

# Assignment 4 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:

- **Shading style**
  - Gouraud, Phong, Phong+NormalMap 세 가지 파이프라인을 각각 별도 GLSL 셋으로 올려 두고, **W** 키를 누를 때마다 순환하도록 구현했다.
  - 모드가 바뀌면 **gRenderer**가 해당 셔이더 즉시 바인딩하고, 공통 유니폼(모델·뷰·투영 행렬, 조명, 텍스처 슬롯 등)을 재설정한다.
  - 노멀맵 모드일 때 모델이 UV나 탄젠트가 없으면 자동으로 Phong으로 fallback해서 화면이 깨지지 않도록 했고, UV가 있는 경우엔 OBJ parser가 삼각형 단위로 탄젠트를 생성·정규화해 VAO 슬롯 3번에 넣는다.
- **Light sources**
  - 방향광(Directional Light) 1개와 점광(Point Light) 3개를 구현했다.
    - **방향광**은 약간 기울어진 하향 방향으로 맵 전체 명암을 깔아 주었다.
    - **점광**에 대한 거리 감쇠에 필요한 값들은 셔이더에 공통 상수로 두어 계산했다.
  - 프레임마다 위치가 업데이트되면, 함수 **set\_lights()**를 통해 유니폼 배열에 일괄 업로드된다. ⇒ 이를 통해, 셔이팅 모드가 바뀌어도 조명 일관성이 유지된다.

- **Detailed Description of Additional Requirements:**

1. **Multiple Lights**

- Player 주위를 공전하는 기본 점광원(Point Light)에 더해, 활성화 된 **Enemy**마다 보조 점광원을 추가했다.
- 이를 위해 **MAX\_POINT\_LIGHTS** (= 4)까지 배열로 전달하도록 Renderer·Shader를 확장하였다.
- GLSL에 uPointLightCount와 포인트 라이트 배열을 추가해 각 광원의 위치·색·세기를 누적 조명에 반영하며, Enemy의 보조광은 상단에 고정 배치했다.

2. **Motion Blur**

- 후처리를 해주는 blur shader를 추가하여 구현하였다.
- 본래 게임화면을 color/velocity texture와 연결된 FBO에 출력한다. 이후, 각각의 texture를 blur shader에서 읽어, velocity에 따른 blur를 구현하여 디스플레이에 출력한다.

3. **Planar Shadow**

- 직접광으로 인해 생기는 그림자를 구현했다.
- 광원의 시점으로 바라보았을 때, 어떤 물체들이 가려지는지에 대한 정보를 depth map texture에 기록한다. 또한 이 광원의 시점에서의 좌표 계산을 위해 lightSpace 행렬을 유지한다.
- 이후 lightSpace 행렬로 구한 광원 시점에서의 fragment 위치와 depth map texture 정보보를 비교하여 그림자가 지는 부분인지를 판단할 수 있다.
- gouraud의 경우 fragment 단위 분석이 어렵기에 phong/phong normal에 대해서만 그림자 기능을 구현했다.

- **Rationale Behind Program's Design:**

- **Texture Loading Stack:**

- 처음엔 PNG 파일을 직접 파싱해 보려 했는데, 컬러 포맷/압축/엔디안 처리까지 신경써야하는 것을 알게 됐다.
  - 그래서 의존성을 하나 추가하여 **libpng**를 설치하고 texture.cpp/.h에서 **<png.h>** 기반 로더를 사용했다. sRGB 해제, 알파 채널, 스트라이드 정렬 같은 귀찮은 부분을 이 라이브러리로 처리했다.

- **Diffuse Color Modulation**

- 프래그먼트 셰이더에서 `vec3 baseColor = texSample * uColor.rgb;`로 텍스처 색과 유니폼 색을 곱하도록 구현했다.
  - 기본적으로 `uColor`를 (1,1,1)로 두면 PNG 원본 색을 그대로 사용할 수 있고, 필요에 따라서 추가 색상 조작이 필요할 때 유니폼 값만 바꿔서 색조 변형이나 팀 색상 효과를 주기 위해 이와 같이 구현했다.

