**Assignment #2**

팀 이름 : 복학왕

팀 멤버 : 20120208 변한섭

20120009 이재엽

1. **프로그램 기능 개요**

이번 프로젝트에서는 Assignment #1을 추가적으로 확장하였다. Assignment #1과 가장 크게 달라진 점은 openGL 내장 함수를 사용하지 않고 shader를 사용하여 그리기를 하였으며 Enemy와 Player를 계층구조로 움직임을 구현한 점이다. 또한, 이 외에도 몇 가지 추가적인 기능을 구현하였다. 먼저, life를 구현하여 Player가 Enemy와 접촉할 시 life를 1씩 감소하도록 하였으며 Remaining life가 화면 왼쪽 상단에 표시되도록 하였다. 그 다음, Player가 총을 쏠 때 총을 쏘는 방향으로 총구를 겨누는 모션을 추가하였으며 r 버튼을 눌러 재시작하는 기능 및 일정시간이 지나면 패배하는 기능도 추가하였다.

1. **프로그래밍 환경**

32비트 환경에서 VS2017, glew2-1-0, freeglut을 이용하여 개발하였다. (LMS에 올라온 버전 사용)

1. **프로그램 설계 및 구현**

먼저, 기존의 Assignment #1의 게임 설계는 동일하게 차용하였다.

또한, textbook의 코드를 다수 사용하였다.

사용한 코드는 아래와 같다.

----------------------------------------------------

<그대로 사용>

Angel.h

mat.h

vec.h

checkError.h

vertices.h

InitShader.cpp

patches.h

vshader\_a9.glsl

fshader\_a9.glsl

<변형하여 사용>

append\_example.cpp

----------------------------------------------------

위의 코드는 textbook의 Appendix A. 9번 example이다. 해당 example은 사람 모델을 계층구조로 설계하는 example이다.

먼저 해당 example의 계층구조 모델을 그대로 사용하고 팔다리의 회전 방향, 크기만 조금 조정하여 Player, Enemy 각각 키보드 입력 및 시간의 흐름에 따라 rotation 값을 바꾸어 팔다리를 움직이며 이동하도록 수정했다.

1. 그리기

이번 프로그램에서는 openGL 내장함수를 이용하지 않고 그리기를 수행해야 했으므로 VAO(Vertex Array Object)를 이용했다.

Player, Enemy, Bullet, Wall, Item1, Item2, 승리화면, 패배화면에 대한 VAO를 각각 만들어 주었고 모든 VAO는 colorcube 메서드를 참고하여 정육면체 형태로 만들어 주었다.

그런 다음 Assignment #1에서 저장해 두었던 각각의 Graphical Object에 대한 정보를 Modelview matrix에 전달해주었고 Projection Matrix에 ortho함수를 통해 윈도우를 정의해주어 화면 출력을 해 주었다.

1. 추가 기능 – 총구 움직임

총을 쏘았을 때 총알이 나가는 방향에 따라 팔의 각도를 해당 방향으로 설정해 주었다. 또한, 시간 count를 통해 일정 시간이(약 0.6초) 지나거나 방향키 입력이 들어올 경우 다시 원래의 자세로 돌아오도록 하였다.

1. 추가 기능 – Life

User class에 life 멤버 변수를 추가하여 패배 조건에 life 감소 관련 코드를 넣어 주었다. 또한, timer 함수와 연동하여 life가 감소된 상황에선 약 1초 동안 User에게 무적 판정을 주어 Enemy와 충돌했을 때 life가 연속적으로 감소하여 한번에 패배하는 것을 방지하였다.

1. 추가 기능 – 재시작

reinit 함수를 만들어 키보드 입력으로 r또는 R이 들어왔을 때 실행해주도록 하였다. reinit 함수는 Enemy 목록을 모두 삭제한 뒤 map 파일을 읽어와 User, Enemy, Item 등 시작 정보를 모두 새로 초기화한다. 또한, (5)에서 언급될 시간 타이머를 리셋하여 시간 제한도 초기화해준다. 따라서 패배, 승리, 게임 도중 언제나 r버튼 또는 R버튼을 누르면 게임을 다시 시작할 수 있다.

