Assignment #4 Report

팀명:	스타벅스	
팀원:	최우석	성창환
학과:	단일계열	컴퓨터공학과
학번:	49003157	20180480
Hemos ID:	f2soundd	chseung

Contents

Abstract

Lighting

- Directional light 및 Point light 를 구현하였다.
- 숫자키를 통해 각 광원의 사용여부를 선택할 수 있도록 하였다.
  - 숫자키 1: 두 광원 모두 사용하여 렌더링
  - 숫자키 2: Directional light 만 렌더링
  - 숫자키 3: Point light 만 렌더링
  - 숫자키 4: 광원 미지원

Shading

- Gouraud shading 및 Phong shading 모드를 구현하였다.
- 두 셰이딩 모드에 대해 각각 GLSL 파일을 작성하였다.
- s 키를 통해 두 셰이딩 모드를 선택할 수 있도록 하였다.
- 두 셰이딩 모드 모두 diffuse, specular, ambient 를 사용한다.

Texture Mapping

- 캐릭터, 배경에 대한 Diffuse 텍스쳐 매핑을 구현하였다.
- 행성계에 대한 Diffuse 텍스쳐 매핑과 Normal 텍스쳐 매핑은 구현하지 못하였다.
- 행성계 대신 캐릭터에 대해 Normal 텍스쳐 매핑을 구현하였다.

Misc.

- 우주 이미지를 이용하여 정면과 바닥을 Diffuse 텍스쳐 매핑하여 배경을 표현했다.
- 배경에 대해 Normal 텍스쳐 매핑은 구현하지 않았다.
- 외부 모델파일(.obj)을 로딩하는데 assimp 라이브러리를 사용했다.
- 외부 이미지를 로딩하는데 stb 라이브러리를 사용했다.

Data Structures

Application

Light Factors

광원과 관련된 변수들을 담는 구조체를 include/core/Light.hpp 에 정의했다. 각 광원과 연관된 인자들을 아래와 같이 어플리케이션에 정의하였으며, 셰이더에서도 아래와 같은 형식을 사용한다.

Directional Light 의 인자
- 빛의 색상
- 빛의 방향
- Ambient 의 세기
- Specular 의 세기
- 반짝임 정도

Point Light 의 인자
- 빛의 색상
- 빛의 위치
- Ambient 의 세기
- Specular 의 세기
- 반짝임 정도
- Attenuation 값을 계산하기 위한 상수항, 1차항, 2차항

Vertex

include/core/Mesh.hpp 에 정의되어있는, 점을 표현하는 구조체이다. 아래와 같은 필드를 가진다.

Vertex 의 필드
- 점의 위치
- 법선 벡터
- 텍스쳐 좌표
- Tangent 벡터
- Bitangent 벡터

Implementation

Implementation - Lighting

Directional Light

씬그래프의 노드로서 작동하는 Object 클래스를 상속하는 Sun 클래스를 정의했다. 캐릭터, 행성계 등과 같이 이 객체 또한 씬그래프 내에서 관리될 수 있으며, 씬그래프의 업데이트시 일정한 궤도를 움직이며 반원운동을 하도록 구현했다. 다른 Object 객체들에 비해 다른 점은 광원으로서 기능하기 위한 Light factor 들이 멤버변수에 정의되어있다는 점이다. 게임 시스템을 총괄하는 GamePlay 객체는 씬그래프의 렌더링을 진행할 때 이 Sun 객체에 저장된 Light factor 구조체를 참조하여 모든 Object들이 각각의 셰이더에 Directional Light에 필요한 변수들을 전달할 수 있도록 했다. 셰이더 프로그램에서 Directional light 는 vec3 getDirectionalLight() 함수를 통해 계산된다.

Point Light

행성계 클래스(Planetary)에 Point light의 계수들이 정의된 구조체를 멤버변수로 두었다. 이 계수들 중 광원의 위치를 위성의 위치로 정의함으로써 위성과 상대적으로 고정된 위치를 가지도록 하였다. 이 계수들 역시 GamePlay 객체가 씬그래프 렌더링을 진행할 때 활용한다. 셰이더 프로그램에서 Point light 는 vec3 getPointLight() 함수를 통해 계산된다.

Implementation - Shading & Texture Mapping

Gouraud 와 Phong 셰이딩은 각각 다른 셰이더프로그램에 작성되었다.

