**Computer Graphics Final Project Report**

**Project Name : Travel of space ball**

**Final Report for Computer Graphics**

**Lee Donghoon**

**13011186**

**glee1228@naver.com**

**2018-06-18**

**Guohua Li (Prof.)**

1. **Title**

**Travel of space ball**

1. **Introduction**

/\* **What I felt while doing the project**

and my own ideas:

The most difficult thing was to implement something. The first idea I wanted to implement was to implement the rubix cube, but it lacked spatial concepts and could not be implemented. I did not even want to use open source.

So, what I thought was the first thing I wanted to do was to create an idea of ​​moving something based on the Android code. So I changed the Android code to the Android code and changed the action to a bouncing motion. I realized how to implement the space forward, I built a skybox and implemented the ball in it. The part I was struggling with was that texture mapping was not working well, The most difficult point was to move the viewpoint based on the speed of the ball.

When I first designed this code, I had the idea of ​​creating a game. So I made a planet around the ball and I wanted to make the planet fly forward by pressing the button.

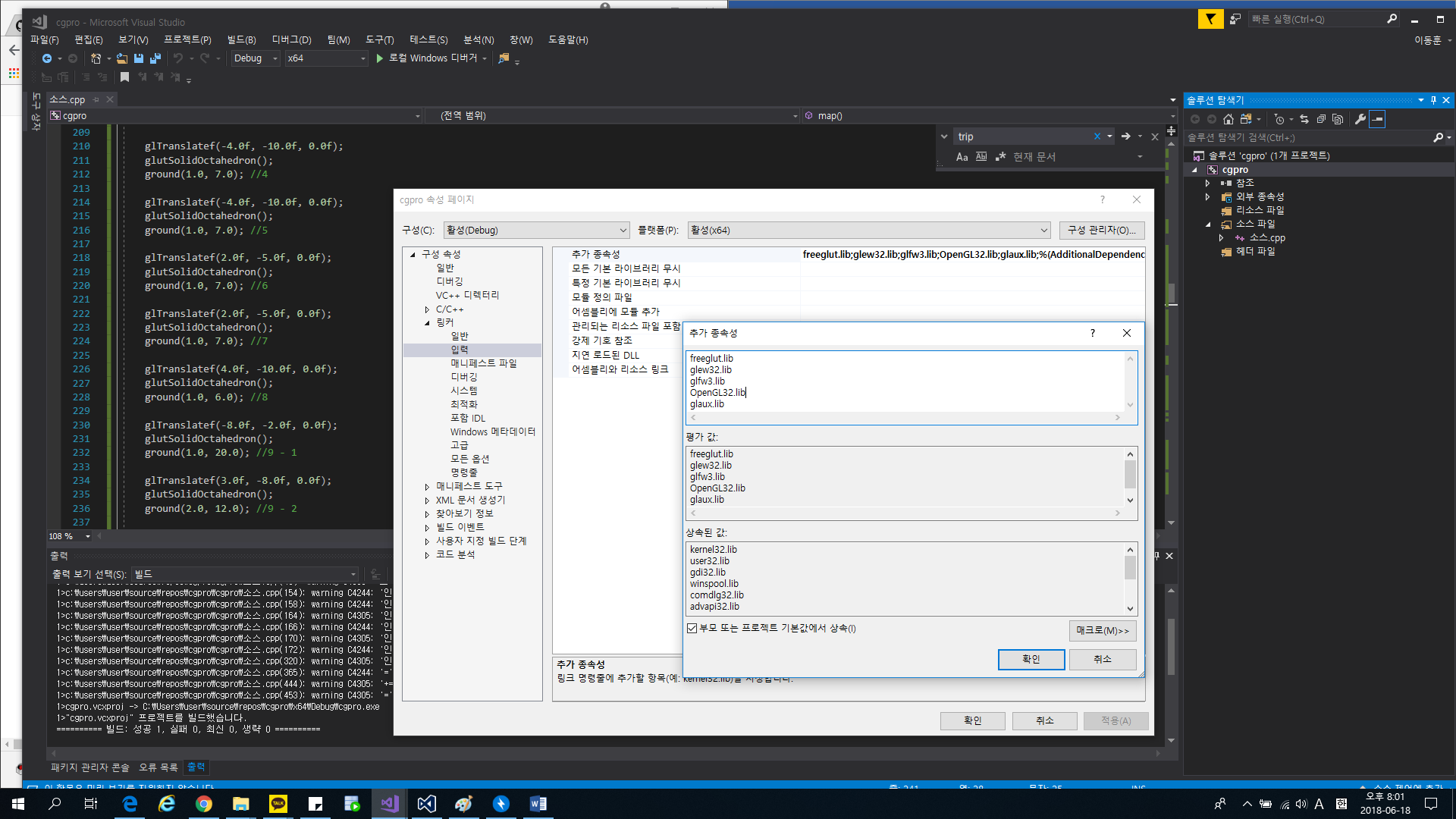
However, it has been difficult to implement the collision function and eventually it has not been implemented.  
It was interesting to be able to create something during the final exam period, but it was a project that took so much time. References are your BMP LOAD and Android code.

