**소스코드**

완성여부: 완성

1. Table.h // 테이블을 그려주는 요소들을 정의한 헤더입니다
2. #include<GL/glut.h>
3. #include<math.h>
4. GLfloat MyVertex[][3] = {
5. { -0.25, -0.25, 0.25 },//v0
6. { -0.25, 0.25, 0.25 },//v1
7. { 0.25, 0.25, 0.25 },//v2
8. { 0.25, -0.25, 0.25 },//v3
9. { -0.25, -0.25, -0.25 },//v4
10. { -0.25, 0.25, -0.25 },//v5
11. { 0.25, 0.25, -0.25 },//v6
12. { 0.25, -0.25, -0.25 },//v7
13. };//탁자그리는데 초기좌표]
14. GLubyte MyQuadList[] = { 0, 3, 2, 1, 2, 3, 7, 6, 0, 4, 7, 3, 1, 2, 6, 5, 4, 5, 6, 7, 0, 1, 5, 4 };//탁자그리는데 쓰이는리스트들
15. void makeTable(){
16. glEnableClientState(GL\_VERTEX\_ARRAY);
17. glVertexPointer(3, GL\_FLOAT, 0, MyVertex);
18. glMatrixMode(GL\_MODELVIEW);
20. glPushMatrix();
21. glScalef(3, 0.5, 3);//상판이다
22. for (int i = 0; i<6; i++)
23. glDrawElements(GL\_LINE\_LOOP, 4, GL\_UNSIGNED\_BYTE, &MyQuadList[4 \* i]);
24. glPopMatrix();
25. glPushMatrix();//상다리 1
26. glScalef(0.5, 1, 0.5);
27. glTranslatef(-1.0, -0.37, 1.1);
29. for (int i = 0; i<6; i++)
30. glDrawElements(GL\_LINE\_LOOP, 4, GL\_UNSIGNED\_BYTE, &MyQuadList[4 \* i]);
31. glPopMatrix();
32. glPushMatrix();//상다리 2
33. glScalef(0.5, 1, 0.5);
34. glTranslatef(1, -0.37, 1.1);
35. for (int i = 0; i<6; i++)
36. glDrawElements(GL\_LINE\_LOOP, 4, GL\_UNSIGNED\_BYTE, &MyQuadList[4 \* i]);
37. glPopMatrix();
38. glPushMatrix(); //상다리3
39. glScalef(0.5, 1, 0.5);
40. glTranslatef(1, -0.37, -1.1);
41. for (int i = 0; i<6; i++)
42. glDrawElements(GL\_LINE\_LOOP, 4, GL\_UNSIGNED\_BYTE, &MyQuadList[4 \* i]);
43. glPopMatrix();
44. glPushMatrix();//상다리4
45. glScalef(0.5, 1, 0.5);
46. glTranslatef(-1.0, -0.37, -1.1);
47. for (int i = 0; i<6; i++)
48. glDrawElements(GL\_LINE\_LOOP, 4, GL\_UNSIGNED\_BYTE, &MyQuadList[4 \* i]);
49. glPopMatrix();
50. }

**2.Objects.h// 탁자위에 올라갈 객체들을 정의한 헤더입니다**

#include<GL/glut.h>

#include<math.h>

float points[][2]={{0.3,0.0},{0.25,0.4},{0.5,0.1},{0.4,0.65}};//컵그리는데 초기좌표

float cupData[760][3];//컵그리기

float circleData[200][3];//원기둥 원

float lineData[200][3];//원기둥 기둥

float quadData[400][3];//구 그리기

int k = 0;

int l = 0;

int c=0;

float radius = 0.5;//원기둥 그리는데 반지름

const double ToRadian = 3.141592 / 180.0;//라디안값

void computeCoord(){ //구 그리기

double phi, theta;

for (phi = -80; phi <= 80.0; phi += 20.0){//위도자르기 위도로 자를떄는 경도가 한번에 잘리기 때문에 180도만 돌리면 된다

double phiR = phi\* ToRadian;

double phiR20 = (phi + 20.0)\*ToRadian;//20도 위의 면

for (theta = -180; theta <= 180.0; theta += 20){//경도자르기

double thetaR = theta\*ToRadian;

double x1 = sin(thetaR)\*cos(phiR);

double y1 = cos(thetaR)\*cos(phiR);

double z1 = sin(phiR);

quadData[k][0] = x1;

quadData[k][1] = y1;

quadData[k][2] = z1;

k++;

x1 = sin(thetaR)\*cos(phiR20);

y1 = cos(thetaR)\*cos(phiR20);

z1 = sin(phiR20);

quadData[k][0] = x1;

quadData[k][1] = y1;

quadData[k][2] = z1;

k++;

