
2017180006 김지영

2018-12-08 보고서

총 6천 코드의 눈물과 땀방울의 결과물


## 2017180006 김지영

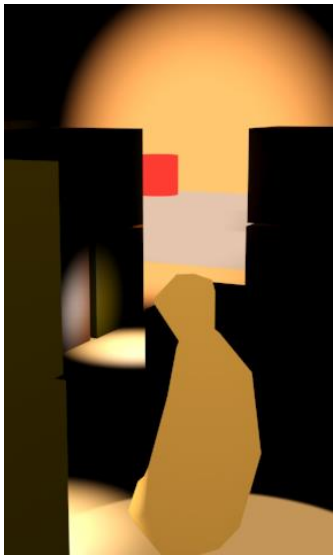
기획 / 그래픽 / 카메라, Scene 프로그래머

## 2017182007 김우빈

메인 프로그래머

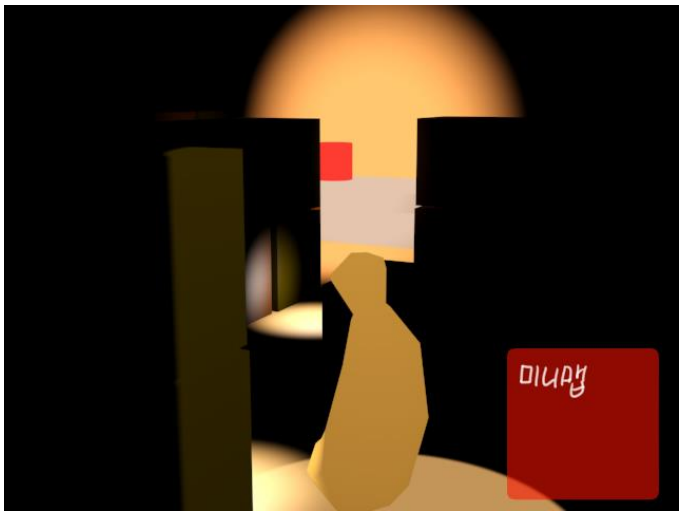
## 1 게임 소개

게임 컨셉



덜 익은 바나나가 완전히 익기 위해서 **붉은 조명 기둥**을 찾아가는 이야기

게임 화면

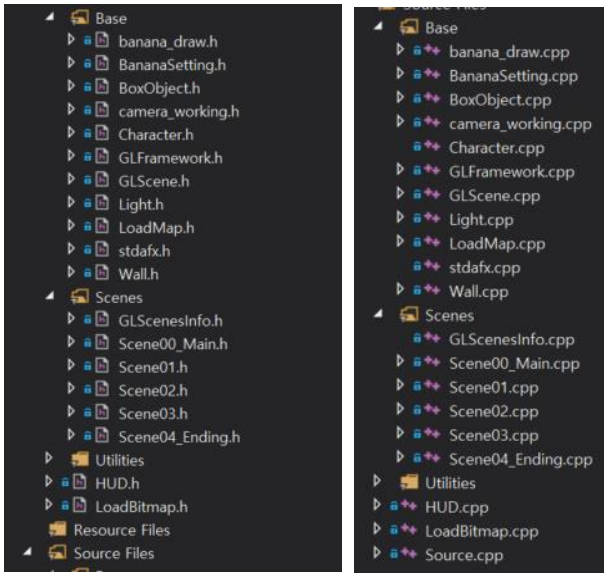



*는 추가 구현(필수 구현이 아님)입니다.