1. **Contents**
2. I put the background into the texture, and the code to read the bmp file referenced Professor.Lee's data.
3. Use glutSpecialFunc() to increase and decrease the speed with the arrow keys, and move left and right.
4. The speed is specified by the lower limit and the upper limit.
5. The background can be changed by keyboard callbacks and menu functions, and game usage is displayed through the console window.
6. To start from scratch, you can restart by clicking the r button and menu 1 button.
7. Press the esc button to exit.
8. We made texture mapping of the main character and the surrounding planet balls and background.
9. Light is specified as defalut, and light change has a section that lights in the middle of the code.
10. Move the ball left or right depending on the direction key(using glutSpecialFunc())
11. Blend Mode( ‘l’keyboard Button ) -> all Blended

Project improvements(I think) :

1. Add a crash function to make the velocity zero when the ball hits ground or obstruction.
2. Space ball can launch the planet to remove obstacles (crash function required)
3. Insert Music

Project Setting:



--The rest is the same as class setting—

PROJECT USER INTERFACE

→ & ← : Move the ball left and right.

↑ & ↓ : Ball's speed control.

space : Jumping the ball.

r : Reset Game.

b : change the background.

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

esc : End game.

1. **Concluding Remarks**

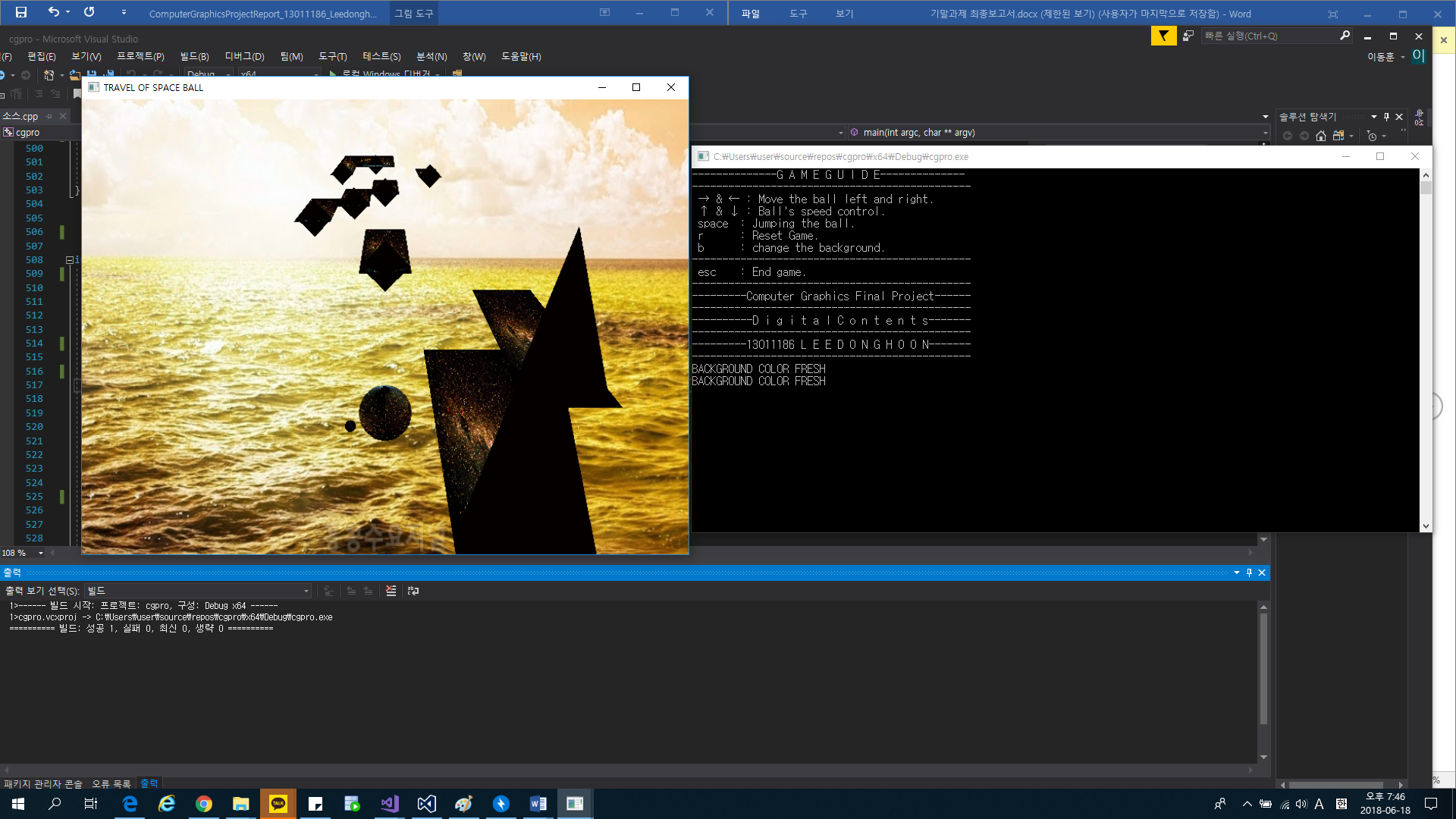
Result:

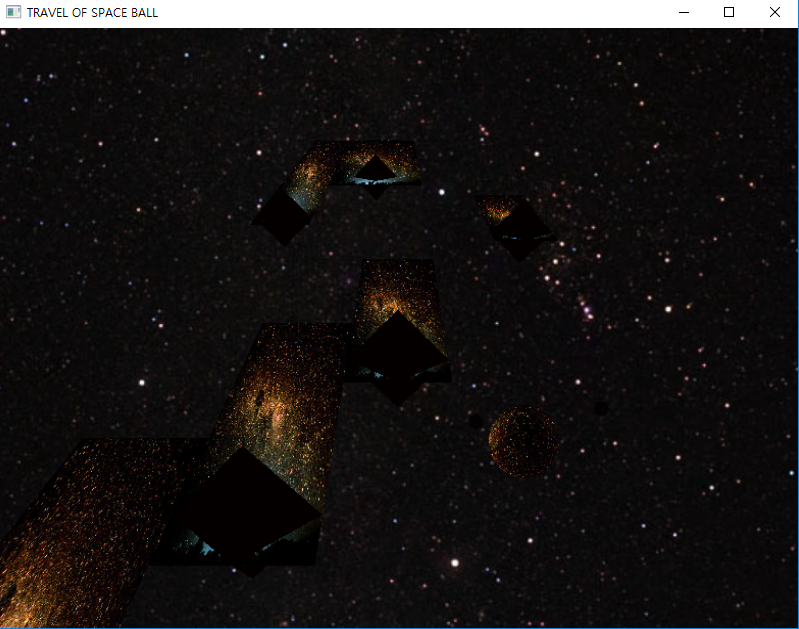
2018/06/18

Writer : LeeDonghoon

FileName: cgpro.cpp

ProjectName: Travel of Space Ball





1. Result Code :

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*S e j o n g U n i v e r s i t y\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*D i g i t a l C o n t e n t s\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1 3 0 1 1 1 8 6 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*P R O V I D E D B Y L e e D o n g h o o n \*\*\*\*\*\*\*\*\*//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2 0 1 8 / 0 6 / 1 8 \* \* \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* C o m p u t e r G r a p h i c s \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* P R O F E S S O R . L E E G U K H W A \*\*\*\*\*\*\*\*\*//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*F i n a l P r o j e c t \* \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*//