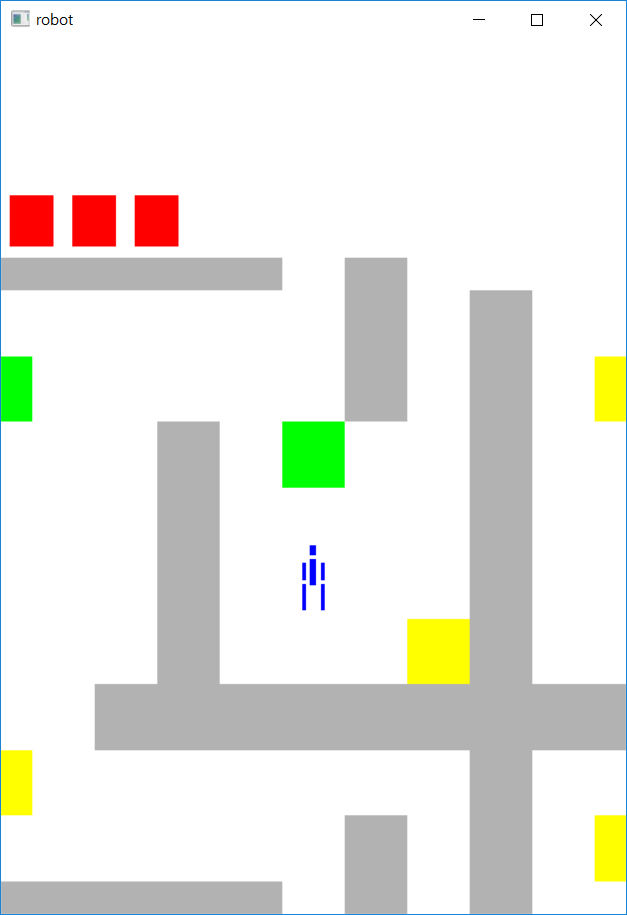
1. 추가 기능 – 시간제한

게임을 시작할 때 init 또는 reinit을 통해 time\_limit을 초기화해주고 timer가 호출될 때마다 time\_limit을 감소시켜 0이 되면 패배하도록 구현하여 일정 시간(100초)이 지나면 패배하게 된다.

