

**숙제3 한국외대 네비게이션**

**과목명 컴퓨터그래픽스**

**담당교수 김상철교수님**

**제출일 20210622**

**전공 컴퓨터전자시스템**

**학번 201904458**

**이름 이준용**

* **소스 코드**

/\* 제목 : 한국외대 컴퓨터 그래픽스 과제

프로그램 : 대학교 네비게이션

학 과 : 컴퓨터 전자시스템 전공

학 번 : 201904458

이 름 : 이준용 \*/

//#include<GL/glut.h>

#include<GL/glew.h>

#include <math.h>

#include<GL/freeglut.h>

#include<GL/gl.h>

#include <iostream>

using std::cout;

#define \_USE\_MATH\_DEFINES // for C++

#define \_CRT\_SECURE\_NO\_WARNINGS

#define windowWidth 1000

#define windowHeight 1000

float x = 0.0f, y = 1.75f, z = 5.0f;

float lx = 0.0f, ly = 0.0f, lz = -1.0f;

int deltaMove = 0;

double rotate\_x = 0.0;

double rotate\_y = 0.0;

# define white 1.0 , 1.0 , 1.0

# define green 0.0 , 0.502, 0.0

# define red 1.0 , 0.0 , 0.0

# define gray 0.502, 0.502,0.502

# define hgray 0.117, 0.180,0.227

# define blue 0.0 , 0.0 , 1.0

# define pi 3.14159265

# define gold 1.0,215.0/255.0,0.0

# define hgreen 0.0,100.0/255.0,0.0

# define brown 210.0/255.0, 105.0/255.0, 30.0/255.0

# define men 244.0/255.0 ,164.0/255.0 , 96.0/255.0

# define menba 139.0/255.0 ,69.0/255.0,19.0/255.0

float fang[8][3];

float san[8][3];

float mat\_ambient[] = { 0.2, 0.2, 0.2, 0.0 };

float mat\_diffuse[] = { 0.6, 0.6, 0.6, 0.0 };

float mat\_specular[] = {0.2, 0.2, 0.2, 0.0};

float mat\_emissive[] = {0.0, 0.0, 0.0, 0.0};

float mat\_shininess[] = {50.0};

GLuint Width = 1000, Height = 1000;

float x\_Translation = 0.0, y\_Translation = 0.0, z\_move = 0.0;

float x\_Rotation = 0.0, y\_Rotation = 0.0;

float ww = 1000, hh = 1000;

static float angle = 0.0, ratio = 0.0;

void display(void);

void bmpReader(void);

static GLint imagewidth;

static GLint imageheight;

void init(void);

void keyboardDemo(unsigned char key, int x, int y);

void rotate(GLfloat ang);

void transla(GLint direct);

void trany(GLint direct);

struct Coordinate {

GLfloat x;

GLfloat y;

GLfloat z;

}cameraPos, sightLin, defaultVal;

void cons(double x, double y, double z, double x1, double y1, double z1) {

san[0][0] = x;

san[0][1] = y;

san[0][2] = z;

san[1][0] = x;

san[1][1] = y;

san[1][2] = z + z1;

san[4][0] = x;

san[4][1] = y + y1;

san[4][2] = z;

san[5][0] = x;

san[5][1] = y + y1;

san[5][2] = z + z1 / 2;

for (int i = 0; i < 3; i++) {

if (i == 0) {

san[3][i] = san[0][i] + x1;

san[2][i] = san[1][i] + x1;

san[6][i] = san[4][i] + x1;

san[7][i] = san[5][i] + x1;

}

else {

san[3][i] = san[0][i];

san[2][i] = san[1][i];

san[6][i] = san[4][i];

san[7][i] = san[5][i];

}

}

}

void constract(double x, double y, double z, double x1, double y1, double z1) {

fang[0][0] = x;

fang[0][1] = y;

fang[0][2] = z;

fang[1][0] = x;

fang[1][1] = y;

fang[1][2] = z + z1;

fang[2][0] = x + x1;

fang[2][1] = y;

fang[2][2] = z + z1;

fang[3][0] = x + x1;

fang[3][1] = y;

fang[3][2] = z;

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 3; j++) {

if (j == 1)

fang[i + 4][j] = fang[i][j] + y1;

else

fang[i + 4][j] = fang[i][j];

}

}

}

void build2() {

glBegin(GL\_POLYGON);

//glColor3f(red);

glNormal3f(0.0, -1.0, 0.0);

glVertex3f(san[0][0], san[0][1], san[0][2]);

glVertex3f(san[1][0], san[1][1], san[1][2]);

glVertex3f(san[2][0], san[2][1], san[2][2]);

glVertex3f(san[3][0], san[3][1], san[3][2]);

glEnd();

glBegin(GL\_POLYGON);

//glColor3f(green);

glNormal3f(-1.0, 0.0, 0.0);

glVertex3f(san[1][0], san[1][1], san[1][2]);

glVertex3f(san[0][0], san[0][1], san[0][2]);

glVertex3f(san[4][0], san[4][1], san[4][2]);

glVertex3f(san[5][0], san[5][1], san[5][2]);

glEnd();

glBegin(GL\_POLYGON);

//glColor3f(green);

glNormal3f(1.0, 0.0, 0.0);

glVertex3f(san[7][0], san[7][1], san[7][2]);

glVertex3f(san[6][0], san[6][1], san[6][2]);

glVertex3f(san[3][0], san[3][1], san[3][2]);

glVertex3f(san[2][0], san[2][1], san[2][2]);

glEnd();

glBegin(GL\_POLYGON);

//glColor3f(green);

glNormal3f(0.0, 0.0, 1.0);

glVertex3f(san[5][0], san[5][1], san[5][2]);

glVertex3f(san[6][0], san[6][1], san[6][2]);

glVertex3f(san[2][0], san[2][1], san[2][2]);

glVertex3f(san[1][0], san[1][1], san[1][2]);

glEnd();

glBegin(GL\_POLYGON);

//glColor3f(green);

glNormal3f(0.0, 0.0, -1.0);

glVertex3f(san[0][0], san[0][1], san[0][2]);

glVertex3f(san[3][0], san[3][1], san[3][2]);

glVertex3f(san[7][0], san[7][1], san[7][2]);

glVertex3f(san[4][0], san[4][1], san[4][2]);

glEnd();

glBegin(GL\_POLYGON);

//glColor3f(red);

glNormal3f(0.0, 1.0, 0.0);

glVertex3f(san[4][0], san[4][1], san[4][2]);

glVertex3f(san[7][0], san[7][1], san[7][2]);

glVertex3f(san[6][0], san[6][1], san[6][2]);

glVertex3f(san[5][0], san[5][1], san[5][2]);

glEnd();

}

void build() {

glBegin(GL\_POLYGON);

//glColor3f(red);

glNormal3f(0.0, -1.0, 0.0);

glVertex3f(fang[0][0], fang[0][1], fang[0][2]);

glVertex3f(fang[1][0], fang[1][1], fang[1][2]);

glVertex3f(fang[2][0], fang[2][1], fang[2][2]);

glVertex3f(fang[3][0], fang[3][1], fang[3][2]);

glEnd();

glBegin(GL\_POLYGON);

//glColor3f(green);

glNormal3f(-1.0, 0.0, 0.0);

glVertex3f(fang[1][0], fang[1][1], fang[1][2]);

glVertex3f(fang[0][0], fang[0][1], fang[0][2]);

glVertex3f(fang[4][0], fang[4][1], fang[4][2]);

glVertex3f(fang[5][0], fang[5][1], fang[5][2]);

glEnd();

glBegin(GL\_POLYGON);

//glColor3f(green);

glNormal3f(1.0, 0.0, 0.0);

glVertex3f(fang[7][0], fang[7][1], fang[7][2]);

glVertex3f(fang[6][0], fang[6][1], fang[6][2]);

glVertex3f(fang[2][0], fang[2][1], fang[2][2]);

glVertex3f(fang[3][0], fang[3][1], fang[3][2]);

glEnd();

glBegin(GL\_POLYGON);

//glColor3f(green);

glNormal3f(0.0, 0.0, 1.0);

glVertex3f(fang[5][0], fang[5][1], fang[5][2]);

glVertex3f(fang[6][0], fang[6][1], fang[6][2]);

glVertex3f(fang[2][0], fang[2][1], fang[2][2]);

glVertex3f(fang[1][0], fang[1][1], fang[1][2]);

glEnd();

glBegin(GL\_POLYGON);

//glColor3f(green);

glNormal3f(0.0, 0.0, -1.0);

glVertex3f(fang[0][0], fang[0][1], fang[0][2]);

glVertex3f(fang[3][0], fang[3][1], fang[3][2]);

glVertex3f(fang[7][0], fang[7][1], fang[7][2]);

glVertex3f(fang[4][0], fang[4][1], fang[4][2]);

glEnd();

glBegin(GL\_POLYGON);