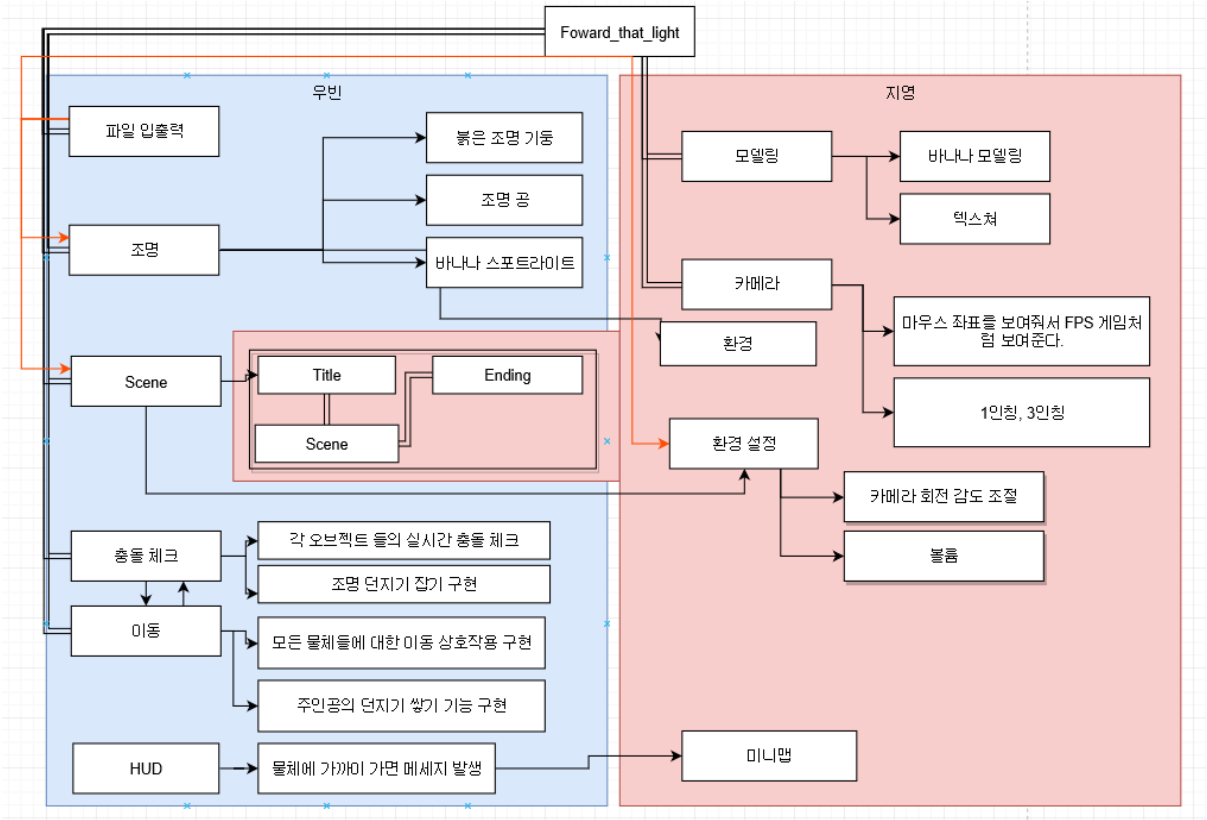
**게임 내 모든 단위: 그리드 1칸 == 5 픽셀 == 10 cm**

## 2 구조 소개

A Git Hub를 이용한 저장소 동기화




B 헤더와 cpp로 구성된 프레임 워크 사용 각각의 기능별 분리

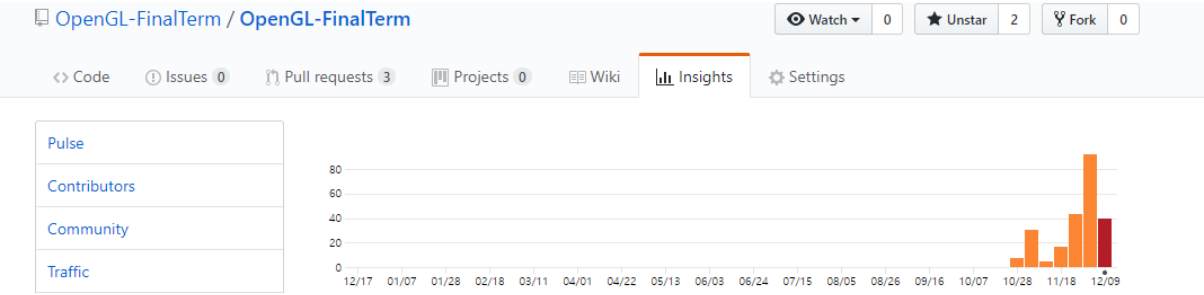



프레임워크 연결 관계도 일부를 나타낸 모습

## 3 팀원간 작업한 내용

| 오브젝트 | 기능 | 설명 | 구현한 사람 |
|---|---|---|---|
| 주인공 | 이동 | x, z방향으로 이동이 가능하다. | 김우빈 |
| | 점프 | y방향으로 키의 2배 점프가 가능하다. | |
| | 하강 | 발판이 없을 시에 -y로 하강이 가능하다.<br>키의 3배 이상 높이에서 떨어지면 죽는다. | |
| | 밀기 | 움직이는 방향으로 물체를 밀어낸다 | |
| | 던지기 | 물체를 집어서 던질 수 있다. | |
| | 쌓기 | 오브젝트를 +y 방향으로 올려서 쌓을 수 있다. | |
| 상자 | 이동 | 타 오브젝트가 힘을 주면 그 방향으로 밀리게 된다. | |
| | 충돌체크 | 앞에 있는 상자가 밀리면 뒤에 있는 상자도 같이 밀린다 | |
| 조명 | 파란 공 | 파란 빛을 비추는 공<br>크기의 10배만큼 주변을 비출 수 있다. | |
| | 빨간 공 | 붉은 빛을 비추는 공<br>파란 공보다 더 먼 곳(0.5배)까지 비출 수 있다. | |
| | 붉은 조명 기둥 | 스테이지 클리어 위치를 알려주는 조명<br>20배 주변을 비출 수 있다. | |
| | 환경 | 시간이 지남에 따라 주변이 모두 밝아진다. | |
| 스테이지 | 선택 | 총 3개의 스테이지를 선택할 수 있다. | 김지영 |
| | 이동 | 스테이지 간 이동이 가능하다 | |
| | 클리어 | 다음 스테이지를 선택할 수 있다. | |
