Assignment 3-1 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:**
  - **Represented by 3D models**:
    - Player·Enemy·Bullet 등 각 entity에 대해 OBJ를 불러와 단일 scene (sceneRoot)에 배치했다.
  - **Graphic style & Camera and Rendering**
    - W키를 입력했을 때, 그래픽의 스타일이 Opaque → Wireframe로 순환되도록 구현했다.
    - C키를 입력했을 때, Top Perspective → Top Orthographic → Third-Person 뷰로 순환하는 전환 구조로 구현했다.
  - **Bounding box**
    - Bounding box를 렌더링하고, Player·Bullet 이동 시 좌표가 범위를 벗어나면 되돌리거나 풀에 반환해 문서에서 언급했던 조건을 만족하도록 구현했다.
- **Detailed Description of Additional Requirements:**
  - **Animated entities**
    - player의 draw_shape() 함수 내에서 Player의 입력 방향에 따라 기체를 좌우/상하로 기울이는 비행 애니메이션을 추가했다.
  - **Destructible enemy body parts**
    - Enemy의 체력이 60% 이하가 되면 양쪽 Upper/Lower Arm에 매달린 Escort Plane이 분리되도록 구현했다.
    - Detached라는 상태 변수를 추가하여, 해당 조건이 활성화 됐을 때, 기존 부모 노드의 역할을 하던 Lower Arm이 아닌 World 기준으로 아래 방향으로 이동하여 사라지게 구현했다.

- **Rationale Behind Program's Design:**
  - **Scene Graph :**
    - "All entities must be represented by a single scene graph." 이 조건을 만족하기 위해서 기존의 모든 객체를 하나로 관리할 수 있는 `class SceneNode`를 정의하고, 전역 변수 `sceneRoot`를 기준으로 각 객체들을 DAG 그래프 구조로 등록했다. main() 로직 안에서 가장 기본적으로 player와 각 enemy들이 sceneRoot의 자식들로 등록되며, 각 객체가 생성될 때 Constructor에서 부모-자식에 대한 세팅이 필수적으로 이뤄지도록 구현했다.
    - 이로써 sceneRoot.draw()를 실행하여 화면에 나오는 모든 object를 출력할 수 있도록 하였고,
      sceneRoot.update()를 실행하여 화면에 나오는 모든 object에 대한 update 로직을 수행하도록 하였다.

- **How Design was Implemented:**
  - **Mesh class** 사용:
    - 3D .obj 파일을 읽고 사용하기 위해 Mesh class를 구현하였다. OBJ 파일에 가지고 있는 positions과 normals 정보들을 벡터로 관리할 수 있다.
    - 각 Entity는 객체가 생성되는 시점인 Constructor 내부에서, 입력받은 OBJ 파일 경로를 참고하여 그에 대한 정보를 loading한다.
    - draw_shape에서 Mesh에 대해 추가로 필요한 Transformation이나 Color 값을 설정하고 최종 결과를 출력하도록 구성했다.

  - **Switch Graphic & Camera Mode**
    - GLUT 콜백에 매핑된 key_down함수 내부를 수정하여 구현했다.
    - 문서에서 언급한대로 w키에 대해 (Graphic) `currentStyle`을 전환하고,
    - c키에 대해 `projectionType`을 전환하게 된다.

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.

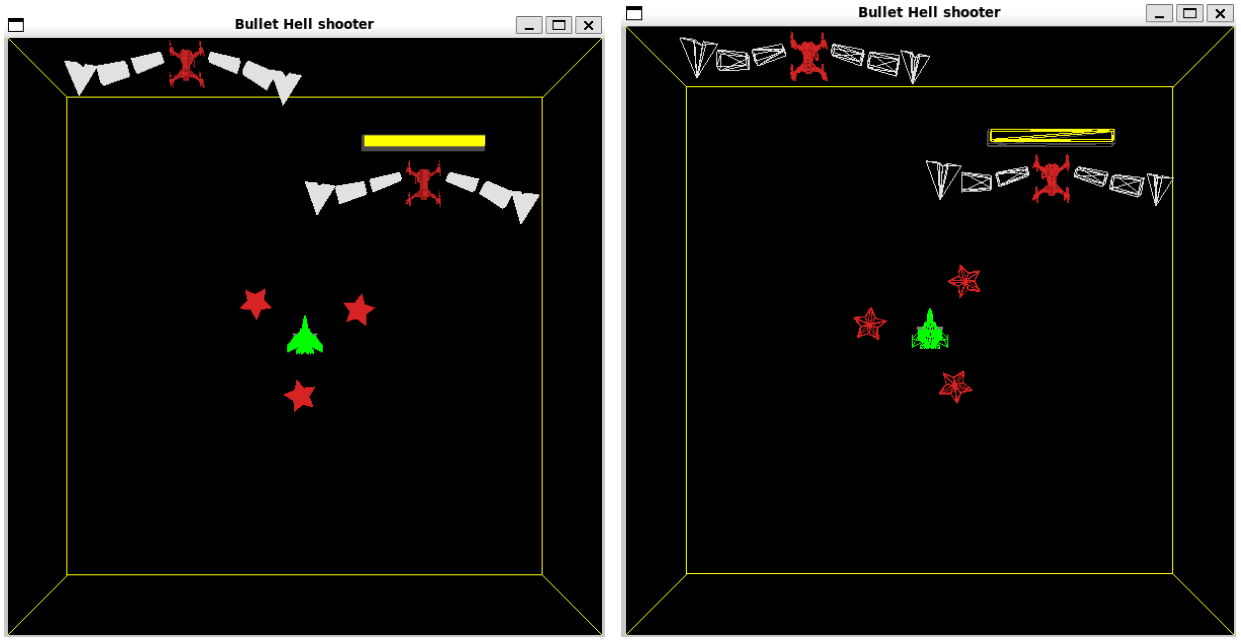
   ```
   >> make run (or ./main)
   ```