//glColor3f(red);

glNormal3f(0.0, 1.0, 0.0);

glVertex3f(fang[4][0], fang[4][1], fang[4][2]);

glVertex3f(fang[7][0], fang[7][1], fang[7][2]);

glVertex3f(fang[6][0], fang[6][1], fang[6][2]);

glVertex3f(fang[5][0], fang[5][1], fang[5][2]);

glEnd();

}

void display(void)

{

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

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glRotatef(rotate\_x, 0.1, 0.0, 0.0); // 위아래 회전

glRotatef(rotate\_y, 0.0, 0.1, 0.0); // 왼 오른쪽 회전

gluLookAt(-500, 200, 500, 500, -10, 500, 0.0, 1.0, 0.0); // 초기 카메라 방향

gluLookAt(cameraPos.x, cameraPos.y, cameraPos.z,

cameraPos.x + sightLin.x, cameraPos.y + sightLin.y, cameraPos.z + sightLin.z,

defaultVal.x, defaultVal.y, defaultVal.z);

glRotatef(x\_Rotation, 1.0, 0.0, 0.0);

glRotatef(y\_Rotation, 0.0, 1.0, 0.0);

glTranslatef(-270, 0, -270);

glScalef(1.5, 1.5, 1.5);

glBegin(GL\_POLYGON);

constract(0, 0, 500, 10000, 10, -49500); // 초록 잔디(왼쪽)

glColor3f(green);

build();

glBegin(GL\_POLYGON);

constract(2700, 0, 100, 1000, 15, -1500); // 명수당

glColor4f(blue, 0.8);

build();

glBegin(GL\_POLYGON); // 중앙도로

constract(0, 0, 500, 10000, 10, 100);

glColor3f(gray);

build();

glBegin(GL\_POLYGON);

constract(0, 0, 600, 10000, 10, 50000); // 초록 잔디(오른쪽)

glColor3f(green);

build();

glBegin(GL\_POLYGON);

constract(330, 20, 250, 140, 1, 250);

glColor3f(gray);

build();

for (int i = 0; i < 50; i++) { //5 도로 흰색 타일 부분

glBegin(GL\_POLYGON);

constract(i \* 100, 10, 500 + 45, 40, 1, 10);

glColor3f(white);

build();

}

glBegin(GL\_POLYGON); // 자연대 하얀색 담장

constract(10, 10, 10, 20, 20, 480);

glColor3f(white);

build();

constract(30, 10, 10, 440, 20, 20);

glColor3f(white);

build();

constract(470, 10, 10, 20, 20, 480);

glColor3f(white);

build();

constract(30, 10, 470, 100, 20, 20);

glColor3f(white);

build();

constract(230, 10, 470, 100, 20, 20);

glColor3f(white);

build();

glBegin(GL\_POLYGON);

constract(10, 10, 10, 20, 50, 20);

glColor3f(white);

build();

glBegin(GL\_POLYGON);

constract(470, 10, 10, 20, 50, 20);

glColor3f(white);

build();

glBegin(GL\_POLYGON);

constract(230, 10, 470, 20, 50, 20);

glColor3f(white);

build();

glBegin(GL\_POLYGON);

constract(10, 10, 470, 20, 50, 20);

glColor3f(white);

build();

glBegin(GL\_POLYGON); // 자연대 입구

constract(470, 10, 470, 20, 50, 20);

glColor3f(white);

build();

glBegin(GL\_POLYGON);

constract(310, 10, 470, 20, 50, 20);

glColor3f(white);

build();

glBegin(GL\_POLYGON);

constract(110, 10, 470, 20, 50, 20);

glColor3f(white);

build();

glColor3f(hgray);

for (int i = 35; i < 470; i += 25) // 검은색 담장

{

constract(15, 20, i, 6, 30, 6);

build();

}

for (int i = 35; i < 470; i += 25)

{

constract(475, 20, i, 6, 30, 6);

build();

}

for (int i = 35; i < 470; i += 25)

{

constract(i, 20, 15, 6, 30, 6);

build();

}

for (int i = 35; i < 110; i += 25)

{

constract(i, 20, 485, 6, 30, 6);

build();

}

for (int i = 250; i < 310; i += 25)

{

constract(i, 20, 485, 6, 30, 6);

build();

}

glColor3f(white);

constract(320, 10, 250, 10, 100, 220);

build();

constract(460, 10, 250, 10, 100, 220);

build();

glColor3f(hgray);

constract(320, 110, 250, 150, 10, 220);

build();

glColor3f(white);

constract(50, 10, 50, 12, 450, 200); //자연대 남쪽벽

build();

glColor3f(white);

constract(1000, 10, 50, 12, 450, 200); // 자연대 북쪽벽

build();

glColor3f(white);

constract(62, 10, 50, 950, 450, 12);

build();

glColor3f(white);

constract(62, 10, 235, 950, 450, 12);

build();

glColor3f(white);

constract(62, 60, 235, 950, 400, 12);

build();

glColor3f(white);

constract(162, 60, 235, 850, 400, 12);

build();

glColor3f(white);

constract(62, 110, 235, 950, 350, 12);

build();

glColor4f(blue, 0.35); //자연대 창문

for (int i = 0; i < 6; ++i) {

glPushMatrix();

glTranslatef(50 \* i, 0, 0);

constract(62 + 50, 60, 235, 50, 50, 30);

build();

}

glPopMatrix();

glColor4f(blue, 0.35); //자연대 창문

for (int i = 0; i < 6; ++i) {

glPushMatrix();

glTranslatef(50 \* i, 0, 0);

constract(62 + 50, 120, 235, 50, 50, 30);

build();

}

glPopMatrix();

glColor4f(blue, 0.35); //자연대 창문

for (int i = 0; i < 6; ++i) {

glPushMatrix();

glTranslatef(50 \* i, 0, 0);

constract(62 + 50, 180, 235, 50, 50, 30);

build();

}

glPopMatrix();

glColor4f(blue, 0.35); //자연대 창문

for (int i = 0; i < 6; ++i) {

glPushMatrix();

glTranslatef(50 \* i, 0, 0);

constract(62 + 50, 240, 235, 50, 50, 30);

build();

}

glPopMatrix();

glColor4f(blue, 0.35); //자연대 창문

for (int i = 0; i < 6; ++i) {

glPushMatrix();

glTranslatef(50 \* i, 0, 0);

constract(62 + 50, 300, 235, 50, 50, 30);

build();

}

glPopMatrix();

glColor4f(blue, 0.35); // 입구 유리문

constract(330, 22, 450, 130, 90, 10);

build();

for (int i = 0; i < 6; i++) // 자연대 가는길

{

glColor3f(brown);

constract(235 - i \* 15, 10, 250 + i \* 40, 40, 2, 20);

build();

}

for (int i = 0; i < 12; i++) // 기숙사 가는길

{

glColor3f(brown);

constract(800 - i \* 15, 10, 1100 - i \* 40, 40, 2, 20);

build();

}

glColor3f(gold);

glTranslatef(250, 400, 0);

glutSolidSphere(35.0, 20, 20);

glTranslatef(-250, -400, 0);

glColor3f(brown); // 니므 몸통

constract(60, 10, 390, 15, 70, 15);

build();

glColor3f(hgreen);

glTranslatef(60, 70, 400); // 나무

glutSolidSphere(25.0, 20, 20);

glTranslatef(20, 0, 0);

glutSolidSphere(25.0, 20, 20);

glTranslatef(-10, 0, -10);

glutSolidSphere(25.0, 20, 20);

glTranslatef(0, 0, 20);

glutSolidSphere(25.0, 20, 20);

glTranslatef(0, 10, -10);

glutSolidSphere(25.0, 20, 20);

//기숙사 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);

glMaterialfv(GL\_FRONT, GL\_SHININESS, mat\_shininess);

glMaterialfv(GL\_FRONT, GL\_EMISSION, mat\_emissive);

glPushMatrix();

glTranslatef(-270, 0, -270);

glScalef(1.5, 1.5, 1.5);

glColor3f(brown);

constract(500, 200, 700, 560, 30, 400);

build();

glColor3f(white);

constract(500, -100, 700, 560, 300, 400);

build();

glColor4f(hgray, 0.35);

for (int i = 0; i < 9; ++i) {

glPushMatrix();

glTranslatef(0, 70 \* i, 0);

constract(500, -100, 680, 700, 10, 2);

build();

glPopMatrix();

}

glColor4f(blue, 0.35);

for (int i = 0; i < 8; ++i) {

glPushMatrix();

glTranslatef(20 \* i, 0, 0);

constract(500, -10, 680, 40, 40, 2);

build();

constract(500, -10, 680, 40, 40, 2);

build();

}

glPopMatrix();

glColor4f(blue, 0.35); // 1층 창문

for (int i = 0; i < 8; ++i) {

glPushMatrix();

glTranslatef(20 \* i, 0, 0);

constract(500, 50, 680, 60, 40, 2);

build();

constract(500, 50, 680, 60, 40, 2);

build();

}

glPopMatrix();

glColor4f(blue, 0.35); // 2층 창문

for (int i = 0; i < 8; ++i) {

glPushMatrix();

glTranslatef(20 \* i, 0, 0);

constract(500, 120, 680, 40, 40, 2);

build();

constract(500, 120, 680, 40, 40, 2);

build();