| 환경 설정 | 스크롤 | 환경설정을 스크롤로 할 수 있다.(직접 구현) | |
| 사운드 | BGM | 게임 내 BGM | 김우빈 |
| | 이펙트 | 바나나 걸어 다니는 효과음 | |
| 파일 입출력 | 맵 불러오기 | 파일 입출력을 통한 맵 부르기 | |
| 안개효과 | | 안개를 넣어서 멀리 있는 물체는 안보이게 했다. | |
| State | | 여러가지 State | |
| 카메라 | 1인칭 | 바나나 시점으로 게임이 전개된다. | 김지영 |
| | 3인칭 | 바나나의 뒤에서 화면을 보여준다. | |
| | *클리어 | 스테이지를 클리어 하면 바나나를 회전해서 올라간다. | |
| 미니 맵 | | 뒷부분이 반투명한 150 X 150 픽셀 사이즈 정사각형이다.<br>각 오브젝트 들의 x, z의 정보를 읽어서 위치를 표시해준다. | |
| 모델링 | 바나나 | 바나나 숨쉬기, 뛰기, 제자리에 있기 모션 | |

# 4 프로젝트 진행 사항




Git hub commit 통계를 보면 하루도 빠짐없이 프로젝트 준비를 했습니다.

프로젝트의 진행방향은 다음과 같습니다.

1 3ds 모델링 후 open GL 에 정점을 입력해서 넘기기(김 지영)

2 박스간 충돌체크 구현 연습(김 우빈)

3 맵 레벨디자인을 해서 프로그래머에게 넘기기(김 지영)

4 전체 기능을 담은 프레임워크 틀 짜기(김 우빈)

5 (코딩 시작)

(김 지영) 카메라 자유시점 1 인칭 FPS GAME 같은 부드러운 카메라 워킹 및 엔딩 오프닝 카메라 워킹 구현

(김 우빈) 자유시점에 따른 360 도 방향 충돌체크 구현, 오브젝트끼리 밀고 던지기, 쌓기 구현,

파일 입출력을 통한 맵 저장, 그 외 주요 기능 구현

6 (각 주요기능 테스트 및 마무리)

(김 지영)

각각의 State 분리하기 시작 화면 / 메인 화면 / 게임 맵 / 환경 설정

(김 우빈)