}

}k = 0;//다음 객체그리는데 쓰기위해 k값 초기화

}

void cylinder()//원기둥 그리기

{

double phi;

double y;

for (y = 0.5; y >= -0.5; y -= 1.0)//원그리기

{

circleData[k][0] = 0.0;

circleData[k][1] = y;

circleData[k][2] = 0.0;

k++;

for (phi = 0.0; phi <= 360.0; phi += 20.0)

{

double phiR = phi \* ToRadian;

double x = radius \* cos(phiR);

double z = radius \* sin(phiR);

circleData[k][0] = x;

circleData[k][1] = y;

circleData[k][2] = z;

k++;

}

}

for (phi = 0.0; phi <= 360.0; phi += 20.0)//원기둥 그리기

{

double phiR = phi \* ToRadian;

double x = radius \* cos(phiR);

double z = radius \* sin(phiR);

lineData[l][0] = x;

lineData[l][1] = 0.5;

lineData[l][2] = z;

l++;

lineData[l][0] = x;

lineData[l][1] = -0.5;

lineData[l][2] = z;

l++;

}

}

void cup(){//컵그리기

double theta;

for(float alpha=0.0f;alpha<=1.0f;alpha+=0.05){

float px=(1-alpha)\*(1-alpha)\*(1-alpha)\*points[0][0]+3\*(1-alpha)\*(1-alpha)\*alpha\*points[1][0]+3\*(1-alpha)\*alpha\*alpha\*points[2][0]+alpha\*alpha\*alpha\*points[3][0];//베지의 곡선공식

float py=(1-alpha)\*(1-alpha)\*(1-alpha)\*points[0][1]+3\*(1-alpha)\*(1-alpha)\*alpha\*points[1][1]+3\*(1-alpha)\*alpha\*alpha\*points[2][1]+alpha\*alpha\*alpha\*points[3][1];

float px1=(1-(alpha+0.05))\*(1-(alpha+0.05))\*(1-(alpha+0.05))\*points[0][0]+3\*(1-(alpha+0.05))\*(1-(alpha+0.05))\*(alpha+0.05)\*points[1][0]+3\*(1-(alpha+0.05))\*(alpha+0.05)\*(alpha+0.05)\*points[2][0]+(alpha+0.05)\*(alpha+0.05)\*(alpha+0.05)\*points[3][0];//0.05위의 면(구와 같은원리)

float py1=(1-(alpha+0.05))\*(1-(alpha+0.05))\*(1-(alpha+0.05))\*points[0][1]+3\*(1-(alpha+0.05))\*(1-(alpha+0.05))\*(alpha+0.05)\*points[1][1]+3\*(1-(alpha+0.05))\*(alpha+0.05)\*(alpha+0.05)\*points[2][1]+(alpha+0.05)\*(alpha+0.05)\*(alpha+0.05)\*points[3][1];

for (theta = -180; theta <= 180.0; theta += 20){

float radian = ToRadian\*theta;

cupData[c][0] = px\*sin(radian);

cupData[c][1] = py;

cupData[c][2] = px\*cos(radian);

c++;

cupData[c][0] = px1\*sin(radian);

cupData[c][1] = py1;

cupData[c][2] = px1\*cos(radian);

c++;

}

}

}

void drawObjects(){

glScalef(0.2, 0.2, 0.2);//구의 크기조정

glTranslatef(-1, 1.55, -1);//구의 위치조정

glBegin(GL\_QUAD\_STRIP);

for (int i = 0; i<304; i++){ //구그리기

glVertex3fv(quadData[i]);