#include <glut.h>

#include <gl/GLU.h>

#include <gl/GL.h>

#include <iostream>

#include <Windows.h>

#include<vector>

#include <fstream>

#include <stdio.h>

#include <cmath>

#include <mmsystem.h>

#include <string.h>

#pragma warning(disable:4996)

#pragma comment(lib, "glaux.lib")

using namespace std;

static double time = 0; // 점프와 동시에 우주공을 컨트롤하는 데 사용하는 time 전역변수

static double time2 = 0; //우주공 주변을 도는 행성공의 시간 time2 전역변수

static bool blending = false;

int angle = 0;

int MenuChk;

GLint width = 800, height = 600;

GLfloat eye = 0.0; //공과 함께 시선이 이동하도록 해주는 전역변수

unsigned int speed = 5; //공의 속도를 조절하는 전역변수

float b\_lightPosition[] = { 0.0f, 4.0f, 16.0f }; //시작 시 공의 좌표 X, Y, Z

bool jump = false, over = false, go = false; //각각 점프여부, 게임오버 여부, 비행기가 앞으로 가는 움직임의 여부이다.

GLfloat g\_light[] = { 0.0, 0.0, 1.5, 1.0 };

GLfloat light0\_ambient[] = { 0.5, 0.4, 0.3, 1.0 };

GLfloat light0\_diffuse[] = { 0.8, 0.7, 0.6, 1.0 };

GLfloat light0\_specular[] = { 1.0, 1.0, 1.0, 1.0 };

GLfloat light1\_ambient[] = { 1.0,0.0,0.0,1.0 };

GLfloat mat1\_ambient[] = { 0.5,0.4,0.3,1.0 };

GLuint MyTextureObject;

GLuint MyBall;

const char \* filename = "bluesea.bmp"; //배경 1 텍스처

const char \* filename3 = "cosmic.bmp"; //ground와 공에 씌우는 텍스처

typedef union PixelInfo

{

std::uint32\_t Colour;

struct

{

std::uint8\_t B, G, R, A;

};

} \*PPixelInfo;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Example Code Start\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class BMP

{

private:

std::uint32\_t width, height;

std::uint16\_t BitsPerPixel;

std::vector<std::uint8\_t> Pixels;

public:

BMP(const char\* FilePath);

std::vector<std::uint8\_t> GetPixels() const { return this->Pixels; }

std::uint32\_t GetWidth() const { return this->width; }

std::uint32\_t GetHeight() const { return this->height; }

bool HasAlphaChannel() { return BitsPerPixel == 32; }

};

BMP::BMP(const char\* FilePath)

{

std::fstream hFile(FilePath, std::ios::in | std::ios::binary);

if (!hFile.is\_open()) throw std::invalid\_argument("Error: File Not Found.");

hFile.seekg(0, std::ios::end);

std::size\_t Length = hFile.tellg();

hFile.seekg(0, std::ios::beg);

std::vector<std::uint8\_t> FileInfo(Length);

hFile.read(reinterpret\_cast<char\*>(FileInfo.data()), 54);

if (FileInfo[0] != 'B' && FileInfo[1] != 'M')

{

hFile.close();

throw std::invalid\_argument("Error: Invalid File Format. Bitmap Required.");

}

if (FileInfo[28] != 24 && FileInfo[28] != 32)

{

hFile.close();

throw std::invalid\_argument("Error: Invalid File Format. 24 or 32 bit Image Required.");