Gouraud: shader/gouraud_vertex.vert | shader/gouraud_fragment.frag
Phong: shader/phong_vertex.vert | shader/phong_fragment.frag

Phong 의 경우, normal 텍스쳐 매핑을 위해 vertex 셰이더에서 TBN 행렬을 계산하고, 이를 이용해 어플리케이션으로부터 전달받은 view position, fragment position, light direction, point light position 을 world space 에서 tangent space 로 변환한 뒤 fragment 셰이더로 전달한다. fragment 셰이더에서는 전달받은 인수들을 통해 directional light와 point light 을 계산한 뒤 이 둘을 합친 광원계수를 diffuse값에 곱하여 fragment 의 색상을 결정하도록 구현하였다.

Gouraud 의 경우, directional light와 point light 을 계산한 뒤 합하는 계산까지 모두 vertex 셰이더에서 수행한 뒤, 이 광원계수를 fragment 셰이더에 전달하고 fragment 셰이더는 전달받은 광원계수를 diffuse 값에 곱해 fragment 의 색상을 결정하는 것만 수행하도록 구현하였다.

행성계에 대한 Diffuse mapping 및 Normal mapping 은 구현하지 못하였으며, 캐릭터의 DIffuse map과 Normal map 은 assimp 라이브러리를 이용해 어플리케이션의 Mesh 클래스에 매핑되도록 했다.

How to Build and Run

Requirements

- Visual Studio 2019 for C++ Desktop

Build and Run

1. Double click file `shmup.vcxproj` to open the project in Visual Studio 2019.
2. Click `Build --> Build Solution` , or press `F7`
3. Click `Debug --> Run without Debug` , or press `Ctrl + F5`

Application Instructions

- ←↑→↓: Move player.

- Space Bar: Fire a bullet.

- c: The player becomes invincible, and the enemy dies in one shot.

- f: Players cannot fire bullets, and the game is over as soon as an enemy bullet is hit.

- r: On/Off wireframe rendering.

- v: Change viewing mode. TPS, FPS, and 2D are supported.

- t: Diffuse mapping on/off

- n: Normal mapping on/off

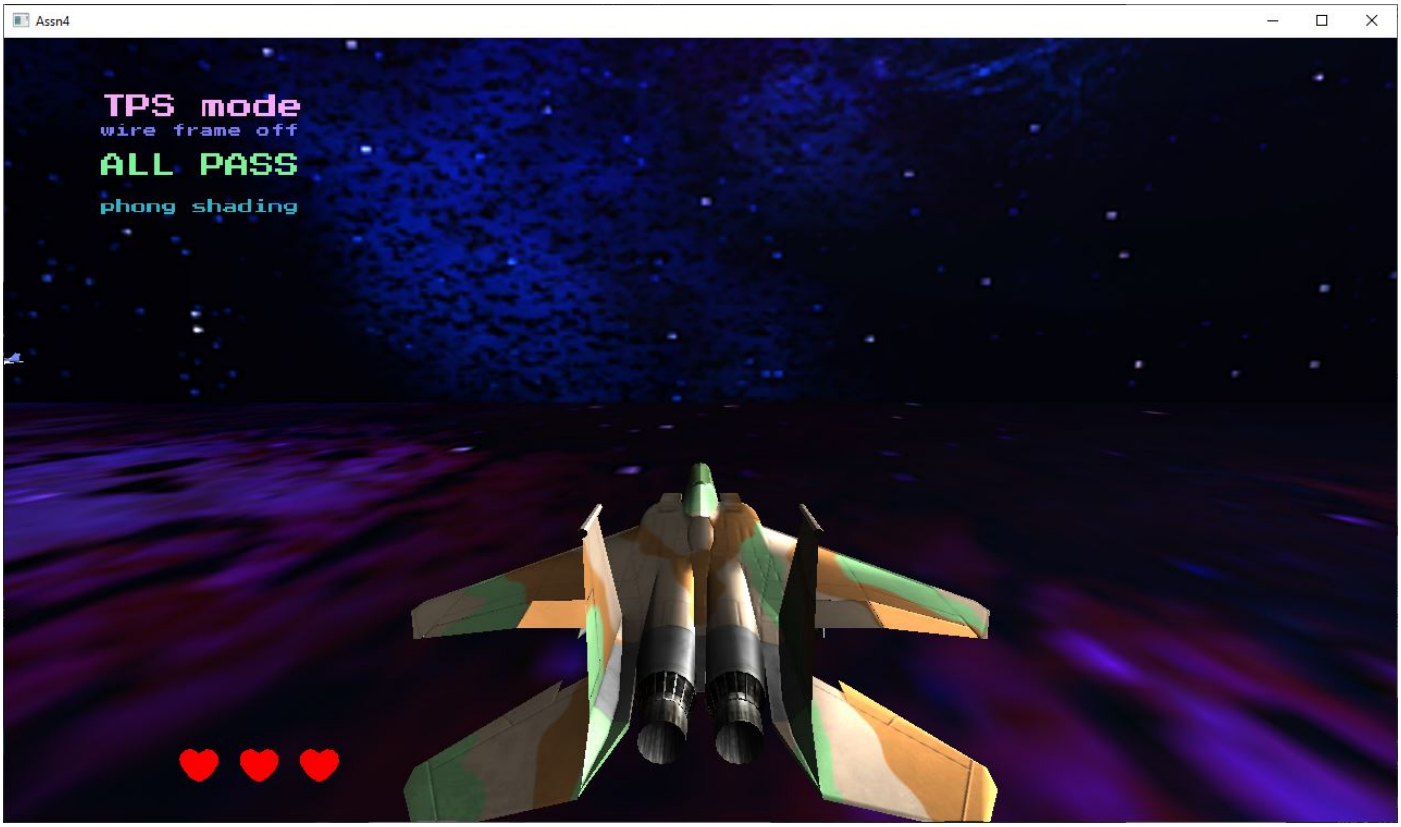
- 1: Use directional light(sun) and point lights(planetary)

