

**숙제1 당구공 애니메이션**

**과목명 컴퓨터그래픽스**

**담당교수 김상철교수님**

**제출일 20210404**

**전공 컴퓨터전자시스템**

**학번 201904458**

**이름 이준용**

* **소스 코드**

/\* 제목 : 한국외대 컴퓨터 그래픽스 과제

프로그램 : 당구공 애니메이션

학 과 : 컴퓨터 전자시스템 전공

학 번 : 201904458

이 름 : 이준용 \*/

#define GL\_SILENCE\_DEPRECATION

#include <GL/glut.h>

#include <GL/gl.h>

#include <GL/glu.h>

#include <math.h>

#include <stdlib.h>

#include <time.h>

GLfloat DeltaX = 0.0; // 수구의 키보드로 이동x 축

GLfloat DeltaY = 0.0; // 수구의 키보드로 이동 y 축

float billiard; // 당구공의 반지름

int value; //당구 게임 시작 변수

int bump; //두 공이 충돌하는 횟수

float hit\_x, hit\_y; // 수구의 x좌표값, 적구 y좌표값

float hit\_moveX, hit\_moveY; //수구 x축 이동 속도, 수구 y축 이동 속도

float red\_x, red\_y; //적구 x좌표값, 적구 y좌표값

float red\_moveX, red\_moveY; //적구 x축 이동 속도, 적구 y축 이동 속도

int cou\_hit, cou\_red; // 수구와 적구가 벽에 부딫힌 횟수

float cou\_reduce; // 당구공 쿠션시 감소하는 속도량

/\*당구대는 x축좌표가 -400 ~ 400이고, y축 좌표가 -500 ~ 500인 좌표평면으로 설정하였습니다.\*/

void initValue() { // 변수 초기화 함수

billiard = 20; // 당구공(billiard ball)의 반지름

/\*(조건2)\*/

while (sqrt(pow(hit\_x - red\_x, 2) + pow(hit\_y - red\_y, 2)) < 700) {// 수구와 적구를 각각 당구대의 아래쪽과 위쪽의 랜덤한 위치에 생성시키고,

//pow(n,p) n의 p제곱근 //이 둘 사이의 거리가 700이상이 되도록 배치한다.

hit\_x = ((rand() % 800) - 400); // 수구의 처음 위치는 당구대 아래쪽 (y축값 -500 ~ 0) 의 랜덤 위치 (x축 값 -400 ~ 400)으로 설정

hit\_y = -1 \* (rand() % 500);

// rand()함수 사용

red\_x = ((rand() % 800) - 400); // 적구의 처음 위치는 당구대 위쪽 (y축 값 0 ~ 500)의 랜덤 위치 (x축 값 -400 ~ 400)으로 설정

red\_y = ((rand() % 500));

}

bump = 0;

value = 0;

red\_moveX = 0; // 적구는 수구와 부딪히기 전까지 움직이지 않음

red\_moveY = 0;

cou\_hit = 0;

cou\_red = 0;

cou\_reduce = 1.3;

}

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

switch (KeyPressed) {

case 'a': //왼쪽

DeltaX -= 0.03;

break;

case 'd': // 오른쪽

DeltaX += 0.03;

break;

case 'w': // 위쪽

DeltaY += 0.03;

break;

case 's': //아래쪽

DeltaY -= 0.03;

break;

case 'p': // 당구 게임 시작

value = 1;

break;

case 'q': // 프로그램 종료

exit(0);

break;

}

}

void DrawCircle\_hit(float hx, float hy, float r) // 수구를 그리는 함수.