}

glBegin(GL\_TRIANGLE\_FAN);

for(int i=0;i<36;i+=2){

glVertex3f(0,0,-1);

glVertex3fv(quadData[i]);

}

for(int i=267;i<304;i+=2){

glVertex3f(0,0,1);

glVertex3fv(quadData[i]);

}

glEnd();

glScalef(1.5, 1.5, 1.5);//원기둥의 크기조정

glTranslatef(0, -0.1, 1.8);//원기둥의 위치조정

glBegin(GL\_TRIANGLE\_FAN);

for (int idx = 0; idx < k / 2; idx++)

glVertex3fv(circleData[idx]);

glEnd();

glBegin(GL\_TRIANGLE\_FAN);

for (int idx = k / 2; idx < k; idx++)

glVertex3fv(circleData[idx]);

glEnd();

glBegin(GL\_LINES);

for (int idx = 0; idx < l; idx++)

glVertex3fv(lineData[idx]);

glEnd();

glScalef(0.7, 0.7, 0.7);//주전자의 크기조정

glTranslatef(2.5, 0, -2);//주전자의 크기조정

glutWireTeapot(1);//주전자 그리기

glScalef(1.5, 1.5, 1.5);

glTranslatef(0,-0.5,1.5);

glBegin(GL\_QUAD\_STRIP);//컵그리기

for (int i = 0; i<760; i++){

glVertex3fv(cupData[i]);

}

glEnd();

glBegin(GL\_TRIANGLE\_FAN);//컵 밑에 닫기

glVertex3f(0,0,0);

for(int i=0; i<40; i+=2) {

glVertex3fv(cupData[i]);

}

glEnd();

}

**3.Table.cpp//정의한 헤더들을 메인에서 불러와 그려줍니다**

#include<GL/glut.h>

#include<math.h>

#include"Table.h"

#include"Objects.h"

GLfloat gXAngle, gYAngle, gZAngle;//로테이트 값을 받을 변수

void myDisplay(){

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 0.0, 1.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glRotatef(gXAngle, 1.0, 0.0, 0.0);//키보드 누를떄마다 로테이트 각이 바뀜

glRotatef(gYAngle, 0.0, 1.0, 0.0);

glRotatef(gZAngle, 0.0, 0.0, 1.0);

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE);

makeTable();

drawObjects();

glutSwapBuffers();

}

void gSetPos(int dir)//키보드 값마다 로테이트값 바뀜

{

switch (dir)

{

case 1:

gYAngle += 10.0f;

break;

case 2:

gYAngle -= 10.0f;

break;

case 3:

gXAngle += 10.0f;

break;

case 4:

gXAngle -= 10.0f;

break;

case 5:

gZAngle += 10.0f;

break;

case 6:

gZAngle -= 10.0f;

break;

}

}

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

{

//q,w,e,a,s,d 을 누르면 테이블 전체가 회전이 된다

//z로 종료하기

switch (KeyPressed)

{

case 'Z':

case 'z':

exit(0);

break;

case 'W':

case 'w':

gSetPos(1);//y++

break;

case 'S':

case 's':

gSetPos(2);//y--

break;

case 'Q':

case 'q':

gSetPos(3);//x++

break;

case 'A':

case 'a':

gSetPos(4);//x--

break;

case 'E':

case 'e':

gSetPos(5);//z++

break;

case 'd':

case 'D':

gSetPos(6);//z--

break;

case 27:

exit(0);

break;

}

glutPostRedisplay();//다시 그려주기

}

void myInit(){//초기화라를 여기서해줌

glClearColor(1.0, 1.0, 1.0, 1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);

}

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

computeCoord();

cup();

cylinder();

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE);

glutInitWindowSize(300, 300);

glutInitWindowPosition(100, 100);

myInit();

glutCreateWindow("Table");

glutDisplayFunc(myDisplay);//display callback function

glLoadIdentity();