충돌체크 점검 및 효과음 BGM 삽입 가까이 가면 클리어가 되게끔 변경

7 보고서 작성 및 마무리 점검

## 4 작업 내용

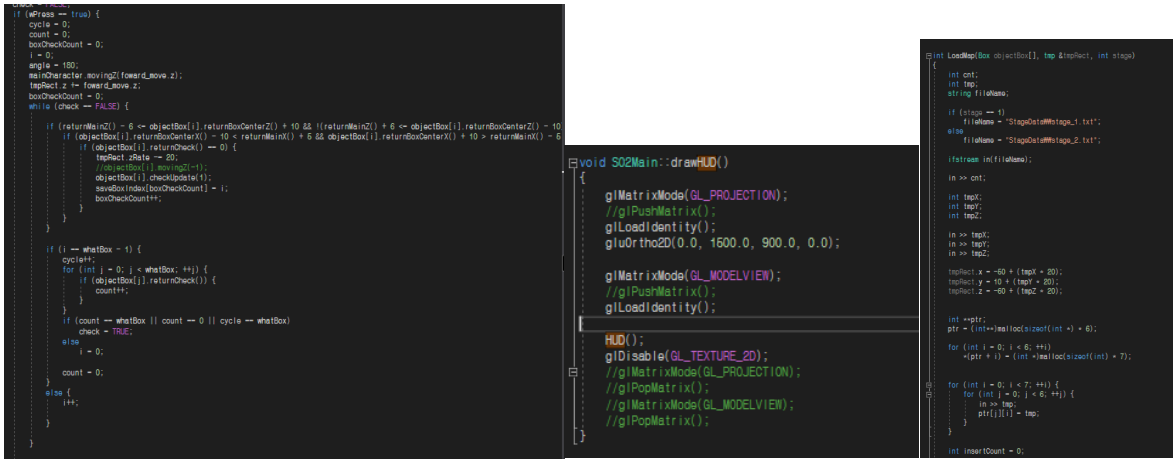
```
void opening_camera_Eye(float* start_x, float* start_y, float* start_z, float* end_x, float* end_y,

    *out_x = (1 - *t) * (*start_x + 1) + *t * *end_x;
    *out_z = (1 - *t) * (*start_z + 1 ) + *t * *end_z;

    //*out_y =
    //  ((*start_y * pow((1 - *t), 3)) + //실린더 위치
    //  (3 * ((contrl_pt + *start_y)/2 ) * *t * pow((1 - *t), 2)) + //제어점 1
    //      (3 * ((contrl_pt + *end_y) / 2) * pow(*t, 2) * (1 - *t)) + //제어점 2
    //      (*end_y * pow(*t, 3)) //도착지점
    //      );

    *out_y =
        ((-pow(*t, 3) + (2 * pow(*t, 2)) - *t)* *start_y +
        ((3 * pow(*t, 3)) - (5 * pow(*t, 2)) + 2) * ((contrl_pt + *start_y) / 2) +
        ((-3 * pow(*t, 3)) + (4 * pow(*t, 2)) + *t)* ((contrl_pt + *end_y) / 2) +
        (pow(*t, 3) - pow(*t, 2)) * *end_y
        ) / 2;
}
```

```
if (pressed == false) {
    //  printf("old x : %f, old y : %f // new x : %f, new y : %f\n", drag_old_postion[0], dra

    difference_new_old[0] = drag_old_postion[0] - drag_new_postion[0];
    difference_new_old[1] = drag_old_postion[1] - drag_new_postion[1];

    //difference_new_old[0] = 750;
    //difference_new_old[1] = 450;

    //difference_new_old nomalized
    //step 1 diffrence vetor size compute
    difference_size = sqrt(pow(difference_new_old[0], 2) + pow(difference_new_old[1], 2));

    //step2 re compute diffrence pos / vector size
    if (difference_size == 0) {
        difference_size = 0.0001;
    }
    difference_normal_pos[0] = difference_new_old[0] / difference_size;
    difference_normal_pos[1] = difference_new_old[1] / difference_size;

    //step3 nomal add to radian range 360

    //if(result_degree[1] < 180)
    result_degree[1] += difference_normal_pos[1];

    if ((result_degree[0] < 360) && (result_degree[0] >= 0)) {
        result_degree[0] += int(difference_normal_pos[0] * 3);
    }
    else if (result_degree[0] >= 360)
        result_degree[0] = 0;
    else
        result_degree[0] = 359;

    if ((result_degree[1] > 360) || (result_degree[1] < -360)) {
        result_degree[1] = 0;
```

### 김지영

원래는 4 방향 시점 변환이었지만 바나나의 움직임을 온전히 보여주질 못해서 '배틀그라운드'나 '오버워치'처럼 FPS 게임에 자주 사용되는 자유 시점 3 인칭 + 1 인칭 카메라를 만들어 보았습니다. 이를 위해 벡터의 정규화 과정도 찾아보고 컴퓨터 그래픽스 이론도 공부했습니다. Eye 와 달리 At 에 -값을 주어야 타 FPS 게임처럼 캐릭터의 등을 계속해서 볼 수 있다는 점도 새롭게 알게 되었습니다.