{

float x1, y1, x2, y2;

float angle;

double radius = r;

x1 = hx; // 현재 수구의 x좌표 받아옴

y1 = hy; // 현재 수구의 y좌표 받아옴

glColor3f(1.0, 1.0, 1.0); // 수구의 색깔: 흰색.

glBegin(GL\_TRIANGLE\_FAN);

glVertex2f(x1, y1); // x1, y1은 원의 중심좌표

for (angle = 0.01f; angle <= 2 \* 3.14f + 0.2; angle += 0.1)

{

x2 = x1 + sin(angle) \* radius;

y2 = y1 + cos(angle) \* radius;

glVertex2f(x2, y2); // x2, y2는 원의 중심을 기준으로 반지름 만큼 원주를 그림(원주 = 2 \* PI \* r )

}

glEnd();

}

void DrawCircle\_red(float rx, float ry, float r) { // 적구를 그리는 함수.

float x1, y1, x2, y2;

float angle;

double radius = r;

x1 = rx;

y1 = ry;

glColor3f(1.0, 0.0, 0.0); // 적구의 색깔: 빨간색

glBegin(GL\_TRIANGLE\_FAN);

glVertex2f(x1, y1); // x1, y1은 원의 중심좌표

for (angle = 0.01f; angle <= 2 \* 3.14f + 0.2; angle += 0.1)

{

x2 = x1 + sin(angle) \* radius;

y2 = y1 + cos(angle) \* radius;

glVertex2f(x2, y2); // x2, y2는 원의 중심을 기준으로 반지름 만큼 원주를 그림(원주 = 2 \* PI \* r )

}

glEnd();

}

void MyTimer(int Value) {

glutPostRedisplay(); //윈도우를 다시 그도록 호출하는 함수

glutTimerFunc(10, MyTimer, 1); // 10ms 이후 반복해서 호출

}

void MyAnimation() { // 당구공 애니메이션 구현 함수

glViewport(0, 0, 800, 1000); // 당구대의 크키를 윈도우의 (0, 0)좌표부터 너비 800, 높이 1000으로 설정.

glClear(GL\_COLOR\_BUFFER\_BIT);

DrawCircle\_hit(hit\_x + hit\_moveX, hit\_y + hit\_moveY, billiard); // 현재 당구공의 위치 + (x, y축 이동값)으로 다음 위치에 수구, 적구를 그림.

DrawCircle\_red(red\_x + red\_moveX, red\_y + red\_moveY, billiard); // billiard = 당구공 반지름

if (value == 1) { // (조건7) 키보드로 'p'를 받을 경우 수구가 적구를 치는 부분

if (sqrt(pow(hit\_x - red\_x, 2) + pow(hit\_y - red\_y, 2)) <= 2 \* billiard) { // 수구의 적구의 길이가 당구공의 반지를 \* 2 보다 작을 때 (두 공이 부딪혔을 때)

if (bump < 2) { // 두 공이 한번도 부딪혔던 적이 없다면 // 제곱근에 루트한 이유는 좌표값이 float형 이므로 값이 너무 작을 경우 프로그램이 0으로 인식방지

hit\_moveX = (0.5 \* hit\_x - (-1) \* (sqrt(3) / 2) \* hit\_y) / 20; // (조건4)

hit\_moveY = ((-1) \* (sqrt(3) / 2) \* hit\_x + 0.5 \* hit\_y) / 20; // 캐롬 당구의 분리각 법칙에 의한 "부딪힌 후"의 각 당구공의 이동값 업데이트.

red\_moveX = ((sqrt(3) / 2) \* red\_x - 0.5 \* red\_y) / 20;

red\_moveY = (0.5 \* red\_x + (sqrt(3) / 2) \* red\_y) / 20;

bump++; // 쿠션 횟수 + 1

}

else { // (조건1)두 공이 이미 한번이상 부딪힌 경우

hit\_moveX = 0; // 더 이상 이동하지 않고 공이 멈추도록 이동값을 0으로 설정.

hit\_moveY = 0;

red\_moveX = 0;

red\_moveY = 0;

}

}

hit\_x += hit\_moveX; // 위의 정의되어 있는 이동값 만큼 수구, 적구를 이동시킴

hit\_y += hit\_moveY;

red\_x += red\_moveX;

red\_y += red\_moveY;