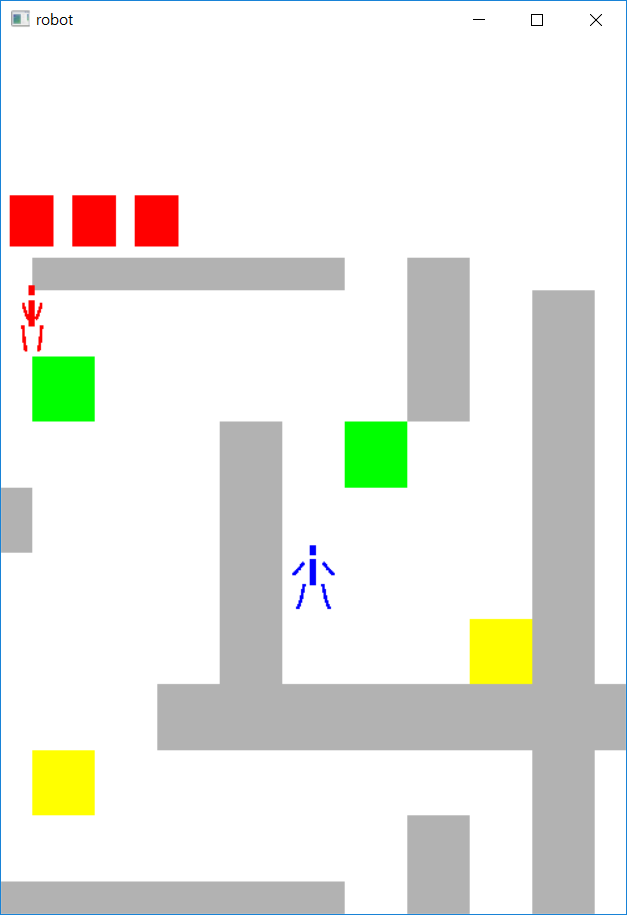
1. **프로그램 실행 방법**

먼저 실행 폴더에 game\_map\_easy.csv 파일을 넣어주어야 해당 파일에서 맵 정보를 받아와 게임을 시작할 수 있다. 이 파일은 벽, 유저, 적, 아이템의 위치를 저장하며 이를 수정할 경우 다른 형태의 맵에서도 플레이 가능하다. 프로그램을 실행할 경우 입력할 수 있는 input은 상, 하, 좌, 우, 스페이스 바, r 버튼이 있는데 각각의 방향키로는 유저를 이동시킬 수 있으며 스페이스 바를 입력할 경우 유저가 마지막으로 이동한 방향 기준으로 총알이 나가게 된다. 그리고 r 버튼을 누를 경우 게임을 재시작하게 된다.

