**중국식 회전테이블 만들기**

1. RotateTable.cpp
2. #include<stdlib.h>
3. #include<GL/glut.h>
4. #include<math.h>
5. #include<GL/GLAux.h>
6. #include"Vector.h"
7. #include<stdio.h>;
8. #include<iostream>
9. using namespace std;
10. GLfloat gXAngle, gYAngle, gZAngle;//로테이트 값을 받을 변수
11. static GLfloat angle;//테이블 원탁을 회전시킬때 받을 값
12. int count = 1;
13. float Delta = 0;
14. const GLfloat red[]={0.8,0.0,0.0,1.0};
15. const GLfloat green[]={0.0,1.0,0.0,1.0};
16. const GLfloat white[]={1.0,1.0,1.0,1.0};
17. const GLfloat lightPos[]={50,100.0,30.0,1.0};
18. const GLfloat brass\_amb[]={0.33,0.22,0.03,1.0};
19. const GLfloat brass\_diff[]={0.78,0.57,0.11,1.0};
20. const GLfloat brass\_diff2[] = {0.2, 0.8, 1.11 , 1.0};
21. const GLfloat brass\_spec[]={0.99,0.91,0.81,1.0};
22. const GLfloat brass\_shine[]={20.8};
23. const GLfloat brass\_diff8[] = {15.78, 1.0, 1.00 , 1.0};
24. const GLfloat brass\_diff3[] = {0.2, 0.4, 0.2 , 1.0};
25. const GLfloat brass\_diff4[] = {0.5, 1, 0 , 1.0};
26. const GLfloat brass\_diff5[] = {1, 0.5, 0.1 , 1.0};
27. const GLfloat brass\_diff6[] = {3, 1.5, 1 , 1.0};
28. const GLfloat brass\_diff7[] = {2, 1, 1 , 1.0};
29. GLfloat MyVertex[][3] = {
30. { -0.25, -0.25, 0.25 },//v0
31. { -0.25, 0.25, 0.25 },//v1
32. { 0.25, 0.25, 0.25 },//v2
33. { 0.25, -0.25, 0.25 },//v3
34. { -0.25, -0.25, -0.25 },//v4
35. { -0.25, 0.25, -0.25 },//v5
36. { 0.25, 0.25, -0.25 },//v6
37. { 0.25, -0.25, -0.25 },//v7
39. };//다리그리는데 좌표
40. 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
41. GLfloat MyTable[][3] = {
42. { -0.25, 0.25, 0.25 },//v0
43. { 0.25, 0.25, 0.25 },//v1
44. { 0.5, 0.25, 0.0 },//v2
45. { 0.25, 0.25, -0.25 },//v3
46. { -0.25, 0.25, -0.25 },//v4
47. { -0.5, 0.25, 0.0 },//v5
48. { -0.25, -0.25, 0.25 },//6
49. { 0.25, -0.25, 0.25 },//v7
50. { 0.5, -0.25, 0.0 },//v8
51. { 0.25, -0.25, -0.25 },//v9
52. { -0.25, -0.25, -0.25 },//v10
53. { -0.5, -0.25, 0.0 }//v11
54. };//테이블그리는데 좌표
55. GLubyte MyTableQuad[] = { 0, 6, 7, 1, 2, 1, 7, 8, 3, 2, 8, 9, 4, 3, 9, 10, 5, 4, 10, 11, 0, 5, 11, 6 };//상판 옆면 쿼드
56. GLuint MyTableQUad2[] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 };//상판 아래 위 쿼드
57. VECTOR normal1[6];//테이블 옆면 벡터값
58. VECTOR legNormal[6];//다리벡터값 저장
59. const double ToRadian = 3.141592 / 180.0;//라디안값
60. float lineData[200][3];//원기둥 기둥
61. float circleData[200][3];//원기둥 원
62. GLubyte MyCylinderQuad[]={1,3,2,0,3,5,4,2,5,7,6,4,7,9,8,6,9,11,10,8,11,13,12,10,13,15,14,12,15,17,16,14,17,19,18,16,19,21,20,18,21,23,22,20,23,25,24,22,25,27,26,24,27,29,28,26,29,31,36,28,31,33,32,30,33,35,34,32,35,1,0,34};
63. VECTOR cylinderNormal[18];
64. float radius = 0.5;//원기둥 그리는데 반지름
65. int k = 0;
66. int l = 0;
67. float dishpoints[][2]={{0.3,0.0},{0.2,0.35},{0.5,0.5},{0.6,0.33}};//접시그리는데 초기좌표
68. GLfloat dishData[760][3];//접시값 좌표넣기
69. float dishNormal2[760][3];//접시값 법선넣기