/\*(조건5)\*/

if ((hit\_moveX < 0 && hit\_x <= -400) || (hit\_moveX > 0 && hit\_x >= 400)) { // 수구가 당구대의 양 옆면에 부딪혔을 경우 ( 쿠션 )

hit\_moveX \*= -1; // 기존 이동값에 -1을 곱해줌

hit\_moveX /= cou\_reduce;

hit\_moveY /= cou\_reduce; // 쿠션시 감속

cou\_hit++;

}

if ((hit\_moveY < 0 && hit\_y <= -500) || (hit\_moveY > 0 && hit\_y >= 500)) { // 수구가 당구대의 위, 아랫면에 부딪혔을 경우

hit\_moveY \*= -1;

hit\_moveX /= cou\_reduce;

hit\_moveY /= cou\_reduce; // 쿠션시 감속

cou\_hit++;

}

if ((red\_moveX < 0 && red\_x <= -400) || (red\_moveX > 0 && red\_x >= 400)) { // 적구가 당구대의 양 옆면에 부딪혔을 경우

red\_moveX \*= -1;

red\_moveX /= cou\_reduce;

red\_moveY /= cou\_reduce; // 쿠션시 감속

cou\_red++;

}

if ((red\_moveY < 0 && red\_y <= -500) || (red\_moveY > 0 && red\_y >= 500)) { // 적구가 당구대의 위, 아랫면에 부딪혔을 경우

red\_moveY \*= -1;

red\_moveX /= cou\_reduce;

red\_moveY /= cou\_reduce; // 쿠션시 감속

cou\_red++;

}

/\*(조건1)\*/

if (cou\_hit >= 2) { // 수구 2쿠션시 서서히 정지

hit\_moveX /= 1.03;

hit\_moveY /= 1.03;

}

if (cou\_red >= 3) { // 적구 3쿠션시 서서히 정지

red\_moveX /= 1.03;

red\_moveY /= 1.03;

}

}

else {

hit\_x += DeltaX;

hit\_y += DeltaY;

/\* 수구가 적구를 향해 직진 적구의 x축 값이 수구보다 작을 시,

수구의 x축값을 감소시키며 접근해야하므로\*/

if (hit\_x > red\_x) { // 수구의 위치가 변경 될때마다 수구가 적구로 무조건 향하는 방향을 계속 연산함 => 키보드에서 'p'(play)누를 경우

// 적구를 무조건 맞추기 위함.

hit\_moveX = -1 \* fabs(hit\_x - red\_x) / 20; // 적구의 x 축 값이 수구보다 작으면 x축 이동값을 음수로 설정

}

else {

hit\_moveX = fabs(hit\_x - red\_x) / 20;

}

hit\_moveY = fabs(hit\_y - red\_y) / 20;

if (((hit\_x + DeltaX) <= -400) || ((hit\_x + DeltaX) >= 400)) { // 수구를 키보드로 위치변경하다가 당구대의 옆면에 부딪혔을 경우

DeltaX = 0;

}

if (((hit\_y + DeltaY) <= -400) || ((hit\_y + DeltaY) >= 400)) { // 수구를 키보드로 위치변경하다가 당구대의 옆면에 부딪혔을 경우

DeltaY = 0;

}

if ((hit\_moveX < 0 && hit\_x <= -400) || (hit\_moveX > 0 && hit\_x >= 400)) { // 수구가 당구대의 양 옆면에 부딪혔을 경우 ( 쿠션 )

hit\_moveX \*= -1; // 기존 이동값에 -1을 곱해줌

hit\_moveX /= cou\_reduce;

hit\_moveY /= cou\_reduce; // 쿠션시 감속

cou\_hit++;