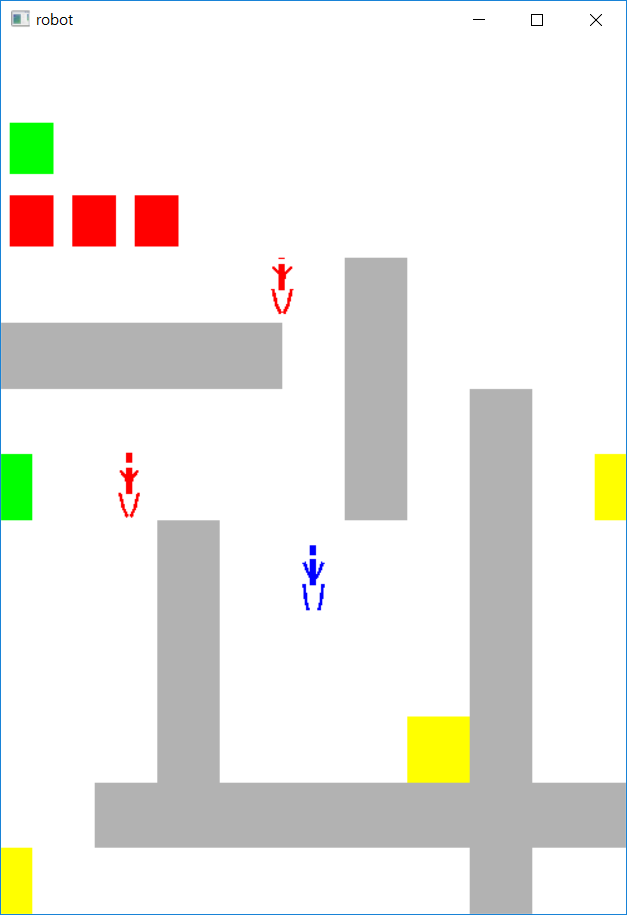
1. **예제**



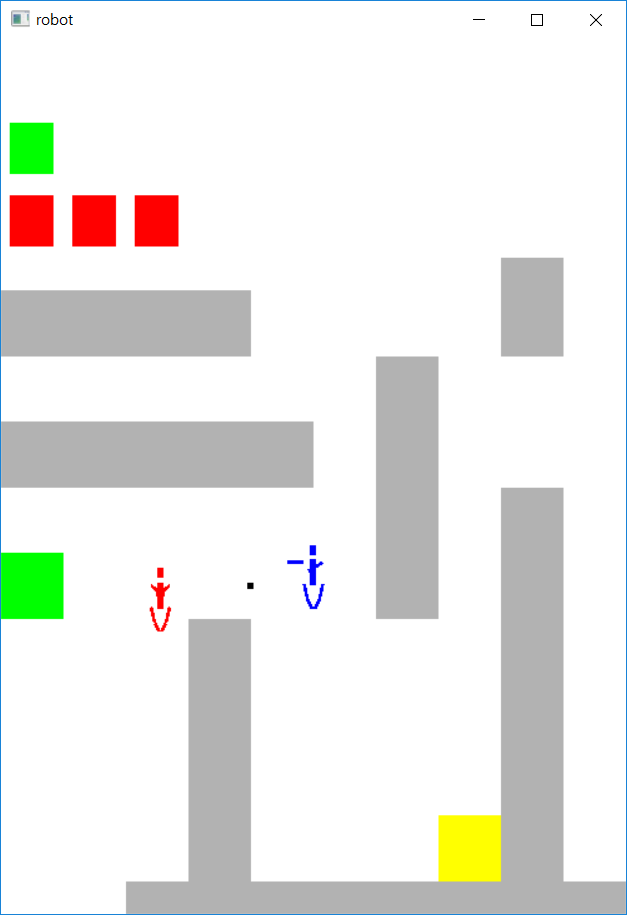
먼저 게임을 실행하면 위 그림과 같은 위치에서 시작하게 된다. 초록색, 노란색 사각형은 각각 아이템을 나타내고, 왼쪽 상단의 빨간색 박스는 remaining life이다.



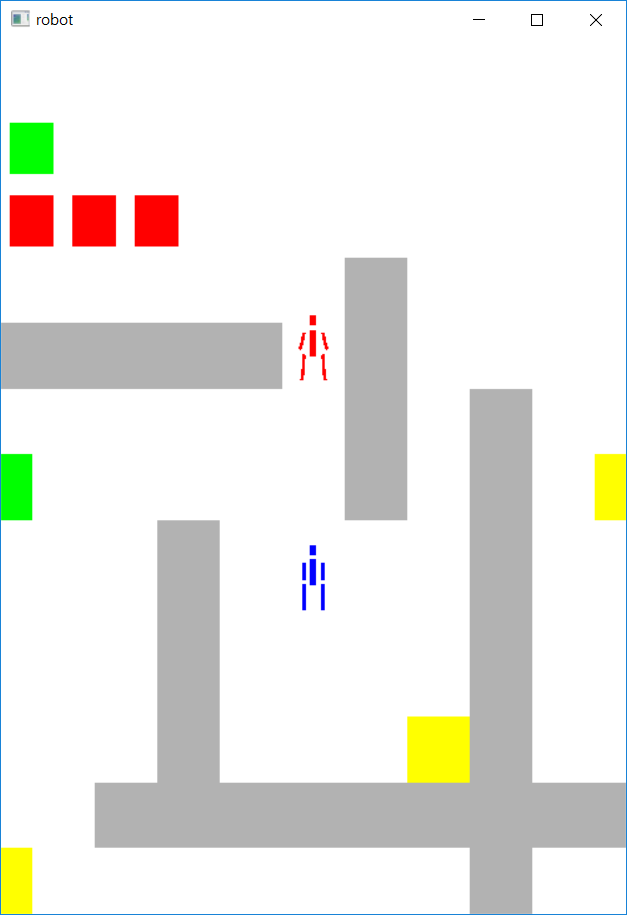
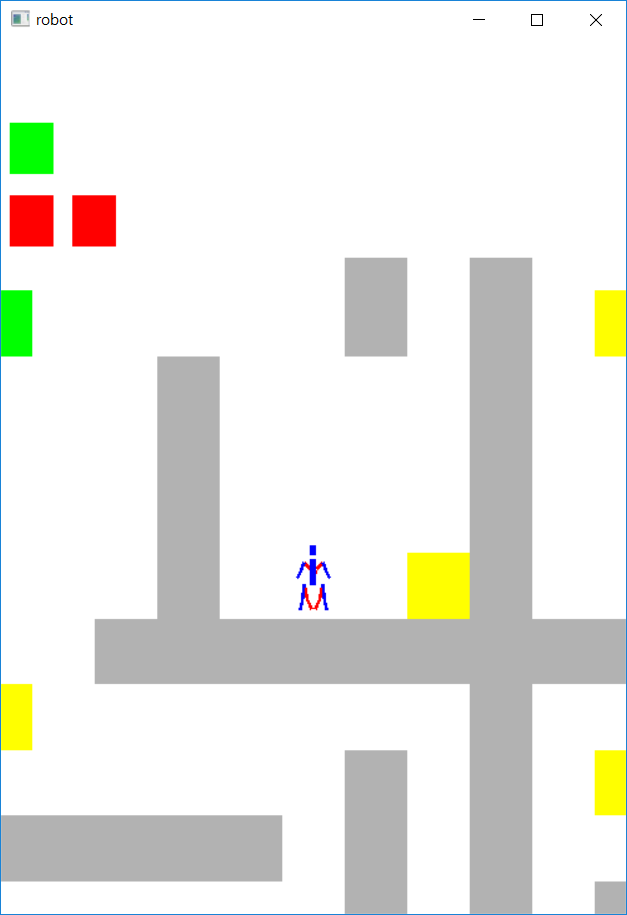
그런 다음 약간 움직여보면 유저의 팔다리가 2계층으로 잘 움직이는 것을 확인할 수 있다.