70. int c,e,d;
71. float oneTwo[3], oneThree[3], dishVector[3], dishNormalize[3];//접시 벡터값 넣을변수들
72. float dish1[3], dish2[3], dish3[3];
73. void ver(int x, int y, int z, GLfloat A[3][3])//VECTOR 를 쓰지 않고 법선구한는법
74. {
75. dish1[0] = A[x][0]; dish1[1] = A[x][1]; dish1[2] = A[x][2];
76. dish2[0] = A[y][0]; dish2[1] = A[y][1]; dish2[2] = A[y][2];
77. dish3[0] = A[z][0]; dish3[1] = A[z][1]; dish3[2] = A[z][2];
78. }
79. void VectorFromPoints(float dish1[3], float dish2[3], float vectoroneTwo[3])//점으로부터 벡터값을 구하기위한 함수
80. {
81. vectoroneTwo[0] = dish2[0] - dish1[0];
82. vectoroneTwo[1] = dish2[1] - dish1[1];
83. vectoroneTwo[2] = dish2[2] - dish1[2];
84. }
85. void VectorCrossProduct(float v1[3], float v2[3], float v1Crossv2[3])//벡터로부터 내적을 구하기위한 함수
86. {
87. v1Crossv2[0] = v1[1] \* v2[2] - v1[2] \* v2[1];
88. v1Crossv2[1] = v1[2] \* v2[0] - v1[0] \* v2[2];
89. v1Crossv2[2] = v1[0] \* v2[1] - v1[1] \* v2[0];
90. }
91. void VectorNormalize(float vector[3], float normalizedVector[3])//단위백터화 시켜주기 위한 작업
92. {
93. float temp;
94. temp = sqrt((vector[0] \* vector[0]) + (vector[1] \* vector[1]) + (vector[2] \* vector[2]));
95. normalizedVector[0] = vector[0]/temp;
96. normalizedVector[1] = vector[1]/temp;
97. normalizedVector[2] = vector[2]/temp;
98. }
99. GLuint dishQuad[760][4];//접시 쿼드리스트 담을 변수
100. VECTOR normal4;
101. void makeTable(){//테이블 만들어주는 함수
102. glEnableClientState(GL\_VERTEX\_ARRAY);
103. glVertexPointer(3, GL\_FLOAT, 0, MyVertex);
104. //glMatrixMode(GL\_MODELVIEW);
106. glPushMatrix();//상다리 1
107. glScalef(0.3, 1, 0.3);
108. glTranslatef(-1.1, -0.23, 2.0);
109. for (int i = 0; i<6; i++){
110. glNormal3f(legNormal[i].x,legNormal[i].y,legNormal[i].z);
111. glDrawElements(GL\_POLYGON, 4, GL\_UNSIGNED\_BYTE, &MyQuadList[4 \*i]);
112. }
113. glPopMatrix();
115. glPushMatrix();//상다리 2
116. glScalef(0.3, 1, 0.3);
117. glTranslatef(1.1, -0.23, 2.0);
118. for (int i = 0; i<6; i++){
119. glNormal3f(legNormal[i].x,legNormal[i].y,legNormal[i].z);
120. glDrawElements(GL\_POLYGON, 4, GL\_UNSIGNED\_BYTE, &MyQuadList[4 \* i]);
121. }
122. glPopMatrix();
124. glPushMatrix(); //상다리3
125. glScalef(0.3, 1, 0.3);
126. glTranslatef(1.1, -0.23, -2.0);
127. for (int i = 0; i<6; i++){
128. glNormal3f(legNormal[i].x,legNormal[i].y,legNormal[i].z);
129. glDrawElements(GL\_POLYGON, 4, GL\_UNSIGNED\_BYTE, &MyQuadList[4 \* i]);
130. }
131. glPopMatrix();
133. glPushMatrix();//상다리4
134. glScalef(0.3, 1, 0.3);
135. glTranslatef(-1.1, -0.23, -2.0);
136. for (int i = 0; i<6; i++){
137. glNormal3f(legNormal[i].x,legNormal[i].y,legNormal[i].z);
138. glDrawElements(GL\_POLYGON, 4, GL\_UNSIGNED\_BYTE, &MyQuadList[4 \* i]);
139. }
140. glPopMatrix();
141. glVertexPointer(3, GL\_FLOAT, 0, MyTable);
142. glPushMatrix();
143. glScalef(1.8, 0.2, 3.2);
144. glTranslated(0,0.33,0);