}

BitsPerPixel = FileInfo[28];

width = FileInfo[18] + (FileInfo[19] << 8);

height = FileInfo[22] + (FileInfo[23] << 8);

std::uint32\_t PixelsOffset = FileInfo[10] + (FileInfo[11] << 8);

std::uint32\_t size = ((width \* BitsPerPixel + 31) / 32) \* 4 \* height;

Pixels.resize(size);

hFile.seekg(PixelsOffset, std::ios::beg);

hFile.read(reinterpret\_cast<char\*>(Pixels.data()), size);

hFile.close();

}

/\*Load BITMAP file to make texture \*/

void LoadGLTextures(const char \* filename) {

glClearColor(0.0, 0.0, 0.0, 0.5);

BMP info = BMP(filename);

glGenTextures(1, &MyTextureObject);

glBindTexture(GL\_TEXTURE\_2D, MyTextureObject);

glTexImage2D(GL\_TEXTURE\_2D, 0, 3, info.GetWidth(), info.GetHeight(), 0, GL\_RGB, GL\_UNSIGNED\_BYTE, info.GetPixels().data());

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glEnable(GL\_TEXTURE\_2D);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Example Code Endline\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void MyInit(void) { //전체 , 지역조명 defalut 설정

glShadeModel(GL\_SMOOTH);

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light0\_ambient);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light0\_diffuse);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, light0\_specular);

}

void PolarView(GLfloat radius, GLfloat elevation, GLfloat azimuth, GLfloat twist) {

glTranslatef(0.0, 0.0, -radius); //극 좌표계 설정

glRotatef(-elevation, 1.0, 0.0, 0.0);

glRotatef(azimuth, 0.0, 0.0, 1.0);

glRotatef(-twist, 0.0, 0.0, 1.0);

}

void Ball() { //공 출력

glFrontFace(GL\_CW);

GLUquadricObj \*obj;

GLUquadricObj \*obj2;

obj = gluNewQuadric();

obj2 = gluNewQuadric();

gluQuadricDrawStyle(obj, GLU\_FILL); //주인공 우주공

gluQuadricDrawStyle(obj2, GLU\_FILL); //공 주변을 도는 행성공

glPushMatrix();

glRotatef(angle, -1, 0, 0);

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light1\_ambient);

glEnable(GL\_TEXTURE\_2D);

gluQuadricTexture(obj, true);

gluSphere(obj, 0.5, 30.0, 10.0);

if (blending == true) {

glMaterialfv(GL\_FRONT, GL\_AMBIENT, mat1\_ambient);

glEnable(GL\_BLEND);

glBlendFunc(GL\_SRC\_ALPHA, GL\_DST\_COLOR);

}

else

glDisable(GL\_BLEND);

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light0\_ambient);

glColor3f(1.0, 1.0, 1.0);

glPushMatrix();

glTranslatef(0, 0.3, 0);

glTranslated(sin(time2+30), 0, 0);

gluSphere(obj2, 0.1, 30.0, 10.0);

glPopMatrix();

glPushMatrix();

glTranslatef(0.35, 0.5, 0);

glTranslated(cos(time2 + 30), 0, 0);

gluSphere(obj2, 0.1, 30.0, 10.0);

glPopMatrix();

glPopMatrix();

}

void ground(float w, float h) { // (w = 길의 폭 , h = 길의 길이)

glColor3f(0.5f, 0.5f, 0.5f);

glDisable(GL\_LIGHTING);

glPushMatrix();

glEnable(GL\_TEXTURE\_2D);

glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_DECAL);

glBegin(GL\_QUADS);

glBindTexture(GL\_TEXTURE\_2D, MyBall);

glTexCoord2f(0.0, 0.0); glVertex3f(-w, w, 0.0);

glTexCoord2f(1.0, 0.0); glVertex3f(w, w, 0.0);

glTexCoord2f(1.0, 1.0); glVertex3f(w, -h, 0.0);

glTexCoord2f(0.0, 1.0); glVertex3f(-w, -h, 0.0);

glEnd();

glPopMatrix();

glEnable(GL\_LIGHTING);

glColor3f(1.0, 1.0, 1.0);