또 이번 기말 텀프에서 지난번 과 달리 유독 곡선을 많이 사용했습니다. 그 이유는 카메라 이동을 할 때 부드러운 움직임을 표현하고 싶었기 때문입니다. 또 메인 화면의 맵 선택 화면에서는 기존에는 X, Z 값을 이용하여 XZ 평면으로만 맵을 나타낼까 하다가 입체적으로 보는게 훨씬 좋은 방법이라 생각해서 파일 입출력을 통해 맵이 저장된 txt 를 읽어서 작은 축소판으로 구현했습니다. 또 카메라 이동을 하면서 포인터에 대해 다시 배우게 되었습니다. 이번 텀프를 진행하면서 프로그램의 구조와 문법에 대해서 다시한번 생각하게 되었습니다.



## 김우빈

처음에는 4 방향 혹은 8 방향 시점 변환으로 알고 충돌체크를 구현해 두었지만, 나중에 자유시점으로 변경되면서 모든 충돌체크를 자유시점은 기반으로 하게 바꾸게 되었습니다. 특히 충돌 체크 코드를 짜면서 불필요한 계산을 최대한 줄이기 위해 체크하지 않는 방향은 아예 연산에서 제외시켰습니다. 그리고 프레임 워크를 이번 텀 프로젝트에서 처음 사용하면서 객체지향 코딩을 하기 위해 노력하게 되었고 프레임워크가 어떻게 돌아가는지를 알게 되었습니다.

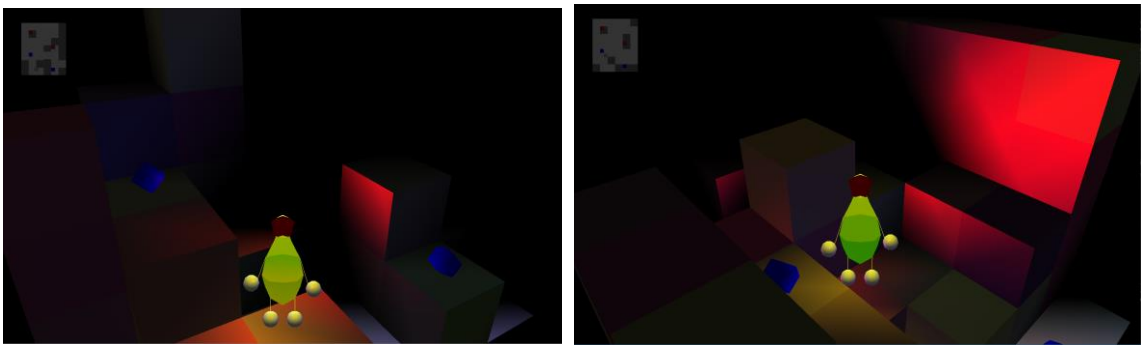
그리고 HUD 라는 시스템을 사용 하였습니다., 3D 화면에서 UI 를 띄우기 위해 2D 화면으로 일시적으로 변환 한 후에 다시 돌아오는 방법을 사용하였는데, 2D 화면으로 변환 후에 다시 3D 화면으로 돌려 주지 않아 버그가 자주 발생 하였지만 어려운 문제는 아니었습니다. 다만 이를 이용하여 미니 맵 말고 다른 UI 를 띄우지 못한 점이 아쉽습니다.

추가적으로 혹시 모를 Map tool 시스템을 대비하여 모든 상자, 조명 값 전부 파일 입출력을 통해 받아왔는데, 메모리의 낭비를 막기 위해 대부분 동적할당을 이용하여 불필요한 공간 낭비를 막았습니다.