}

glPopMatrix();

glColor4f(blue, 0.35); // 3층 창문

for (int i = 0; i < 8; ++i) {

glPushMatrix();

glTranslatef(20 \* i, 0, 0);

constract(500, 200, 680, 40, 40, 2);

build();

constract(500, 200, 680, 40, 40, 2);

build();

}

glPopMatrix();

glPushMatrix();

glTranslatef(500, 170, 680);

glScalef(0.4, 0.5, 2);

glColor3f(red); // hufs 로고

glLineWidth(5);

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, 'H');

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, ' ');

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, 'U');

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, ' ');

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, 'F');

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, ' ');

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, 'S');

glLineWidth(1);

glPopMatrix();

glPopMatrix();

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//공학관\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);

glMaterialfv(GL\_FRONT, GL\_SHININESS, mat\_shininess);

glMaterialfv(GL\_FRONT, GL\_EMISSION, mat\_emissive);

glPushMatrix();

glTranslatef(800, 0, -1370);

glScalef(1.5, 1.5, 1.5);

glColor3f(white);

constract(500, 200, 1000, 700, 30, -1000);

build();

glColor4f(white, 0.99);

constract(500, -100, 1000, 700, 300, -1000);

build();

glColor4f(brown, 0.85);

for (int i = 0; i < 6; ++i) {

glPushMatrix();

glTranslatef(0, 60 \* i, 0);

constract(500, -100, 1000, 700, -20, 5);

build();

glPopMatrix();

}

glColor4f(blue, 0.35); // 1층 창문

for (int i = 0; i < 8; ++i) {

glPushMatrix();

glTranslatef(20 \* i, 0, 0);

constract(500, 50, 1000, 40, -20, 5);

build();

constract(500, 50, 1000, 40, -20, 5);

build();

}

glPopMatrix();

glColor4f(blue, 0.35); // 2층 창문

for (int i = 0; i < 8; ++i) {

glPushMatrix();

glTranslatef(20 \* i, 0, 0);

constract(500, 120, 1000, 60, -20, 5);

build();

constract(500, 120, 1000, 60, -20, 5);

build();

}

glPopMatrix();

glColor4f(blue, 0.35); // 3층 창문

for (int i = 0; i < 8; ++i) {

glPushMatrix();

glTranslatef(20 \* i, 0, 0);

constract(500, 180, 1000, 40, -20, 5);

build();

constract(500, 180, 1000, 40, -20, 5);

build();

}

glPopMatrix();

glPopMatrix();

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//도서관 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);

glMaterialfv(GL\_FRONT, GL\_SHININESS, mat\_shininess);

glMaterialfv(GL\_FRONT, GL\_EMISSION, mat\_emissive);

glPushMatrix();

glTranslatef(1000, 0, -70);

glScalef(1.5, 1.5, 1.5);

glColor3f(white);

constract(500, 200, 700, 660, 10, 400);//constract(-15, 20, 0, 30, 70, -30); 첫번째 인자 x축 정면

build();

glColor3f(brown);

constract(500, -100, 700, 660, 300, 400);

build();

glColor4f(white, 0.9);

for (int i = 0; i < 5; ++i) {

glPushMatrix();

glTranslatef(0, 70 \* i, 0);

constract(500, -100, 640, 640, 10, 2);

build();

glPopMatrix();

}

glColor4f(blue, 0.35); // 1층 창문

for (int i = 0; i < 8; ++i) {

glPushMatrix();

glTranslatef(20 \* i, 0, 0);

constract(500, -10, 680, 40, 40, 2);

build();

constract(500, -10, 680, 40, 40, 2);

build();

}

glPopMatrix();

glColor4f(blue, 0.35); // 2층 창문

for (int i = 0; i < 8; ++i) {

glPushMatrix();

glTranslatef(20 \* i, 0, 0);

constract(500, 50, 680, 60, 40, 2);

build();

constract(500, 50, 680, 60, 40, 2);

build();

}

glPopMatrix();

glColor4f(blue, 0.35); // 3층 창문

for (int i = 0; i < 8; ++i) {

glPushMatrix();

glTranslatef(20 \* i, 0, 0);

constract(500, 120, 680, 40, 40, 2);

build();

constract(500, 120, 680, 40, 40, 2);

build();

}

glPopMatrix();

glPushMatrix();

glTranslatef(500, 170, 680);

glScalef(0.4, 0.5, 2);

glColor3f(red); // 도서관 로고

glLineWidth(5);

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, 'L');

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, 'I');

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, 'B');

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, 'R');

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, 'A');

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, 'R');

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, 'Y');

glLineWidth(1);

glPopMatrix();

glPopMatrix();

GLUquadricObj\* pObj; // Quadric Object

pObj = gluNewQuadric();

glPushMatrix();

glColor3f(white); // 도서관 중앙홀(원기둥으로 표현)

glTranslatef(2000, 0, 900);

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

gluCylinder(pObj, 100,100, 400, 260, 130);

glPopMatrix();

glPushMatrix();

glColor4f(brown,0.8); // 도서관 중앙홀(원기둥으로 표현

glTranslatef(2000, 0, 900);

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

gluCylinder(pObj, 110, 110, 150, 260, 130);

glPopMatrix();

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

glFlush();

glutSwapBuffers();

}

void init(void)

{

GLfloat sun\_direction[] = { 100.0, 100.0, 100.0, 0.0 }; // 태양 조명

GLfloat sun\_intensity[] = { 1.0, 1.0, 1.0, 1.0 }; // 태양 밝기

GLfloat ambient\_intensity[] = { 0.5, 0.5, 0.5, 1.0 }; // 태양 어두운 부분

glEnable(GL\_LIGHTING); // Set up ambient light.

glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, ambient\_intensity);

glEnable(GL\_LIGHT0); // Set up sunlight.

glLightfv(GL\_LIGHT0, GL\_POSITION, sun\_direction);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, sun\_intensity);

glEnable(GL\_COLOR\_MATERIAL); // Configure glColor().

glColorMaterial(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE);

cout << "The OpenGL version is: " << glGetString(GL\_VERSION) << "\n";

cout << glGetString(GL\_VENDOR) << "\n";

glLineWidth(10);

glClearColor(0.0, 0.0, 0.0, 0.0); // 전체 배경색 검은색으로 표현

glEnable(GL\_BLEND);

glDisable(GL\_DEPTH\_TEST);

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE);

glMatrixMode(GL\_PROJECTION);

glOrtho(-ww,ww, -hh, hh, -ww, ww); // specify clipping volume

cameraPos.x = 0.0; cameraPos.y = 1.75; cameraPos.z = 5.0;

sightLin.x = 0.0; sightLin.y = 0.0; sightLin.z = -1.0;

defaultVal.x = 0.0; defaultVal.y = 1.0; defaultVal.z = 1.0;

}

void sightLine(int w, int h) { // reshape 부분

ratio = (w \* 5)/ h;

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glViewport(0, 0, w, h);

gluPerspective(90, ratio, 1, 100000);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluLookAt(0.1, 1, 500, 500, 10, 100,defaultVal.x, defaultVal.y, defaultVal.z);

}

void keyboardDemo(unsigned char key, int x, int y) { // 키보드조작(첨부파일내용에 오류가 있어서 제 마음대로 고쳤습니다.)

switch (key) {

case 'w': // 전진

transla(1);

break;

case 's': // 후진

transla(-1);

break;

case 'd': // 왼쪽(반시계방향)회전

rotate\_y += 5;

break;

case 'a': // 오른쪽(시계방향)회전

rotate\_y -= 5;

break;

case 'e': // 고개 위로

rotate\_x += 5;

break;

case 'q': // 고개 아래로

rotate\_x -= 5;

break;

case 27:

exit(0);

break;

default:

break;

}

glutPostRedisplay();

}

void transla(GLint direct) {

cameraPos.x = cameraPos.x + direct \* (sightLin.x+1) \* 10.3;

cameraPos.z = cameraPos.z + direct \* sightLin.z \* 0.3;

glLoadIdentity();

gluLookAt(cameraPos.x, cameraPos.y, cameraPos.z,

cameraPos.x + sightLin.x, cameraPos.y + sightLin.y, cameraPos.z + sightLin.z,

defaultVal.x, defaultVal.y, defaultVal.z);

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(windowWidth, windowHeight);

glutInitWindowPosition(100, 100);

glutCreateWindow("한국외대 네비게이션\_201904458\_이준용");

init();

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);

glEnable(GL\_DEPTH\_TEST);

glutReshapeFunc(sightLine);

glutDisplayFunc(display);

glutKeyboardFunc(keyboardDemo);

glutMainLoop();

return 0;