145. for (int i = 0; i<6; i++){//상판옆에 그려주기
146. glNormal3f(normal1[i].x,normal1[i].y,normal1[i].z);//상판옆면에대한 법선
147. glDrawElements(GL\_POLYGON, 4, GL\_UNSIGNED\_BYTE, &MyTableQuad[4 \* i]);
148. }
149. glPopMatrix();
150. glPushMatrix();
151. glScalef(1.8, 0.2, 3.2);
152. glTranslated(0,0.33,0);
153. glBegin(GL\_TRIANGLE\_FAN);
154. glNormal3f(0,1.25,0);//상판위에대한 법선
155. glColor3f(0.8,0.3,0.8);
156. glVertex3f(0,0.25,0);
157. for(int i=0;i<5;i+=1){//상판그려주기
158. glVertex3fv(MyTable[i]);
159. glVertex3fv(MyTable[i+1]);
161. }
163. glEnd();
164. glBegin(GL\_TRIANGLE\_FAN);{
165. glNormal3f(0,1.25,0);//위에 상판에대한 법선
166. glVertex3f(0,0.25,0);
167. glVertex3fv(MyTable[5]);//상판그려주기
168. glVertex3fv(MyTable[0]);
169. }
170. glEnd();
172. glBegin(GL\_TRIANGLE\_FAN);
173. glNormal3f(0,-1.25,0);//상판 아랫면에대한 법선
174. glVertex3f(0,-0.25,0);
175. for(int i=6;i<12;i++){
177. glVertex3fv(MyTable[i]);//아랫면 상판그려주기
178. glVertex3fv(MyTable[i+1]);
180. }
181. glBegin(GL\_TRIANGLE\_FAN);{
182. glVertex3f(0,0.25,0);
183. glVertex3fv(MyTable[11]);
184. glVertex3fv(MyTable[6]);
185. }
187. glEnd();
188. glPopMatrix();
189. }
190. void MyTableNormal(){//테이블 법선구해주기
191. int j=0;
192. for(int i=6;i<11;i++){
193. VECTOR v1(MyTable[i+1],MyTable[i]);
194. VECTOR v2(MyTable[i-6],MyTable[i]);
195. Cross\_Product(v1,v2,normal1[j]);//내적하기
196. j++;
197. }
198. VECTOR v1(MyTable[6],MyTable[11]);
199. VECTOR v2(MyTable[5],MyTable[11]);
200. Cross\_Product(v1,v2,normal1[j]);
201. for(int i=0;i<6;i++)
202. normal1[i].normalizer();//단위벡터화 시키기
204. }
205. void LegNormal(){//책상다리 법선구해주기
206. VECTOR v1(MyVertex[3],MyVertex[0]), v2(MyVertex[1],MyVertex[0]);
207. Cross\_Product(v1,v2,legNormal[0]);
208. VECTOR v3(MyVertex[7],MyVertex[3]), v4(MyVertex[2],MyVertex[3]);
209. Cross\_Product(v3,v4,legNormal[1]);
210. VECTOR v5(MyVertex[3],MyVertex[0]), v6(MyVertex[4],MyVertex[0]);
211. Cross\_Product(v6,v5,legNormal[2]);
212. VECTOR v7(MyVertex[2],MyVertex[1]), v8(MyVertex[5],MyVertex[1]);
213. Cross\_Product(v7,v8,legNormal[3]);
214. VECTOR v9(MyVertex[4],MyVertex[7]), v10(MyVertex[6],MyVertex[7]);
215. Cross\_Product(v9,v10,legNormal[4]);
216. VECTOR v11(MyVertex[0],MyVertex[4]), v12(MyVertex[5],MyVertex[4]);
217. Cross\_Product(v11,v12,legNormal[5]);
218. for(int i=0;i<6;i++)
219. legNormal[i].normalizer();
220. }
221. void cylinder()//원기둥 그리기
222. {
223. double phi;
224. double y;
226. for (y = 0.5; y >= -0.5; y -= 1.0)//원그리기
227. {
228. circleData[k][0] = 0.0;
229. circleData[k][1] = y;
230. circleData[k][2] = 0.0;
231. k++;
232. for (phi = 0.0; phi <= 360.0; phi += 20.0)
233. {
234. double phiR = phi \* ToRadian;
235. double x = radius \* cos(phiR);
236. double z = radius \* sin(phiR);
237. circleData[k][0] = x;
238. circleData[k][1] = y;
239. circleData[k][2] = z;
240. k++;
241. }
242. }
243. for (phi = 0.0; phi <= 360.0; phi += 20.0)//원기둥 그리기