- **Debug-Friendly Toggles**

- 이번에는 키보드 **T**와 **B** 키에 새롭게 기능을 추가했다.
  - **t/T 일시정지 키:** 게임 로직을 멈춘 상태에서 셰이딩·카메라·조명 전환을 비교하려고 넣었다. 업데이트 루프가 멈춰야 광원이 도는지, 노멀맵 하이라이트가 어떻게 바뀌는지 한눈에 확인할 수 있어서 데모 때 매우 유용했다.
  - **b/B 배경 전환 키:** 조교님이 제공해주신 `diffuse_ocean_*`, `diffuse_sky_*`의 day/night 페어를 전부 활용하고자 위 기능을 구현했다. 단순히 색만 바꾸는 대신 텍스처 세트를 교체하는 방식으로 구현했다.

- **Frame buffer object (FBO)**

- `texture`를 생성할 때, 게임화면의 이미지를 디스플레이가 아닌 다른 버퍼에 출력할 수 있는 기술이 필요했고, 이를 FBO가 제공한다. `texture`와 FBO를 attach해놓으면, 이후 `glBindFramebuffer()` 함수를 통해 출력할 버퍼(FBO or 메인 디스플레이)를 선택하고, `texture`에 기록할 수 있다.

- **How Design was Implemented:**
  - **Renderer & Mesh Modules**
    - core/render/renderer.cpp에서 셰이딩 모드별 프로그램을 로드해 유니폼 위치를 캐싱하고, `set_shading_mode()`로 활성 셰이더만 바꾸도록 구현했다.
    - 같은 파일의 `draw_mesh()`, `draw_raw()`가 diffuse-normal 텍스처 슬롯(0/1)과 라이트 유니폼을 일괄 세팅하며, UV·탄젠트가 없을 때는 Phong으로 폴백하는 기능도 작성했다.
    - OBJ 파싱과 UV·탄젠트 생성은 core/render/mesh.cpp에서 담당하며, VAO 슬롯 0/1/2/3(위치/법선/UV/탄젠트)로 올려 준다.
  - **Vertex & Fragment Shaders**
    - Gouraud, Phong, Phong+NormalMap 세 가지 파이프라인에 대한 셰이더를 구현할 때 수업 자료(Lec13. Illumination and Shading)의 내용을 적극 활용하였다.

## End-User Guide

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

### 1. 컴파일 (Compilation):

간단한 명령어만으로 동작할 수 있도록 Makefile코드를 작성했다. `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 run`을 입력하면 빌드 및 실행까지 한 번에 할 수 있다. `make` 명령어를 통해 이미 실행파일을 생성했다면, 간단히 `./main`을 통해서도 실행 가능하다.

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

### 3. 실행파일 정리:

```
>> make clean (/build 폴더와 main 실행파일 삭제)
```

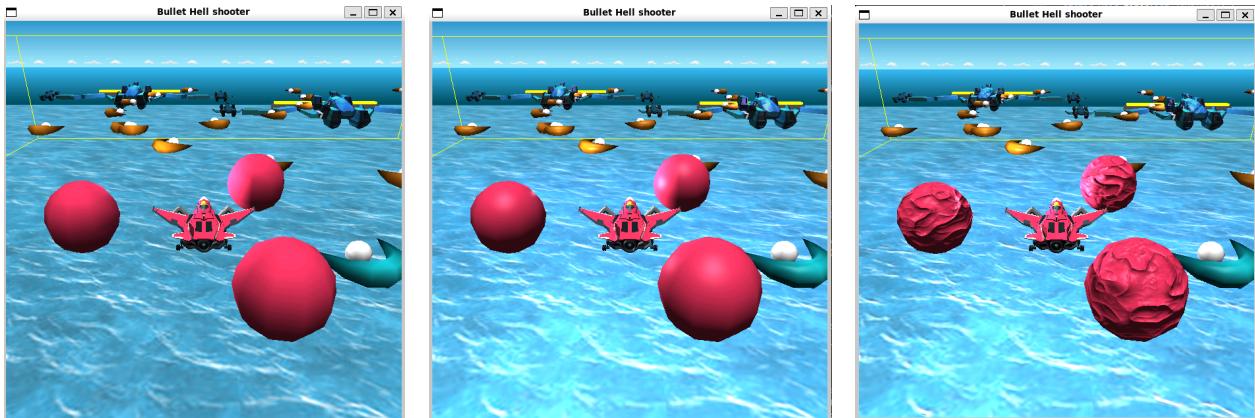
### 4. 조작법 (Controls):