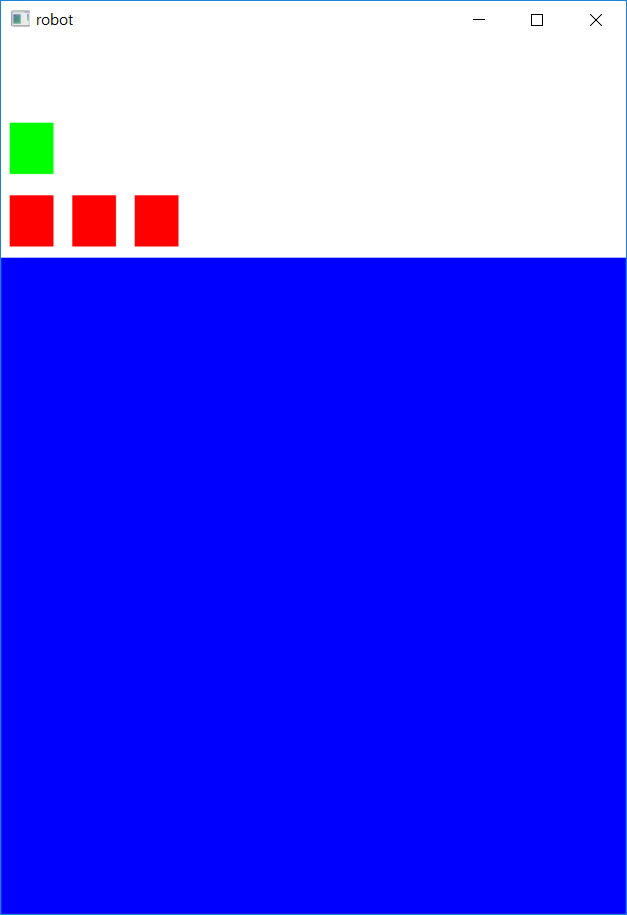
Assignment #1과 동일하게 움직여서 아이템을 먹으면 왼쪽 상단에 아이템이 출력되며 Enemy도 시간에 따라 스스로 잘 움직인다.