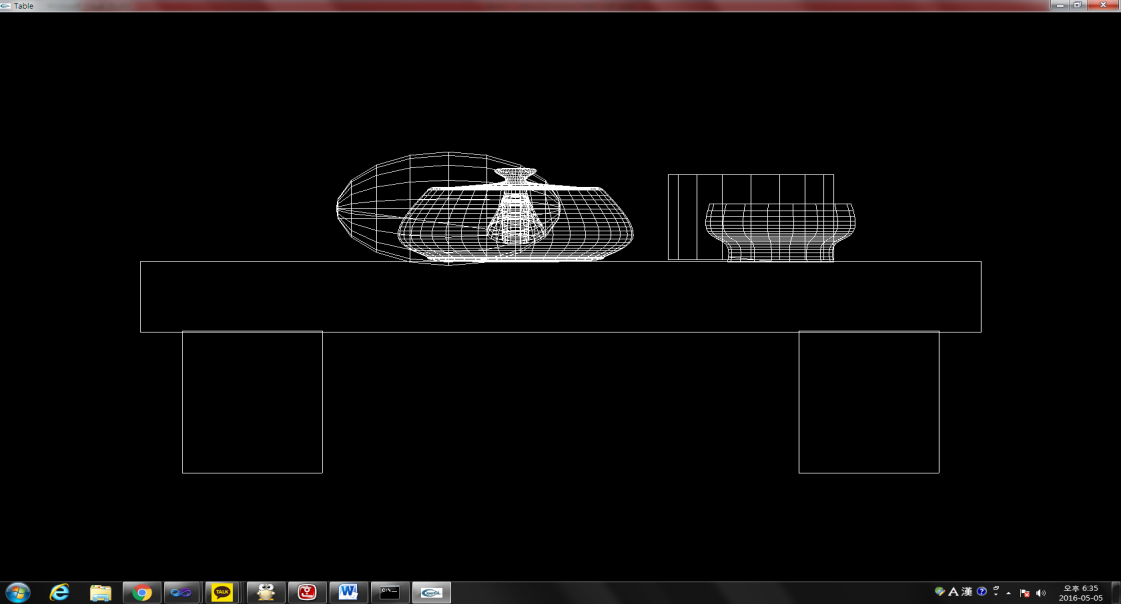
glutKeyboardFunc(MyKeyboard);

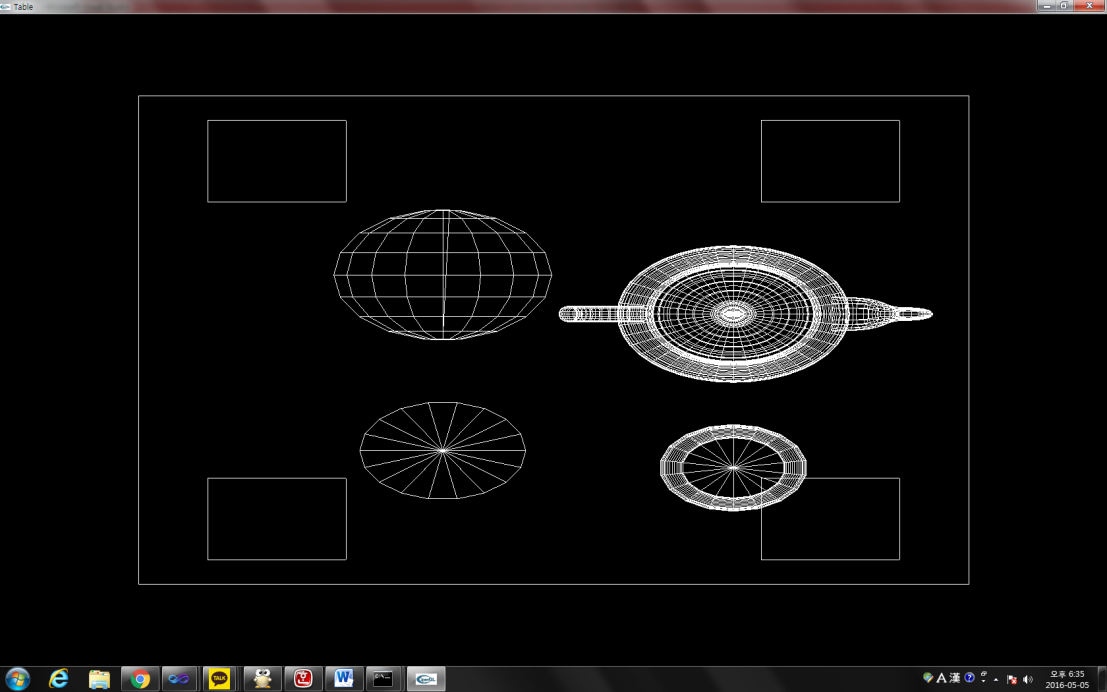
glutMainLoop();