또 이번 기말 텀 프로젝트에선 2D 게임 프로그래밍 시간에 배운 프레임워크를 처음으로 수정, 제작, 사용을 하게 되었는데 이전에는 프레임 상관없이 캐릭터의 이동 값이 고정되어 있었던 것에 비해 이번 프로젝트 에서는 시간에 따른 이동이 결정되어 있기에 이전 학기 윈도우 프로그래밍 시간 프로젝트와는 다른 양상을 보였다고 생각합니다. 특히 이전 까지는 하드 코딩이 주를 이뤘지만, 이번 프로젝트 에서는 부분적으로 코드를 짤 수 있었고, 그로 인해 무식한 코딩 즉 하드 코딩의 빈도가 줄어 들게 되었습니다. 이번 텀 프로젝트를 진행 하면서 변수이름, 함수이름, 객체지향을 공부해 보는 시간으로 이용하려고 노력하였습니다. 비록 완벽하게 객체지향적으로 구현된 것도 아니지만, 추후에 코딩을 할 때 좋은 경험이 될 수 있는 밑 거름이 되었다고 생각합니다. 그리고 프로그램의 구조가 엄청 큰 정도가 아닌 거의 대다수에 영향을 미친다는 것을 다시 한번 뼈저리게 알게 되었습니다.

## 5 결과물 분석

**[결과물 사진]**



**[잘한 점]**

카메라 자유시점 구현에 성공한 점이 잘 되었다고 생각합니다. 또한 최적화를 위해 누른 방향으로만 충돌체크를 한다는 발상도 좋았습니다. 안개를 이용하여 시야에 제한을 둔다는 연출도 좋았던 것 같습니다.

**[아쉬운 점]**

그림자를 못 넣은 것이 아쉽습니다. 그림자 대신에 안개를 사용했습니다만 나중에 과제전에서는 그림자를 넣어 더 생동감 넘치는 게임으로 만들고 싶습니다.

색감이 예상과 다르게 나와서 당황했습니다. 또 파일 입출력을 통해 맵을 로드 하는데 더 나아가 맵 툴까지 발전하지 못한 점이 아쉽습니다.

## 6 필요한 명령어 소개

| | | |
|---|---|---|
| **캐릭터** | **이동** | **마우스 움직임으로 회전각도 구함**<br>**바라보고 있는 시점 쪽으로 움직임** |
| | 점프 | Space |
| | 잡기 | F<br>잡은 물체는 마우스 왼쪽 클릭을 통해 놓을 수 있음 |
| | 쌓기 | Q 상자 집기 |
| **Active** | | 좌 클릭<br>바라보고 있는 정면을 향해서 잡은 물건을 던지거나 쌓기를 한다. |
| **카메라** | 시점 변환 | U<br>(1 인칭 3 인칭 간에 시점 변환) |
| | 카메라 회전 | 마우스 움직임<br>카메라가 360 방향으로 회전한다. |
| **State 전환** | R | Game Over 나 Clear 시 state 전환을 위해 누르는 버튼입니다. |

자유시점 카메라로 구현했기 때문에 불필요한 키보드 명령어는 과감하게 정리했습니다.