}

void map() { //맵을 구현하는 메소드

glTranslatef(0.0f, 3.5f, 16.0f);

glColor3f(1.0, 0.0, 0.0);

glRotatef(90.0f, 1.0f, 0.0f, 0.0f);

ground(10.0, 1.0); //1

glTranslatef(2.0f, -5.0f, 0.0f);

glutSolidCone(1.0, -3.5, 30, 10);

ground(1.0, 7.0); //2

glTranslatef(2.0f, -5.0f, 0.0f);

glutSolidCone(1.0,-3.5,30,10);

ground(1.0, 7.0); //3

glTranslatef(-4.0f, -10.0f, 0.0f);

glutSolidOctahedron();

ground(1.0, 7.0); //4

glTranslatef(-4.0f, -10.0f, 0.0f);

glutSolidOctahedron();

ground(1.0, 7.0); //5

glTranslatef(2.0f, -5.0f, 0.0f);

glutSolidOctahedron();

ground(1.0, 7.0); //6

glTranslatef(2.0f, -5.0f, 0.0f);

glutSolidOctahedron();

ground(1.0, 7.0); //7

glTranslatef(4.0f, -10.0f, 0.0f);

glutSolidOctahedron();

ground(1.0, 6.0); //8

glTranslatef(-8.0f, -2.0f, 0.0f);

glutSolidOctahedron();

ground(1.0, 20.0); //9 - 1

glTranslatef(3.0f, -8.0f, 0.0f);

glutSolidOctahedron();

ground(2.0, 12.0); //9 - 2

glColor3f(0.5f, 0.5f, 0.5f);

glTranslatef(0.0f, -9.0f, 1.0f);

ground(1.0, 12.0); //10

}

void MyDisplay() {

GLfloat LightPosition[] = { 0.0, 0.0, 1.5, 1.0 };

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glDisable(GL\_LIGHTING);

LoadGLTextures(filename); //텍스처 매핑 로드

float r = 999.0f;

glPushMatrix();

glBindTexture(GL\_TEXTURE\_2D, MyTextureObject); //sky 전면

glBegin(GL\_QUADS);

glTexCoord2f(0.0f, 0.f); glVertex3f(-r, -r / 2, -r);

glTexCoord2f(1.f, 0.f); glVertex3f(r, -r / 2, -r);

glTexCoord2f(1.f, 1.f); glVertex3f(r, r, -r);

glTexCoord2f(0.f, 1.f); glVertex3f(-r, r, -r);

glEnd();

glBindTexture(GL\_TEXTURE\_2D, MyTextureObject);

glBegin(GL\_QUADS);

glTexCoord2f(0.f, 0.f); glVertex3f(r, -r / 2, r);

glTexCoord2f(1.f, 0.f); glVertex3f(-r, -r / 2, r);

glTexCoord2f(1.f, 1.f); glVertex3f(-r, r, r);

glTexCoord2f(0.f, 1.f); glVertex3f(r, r, r);

glEnd();

glBindTexture(GL\_TEXTURE\_2D, MyTextureObject);

glBegin(GL\_QUADS);

glTexCoord2f(0.f, 0.f); glVertex3f(r, -r / 2, -r);

glTexCoord2f(1.f, 0.f); glVertex3f(r, -r / 2, r);

glTexCoord2f(1.f, 1.f); glVertex3f(r, r, r);

glTexCoord2f(0.f, 1.f); glVertex3f(r, r, -r);

glEnd();

glBindTexture(GL\_TEXTURE\_2D, MyTextureObject);

glBegin(GL\_QUADS);

glTexCoord2f(0.f, 0.f); glVertex3f(-r, -r / 2, r);

glTexCoord2f(1.f, 0.f); glVertex3f(-r, -r / 2, -r);

glTexCoord2f(1.f, 1.f); glVertex3f(-r, r, -r);

glTexCoord2f(0.f, 1.f); glVertex3f(-r, r, r);

glEnd();

glBindTexture(GL\_TEXTURE\_2D, MyTextureObject);

glBegin(GL\_QUADS);

glTexCoord2f(0.f, 0.f); glVertex3f(-r, r, -r);