- **이동:** 방향키
- **전투(발사):** 스페이스바 (누르고 있기)
- **음영 모드:** W / w
  - **순환:** Gouraud → Phong → Phong Normal Map
- **카메라 시점:** C / c
  - **순환:** Perspective ↔ Third-person (no orthographic mode)
- **일시정지/재개:** T / t
- **주/야간 전환 (환경 텍스처):** B / b
  - **전환:** 주간 ↔ 야간 하늘/바다 텍스처
- **그림자 전환:** S / s
- **모션 블러 전환:** M / m
- **게임 오버:**
  - **다시 시작:** R / r
  - **종료:** Q / q / ESC

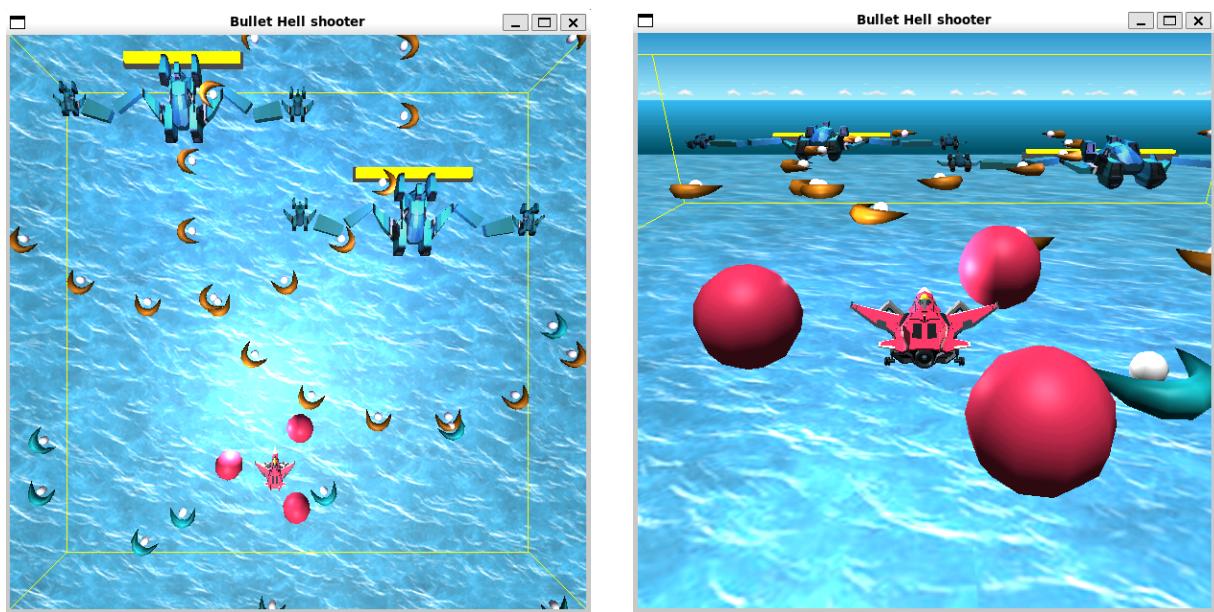
[Assn4](#)의 README를 참고하여, 직접 git clone하고 컴파일 및 실행하는 것도 가능하다.

- Screenshots of Each Feature:

1. gouraud (왼쪽) / phong(중앙) / phong normal(오른쪽)



