Computer Graphics Homework 2

<PackMan>

학번: 2015147506
전공: 컴퓨터 과학과
이름: 김기현
제출일: 6월 25일

목차

I. 구현한 표

기능	구현 여부	구현 위치	비고
Vertex Shader	O	main.cpp(line 129, 133, 134, 138, 156) modelLoading.vs/fs ground.vs/fs background.vs/fs text.vs/fs global.vs/fs	129: modelloading 133: ground 134: background 138: text 156: global
Fragment Shader	O		
Transformation	O	main.cpp(line 615~, 466)	615:object transform 466:object transform
Viewing	O	main.cpp(line 287)	287:render setview
Texture	O	main.cpp(line 135, 243)	135:background 243: loadTexture
Lighting	O	main.cpp (line 160) global.vs/fs	160: allocate light
Model Loading	O	main.cpp(line 130, 471) modelLoading.vs/fs	130: model call 471: model draw
Text	O	main.cpp(line 138, 425) text.vs/fs	138: call textstyle 425: rendertext
Animation (Keyframe)	O	main.cpp(line.441, 704, 749)	441:render part 704:initKeyframes 749:updateAnimData
PlaySound	O	main.cpp(line.590)	590: playsound

PlaySound 추가 구현

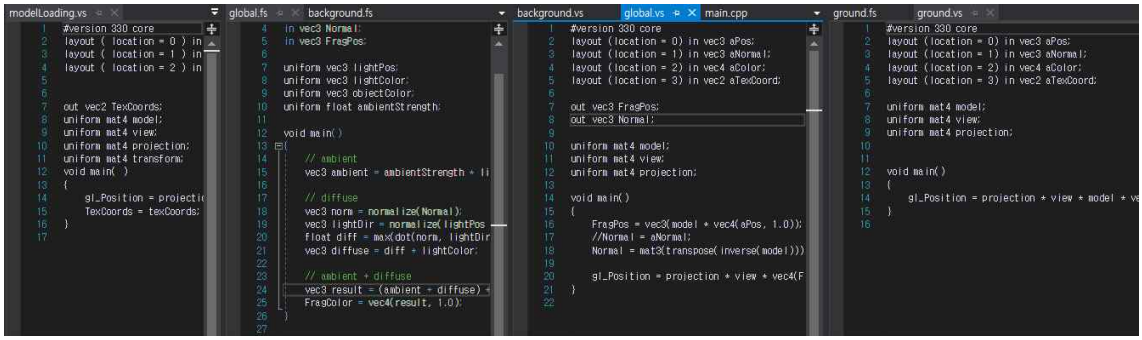
II. 구현한 기능들에 대한 상세 설명

0) 기본적인 게임의 룰

팩맨 게임은 1980년대 2D 감성을 3D에 담아 제작한 게임입니다. 위에서 유령 혹은 사과가 내려오며, 각각이 random하게 내려와 플레이어를 혼동시킵니다. 팩맨에서 구현되는 소리를 모두 구현하였으며, 요구하는 과일의 수를 모두 모은다면 스테이지 클리어 음악과 함께 게임에서 승리를 하며 시간 기록이 되고, 유령과 부딪히게 된다면 죽음을 뜻하는 소리와 함께 게임 오버를 하게 되는 간단한 게임입니다. 좌우 버튼을 통해 팩맨을 움직일 수 있습니다.

1,2) Vertex Shader, Fragment Shader

Vertex Shader과 Fragment Shader은 각각의 Object특성에 맞춰서 modelLoading, ground. background, text, global 총 5가지 shader을 추가하였습니다. 추후에 object별로 다르게 변경을 할 수 있었기 때문에, global, background, ground를 추가적으로 생성하였으며, text shader은 text를 구현하기 위해서, ModelLoading을 위한 shader음 ModelLoading을 위해서 모두 주어진 shader을 그대로 이용하였습니다. 각 shader별로 texcoord, transformation, color등을 주어져서 각 model에게 적용을 하였습니다. 한가지 차이점은, 1980감성을 살리기 위하여 각 object에게 lighting을 모두 준 것이 아니라, global shader에만 lighting을 적용해 주었습니다.