## 5 프로젝트 개발 소감 및 후기

## 김 지영

"곡선 마스터와 모델링 직접 그리기 그리고 카메라 너무 쉬운데...."

```
// 번호 / (몇 줄)행 / 라인 갯수

index.pos[0].x = 0.0f;
index.pos[0].y = -2.761;
index.pos[0].z = 17.274;

index.pos[1].x = 0.0f;
index.pos[1].y = -4.338;
index.pos[1].z = 13.761;

index.pos[2].x = -3.034;
index.pos[2].y = -2.825;
index.pos[2].z = 14.274;

index.pos[3].x = -4.181;
index.pos[3].y = -0.4f;
index.pos[3].z = 15.601;

index.pos[4].x = -2.728;
index.pos[4].y = 1.37f;
index.pos[4].z = 17.539;

index.pos[5].x = 0.0f;
index.pos[5].y = 1.842;
index.pos[5].z = 18.588f;

//머리 2
index.pos[1 + 1 + 5].x = 0.0f;
index.pos[1 + 1 + 5].y = 1.258;
index.pos[1 + 1 + 5].z = 9.247;

index.pos[2 + 1 + 5].x = -1.233;
index.pos[2 + 1 + 5].y = 1.589;
index.pos[2 + 1 + 5].z = 9.635;

index.pos[3 + 1 + 5].x = -2.411;
index.pos[3 + 1 + 5].y = 2.686;
index.pos[3 + 1 + 5].z = 10.71;
```

```
for (int j = 0; j < 1; j++) {

    glColor3f(j + 0.5 + 0.5, j + 0.2, 0);

    for (int i = 4; i > 0; i--) {


        glBegin(GL_POLYGON); {//아래

            glVertex3f(temp.pos[i + (j + 5)].x, temp.pos[i + (j + 5)].y, temp.pos[i + (j + 5)].z);//중심

            glVertex3f(temp.pos[(i + 1) + (j + 5)].x, temp.pos[(i + 1) + (j + 5)].y, temp.pos[(i + 1) + (j + 5)].z);//그외

            glVertex3f(temp.pos[(i + 1) + ((j + 1) + 5)].x, temp.pos[(i + 1) + ((j + 1) + 5)].y, temp.pos[(i + 1) + ((j + 1

            glVertex3f(temp.pos[(i)+((j + 1) + 5)].x, temp.pos[(i)+((j + 1) + 5)].y, temp.pos[(i)+((j + 1) + 5)].z);//그외

        }
        glEnd();
    }
}
```

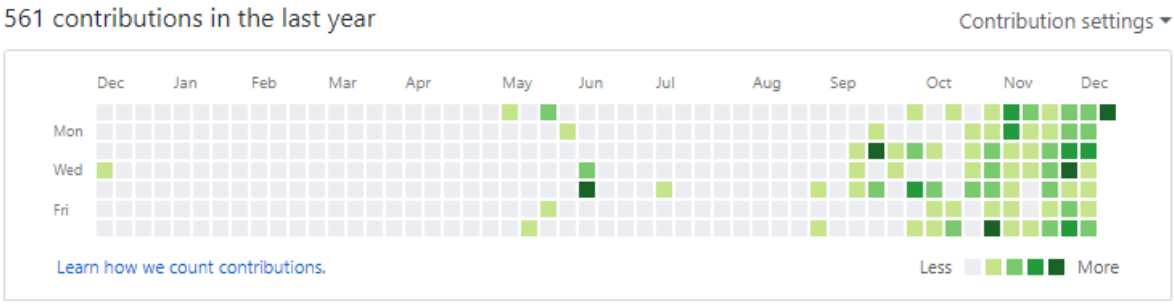
바나나의 귀여움을 설명하는 게임을 만들어야 했기 때문에 모델링에서부터 많은 준비과정이 있었던 것 같습니다. 우선 살아있는 바나나를 만들기 위해 모델링의 각 정점을 움직여야 했기 때문에 과감하게 OBJ, FBX 파일 같은 친구들을 버리고 일일이 손으로 50 개의 float(소수점 숫자 3 자리 까지) 직-접 입력했습니다. 신선한 경험이었지만 이걸 본 팀원은 아직도 이해를 못하겠다고 합니다. 아무래도 제 기준에서 적은 코딩보다는 남도 알아볼 수 있는 코딩을 하는게 중요하다는 점을 알게 되었습니다.

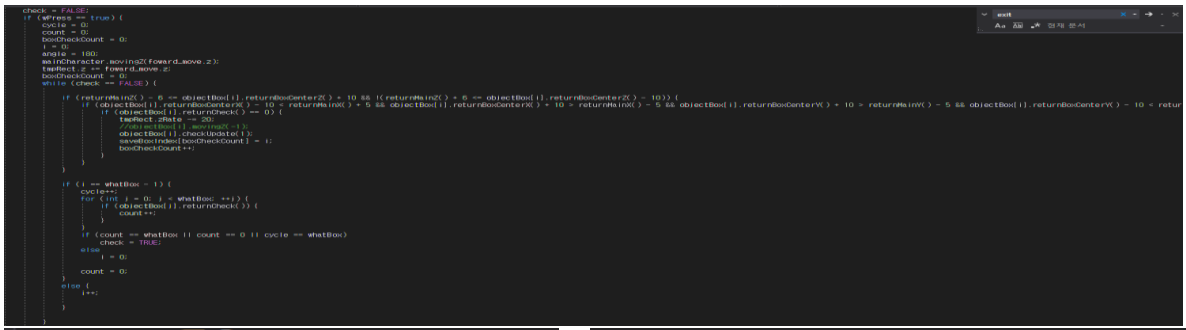
2 디 게임과 3D 게임의 구조는 비슷하면서도 다릅니다. 2D 게임에서 하지못하는 구조나 레벨 디자인을 3D 에서는 더 많이 구현할 수 있었습니다.