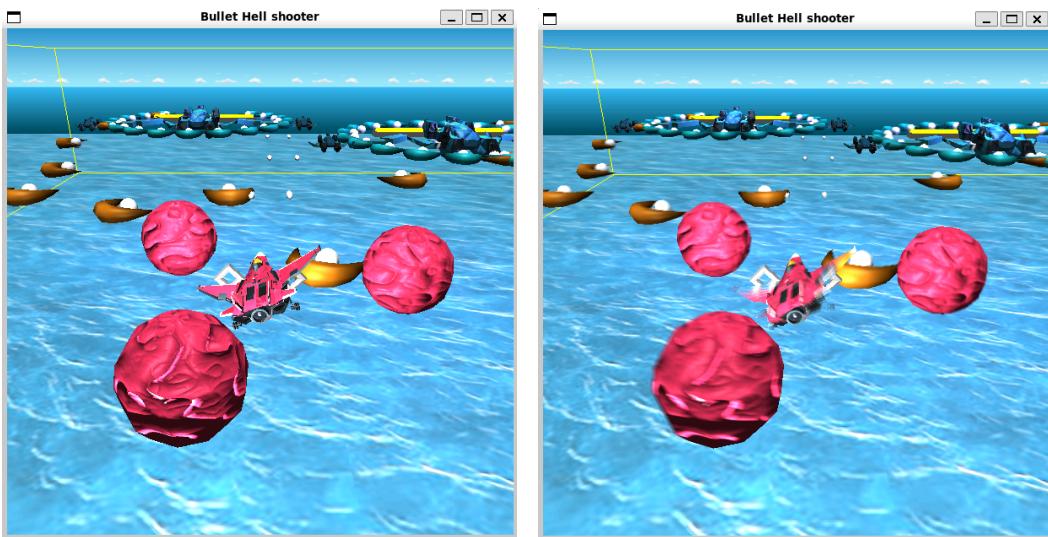
2. perspective view (왼쪽) / third-person view with perspective projection(오른쪽)



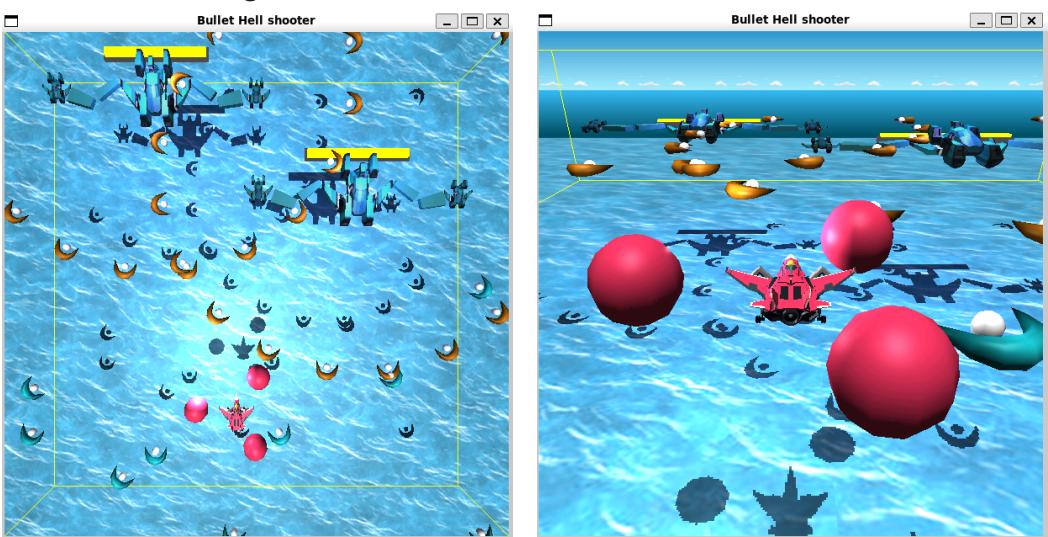
3. Day Background(왼쪽) / Night Background(오른쪽)