244. {
245. double phiR = phi \* ToRadian;
246. double x = radius \* cos(phiR);
247. double z = radius \* sin(phiR);
248. lineData[l][0] = x;
249. lineData[l][1] = 0.5;
250. lineData[l][2] = z;
251. l++;
252. lineData[l][0] = x;
253. lineData[l][1] = -0.5;
254. lineData[l][2] = z;
255. l++;
256. }
258. }
259. void drawCylinder(){
260. glScalef(0.3,0.3,0.3);
261. glScalef(4.5,0.5,4.5);
262. glTranslatef(0.0,1.3,0.0);
263. glBegin(GL\_TRIANGLE\_FAN);
264. glNormal3f(0,1.5,0);//원기둥 위에대한 법선
266. for (int idx = 0; idx < k / 2; idx++)
268. glVertex3fv(circleData[idx]);
270. glEnd();
271. glBegin(GL\_TRIANGLE\_FAN);
272. glNormal3f(0,-1.5,0);//원기둥 아래에 대한 법선
273. for (int idx = k / 2; idx < k; idx++)
275. glVertex3fv(circleData[idx]);
276. glEnd();
278. glEnableClientState(GL\_VERTEX\_ARRAY);
279. glVertexPointer(3, GL\_FLOAT, 0, lineData);
280. for(int i=0;i<18;i++){
281. glNormal3f(cylinderNormal[i].x,cylinderNormal[i].y,cylinderNormal[i].z);
282. glDrawElements(GL\_POLYGON, 4, GL\_UNSIGNED\_BYTE, &MyCylinderQuad[4 \* i]);
283. }
285. }
286. void CylinderNormal(){//원기둥 옆면에 대한 법선 구하기
287. int j=0;
288. for(int i=1;i<=33;i+=2){
289. VECTOR v1(lineData[i+2],lineData[i]);
290. VECTOR v2(lineData[i-1],lineData[i]);
291. Cross\_Product(v1,v2,cylinderNormal[j]);
292. j++;
293. }
294. VECTOR v1(lineData[1],lineData[35]);
295. VECTOR v2(lineData[34],lineData[35]);
296. Cross\_Product(v1,v2,cylinderNormal[j]);
297. for(int i=0;i<j;i++){
298. cylinderNormal[i].normalizer();
300. }
302. }
303. void dish(){//접시그리기
304. double theta;
305. for(float alpha=0.0f;alpha<=1.0f;alpha+=0.05){
306. float px=(1-alpha)\*(1-alpha)\*(1-alpha)\*dishpoints[0][0]+3\*(1-alpha)\*(1-alpha)\*alpha\*dishpoints[1][0]+3\*(1-alpha)\*alpha\*alpha\*dishpoints[2][0]+alpha\*alpha\*alpha\*dishpoints[3][0];//베지의 곡선공식
307. float py=(1-alpha)\*(1-alpha)\*(1-alpha)\*dishpoints[0][1]+3\*(1-alpha)\*(1-alpha)\*alpha\*dishpoints[1][1]+3\*(1-alpha)\*alpha\*alpha\*dishpoints[2][1]+alpha\*alpha\*alpha\*dishpoints[3][1];
308. float px1=(1-(alpha+0.05))\*(1-(alpha+0.05))\*(1-(alpha+0.05))\*dishpoints[0][0]+3\*(1-(alpha+0.05))\*(1-(alpha+0.05))\*(alpha+0.05)\*dishpoints[1][0]+3\*(1-(alpha+0.05))\*(alpha+0.05)\*(alpha+0.05)\*dishpoints[2][0]+(alpha+0.05)\*(alpha+0.05)\*(alpha+0.05)\*dishpoints[3][0];//0.05위의 면(구와 같은원리)
309. float py1=(1-(alpha+0.05))\*(1-(alpha+0.05))\*(1-(alpha+0.05))\*dishpoints[0][1]+3\*(1-(alpha+0.05))\*(1-(alpha+0.05))\*(alpha+0.05)\*dishpoints[1][1]+3\*(1-(alpha+0.05))\*(alpha+0.05)\*(alpha+0.05)\*dishpoints[2][1]+(alpha+0.05)\*(alpha+0.05)\*(alpha+0.05)\*dishpoints[3][1];
311. for (theta = -180; theta <= 180.0; theta += 20){
312. float radian = ToRadian\*theta;
313. dishData[e][0] = px\*sin(radian);
314. dishData[e][1] = py;
315. dishData[e][2] = px\*cos(radian);
316. e++;

