
  - Hidden-line 모드를 구현하면서 `glPolygonMode`, `glDepthMask`, `glColorMask` 등 전역 GL 상태를 덮어쓰는 방식으로 구현하다 보니, 렌더 스타일을 여러 번 전환하거나 다른 경로가 뒤이어 호출될 때 의도치 않은 상태가 전파되는 문제가 있었다. 유저 관점에서 크게 의식하지 않을 정도의 문제지만, 디버깅과 이후 확장을 생각하면 Renderer 내부에서 변경 전 상태를 저장해 두었다가 각 draw 호출이 끝날 때 원복하는 편이 안전하다 판단해 capture_state, restore_state 유틸을 추가했다.

- **Ideas for Future Program Improvement:**
  - 현재 object들을 그리는 draw_mesh 함수와, 별들을 그리는 draw_raw 함수를 분리해서 사용했는데 하나의 함수로 일원화해서 구현이 가능할 것 같다.
  - 추후 vertex shader, fragment shader의 수정을 통해, illumination과 texture 처리 등으로 좀 더 발전된 형태의 그래픽을 다룰 수 있을 것이다.
- **Learnings and Conclusions from Assignment:**
  - Shader의 개념과 장점, 동작 원리를 코드를 통해 상세히 찾아보고 공부하게 되었다.

References

- **OpenGL Official Website** (https://www.opengl.org/)
- **OpenGL Extension Wrangler Library** (http://glew.sourceforge.net/)
- **freeglut** (http://freeglut.sourceforge.net/)
- **OpenGL Mathematics (GLM)** (https://glm.g-truc.net/0.9.9/index.html)

AI-Assisted Coding References

- **SunHo Cha**
  - **Tools Used:** Microsoft Copilot
  - **How Used:**
    - hidden line removal wire frame을 구현하는 방식을 찾을 때 사용했다.
  - **Application Location:**
    - Renderer의 draw_mesh 부분을 수정할 때, AI 코드를 참고해서 작성했다.
  - **Estimated AI Assistance Percentage:** 40%

- **SoonHo KIM**
  - **Tools Used:** ChatGPT Codex CLI, Gemini Pro
  - **How Used:**
    - shader 예제를 생성하고 workflow를 이해하기 위해서 사용
    - shader 관련 클래스 스켈레톤 구현하는데 사용
  - **Application Location:**
    - renderer, shader_program Class 스켈레톤 구현
    - basic.vert, .frag 파일 생성 및 검토
    - global GL state 관련 문제 디버깅
  - **Estimated AI Assistance Percentage:** 60%