- 2: Use directional light only

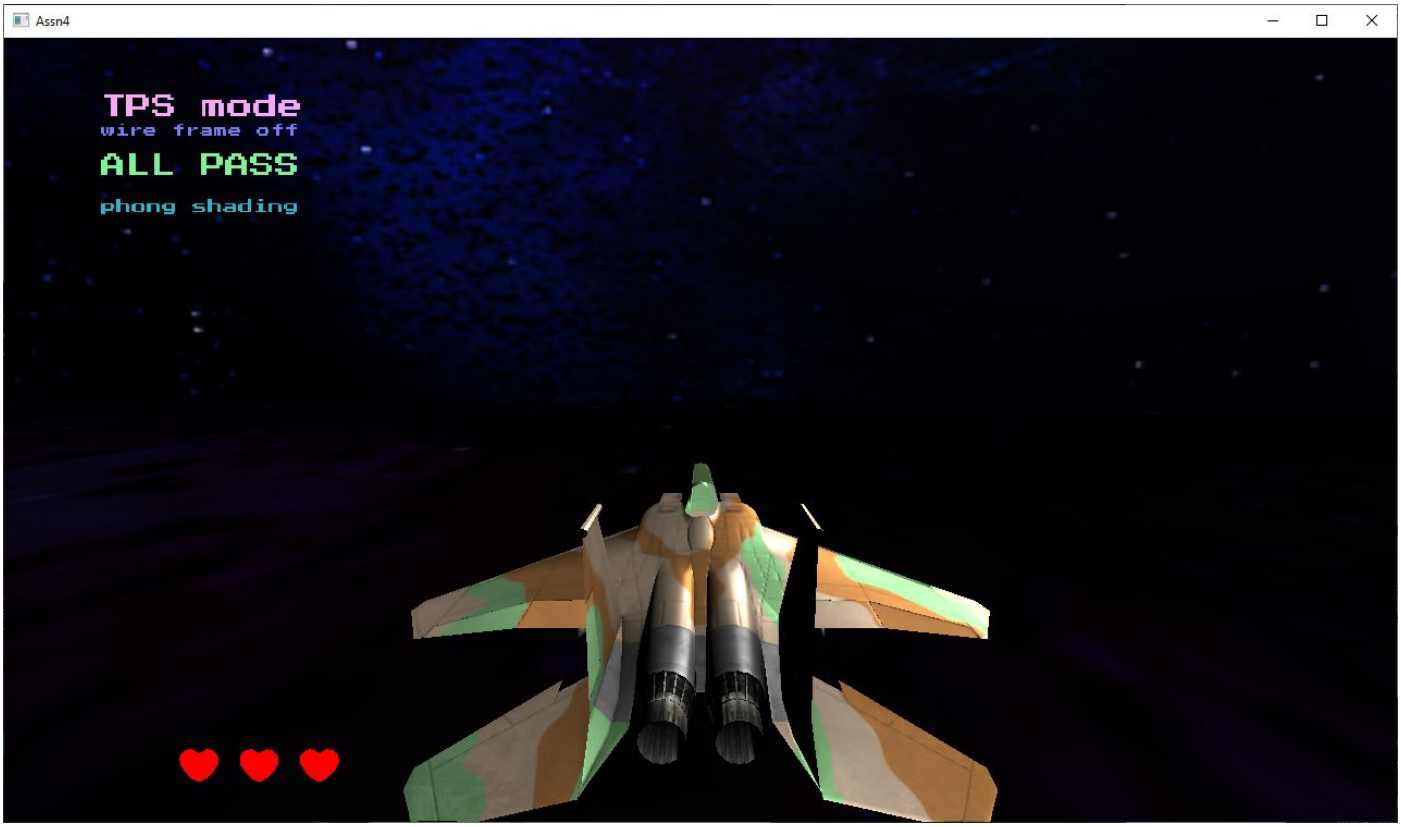
- 3: Use point lights only

- 4: No lighting

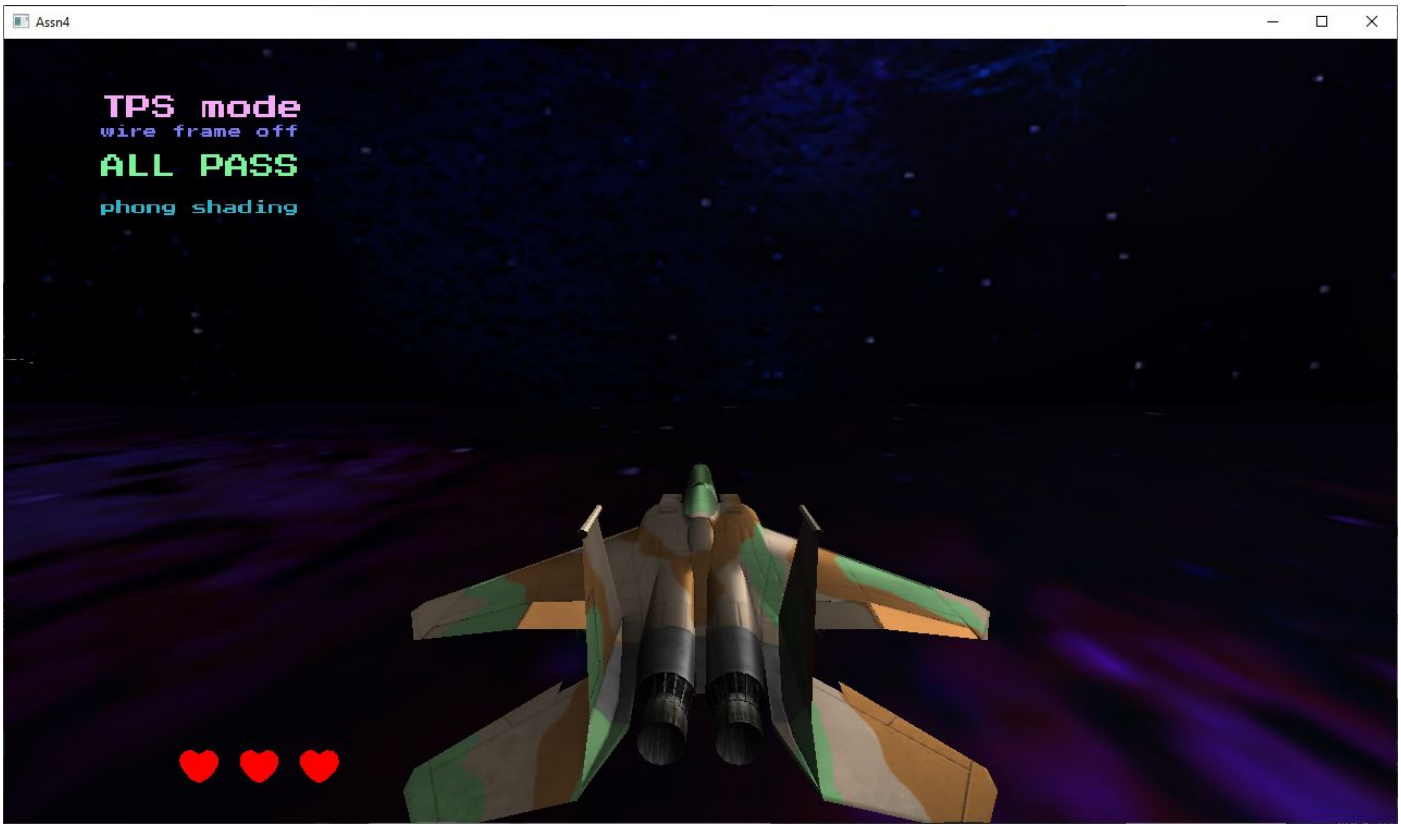
Result Screen