}

if ((hit\_moveY < 0 && hit\_y <= -500) || (hit\_moveY > 0 && hit\_y >= 500)) { // 수구가 당구대의 위, 아랫면에 부딪혔을 경우

hit\_moveY \*= -1;

hit\_moveX /= cou\_reduce;

hit\_moveY /= cou\_reduce; // 쿠션시 감속

cou\_hit++;

}

}

glutSwapBuffers(); // 두개 버퍼를 바꿈

}

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

srand(static\_cast<unsigned int>(time(nullptr))); // 랜덤 위치를 선정하기위한 srand

initValue(); // 초기값 설정 함수로. 사용할 변수 초기화

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE); // 애니메이션을 사용 할때 더블 버퍼링을 사용하는게 좋다. 더블버퍼링 안하면 ( 깜빡이는현상 발생)

//전면 버퍼를 화면에 뿌리고, 애니메이션은

// 그위에 뿌리는 형식

glutInitWindowSize(800, 1000); // 윈도우의 사이즈를 800, 1000으로 설정

glutInitWindowPosition(400, 0); // 프로그램 실행 시킬 경우 나타나는 화면을 가운데로 위치시킴.

glutCreateWindow("OpenGl\_숙제1 당구공 애니메이션\_201904458\_이준용"); //새로운 윈도우 상단에 title 바에 나타남

glClearColor(0.05, 0.45, 0.0, 1.0); //float형 색상 값이 들어감 인자는 순서대로 RGB 마지막 인자는 알파(투명도) (1.0,1.0,1.0,1.0)으로 바꾸면 배경화면이 흰색으로 바뀜

// glClearColor 함수가 배경화면 설정하는 함수임.

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-400.0, 400.0, -500.0, 500.0, -1.0, 1.0); // 윈도우 크기에 따른 도형의 왜곡을 방지하기위해 뷰포트 크기에 맞춰 x, y 좌표 최대, 최소값을 argument로 이용하여 비율을 맞춰줌.

glutKeyboardFunc(MyKeyboard);

glutDisplayFunc(MyAnimation); // MyAnimation에 정의한 내용 실행.

glutTimerFunc(40, MyTimer, 1);