3) Transformation

Transformation은 이번 과제에 정말 다양한 파트에서 적용을 하였습니다. 모든 Object들을 create()함수를 통해 생성 해줄때마다 translate, rotate, scale들을 조절하면서 각 object들을 적합한 사이즈로 만들었을 뿐만 아니라, plane같은 경우, 어떨 때는 길이를 scale조절하여 긴 바닥으로 만들기도 하였으며, rotate를 통해 벽처럼 보이도록 설정하기도 하였습니다. 이러한 transformation들을 각 object들에 속해 있는 shader에 전달하여 transform하여 좌표 변경하도록 하였습니다.

```
void create_G() {
    gmodel = glm::mat4(1.0);
    groundShader->use();
    groundShader->setMat4("view", view);
    gmodel = glm::translate(gmodel, glm::vec3(0.0f, 0.0f, groundY));
    gmodel = glm::scale(gmodel, glm::vec3(0.5f, 5.0f, 1.0f));
    //gmodel = glm::rotate(gmodel, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 0.0f));
    groundShader->setMat4("model", gmodel);
    groundShader->setVec3("gcolor", gcolor);
    ground->draw(groundShader);
}
```

4) Viewing

Viewing은 이번 과제에서는 object 이동과는 크게 차이를 주지 않았습니다. 처음에 intro를 실행하기 위해 viewing을 설정 한 것과, 실제 게임을 함에 있어서 view를 설정하여 다른 object가 현재 있는 view로 이동하도록, 기준 좌표가 되도록 설정하였습니다. 또한, 위에서 내려오는 입체감을 주기 위해서 view의 y좌표를 수정하여 packman이 ghost를 아래에서 피하도록 시점 변경을 하였습니다.

```
view = glm::lookAt(cameraPos, glm::vec3(.0f, .0f, 0.0f), glm::vec3(0.0f, 1.0f, 0.0f));
view = view * camArcBall.createRotationMatrix();
```

5) Texture

Texture은 배경에 이용하기 위해 LoadTexture()함수를 이용하였습니다. 처음에 게임을 시작하기전에 Plane에 Texture을 입혀 팩맨의 모양을 보여주었으며, 팩맨 게임에 어울리는 푸르스름한 어두운 배경을 plane에 입혀 텍스쳐가 보이도록 하였습니다

```
background = loadTexture("background/bam.jpg");
background2 = loadTexture("background/pack.png");
```

6) Lighting

Lighting은, 이 게임에서의 모토가 2D감성을 살리는 것이였기 때문에, 게임을 직접 플레이 하는 데에 잇는 object에는 1980년대처럼 Lighting의 효과를 받지 않도록 하였습니다. 그러나, Keyframe animation을 통해 움직이는 후방에 있는 거대 팩맨은 2D와의 연관이 없도록 했기 때문에 Light를 좌측 상단에 배치하여 거대 팩맨이 지나가면서 빛이 비치는 것을 표현하도록 하였습니다. global.fs/vs를 통해 Light를 받게 하여 움직이면서 음영이 변하는 것을 관찰할 수 있도록 하였습니다.

```
pack = new Model((GLchar *)"res/models/pacman/PacMan.fbx");
globalShader->use();
globalShader->setVec3("objectColor", objectColor);
globalShader->setVec3("lightColor", lightColor);
globalShader->setVec3("lightPos", lightPos);
globalShader->setFloat("ambientStrenth", ambientStrenth);
```

7) Model Loading

Model loading은 인터넷에서 각각 Ghost, PackMan, Apple 모델을 다운로드 받아서 model loading을 하였습니다. 각각의 모델들을 지정한 주소에서 받아온 후 create_모델()함수에 각각 적용시켜서 Draw를 하여서 구현을 할 수 있도록 하였습니다

```
ourModel = new Model((GLchar *)"res/models/pacman/PacMan.fbx");
enemy = new Model((GLchar *)"res/models/pacman/ghost.fbx");
fruit = new Model((GLchar *)"res/models/pacman/Apple_Final.fbx");
void create_E() {
    glm::mat4 emodel(1.0);
    // Rotate model by arcball and panning
    emodel = glm::translate(emodel, enemyCoord[0]);
    emodel = emodel * modelArcBall.createRotationMatrix();
    // It's a bit too big for our scene, so scale it down
    emodel = glm::scale(emodel, glm::vec3(0.03f, 0.03f, 0.03f));
    shader->setMat4("model", emodel);
    enemy->Draw(shader);
}
```

8) Text

Text를 실행시키기 위해서는 별도의 text.vs/fs를 설정하여서 만들어야 했습니다. 주어진 글씨체를 입히고, RenderText()에 지정된 좌표에 지정된 크기의 지정된 시간에 맞게 Text를 출력하도록 하였습니다.

```
textShader = new Shader("text.vs", "text.fs");
text = new Text((char*)"fonts/arial.ttf", textShader, SCR_WIDTH, SCR_HEIGHT);
```

```
else if (startflag == 4) { // the good end
    text->RenderText("Congratulation! Your Record is", 300.0f, 550.0f, 0.5f, glm::vec3(1.0, 1.0f, 1.0f));
    text->RenderText(to_string(endTime), 400.0f, 450.0f, 0.5f, glm::vec3(0.0, 1.0f, 1.0f));
}
```