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

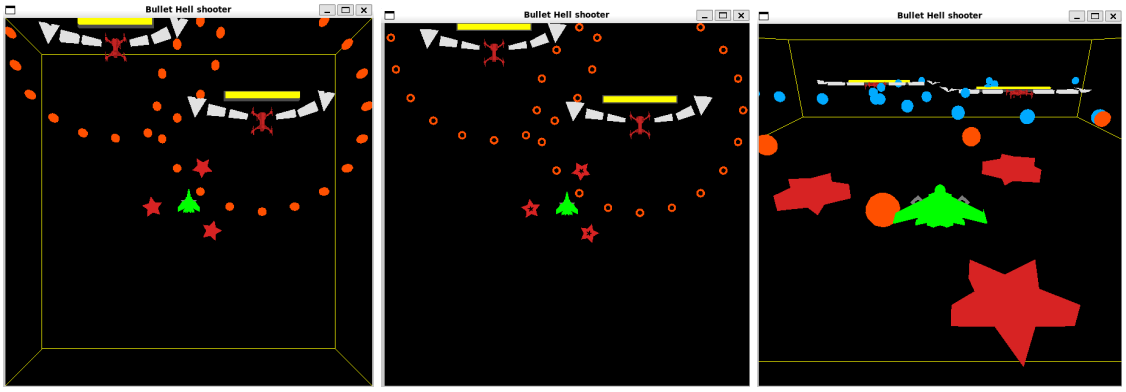
Assn3_1의 README를 참고하여, 직접 git clone하고 컴파일 및 실행하는 방법도 있다.

- **Screenshots of Each Feature:**
  1. 게임 플레이 화면 (그래픽 스타일)



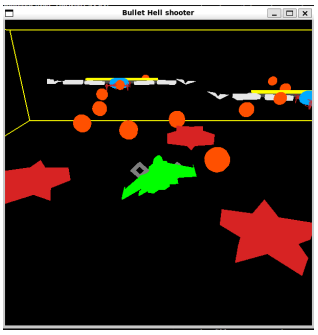
(왼쪽) opaque polygon, (오른쪽) wireframe style

  2. Camera Mode

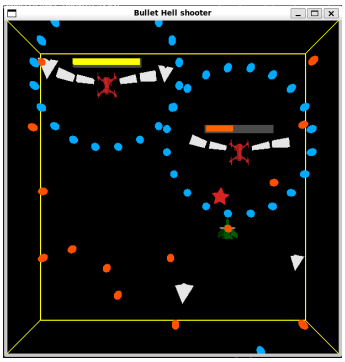


(왼쪽) Perspective, (중앙) Orthographic, (오른쪽) third-person view

  3. Animated entity(왼쪽 이동 중)



  4. Destructible body part(enemy hand)



Discussions/Conclusions

- **Obstacles Encountered and Resolutions:**
  - Destructible enemy body parts와 관련된 기능을 만들기 위해 기존 assn2에서 만들었던 Hand 클래스를 변환하여 구현했다. 이 때 마지막 Hand에 위치한 객체가 일관적으로 아래를 바라보도록 만드는 과정에서 어려움이 있었다. Upper와 Lower의 이동/회전을 상쇄하고, 왼쪽/오른쪽을 구별할 수 있는 인자 값을 추가하여 해결할 수 있었다.
  - 체력바를 구현할 때, 기존에 배포한 assets/square.obj를 기반으로 구현했는데, 원하는 모양을 연출하기 어려웠다. 이를 위해 아주 간단한 직육면체 .obj를 따로 정의하여 구현했다.
  - 기존의 update 함수를 update와 update_logic으로 구분하고, update_logic에서 각 클래스에 따른 동작을 구현하여 코드를 간결화할 수 있었다.
- **Ideas for Future Program Improvement:**
  - 현재는 camera를 전역변수로 관리를 했는데, 이를 Object 또는 다른 클래스로 감싸면 추후에 다중 카메라를 관리할 수 있을 것이다.
- **Learnings and Conclusions from Assignment:**
  - Graphic 스타일과 Camera View 전환 매커니즘을 명확하게 이해하게 됐다.
  - Destructible enemy body 기능을 구현하면서 부모-자식 간의 연쇄 변환 적용 관계를 제대로 이해할 수 있었다.

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:**
    - glm::perspective, glm::ortho의 사용법을 알기 위해 사용
    - 오류가 발생할 때 그 원인을 빠르게 파악하기 위해 사용
  - **Application Location:**
    - 이번 과제에는 AI로 생성한 코드가 없다.
  - **Estimated AI Assistance Percentage:** 20%

- **SoonHo KIM**
  - **Tools Used:** GitHub Copilot
  - **How Used:**
    - pdf 자료를 이해하기 위해서 사용
    - Health bar를 표현하기 위한 간단한 직육면체 .obj 파일 생성을 위해 사용
  - **Application Location:**
    - GL_CULL_FACE를 활성화 했을 때, 몇 객체들이 보이지 않는 문제에 대한 디버깅 요청
      → Vertex를 정의하는 순서에 따라 도형의 z방향(법선 벡터)가 정해지는 것을 알게 됨.
    - Mesh Class 스켈레톤 구현
  - **Estimated AI Assistance Percentage:** 40%