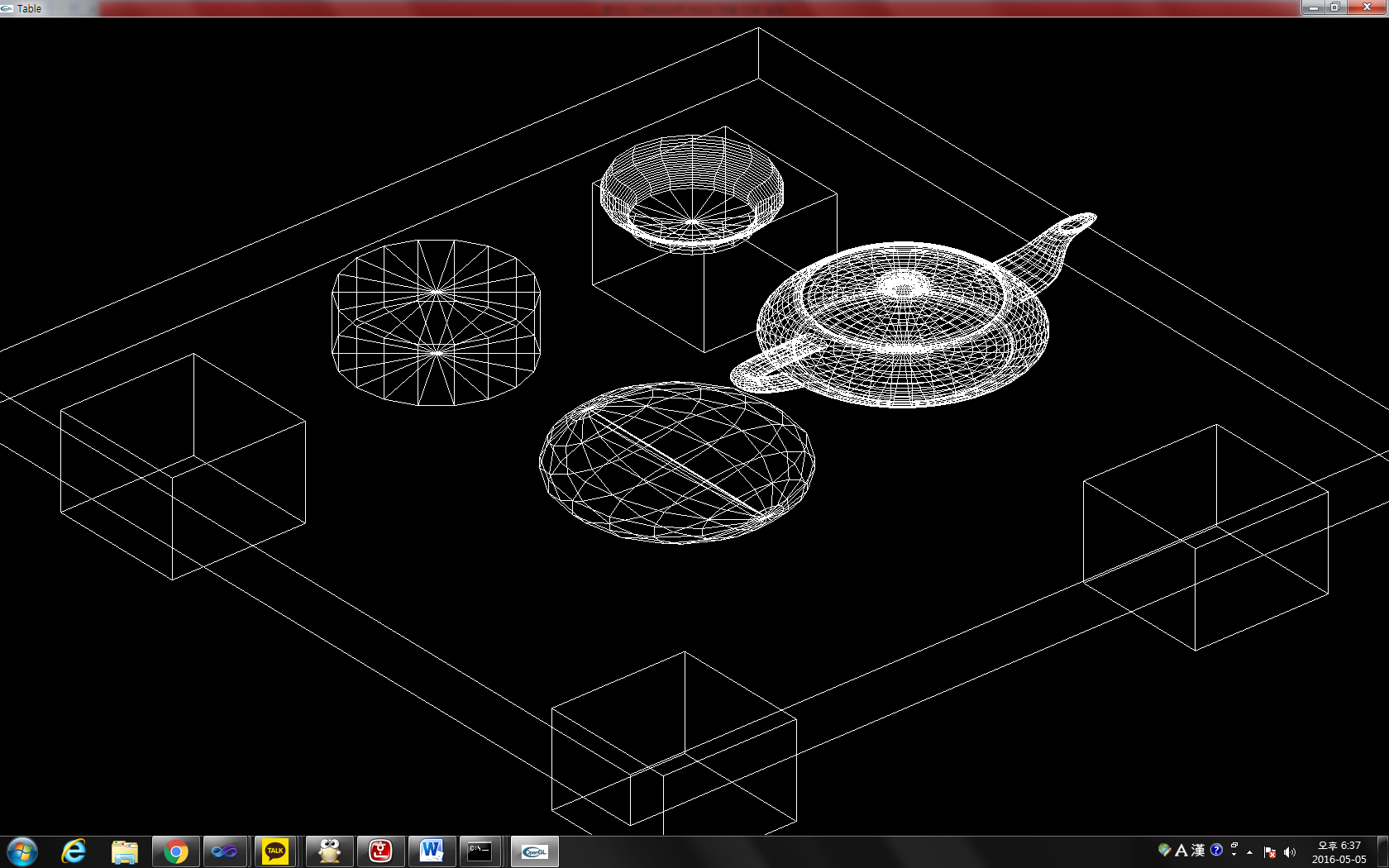
return 0;