9) Animation (Keyframe)

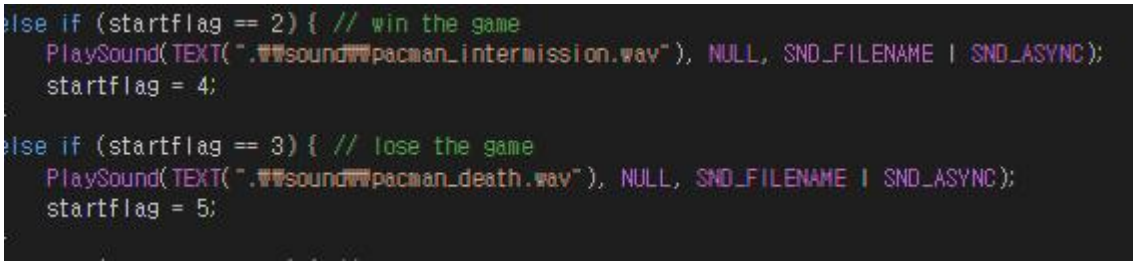
Object가 부드럽게 움직이기 위해서는 Keyframe animation을 이용해야 하는데, 거대 팩맨이 반복적으로 배경에서 움직이기 위해서, updateAnimData()와 initKeyframes()를 이용하였습니다. 특별히 다른 input key를 이용하는 것이 아닌, 자동적으로 배경을 지나가는 것을 표현하였으며, initKeyframes()함수에서 경로 좌표와 rotate하는 각도 등을 수정해 주며 거대 팩맨이 이동을 할 수 있도록 하였습니다.

```
if (renderMode == INIT) {
    float cTime = (float)glfwGetTime(); // current time
    timeT = cTime - beginT;
    updateAnimData();
}
if (renderMode == STOP) {
    renderMode = INIT;
    beginT = glfwGetTime();
}
globalShader->use();
globalShader->setMat4("view", view);
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(xTrans, yTrans, zTrans));
glm::vec3 eulerAngles(glm::radians(xAngle), glm::radians(yAngle), glm::radians(zAngle));
glm::quat q(eulerAngles);
glm::mat4 rotMatrix = q.operator glm::mat4x4();
model = model * rotMatrix;
globalShader->setMat4("model", model);
pack->Draw(globalShader);
```

10) PlaySound

이번 과제에서 별도로 추가한 기능은 PlaySound입니다. **PlaySound 함수를 실행하기 위해서는 windows.h와 mmsystem.h, winmm.lib를 추가해주고 환경설정을 추가로 해주어야 했습니다.** 이때, 있었던 애로사항으로는, Windows보다 mmsystem 헤더를 먼저 추가해 주면 오류가 나서 실행을 하지 못하는 점이 있었습니다. PlaySound 함수를 통해 필요한 wav를 다운받고, 이를 이용해 게임 인트로 음악과, 어떠한 이벤트가 발생했을 때, 혹은 지속적으로 나오는 게임의 배경음을 구현하기 위해 SND_ASYNC, SND_LOOP과 같은 condition을 추가해 주었습니다.

```
#include <Windows.h>
#include <mmsystem.h>
#pragma comment(lib,"winmm.lib")
```



III. 결론 및 느낀 점

이번 학기에 배운 모든 내용을 집약적으로 실행해 보려고 하였습니다. 간단한 object 생성에서부터, texture을 입히는 것, 그리고 keyframe animation까지 실행해 보았을 뿐만 아니라, 추가적으로 PlaySound함수를 구현해 실제 게임과 같이 난이도 조정도 하면서 게임을 즐길 수 있도록 하였습니다. 애로사항은 게임에서 Collision을 구현하기가 어려웠는데, 주어진 좌표와 모델 사이즈를 계산해 coordinate의 거리 계산을 통해 Collision을 구현할 수 있었습니다. 게임을 구현하는데에 있어서 1980년대 그래픽에서 조금 더 벗어나서 즐길 수 있도록 하였으나, 구현을 매끄럽게 하지 못한 점이 아쉽다고 할 수 있겠습니다. 그럼에도 불구하고, 주어진 조건을 조금씩이라도, 전부 동원하여 어느 정도의 컴퓨터 그래픽스에서 배운 바를 행할 수 있어서 좋은 경험이었던 것 같습니다.