}

* **화면 dump**

**과제3으로 주신 ppt과제 설명에 조작키 관련 부분에서 오타가 있어서 다음과 같이 수정했습니다.**

**처음 실행했을 경우 기숙사 건물과 다른 건물의 창문이 정렬이 안되어 있습니다. 실행 화면을 클릭하거나 조작키를 누를 경우 정상적으로 보이게 됩니다.**

**★ w => 앞으로 전진**

**★ s => 뒤로 후진**

**★ a => 왼쪽(반시계방향)회전**

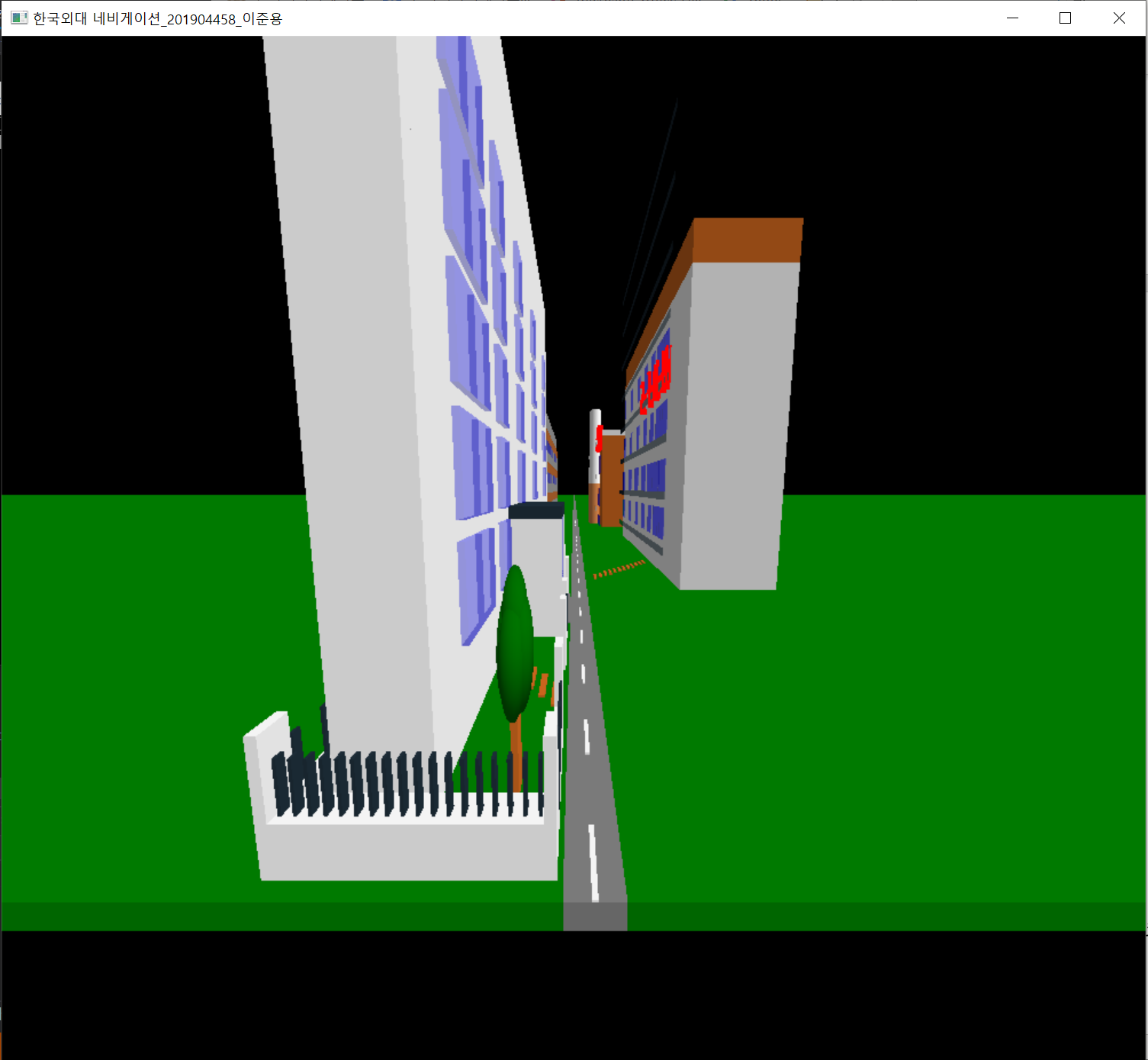
**★ d => 오른쪽(시계방향)회전**

**★ q => 고개 위로**

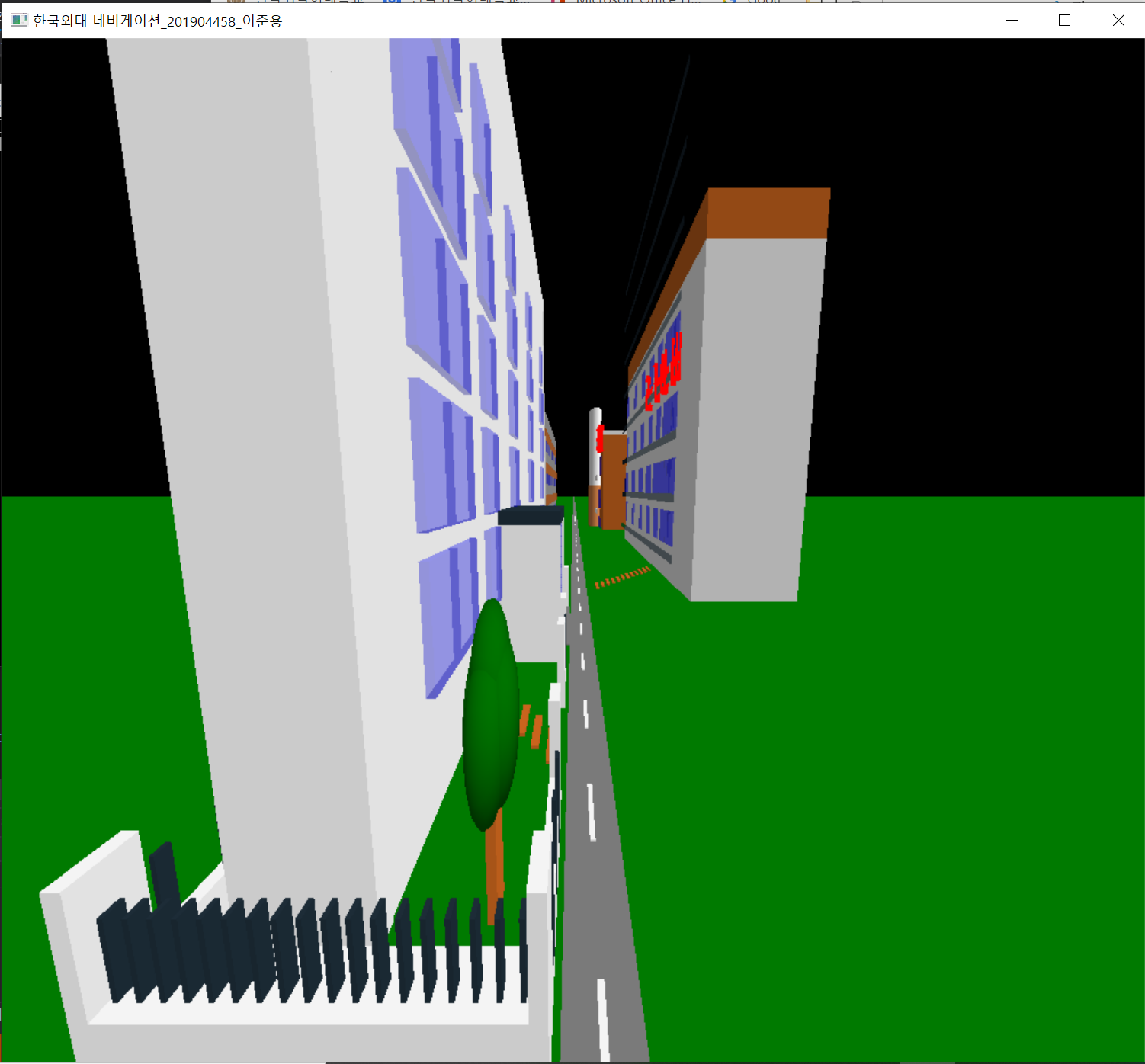