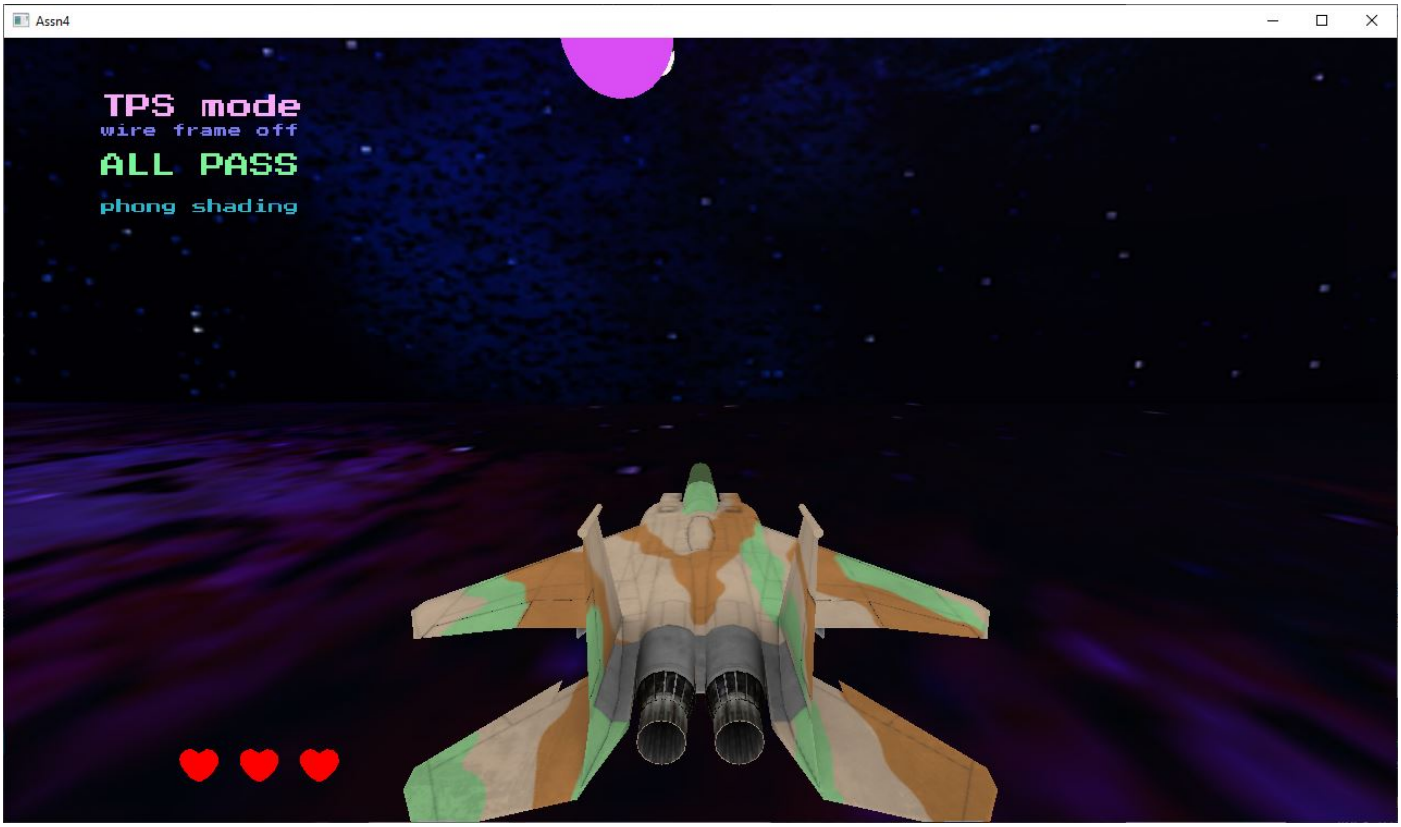
<Directional Light, Point Light>



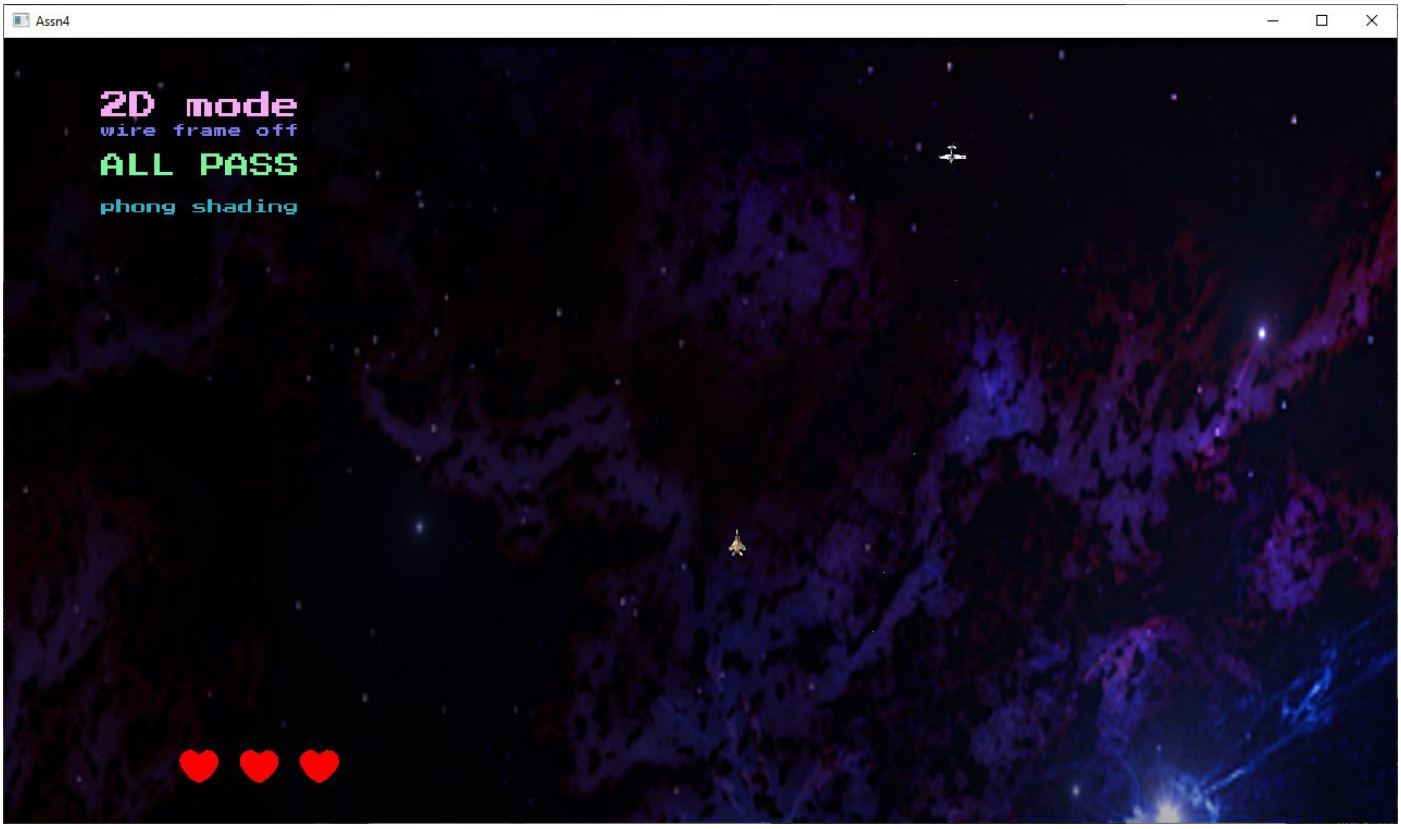
<Directional Light only>



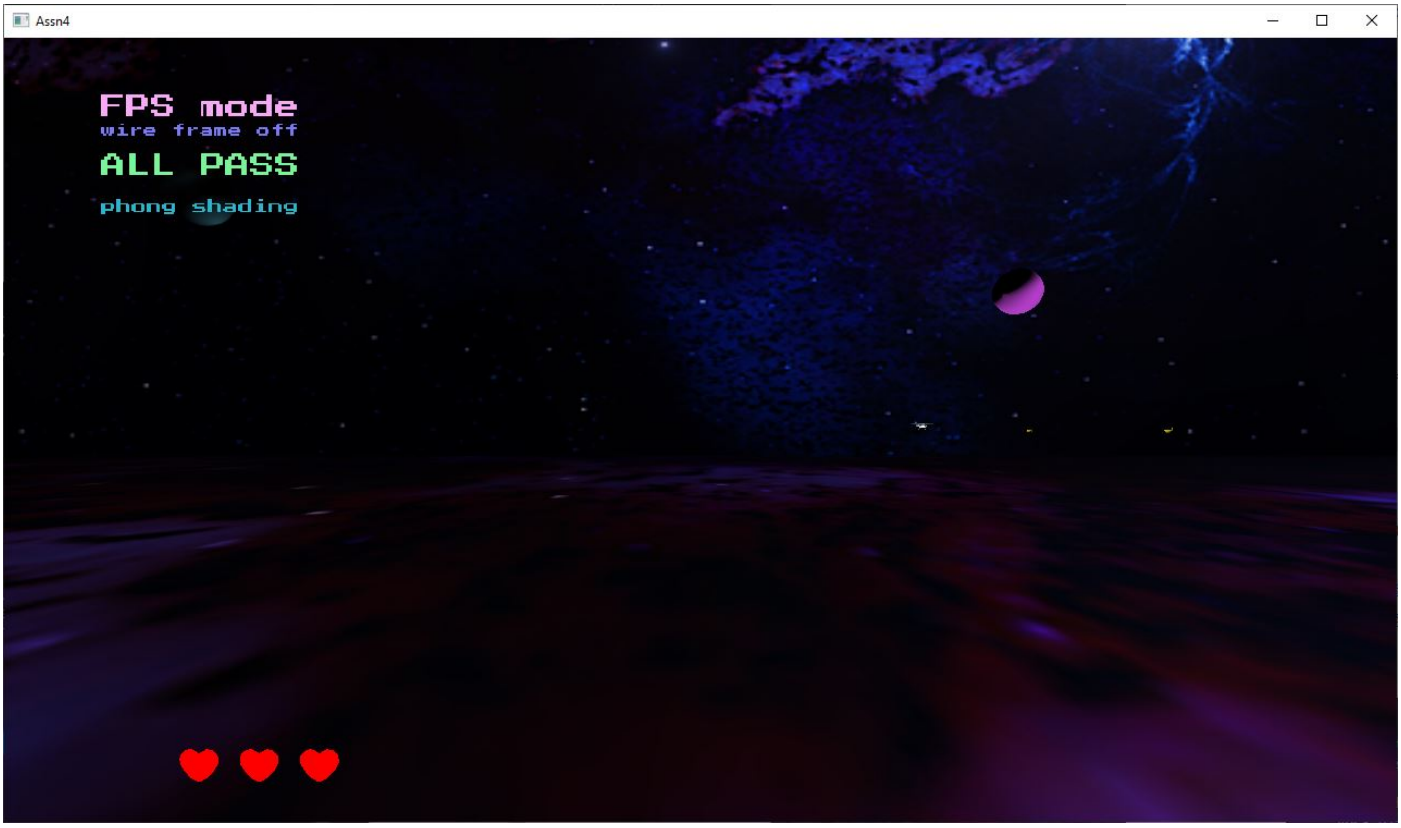
<Point Light only>



<No lighting>



<Top-view>



<FPS view>

Appendix

Development Environment

- Windows (x64)
- Visual Studio 16.9.1
- freeglut 3.0.0
- glew 2.1.0
- glm 0.9.9.7
- assimp 5.0.1

Resources

Assets

3D Models
https://www.cgtrader.com/free-3d-models

sText Models (“assets/models/text3d_*”)
http://profilki.pl/en/generators/3d-texts

Others

The Open Asset Import Library (assimp)
https://github.com/assimp/assimp

Image loader (stb)
https://github.com/nothings/stb

How to Normal Mapping
https://learnopengl.com/Advanced-Lighting/Normal-Mapping