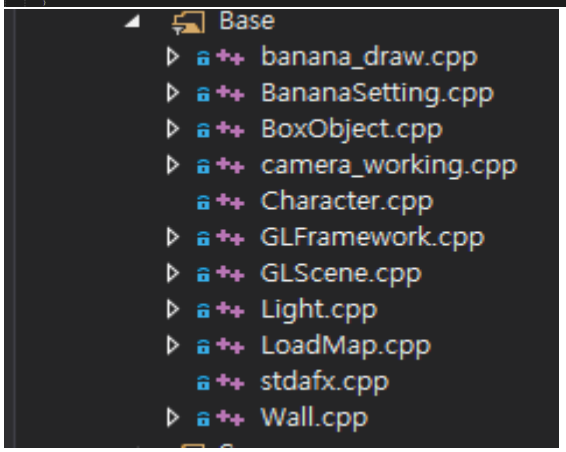
그리고 프로젝트 작업 도중에 개인 사정이 생겨 일주일 정도의 공백이 생겼었는데 계속 작업하면서 기다려준 팀원인 우빈이에게 고마움을 전합니다.

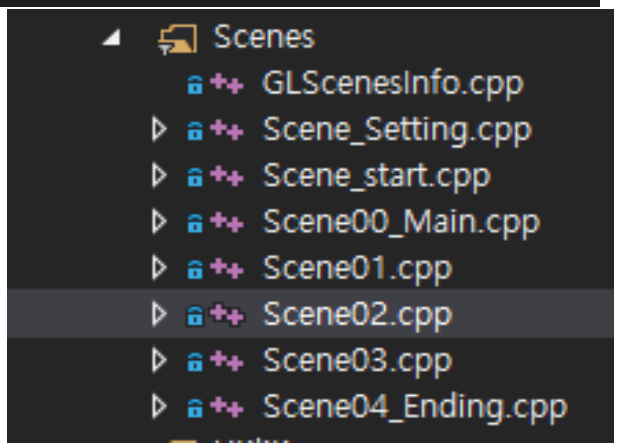
1 년동안 고생해주신 송 인희교수님께도 많은 감사를 드립니다.

## 김 우빈






개발을 하면서 이번에는 최대한 분리를 하면서 이해하기 쉬운 코드를 짜보자 라는 생각으로 컴그 기말 프로젝트를 하게 되었습니다. 그리고 이전 윈도우 프로그래밍 기말 프로젝트와는 달리 이번 프로젝트에서는 프레임워크를 적극적으로 사용하여 프로젝트를 진행하게 되었습니다. 2D 게임 프로그래밍 시간에 배운 프레임 워크를 기반으로 이번 기말 프로젝트도 유사하게 구현하였습니다. 특히 이번 기말 프로젝트에서 어려운 점은 개인적으로는 충돌 체크라고 생각을 합니다. 원래는 4,8 방향을 기반으로 충돌체크를 구현해 두었지만, 최종적으로는 마우스 회전을 이용한 자유시점으로 변경되면서 4,8 방향 충돌체크를 넘어서게 되었습니다. 특히나 그 외 조명이 날라가는 곳, 박스가 떨어지는 위치 등등 모든 오브젝트들이 자유시점으로 바뀌자 마자 경우의 수를 계산하기 어려움이 있었습니다. 그러나 게임수학 시간에 배운 내용을 활용하여 최대한 간결하게 코드를 짜기 위해 노력하였고, HUD 기능을 이용하기 위해 3D 화면과 2D 화면의 자유로운 전환을 구현 해낸 것이 만족스러웠습니다. OBJ 같은 모델링 데이터를 불러오지는 않았지만, 수업시간에 배운 내용을 모두 활용하여 최고의 결과를 만들어 냈다고 생각합니다.