319. dishData[e][0] = px1\*sin(radian);
320. dishData[e][1] = py1;
321. dishData[e][2] = px1\*cos(radian);
322. e++;
323. }cout<<e<<endl;
324. }
325. for(int i=0;i<723;i++){//접시의 쿼드값 잡아주기
326. dishQuad[i][0]=i;
327. dishQuad[i][1]=i+1;
328. dishQuad[i][2]=i+38;
329. dishQuad[i][3]=i+39;
331. }
332. }
333. int p=0;
334. void drawDish(){//접시그려주기
335. glEnableClientState(GL\_VERTEX\_ARRAY);
337. glVertexPointer(3,GL\_FLOAT,0,dishData);

340. for (int i = 0; i<721; i++){
342. glNormal3f(dishNormal2[i][0],dishNormal2[i][1],dishNormal2[i][2]);//접시 법선
343. glDrawElements(GL\_QUAD\_STRIP,4,GL\_UNSIGNED\_INT,&dishQuad[i]);
344. }
345. glBegin(GL\_TRIANGLE\_FAN);//접시 밑에 닫기
346. glVertex3f(0,0,0);
347. glNormal3f(0,1,0);//접시 밑에 법선
348. for(int i=0; i<40; i+=2) {
349. glVertex3fv(dishData[i]);//접시밑바닥 그려주기
350. }
351. glEnd();
352. }
353. void DishNormal(){//접시 법선잡아주기

356. for (int i = 0; i < 721; i++)
357. {
359. ver(i, i + 38, i + 1,dishData);
360. VectorFromPoints(dish1, dish2, oneTwo);
361. VectorFromPoints(dish1, dish3, oneThree);
362. VectorCrossProduct(oneTwo, oneThree, dishVector);
363. VectorNormalize(dishVector, dishNormalize);
365. dishNormal2[i][0] = dishNormalize[0];
366. dishNormal2[i][1] = dishNormalize[1];
367. dishNormal2[i][2] = dishNormalize[2];
368. ver(i + 1, i, i + 39,dishData);
369. VectorFromPoints(dish1, dish2, oneTwo);
370. VectorFromPoints(dish1, dish3, oneThree);
371. VectorCrossProduct(oneTwo, oneThree, dishVector);
372. VectorNormalize(dishVector, dishNormalize);
374. dishNormal2[i][0] += dishNormalize[0];
375. dishNormal2[i][1] += dishNormalize[1];
376. dishNormal2[i][2] += dishNormalize[2];
377. ver(i + 38, i + 39, i,dishData);
378. VectorFromPoints(dish1, dish2, oneTwo);
379. VectorFromPoints(dish1, dish3, oneThree);
380. VectorCrossProduct(oneTwo, oneThree, dishVector);
381. VectorNormalize(dishVector, dishNormalize);
383. dishNormal2[i][0] += dishNormalize[0];
384. dishNormal2[i][1] += dishNormalize[1];
385. dishNormal2[i][2] += dishNormalize[2];
386. ver(i + 39, i + 1, i + 38, dishData);
387. VectorFromPoints(dish1, dish2, oneTwo);
388. VectorFromPoints(dish1, dish3, oneThree);
389. VectorCrossProduct(oneTwo, oneThree, dishVector);
390. VectorNormalize(dishVector, dishNormalize);
391. ;
392. dishNormal2[i][0] += dishNormalize[0];
393. dishNormal2[i][1] += dishNormalize[1];
394. dishNormal2[i][2] += dishNormalize[2];
395. dishNormal2[i][0] / 4.0;
396. dishNormal2[i][1] / 4.0;
397. dishNormal2[i][2] / 4.0;
399. }