**★ e => 고개 아래로**

**★종료(esc)**

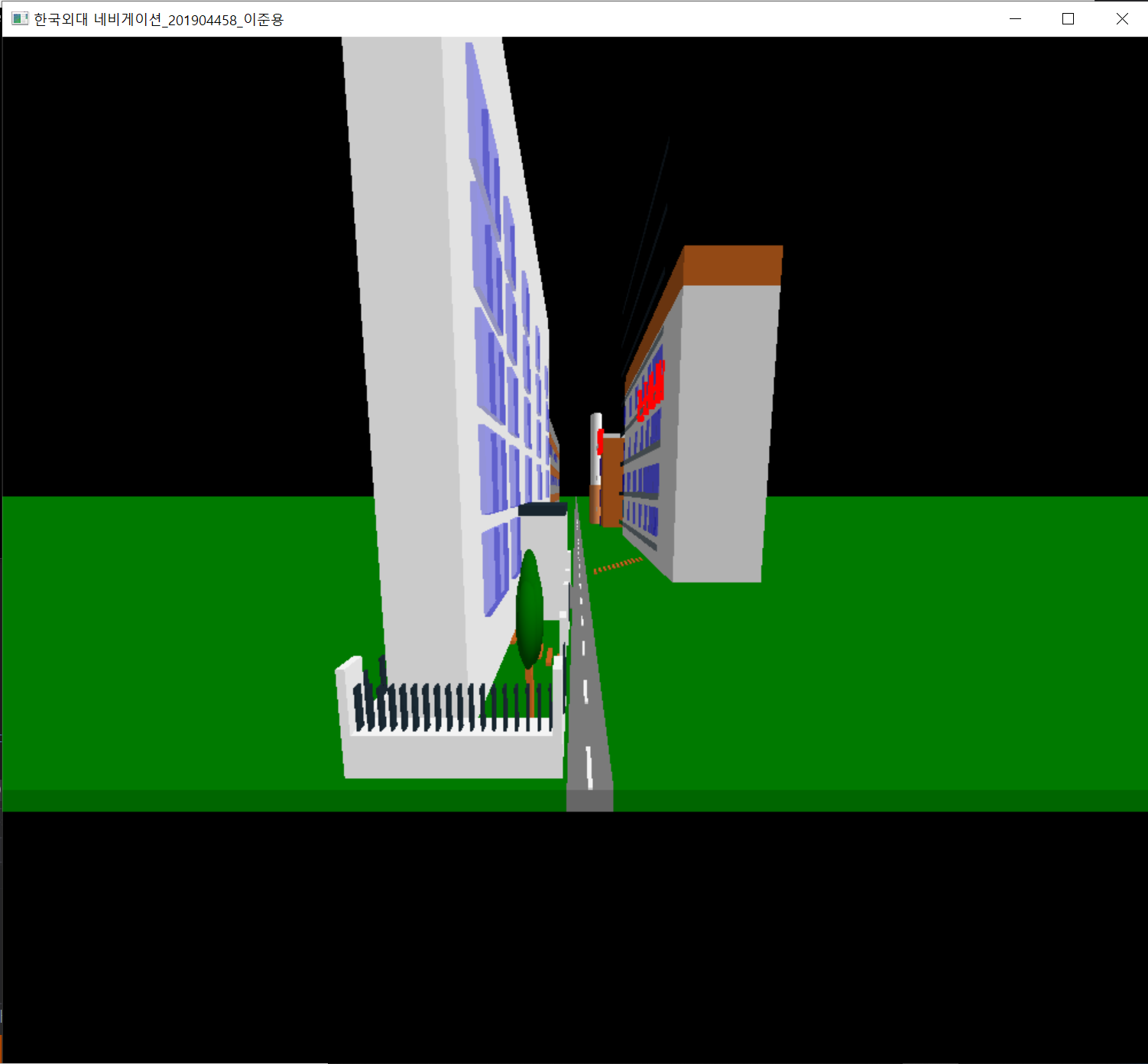
<프로그램 시작 화면>



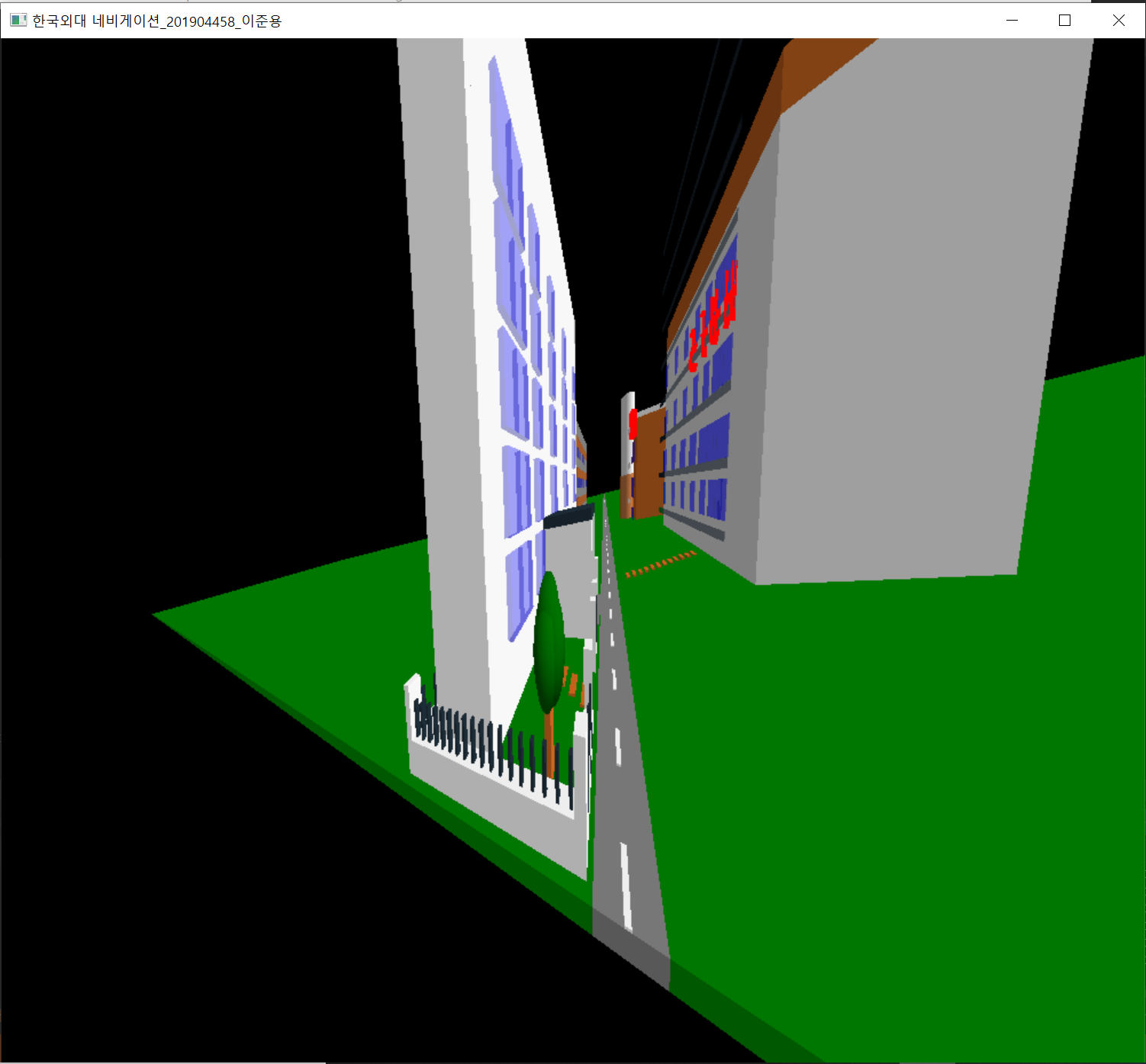
<프로그램 시작 후 키보드로 ‘w’키 10번 눌렀을 경우>



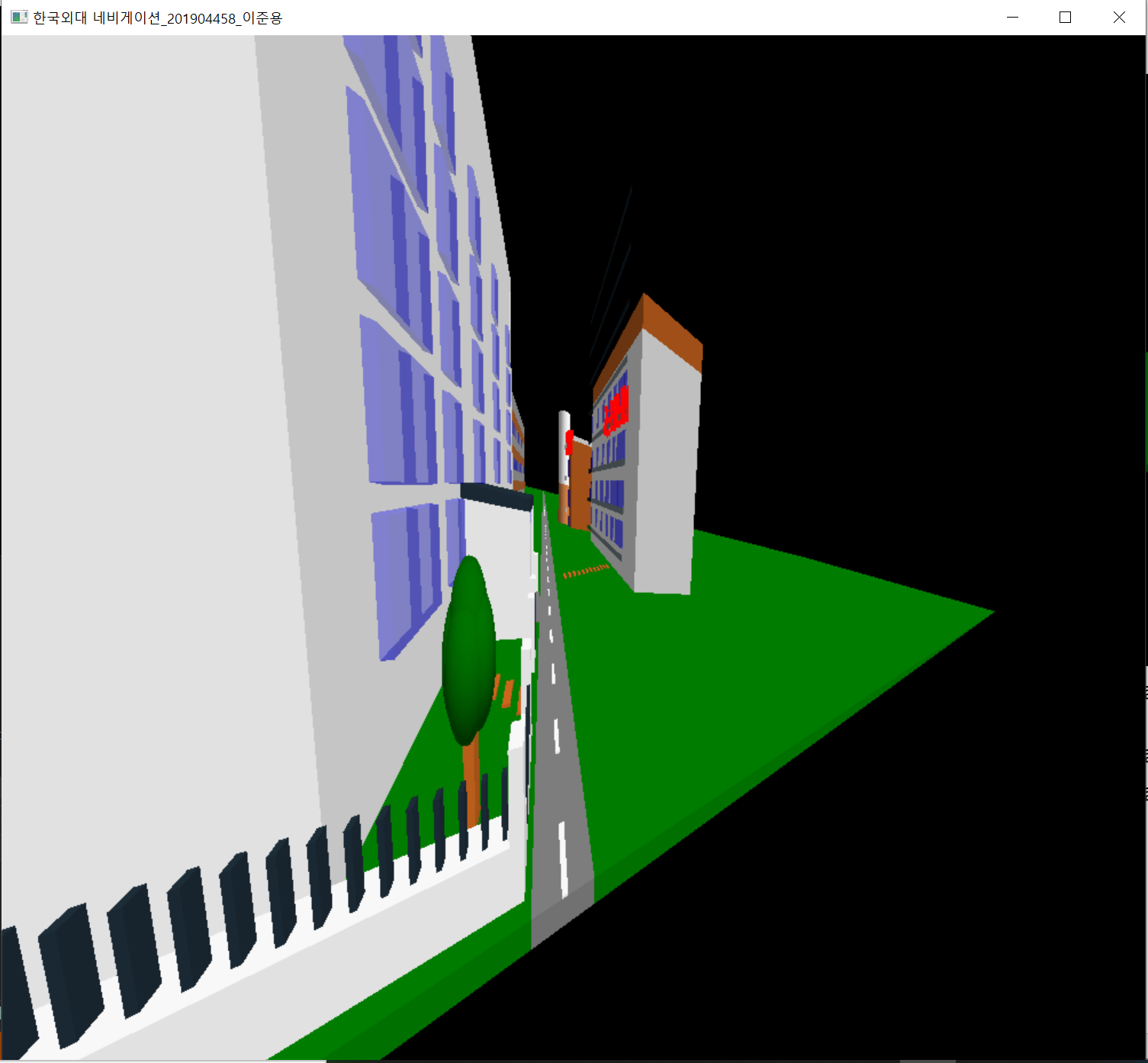
<프로그램 시작 후 키보드로 ‘s’키 10번 눌렀을 경우 >



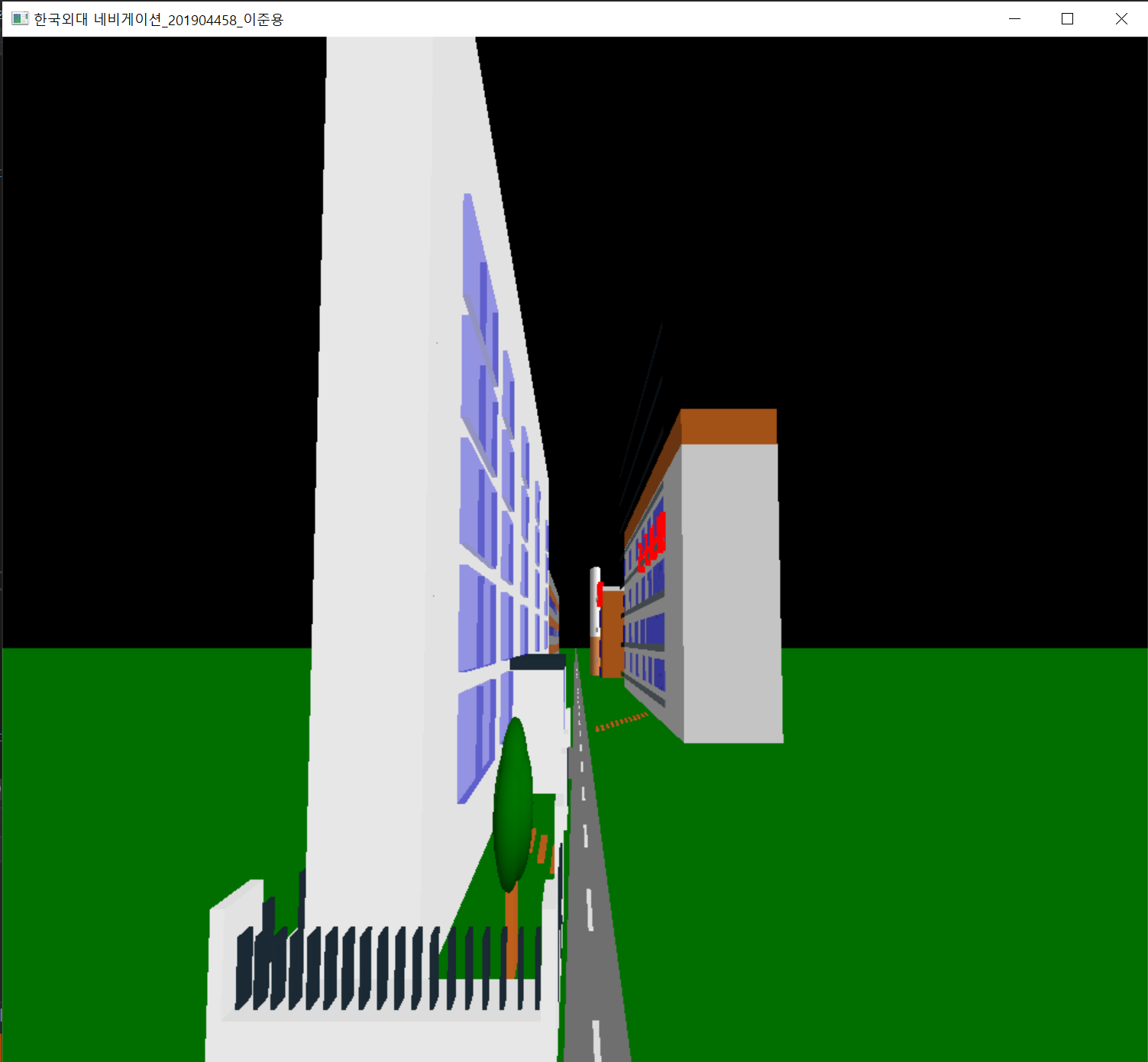