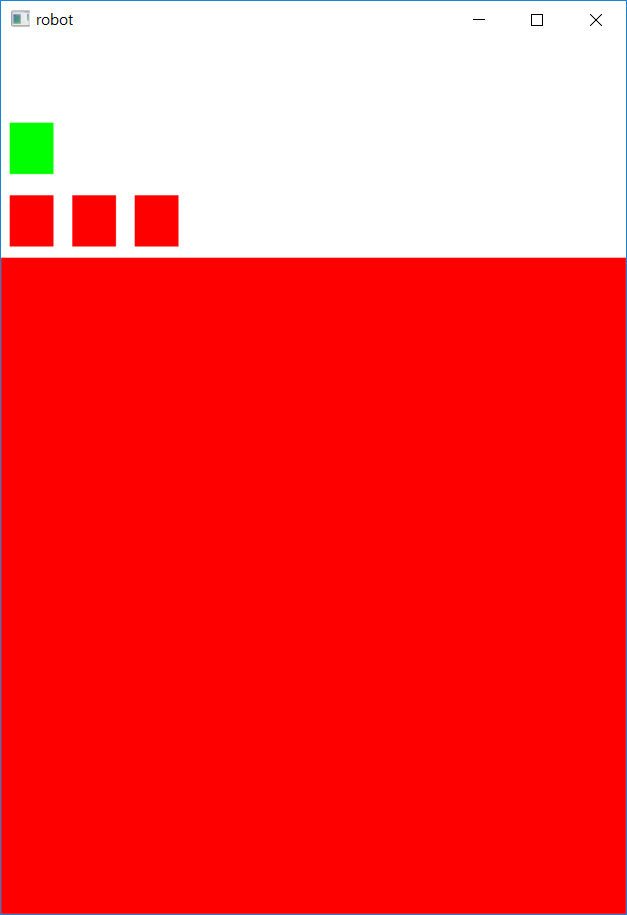
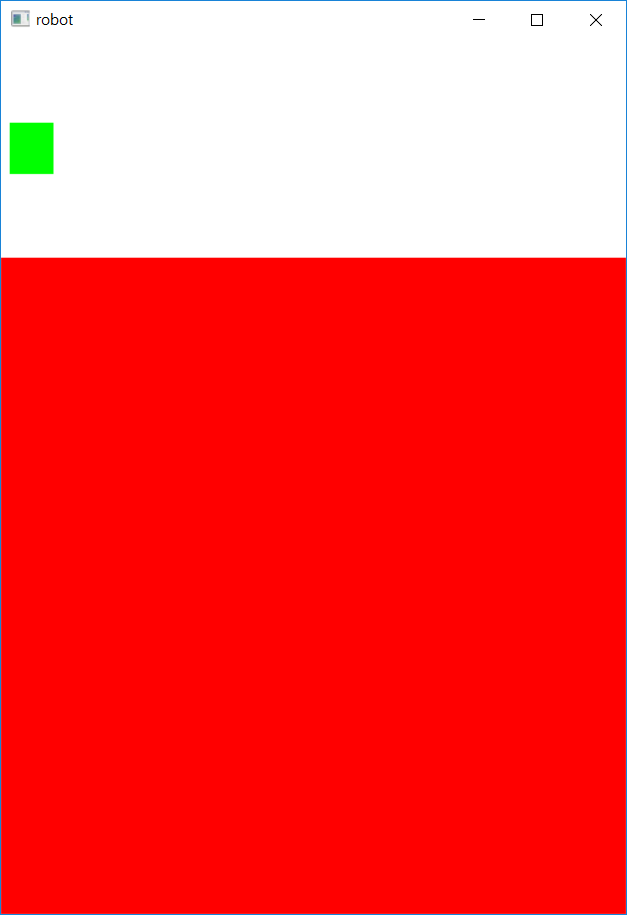
총알을 발사할 때는 해당방향으로 팔을 뻗어 총구를 겨눈다.