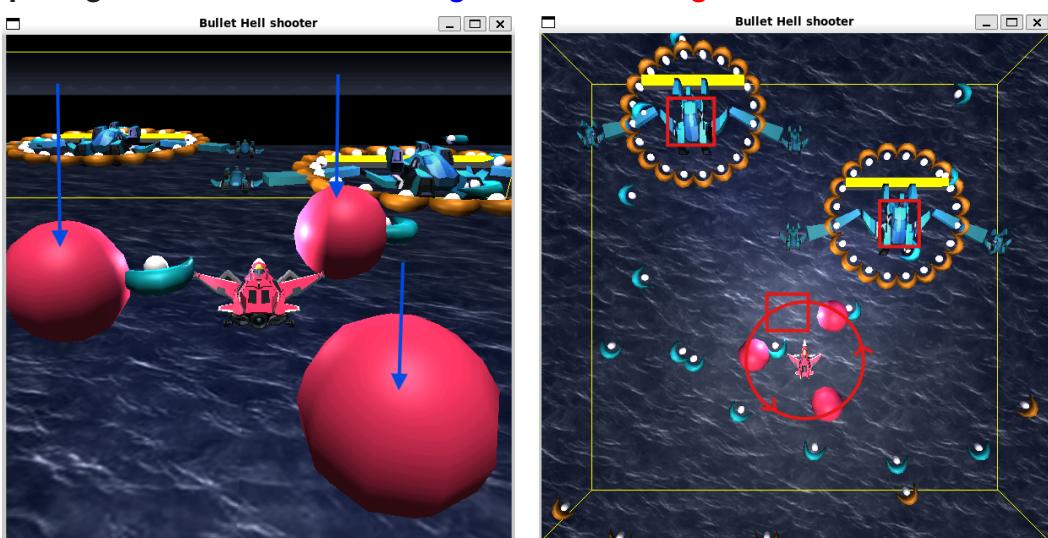
#### 4. motion blur 적용 전 / 적용 후



#### 5. planar shadow casting



#### 6. Multiple Light source : Directional Light(1개) / Point Light(3개)



# Discussions/Conclusions

- **background 구현 과정:**
  - 3인칭 시야로 전환했을 때 전방 벽이 검은 면으로 사라지는 문제가 발생했다.
  - 여러 시도 끝에, 텍스처 면을 한쪽 면만 렌더링하는 기본 백페이스 컬링 때문이라는 것을 알게 됐다.
  - 배경을 그리는 동안 GL\_CULL\_FACE를 잠시 끄고 양면 렌더링하도록 수정하여, 카메라 위치·각도에 상관없이 벽 텍스처가 항상 보이도록 수정했다.
- **Gouraud Lighting Insight:**
  - 게임을 실행해보면 Gouraud 셰이딩이 다른 셰이딩 모드에 비해서 상대적으로 어두운 것처럼 보인다.
  - 현재 바닥에 깔린 면은 두 개의 거대한 삼각형들로 채워져있다. Gouraud는 Phong과 다르게 정점 단위로 조명을 계산하기 때문에, 바닥에서는 정점들이 점광원으로부터 멀리 떨어져 감쇠가 크게 적용된다.
  - 그 결과 매우 작은 조명값이 면 전체로 보간되어 하이라이트가 거의 사라지며, 동시에 Phong/Phong+Normal Map 모드처럼 픽셀 단위로 재계산하는 경우와 뚜렷한 대비를 만드는 것을 확인할 수 있었다.
- **Write Two Texture:**
  - 두 개 이상의 texture를 FBO에 attach하여 사용할 때 첫번째로 attach한 texture에만 기록이 되는 일이 있었다. 두 texture의 포맷이 달라서 (GL\_RGBA16F vs GL\_RG16F) 발생한 문제였고, 두개 이상의 texture를 FBO에 연결해서 사용할 경우 포맷을 통일하는 것이 중요하다는 것을 확인했다.
- **prevModelMatrix 계산:**
  - Motion Blur를 구현할 때, blur 효과가 이상하게 적용되는 경우가 있었다. 이는 각 object마다 prevModelMatrix를 계산할 때, draw\_shape 함수 내에서의 일시적 transform을 다 고려하지 않고 계산을 하여 발생하는 오류였다.

## 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 Google Gemini
  - **How Used:**
    - blur와 shadow를 shader 내에서 구현하는 아이디어를 얻기 위해 사용
    - FBO의 연결방법을 공부하기 위해 사용
  - **Application Location:**
    - shader들 내 shadow + motion blur 관련 부분
    - FBO 초기 설정 부분
  - **Estimated AI Assistance Percentage:** 70%
- **SoonHo KIM**
  - **Tools Used:** ChatGPT Codex CLI
  - **How Used:**
    - .vert, .frag 예제를 생성하고 workflow를 이해하기 위해서 사용
    - texture 관련 클래스 스켈레톤 구현하는데 사용
  - **Application Location:**
    - main.cpp 코드 refactoring
    - background 구현 과정 중, back face culling 문제 발견
    - [README.md](#) 업데이트
  - **Estimated AI Assistance Percentage:** 70%