<프로그램 시작 후 키보드로 ‘a’키 3번 눌렀을 경우>



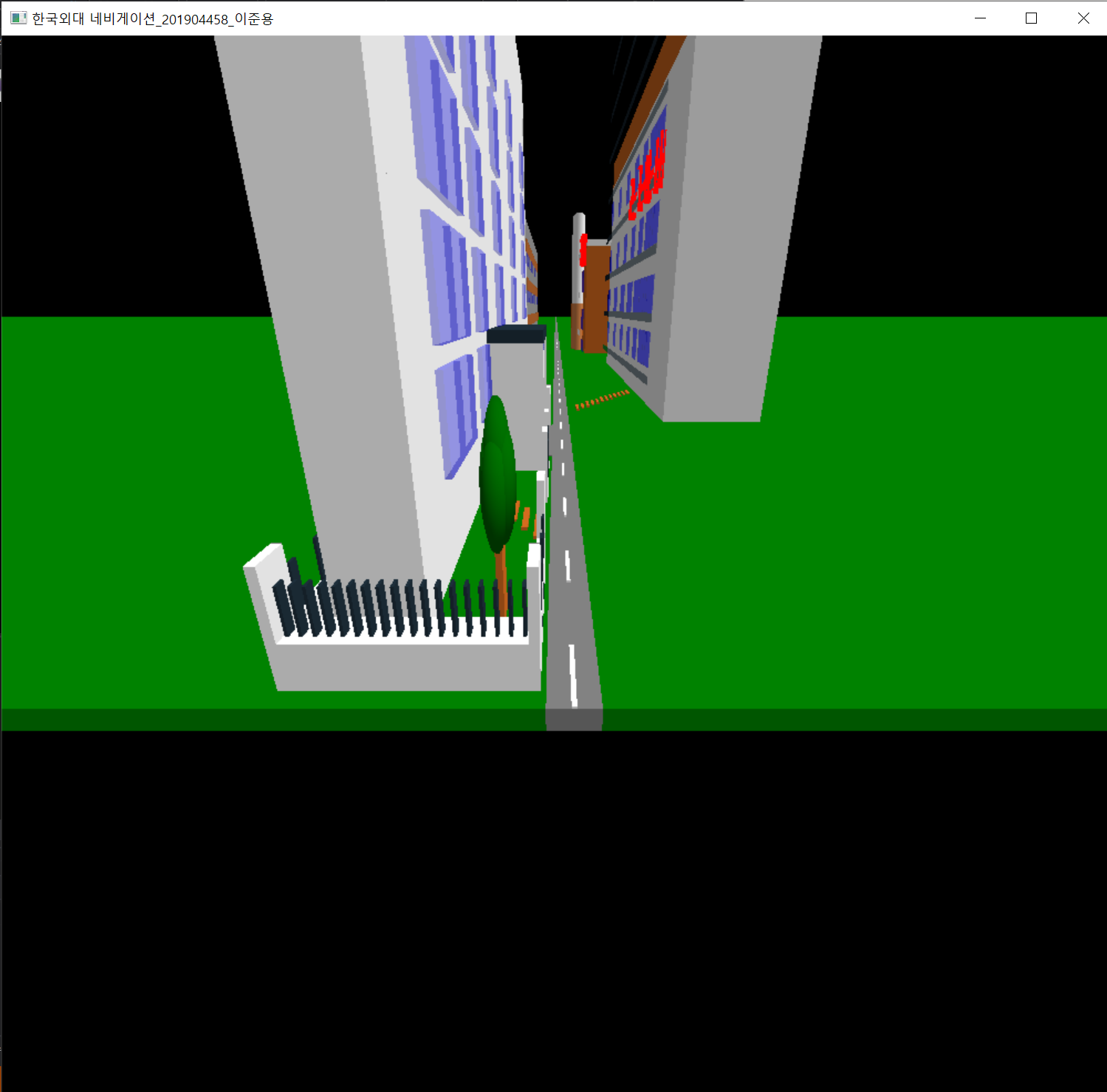
<프로그램 시작 후 키보드로 ‘d’키 3번 눌렀을 경우>



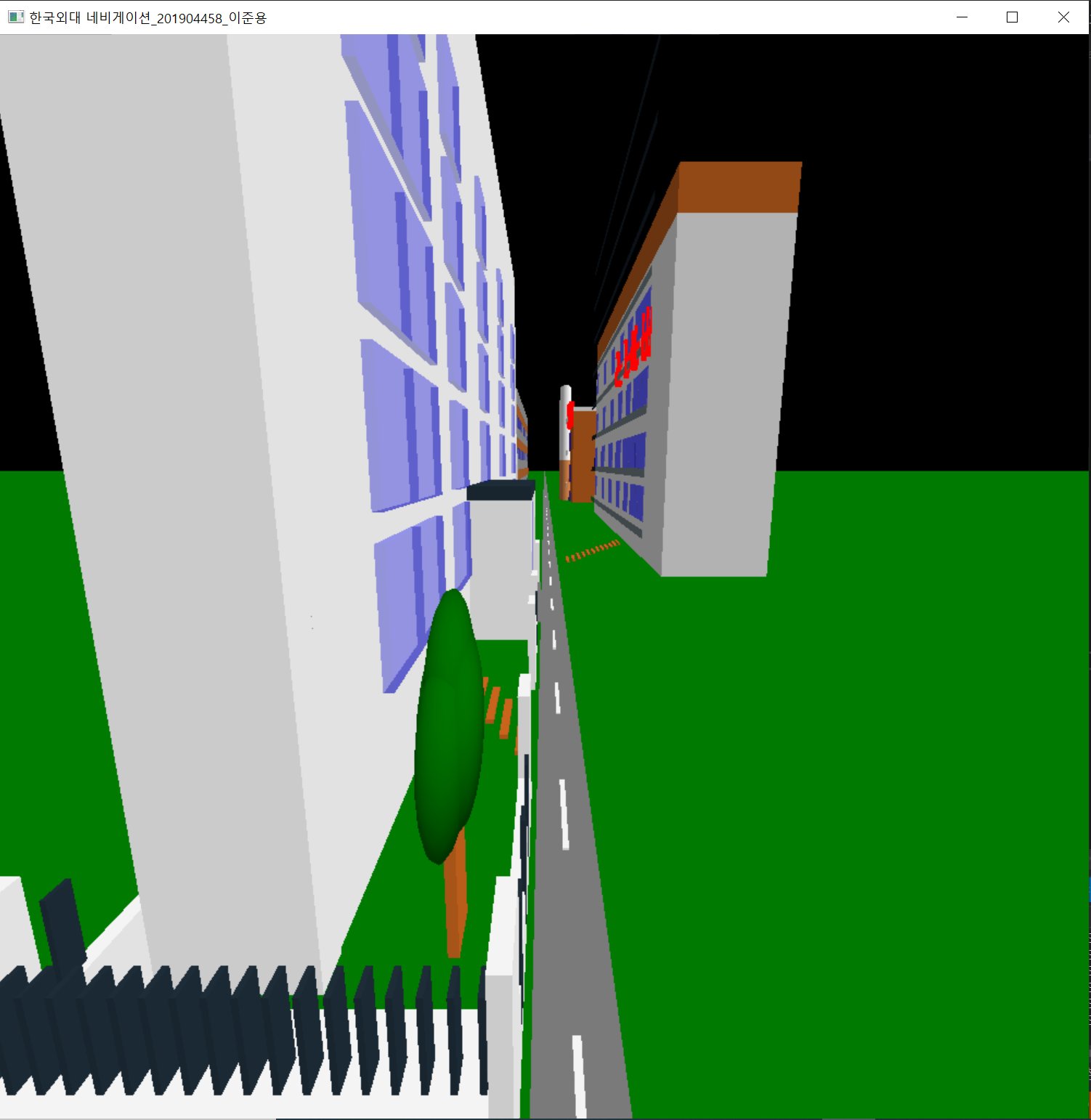
<프로그램 시작 후 키보드로 ‘q’키 눌렀을 경우>



<프로그램 시작 후 키보드로 ‘e’키 3번 눌렀을 경우>



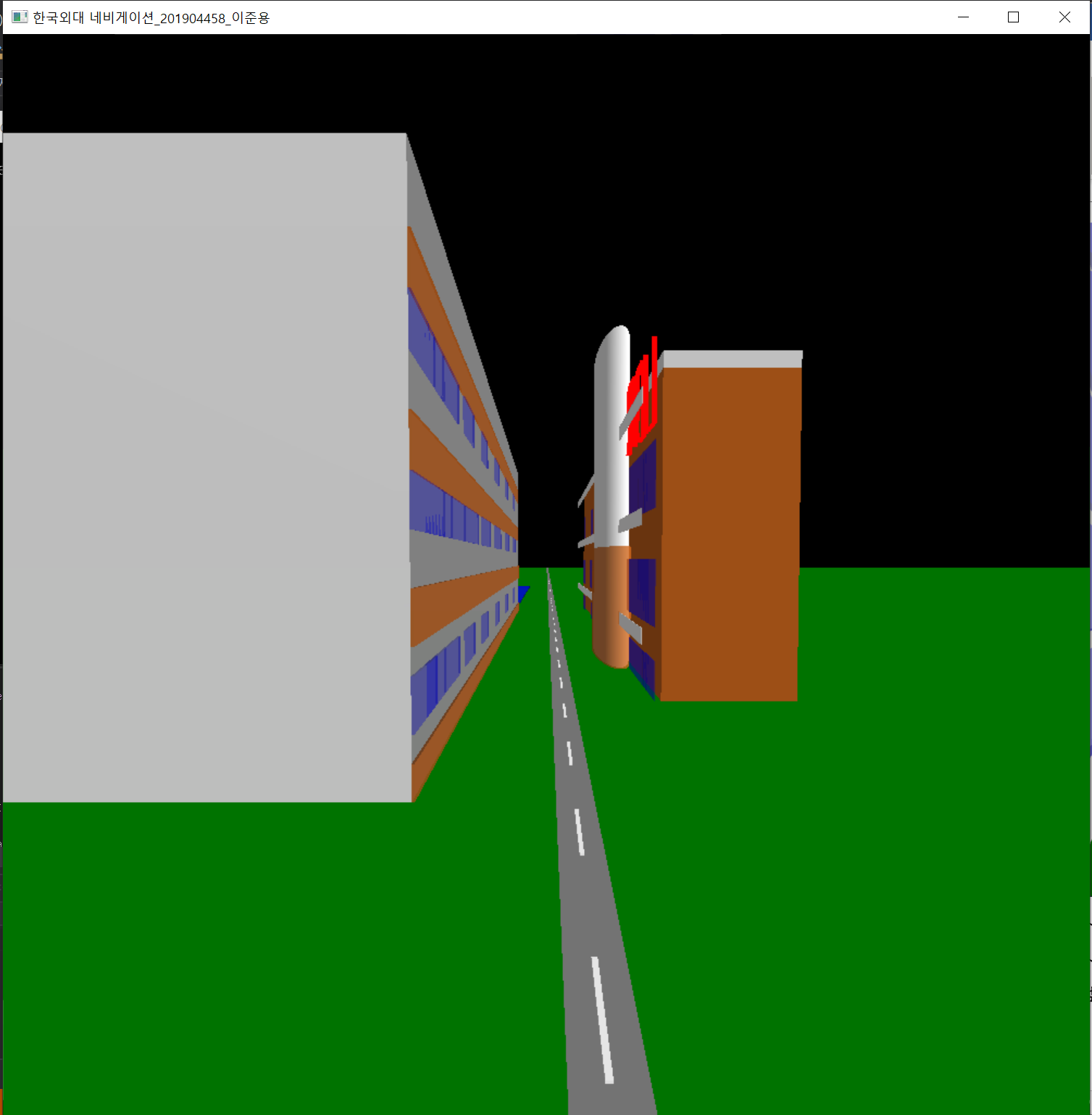
<프로그램 시작 후 키보드로 ‘w‘전진 후 자연대 건물(왼쪽)>



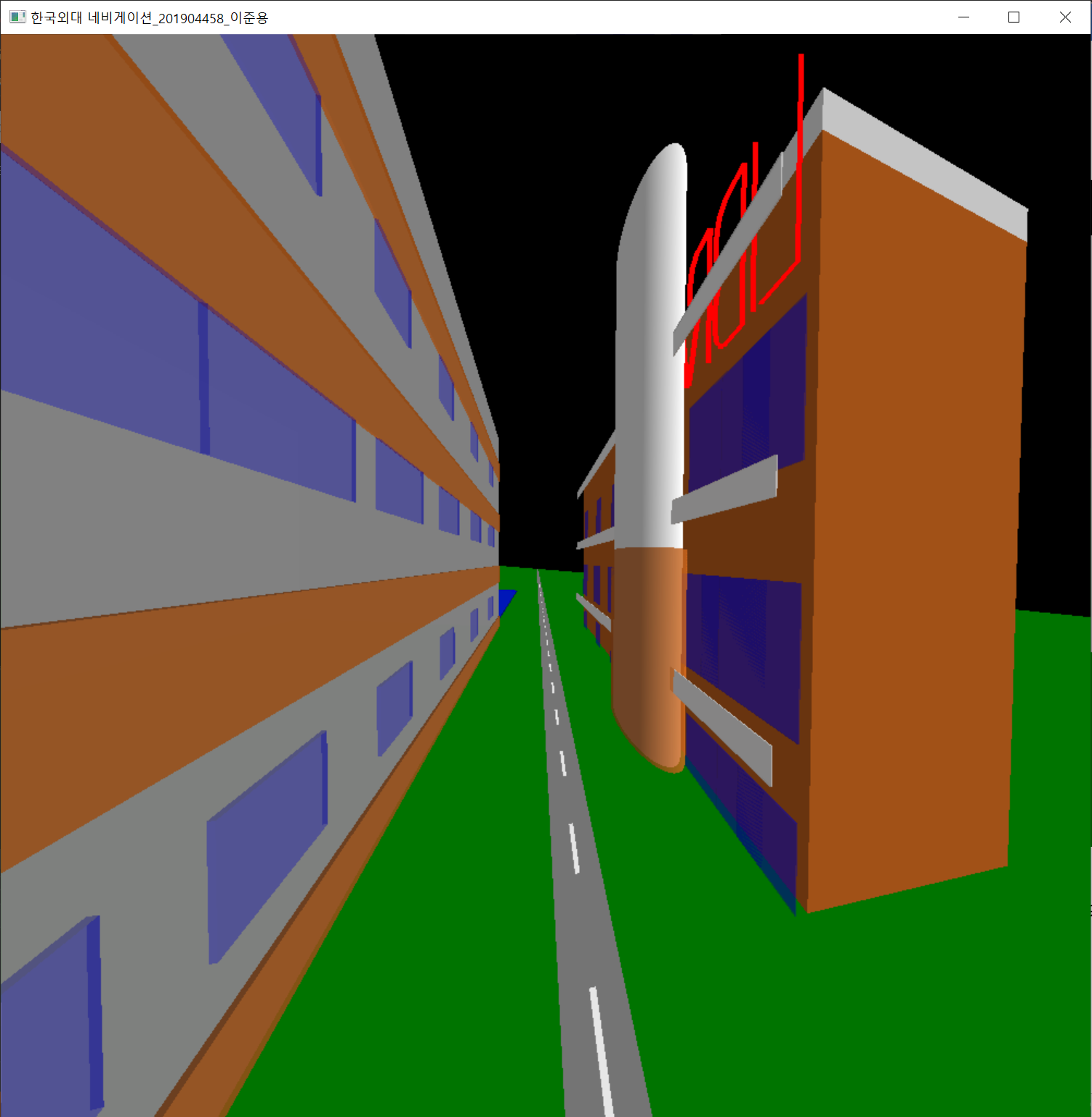