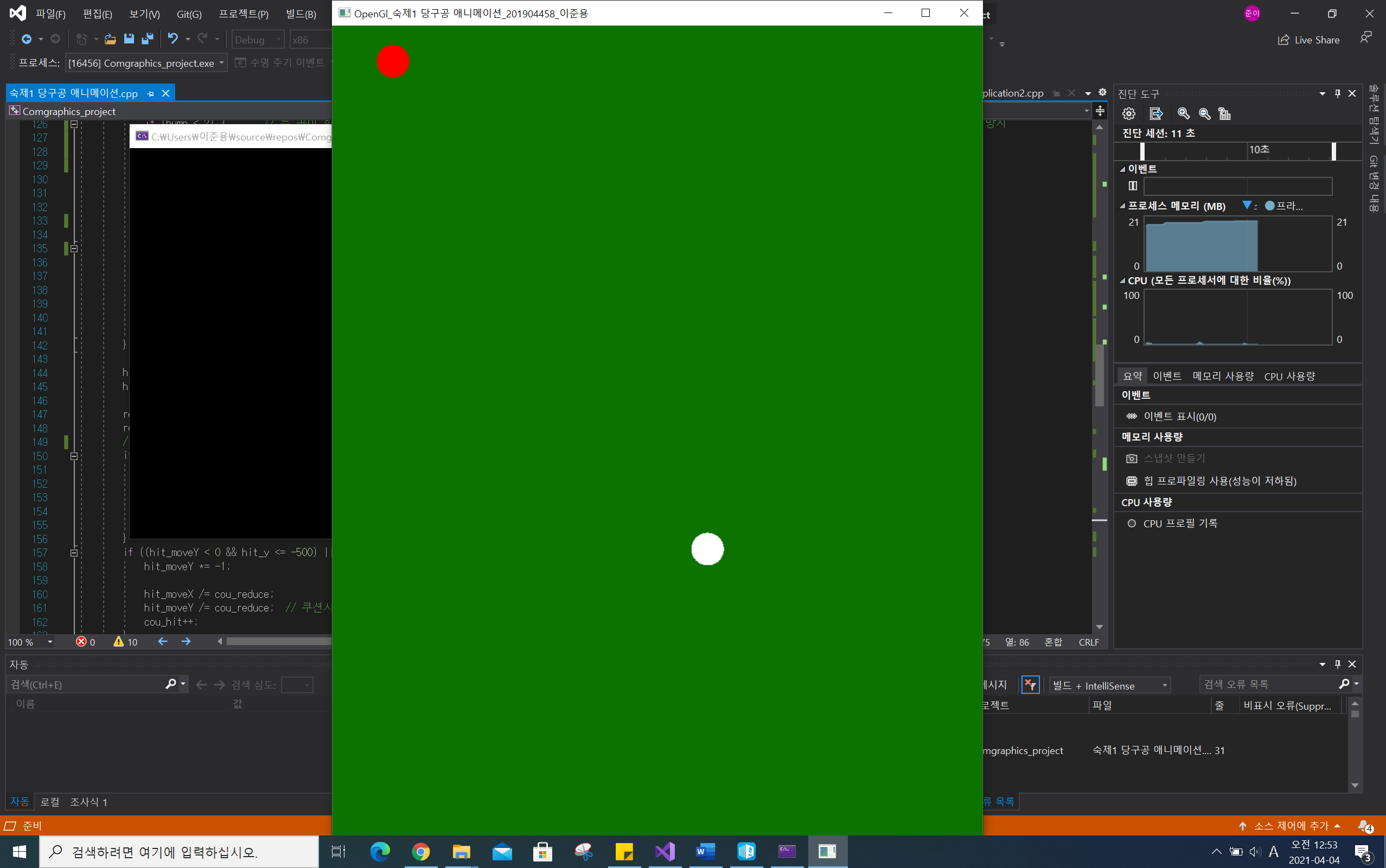
glutMainLoop(); // 끝났을 경우 무조건 glutMainLoop() 실행

return 0;