402. }
403. void DrawfiveDishes(void)//접시 다섯개 그리기
404. {
405. glPushMatrix();
406. glScalef(0.3,1.3,0.3);
407. glTranslatef(1.0,0.4,0);
408. drawDish();
409. glPopMatrix();
410. glPushMatrix();
411. glScalef(0.3,1.3,0.3);
412. glTranslatef(0.2,0.4,-1.0);
413. drawDish();
414. glPopMatrix();
415. glPushMatrix();
416. glScalef(0.3,1.3,0.3);
417. glTranslatef(-0.9,0.4,-0.4);
418. drawDish();
419. glPopMatrix();
420. glPushMatrix();
421. glScalef(0.3,1.3,0.3);
422. glTranslatef(-0.6,0.4,0.8);
423. drawDish();
424. glPopMatrix();
425. glPushMatrix();
426. glScalef(0.25,1.3,0.25);
427. glTranslatef(0.6,0.4,1.2);
428. drawDish();
429. glPopMatrix();
431. }
432. void DrawSolid(){// 원다섯개 그려주기
433. glMaterialfv(GL\_FRONT,GL\_DIFFUSE,brass\_diff5);
434. glPushMatrix();
435. glScalef(0.5,5,0.5);
436. glTranslatef(0.6,0.3,0);
437. glutSolidSphere(0.2,200,200);
438. glPopMatrix();
439. glMaterialfv(GL\_FRONT,GL\_DIFFUSE,brass\_diff4);
440. glPushMatrix();
441. glScalef(0.5,5,0.5);
442. glTranslatef(-0.55,0.3,-0.28);
443. glutSolidSphere(0.2,200,200);
444. glPopMatrix();
445. glMaterialfv(GL\_FRONT,GL\_DIFFUSE,brass\_diff2);
446. glPushMatrix();
447. glScalef(0.5,5,0.5);
448. glTranslatef(-0.35,0.3,0.5);
449. glutSolidSphere(0.2,200,200);
450. glPopMatrix();
451. glMaterialfv(GL\_FRONT,GL\_DIFFUSE,brass\_diff5);
452. /\*glPushMatrix();
453. glScalef(0.5,5,0.5);
454. glTranslatef(0.32,0.3,0.6);
455. glutSolidSphere(0.2,200,200);
456. glPopMatrix();\*/
457. glMaterialfv(GL\_FRONT,GL\_DIFFUSE,brass\_diff);
458. glPushMatrix();
459. glScalef(0.5,5,0.5);
460. glTranslatef(0.15,0.3,-0.65);
461. glutSolidSphere(0.2,200,200);
462. glPopMatrix();
464. }
466. void myDisplay(){
467. glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT);
468. glColor3f(1.0,0.0,0.0);
469. glLoadIdentity();
470. glRotatef(gXAngle, 1.0, 0.0, 0.0);//키보드 누를떄마다 로테이트 각이 바뀜
471. glRotatef(gYAngle, 0.0, 1.0, 0.0);
472. glRotatef(gZAngle, 0.0, 0.0, 1.0);
473. glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL);
474. glMaterialfv(GL\_FRONT,GL\_DIFFUSE,brass\_diff);
475. glLightfv(GL\_LIGHT7,GL\_DIFFUSE,red);
476. makeTable();//테이블 만든것
477. glRotatef(angle, 0, angle,0);//원기둥 회전을 위한것
479. drawCylinder();//원기둥 그려주기
481. glMaterialfv(GL\_FRONT,GL\_DIFFUSE,brass\_diff3);//색을너줌
482. DrawfiveDishes();//접시그려주기
483. DrawSolid();//원들 그려주기
484. glFlush();
485. glutSwapBuffers();
486. }
487. void RotationAngle()
488. {
489. angle += Delta;
490. if (angle>360)
491. {
492. angle = 0;
493. }
494. myDisplay();
495. }
496. void myMouse(int btn, int state, int x, int y)
497. {
498. if (btn == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN) count = 1;
499. if (btn == GLUT\_MIDDLE\_BUTTON && state == GLUT\_DOWN) count = 2;