<프로그램 시작 후 키보드로 ‘w‘전진 후 기숙사 건물(오른쪽)>



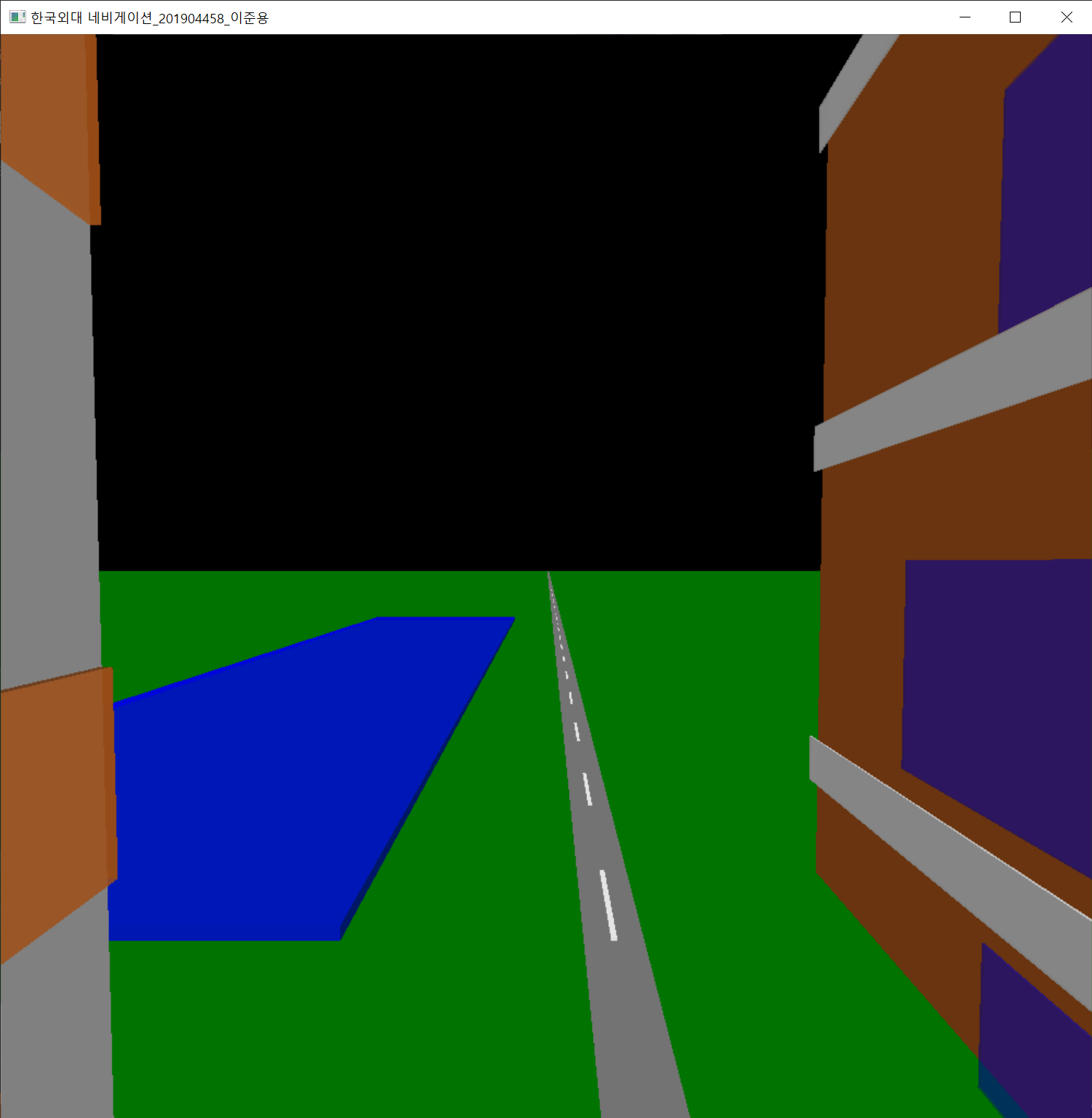
<프로그램 시작 후 키보드로 ‘w‘전진 후 공학관 건물(왼쪽)>



<프로그램 시작 후 키보드로 ‘w‘전진 후 도서관 건물(오른쪽)>



<프로그램 시작 후 키보드로 ‘w‘전진 후 명수당>



* **소감**

한국외대로 편입하고 컴퓨터 그래픽스 수업을 수강하면서OpenGL의 개념과 명령어들을 활용하는 법과 종류들과 함수들의 인자들 등 많은 것들을 습득하고 활용할 수 있게 되었습니다. 특히 glVertex3f 사용법, gluLokkAt함수, glColor3f, glColor4f, glBegin(GL\_POLYGON), glPushMatrix, glPopMatrix함수들을 이번 과제를 하면서 자유롭게 다룰 수 있게 되었습니다.

한국외대 네비게이션 과제를 하면서 편입이후로 비대면 수업 때문에 대학교를 한번도 가보지 못했습니다. 그렇기에 대학교 건물을 사진만 보고 코딩해야 된다는 점에서 어려움이 있었습니다. 그렇기 때문에 더더욱 사진과 비슷하게 만들려고 노력했습니다. 2학기에 대면 수업이 된다면 제가 코딩했던 건물들의 외관과 강의실들을 실제로 보고 싶습니다.