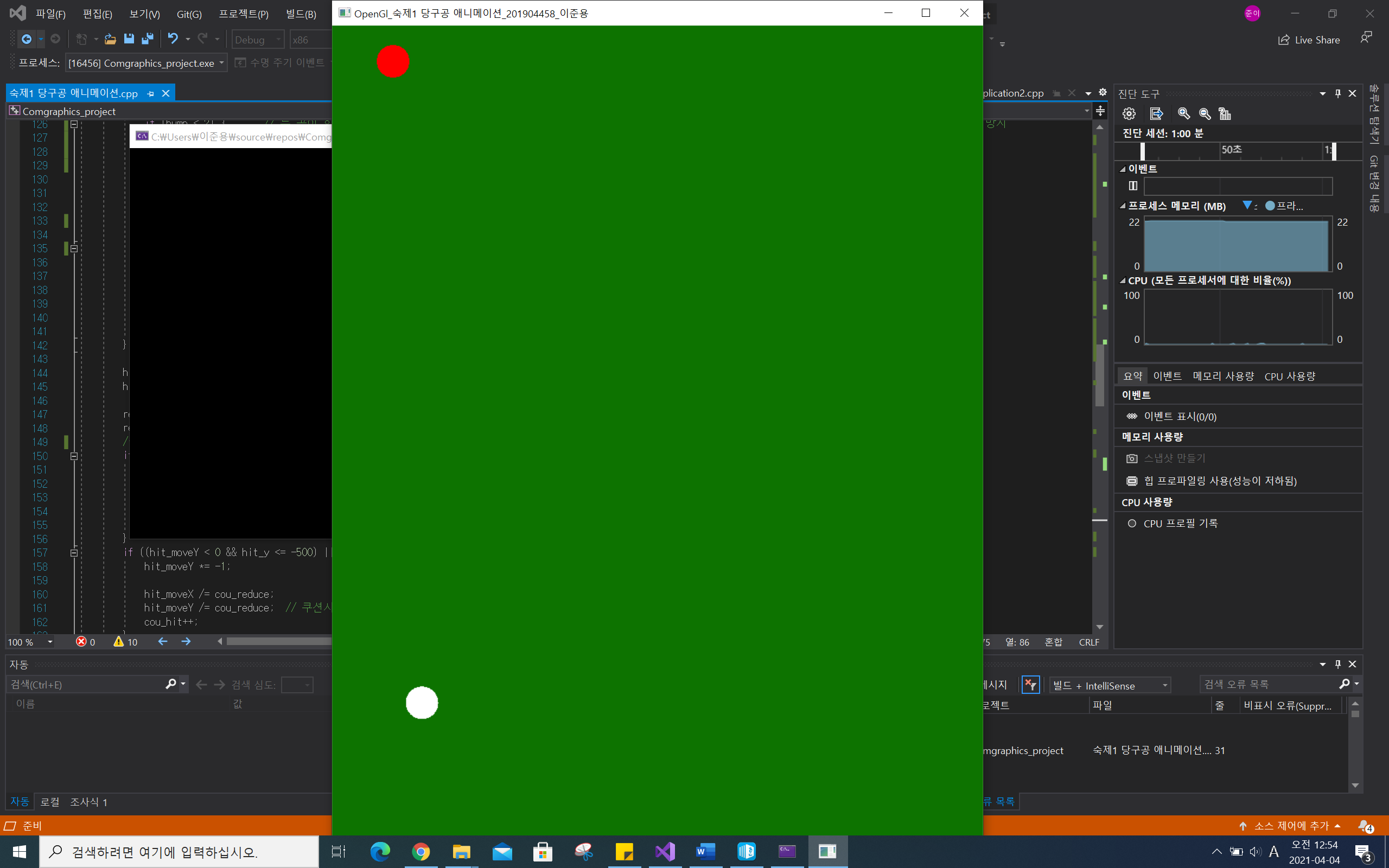
}

* **화면 dump ★ 수구방향키(awsd) 당구플레이(p) 종료(q) ★**

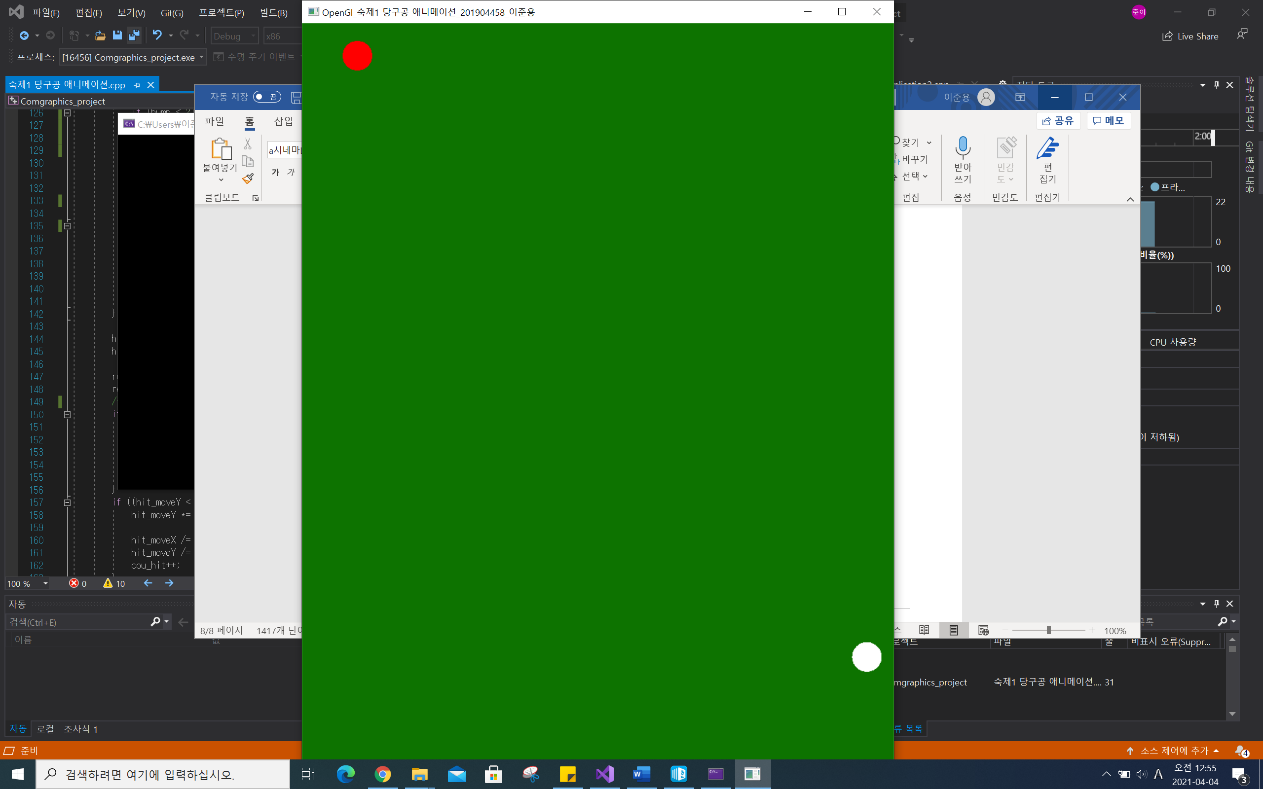
<프로그램 시작 화면>



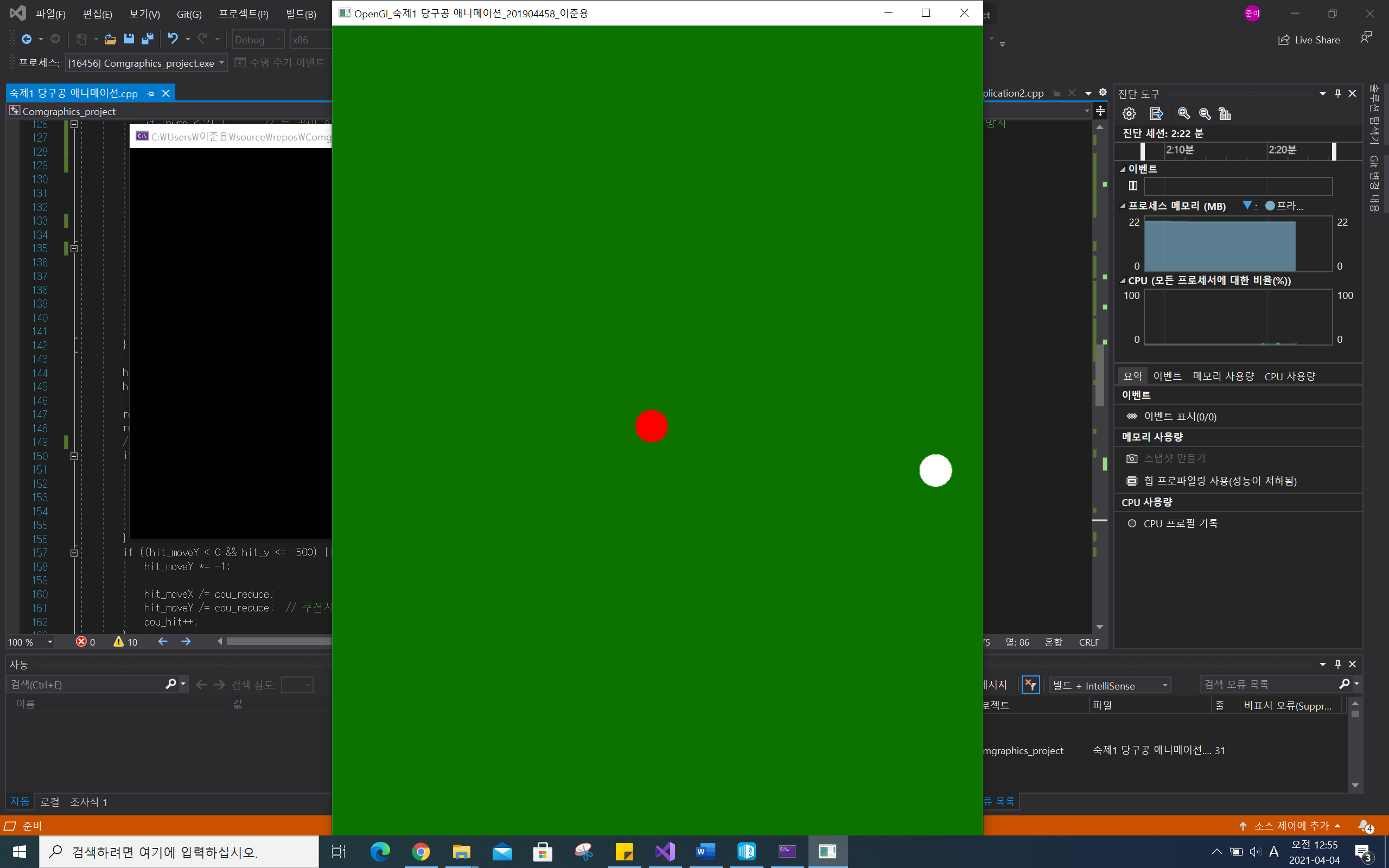
<프로그램 시작 후 키보드로 수구 움직임1>



<키보드로 수구움직임1 후 키보드로 수구 움직임 2>



<키보드에서 ‘**p**’누르면 당구공 애니메이션 시작 후 종료 화면>



* **소감**