500. if (btn == GLUT\_RIGHT\_BUTTON && state == GLUT\_DOWN) count = 3;
501. }
502. void gSetPos(int dir)//키보드 값마다 로테이트값 바뀜
503. {
504. switch (dir)
505. {
506. case 1:
507. gYAngle += 10.0f;
508. break;
509. case 2:
510. gYAngle -= 10.0f;
511. break;
512. case 3:
513. gXAngle += 10.0f;
514. break;
515. case 4:
516. gXAngle -= 10.0f;
517. break;
518. case 5:
519. gZAngle += 10.0f;
520. break;
521. case 6:
522. gZAngle -= 10.0f;
523. break;
524. }
525. }
526. void MyKeyboard(unsigned char KeyPressed, int X, int Y)
527. {
528. //q,w,e,a,s,d 을 누르면 테이블 전체가 회전이 된다
529. //z로 종료하기
530. switch (KeyPressed)
531. {
533. case 'W':
534. case 'w':
535. gSetPos(1);//y++
536. break;
537. case 'S':
538. case 's':
539. gSetPos(2);//y--
540. break;
541. case 'Q':
542. case 'q':
543. gSetPos(3);//x++
544. break;
545. case 'A':
546. case 'a':
547. gSetPos(4);//x--
548. break;
549. case 'E':
550. case 'e':
551. gSetPos(5);//z++
552. break;
553. case 'd':
554. case 'D':
555. gSetPos(6);//z--
556. break;
558. }
560. glutPostRedisplay();//다시 그려주기
561. }
562. void myKeyboard(int key, int x, int y) {//방향버튼을 클릭하면 원탁이 회전한다
563. switch (key) {
564. case GLUT\_KEY\_LEFT:
565. Delta -= 0.01;
566. break;
567. case GLUT\_KEY\_RIGHT:
568. Delta += 0.02;
569. break;
570. case GLUT\_KEY\_UP:
571. Delta += 0.02;
572. break;
573. case GLUT\_KEY\_DOWN:
574. Delta = 0;
575. break;
576. }
578. }
579. void myInit(){
580. glClearColor(1.0,1.0,1.0,1.0);
581. glMatrixMode(GL\_PROJECTION);
582. glLoadIdentity();
584. gluPerspective(0.0,1.0,1.0,10.0);
585. glMatrixMode(GL\_MODELVIEW);
586. gluLookAt(0.0,0.0,20.0,0.0,0.0,0.0,0.0,1.0,0.0);
587. glLightfv(GL\_LIGHT0,GL\_DIFFUSE,white);
588. glLightfv(GL\_LIGHT0,GL\_POSITION,lightPos);
589. glMaterialfv(GL\_FRONT,GL\_AMBIENT,brass\_amb);
590. glMaterialfv(GL\_FRONT,GL\_SPECULAR,brass\_spec);
591. glMaterialfv(GL\_FRONT,GL\_SHININESS,brass\_shine);
593. glEnable(GL\_LIGHTING);
594. glEnable(GL\_LIGHT0);
595. glShadeModel(GL\_SMOOTH);
596. glEnable(GL\_DEPTH\_TEST);
597. }
598. int main(int argc, char\*\* argv){
600. glutInit(&argc, argv);
601. MyTableNormal();//법선만들어주는 함수들 실행
602. LegNormal();
603. cylinder();
604. CylinderNormal();
605. dish();
606. DishNormal();
607. glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGB | GLUT\_DEPTH);
608. glutInitWindowSize(500, 500);
609. glutInitWindowPosition(100, 100);
610. glutCreateWindow("RotateTable");
611. myInit();
612. glutSpecialFunc(myKeyboard);
613. glutKeyboardFunc(MyKeyboard);
614. glutDisplayFunc(myDisplay);//display callboneThreek function
615. glutIdleFunc(RotationAngle);
616. glutMouseFunc(myMouse);
617. glutMainLoop();
618. return 0;
619. }