glTexCoord2f(1.f, 0.f); glVertex3f(r, r, -r);

glTexCoord2f(1.f, 1.f); glVertex3f(r, r, r);

glTexCoord2f(0.f, 1.f); glVertex3f(-r, r, r);

glEnd();

glBindTexture(GL\_TEXTURE\_2D, MyTextureObject);

glBegin(GL\_QUADS);

glTexCoord2f(0, 0); glVertex3f(-r, -r / 2, r);

glTexCoord2f(1, 0); glVertex3f(r, -r / 2, r);

glTexCoord2f(1, 1); glVertex3f(r, -r / 2, -r);

glTexCoord2f(0, 1); glVertex3f(-r, -r / 2, -r);

glEnd();

glPopMatrix();

glEnable(GL\_LIGHTING);

glDisable(GL\_TEXTURE\_2D);

PolarView(27.0, -20.0, 0.0, 0.0); //극 좌표계 설정

glTranslatef(0.0, 0.0, eye); //시점이 공과 함께 움직이도록 설정

glFrustum(-1.0, 1.0, -1.0, 1.0, 0.0, 5.0); //원근 투영

glPushMatrix(); //지역조명 설정

glTranslatef(0.0, 15.0, eye);

glLightfv(GL\_LIGHT0, GL\_POSITION, g\_light);

glDisable(GL\_LIGHTING);

glColor3f(0.9, 0.9, 0.9);

glutWireSphere(0.06, 10, 10);

glEnable(GL\_LIGHTING);

glPopMatrix();

LoadGLTextures(filename3); //ground , ball 텍스처 로드

glPushMatrix();

glTranslatef(b\_lightPosition[0], b\_lightPosition[1], b\_lightPosition[2]); //공의 위치 설정

Ball(); // 공 설정

glPopMatrix();

glDisable(GL\_LIGHTING);

map(); //맵 설정

glEnable(GL\_LIGHTING);

glEnable(GL\_TEXTURE\_2D);

glutSwapBuffers();

}

void MyReshape(int NewWidth, int NewHeight) {

glViewport(0, 0, NewWidth, NewHeight);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0, (GLfloat)NewWidth / (GLfloat)NewHeight, 0.5, 1000.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void reset() { //메뉴에서 reset항목을 선택하거나 r 버튼을 누를 시 실행되는 함수

cout << "게임을 새로 시작합니다." << endl;

b\_lightPosition[0] = 0.0f; //시작시 우주공 세팅.

b\_lightPosition[1] = 4.0f;

b\_lightPosition[2] = 16.0f;

eye = 0.0f;

over = false;

go = false;

speed = 5;

}

void myjump(int Value) { //공을 점프시키는 메소드

if (jump) {

b\_lightPosition[1] = 4.0f+abs((sin(time) \* 45.0)); //sin함수를 사용하여 점수 구현.

if (time = 50) //시간에 따라 점프를 false시킴

jump = false;

}

else {

b\_lightPosition[1] = 4.0f;

}

glutTimerFunc(500, myjump, 1);

glutPostRedisplay();

}

void background()

{

if (filename == "bluesea.bmp") //배경 1

filename = "cosmo.bmp"; //배경 2

else

filename = "bluesea.bmp";

cout << "BACKGROUND COLOR FRESH" << endl;

glutPostRedisplay();

}

void blend()

{

if (blending == true)

blending = false;

else

blending = true;

}

void MyKeyboard(unsigned char KeyPressed, int X, int Y) { //키보드 이벤트

switch (KeyPressed) {

case 'r':

reset(); //리셋 메소드

break;

case 'b': // 배경 텍스처매핑 변경

background();

break;

case 'l':

blend();

break;

case 32: //스페이스바(return 32) 입력

jump = true; //위의 myjump함수에서 점프를 할 수 있도록 하는 트리거

break;

case 27: //esc버튼 입력시 프로그램 종료

cout << "PROGRAM EXIT" << endl;

exit(0);

break;

default:

break;

}

glutPostRedisplay();