평소에 친구들과 당구를 치는 것을 좋아합니다. 그렇기에 더욱더 당구공 애니메이션은 저에게 재미있게 다가왔던 것 같습니다. 당구공 애니메이션 프로그램 과제를 하면서 OpenGl의 기초적인 함수와 메소드를 많이 알게 되었습니다. 특히 캐롬 당구의 분리각 90도 법칙을 어떻게 적용하는가에 대해 고민을 많이 하였습니다. 당구의 원리를 프로그램을 짜보면서 수학적인 원리가 많이 들어간다는 것이 흥미로웠습니다.

여러 번의 시행착오의 끝에 프로그래밍 결과물이 시각적으로 전달되다 보니 다른 프로그래밍에 비해 뿌듯함도 두배 이상이 되었습니다. 게임프로그래밍에서는 아직은 초보 단계이지만 이번 과제를 통해서 전반적인 게임 그래픽의 원리의 이해와 더불어 전반적인 OpenGl컴퓨터 그래픽스의 기초에 한발자국 다가가는 그런 시간이었습니다. 특히 웹 프로그래밍도 OpenGl과는 비슷하게 프로그램의 결과물이 실제로 시각적으로 전달하는 것이 비슷하지만 게임과 같이 사용자가 키보드와 마우스로 프로그램 실행화면에서 직접 변화를 주는 부분에서 간접적으로 게임의 한부분을 경험하는 것 과 같아 특별했던 것 같습니다.

당구공 프로그램과 보고서를 끝내고 실제 게임을 플레이하는 것이 아닌 게임 프로그래밍 제작에 대해 어떤 것들이 있을까 흥미를 가지게 되는 그런 시간이었습니다.

감사합니다.