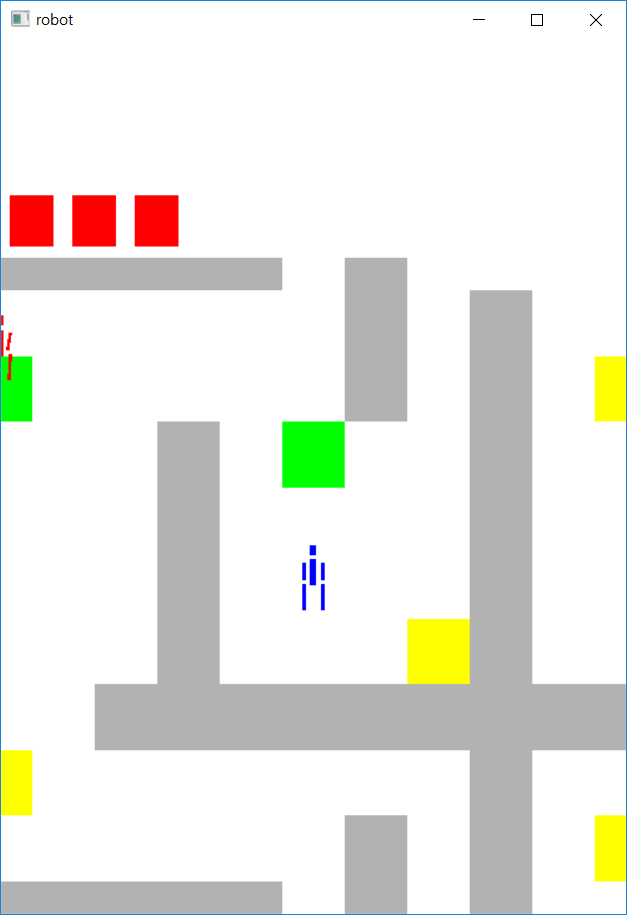
Enemy에게 잡히면 체력이 감소하게 된다.



모든 적을 죽인 경우 위와 같이 파란색 승리 화면이 출력된다.

시간 제한이 지난 경우 왼쪽과 같이 패배 화면이 출력되며 life가 모두 소모된 경우에도 마찬가지로 오른쪽과 같이 패배 화면이 출력된다.



언제든지 r 버튼을 누르면 위와 같이 게임이 다시 시작 된다.

1. **토론**

강의시간에만 배웠던 Vertex Shader, Fragment Shader의 기능을 직접 구현하는 것은 훨씬 더 복잡했다. 특히 Application의 소스와 Shader들과 연결되는 방식을 정확히 이해하기 위해 많은 시간이 소모되었다. 본 과제를 하기 이전에는 Shader 자체에서 많은 기능을 구현해야 한다고 생각했지만 대부분의 shader는 다음과 같은 형태로 이루어진 것을 확인했다.

위 Vertex Shader는 Application에서 보내는 정점(vPosition), 색(vColor) 정보와

ModelView\_matrix, Projection\_matrix를 정점에 곱해 정점의 최종 결과값을

fragment Shader에게 보낸다.

Fragment Shader는 Vertex Shader에게 받은 색 정보를 다시 저장해 그대로 보낸다.

본 과제에서는 그림자나 텍스쳐 매핑 같은 추가 기술의 구현이 없었으므로 Fragment

Shader가 가장 간단한 형태를 가지게 된다.

코드를 직접 작성하면서 Shader에 대한 이해를 크게 높일 수 있었다.

그리고 VAO(Vertex Array Object)를 object마다 다르게 설정해주고 필요할 때마다 재사용 가능한 것을 직접 확인 하면서 VAO의 유용함을 알았다.

마지막으로 계층구조를 표현하기 위해 효과적인 방법 중 하나로 “Left Child Right Sibling” 방식의 링크드 리스트 구조를 이용했다. 이후에는 preorder traverse 방식을 사용하여 전체 계층 구조를 그릴 수 있었다. 계층구조 표현에 상당히 유용한 방법임을 확인 할 수 있었다.

1. **결론**

이번 Assignment를 통해 shader 프로그래밍과 VAO, VBO의 사용법, 그리고 계층 구조에 대해 잘 이해하게 되었다.

1. **개선 방향**

플레이어의 이동이 부드럽지 않은 점이나 적의 인공지능 부분은 여전히 개선되지 않았으며 플레이어의 움직임이 실제 사람이 걷는 것과는 조금 차이가 있다.

1. **참고 문헌**
2. 그래픽스 Lecture Note
3. Interactive Computer Graphics 6th Edition
4. <https://www.cs.unm.edu/~angel/BOOK/INTERACTIVE_COMPUTER_GRAPHICS/SIXTH_EDITION/CODE/>