}

void MySpecial(int key, int x, int y) {

if (key == GLUT\_KEY\_UP) {

go = true; //공을 앞으로 움직이게 한다는 트리거 실행

if (speed >= 140) { //스피드의 값이 140이 넘을 경우 더이상 증가되지 않음

cout << "Can't be accelerated anymore" << endl;

speed = 140;

}

else {

speed += 5; //스피드 증가

}

}

if (key == GLUT\_KEY\_DOWN) {

if (speed <= 5) { //스피드의 값이 5보다 낮을 경우 더이상 감소되지 않고 공을 멈추는 트리거 실행

cout << "Can't be slowed anymore" << endl;

speed = 5;

go = false;

}

else {

speed -= 5; //스피드 감소

}

}

if (key == GLUT\_KEY\_LEFT) {

b\_lightPosition[0] -= 0.1f; //공을 왼쪽으로 이동시킴

}

else if (key == GLUT\_KEY\_RIGHT) {

b\_lightPosition[0] += 0.1f; //공을 오른쪽으로 이동시킴

}

glutPostRedisplay();

}

void MyTimer(int Value) {

time2 = time2 + 0.05;

if (go) { //go 가 참일 경우에만 실행

if (b\_lightPosition[2] > -74.7) {

b\_lightPosition[2] -= 0.1f; //공을 Z축 방향으로 이동시킴

eye += 0.1; //시점도 공과 함께 이동

angle = (angle + 8)%360;

if (jump != false)

{

time = 0;

time = time + 0.05;

}

}

else {

b\_lightPosition[2] = -74.7; //맵의 끝까지 도달하면 그자리에서 멈춤

cout << ">>> End <<<" << endl;

over = true;

}

}

glutTimerFunc(150 - speed, MyTimer, 1);

glutPostRedisplay();

}

void Guide() { //콘솔창에 출력하는 게임 설명서

cout << "--------------G A M E G U I D E--------------" << endl;

cout << "----------------------------------------------" << endl;

cout << " → & ← : Move the ball left and right." << endl;

cout << " ↑ & ↓ : Ball's speed control." << endl;

cout << " space : Jumping the ball." << endl;

cout << " r : Reset Game." << endl;

cout << " b : change the background." << endl;

cout << "----------------------------------------------" << endl;

cout << " esc : End game." << endl;

cout << "----------------------------------------------" << endl;

cout << "---------Computer Graphics Final Project------" << endl;

cout << "----------------------------------------------" << endl;

cout << "----------D i g i t a l C o n t e n t s-------" << endl;

cout << "----------------------------------------------" << endl;

cout << "---------13011186 L E E D O N G H O O N-------" << endl;

cout << "----------------------------------------------" << endl;

}

void MenuPoint(int entryID) {

MenuChk = entryID;

if (entryID == 1) {

reset(); //리셋

}

if (entryID = 2) {

Guide(); //게임 사용법 콘솔창에 입력

}

if (entryID = 3) {

background(); // 배경 변경

}

if (entryID == 4) {

blend();

}

glutPostRedisplay();

}

void MenuFunc() {

GLint mainMenu = glutCreateMenu(MenuPoint); //메인 메뉴

glutAddMenuEntry("Reset", 1);

glutAddMenuEntry("Game Guide", 2);

glutAddMenuEntry("Change BackGround", 3);

glutAddMenuEntry("BLEND MODE", 4);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

}

int main(int argc, char\*\* argv) {

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGBA | GLUT\_DOUBLE | GLUT\_DEPTH);

glutInitWindowSize(width, height);

glutInitWindowPosition(100, 100);

glutCreateWindow("TRAVEL OF SPACE BALL");

MyInit();

glutDisplayFunc(MyDisplay);

glutReshapeFunc(MyReshape);

glutKeyboardFunc(MyKeyboard);

glutSpecialFunc(MySpecial);

glutTimerFunc(150 - speed, MyTimer, 1);

glutTimerFunc(500, myjump, 1);

MenuFunc();

glutMainLoop();

return 0;

}

1. **Reference**

[1] Class example code( Android code) => How to use the sin,cos function for recoil

[2] BMP.cpp -> Professor.Lee ‘s Example Code