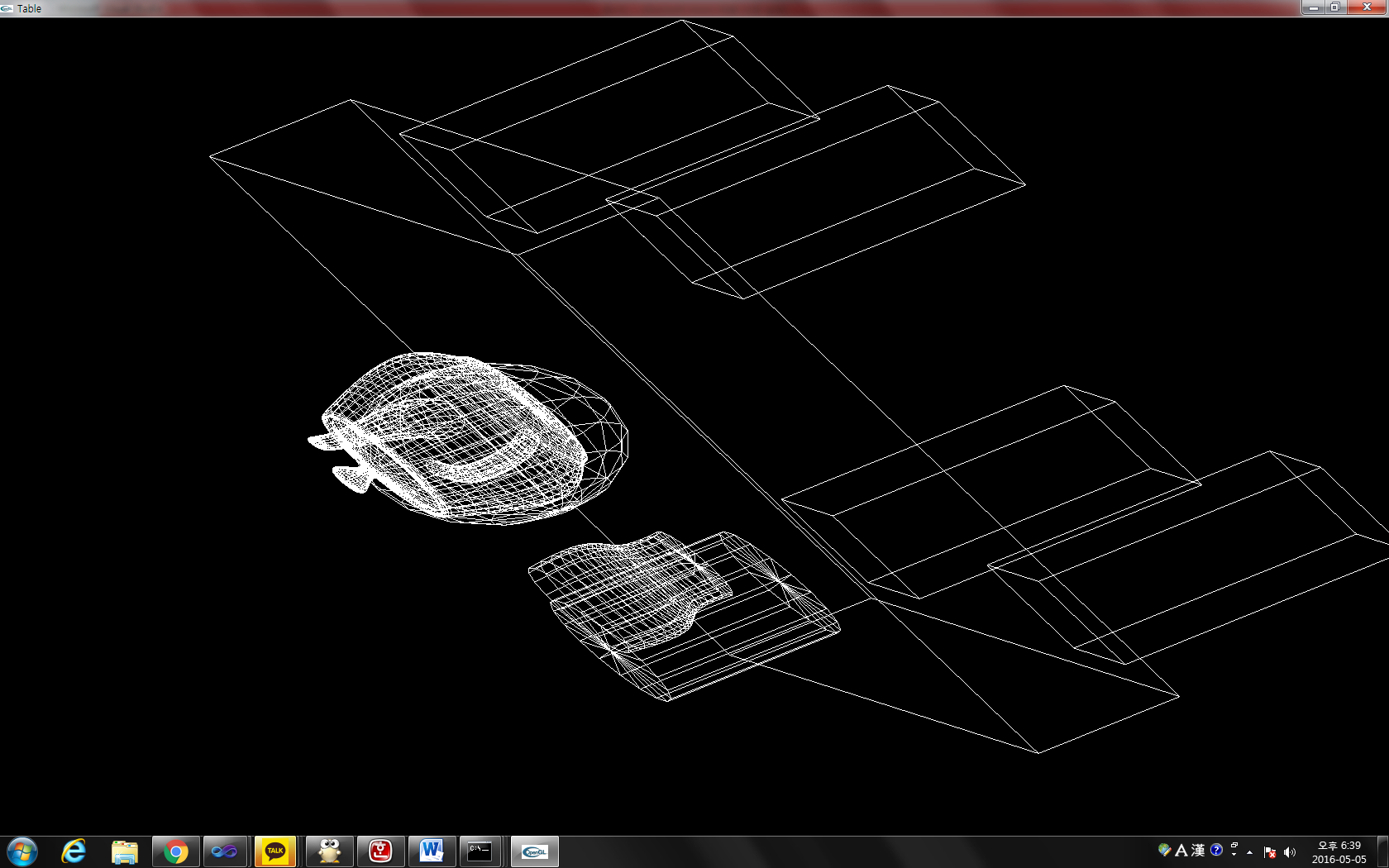
}

결과화면









사용된 알고리즘

1. 테이블 만들기

-pushMatrix, popMatrix

처음에 임의의 정육각형을 정의를 한 후 pushMatrix를 써주면 스택안에 들어가 저장이 된다 따라서 내가 translate 하고 scale을 해도 popMatrix를 통해 원래처음 저장한 정육각형을 다시 불러올 수 있다.

-translate , scale

Translate 를 하면 x, y, z에 주어진 값 만큼 더하기를 해준다 따라서 정육면체의 위치를 원하는 장소로 옴길 수 있다. Scale은 x, y, z에 곱해준다 따라서 비율의 변화로 인해 정육면체의 모양을 변화시킬 수 있다 따라서 원래 초기의 정육각형 값으로부터 높이가 짧은 상판과 높이가 긴 상다리들을 만들 수 있다.

1. 컵만들기

-베지의 방정식

수업시간에 한 베지의 방정식으로 곡선을 만들 수 있다. 하지만 수업시간에 한 곡선은

이런 곡선이었다 이 곡선을 이렇게 와인잔 처럼 들어간 곡선으로 만들기 위해서는 점의 순서를 바꾸어야 한다

P3

p1 p2 기존에 수업시간에 했던 방식이 아닌, 이 순서와 위치대로 점을 넣어주

P0 고 베지의 방정식을 그대로 써주면 된다

float points[][2]={{0.3,0.0},{0.25,0.4},{0.5,0.1},{0.4,0.65}};//컵그리는데 초기좌표

이 점의 위치가 위의 점 위치와 비슷함을 확인할 수 있다

floatpx=(1-alpha)\*(1-alpha)\*(1-alpha)\*points[0][0]+3\*(1-alpha)\*(1-alpha)\*alpha\*points[1][0]+3\*(1-alpha)\*alpha\*alpha\*points[2][0]+alpha\*alpha\*alpha\*points[3][0]; 베지의 곡선 공식이다

-곡선으로 컵만들기

구 그리기와 비슷한 원리이다 먼저 배지의 곡선공식을 이용하여 px,py 좌표를 구하고 0.05위의 점 px1,py1를 잡아주어 가로의 선들을 그려주었다.그 밑에 for문에서는 -180에서 180도 까지 20도씩 증가하면서 곡선을 회전시켜준다. cupData에 현재 곡선 값과 0.05위의 값을 그 다음 값으로 넣어줘 그릴 때 배열의 있는 값을 차례로 뽑아와 그린다 컵의 밑바닥은 원기둥에서한 TRIANGLE\_FAN을 이용하여 그려준다 이때 I값은 반드시 2 만큼 증가 해야 된다 왜냐하면 TRIANGLE\_FAN은 중점을 기준으로 삼각형을 그리기 위해 점 2개씩 묶어주기 때문이다.

**어려웠던 문제**

좌표 찾는 게 가장 어려웠었다. 특히 컵을 그리기 위한 곡선 찾기가 어려웠다. 중간에 볼록하게 들어가게 만드는 게 수업시간에 한 곡선과 달라 어려웠다. 또한 곡선을 찾고 360도 돌리는 게 어려웠다.

완성이 되었다 싶어도 이리저리 rotate 값을 바꾸어서 돌려보니 객체가 떠있다든가 테이블의 다리가 안 붙어있는 경우가 많았다 이를 확인하기 위해 일일이 rotate값을 바꾸기가 힘들었다

**해결방법**

처음엔 수업시간에 했던 곡선의 좌표를 비율을 같게 하여 크게도 해보고 작게도 해봤는데 그냥 평범한 곡선이 나왔다. 그래서 좌표의 위치를 바꿔보니 중간이 잘록하게 들어간 곡선이 나왔고 점이 곡선을 땅겨준다는 사실을 알았다

그 곡선을 360도 돌리는 것은 수업시간에 한 구를 참고를 많이 했다 -180에서 180까지 잘게 나누어 줄수록 원형에 가깝게 된다 90도를 넣었더니 사각형 모양의 특이한 겁이 나왔었다.

Rotate를 내 맘대로 바꾸기 위해서 인터넷에서 key를 눌러 Rotate를 사용자 마음대로 바꾸는 방법을 찾았다. 내가 원하는 각도를 볼 수 있어 편했다.