2. Vector.h

#include <math.h>

class VECTOR

{

public:

float x, y, z;

VECTOR()

{

x = 0.0f;

y = 0.0f;

z = 0.0f;

}

VECTOR(const float\* v)

{

x = v[0];

y = v[1];

z = v[2];

}

VECTOR(const float\* p1, const float\* p2)

{

x = p2[0] - p1[0];

y = p2[1] - p1[1];

z = p2[2] - p1[2];

}

void normalizer() // 정규화. 벡터의 크기를 1로 고정시킨다.

{ // 라이팅이 적용되려면 모든 벡터는 정규화 과정을 꼭 거쳐야 한다.

float len;

len = sqrt(x\*x + y\*y + z\*z);

x = x / len;

y = y / len;

z = z / len;

}

};

void Cross\_Product(const VECTOR& v1, const VECTOR& v2, VECTOR& ans)

{

// a \* b = (a2b3 - a3b2, a3b1 - a1b3, a1b2 - a2b1)

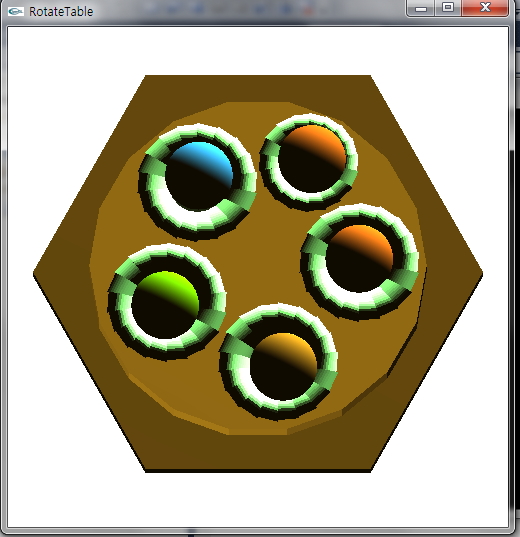
ans.x = v1.y\*v2.z - v1.z\*v2.y;

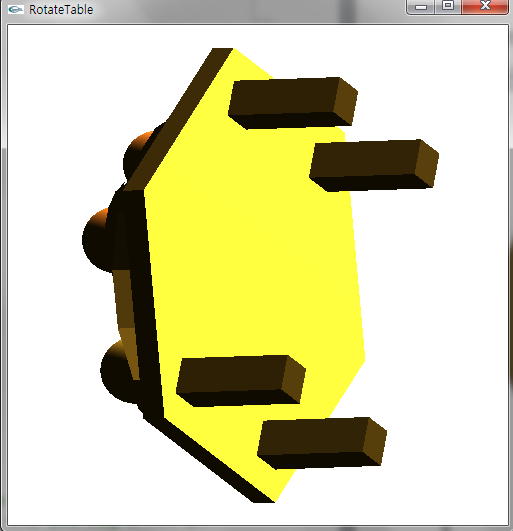
ans.y = v1.z\*v2.x - v1.x\*v2.z;

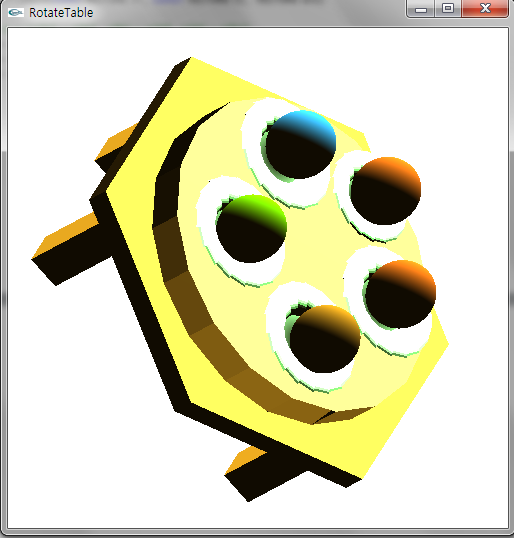
ans.z = v1.x\*v2.y - v1.y\*v2.x;

}

**3. 결과화면**









접시 밑바닥입니다.

4.과정해결 설명

1.VECTOR.h

이 헤더파일은 인터넷어서 만들어둔 헤더파일을 가져왔습니다

VECTOR 이라는 객체는 x,y,z 값을 가지고있고

VECTOR v1 안에 두 점을 넣어주면 안에서 두 점에 대해 빼기를 통해 벡터의 크기를 찾아준다

이렇게 두 벡터를 찾으면 Cross\_Product 라는 함수를 통해 두 벡터를 넣어줘 외적을 시켜준다

Normalize() 함수를 통해 단위벡터화도 시켜준다

2. VECTOR. 헤더를 이용하지 않고 벡터를 구하는 함수

void ver(int x, int y, int z, GLfloat A[3][3])//VECTOR 를 쓰지 않고 법선구한는법

{

dish1[0] = A[x][0]; dish1[1] = A[x][1]; dish1[2] = A[x][2];

dish2[0] = A[y][0]; dish2[1] = A[y][1]; dish2[2] = A[y][2];

dish3[0] = A[z][0]; dish3[1] = A[z][1]; dish3[2] = A[z][2];

}

void VectorFromPoints(float dish1[3], float dish2[3], float vectoroneTwo[3])//점으로부터 벡터값을 구하기위한 함수

{

vectoroneTwo[0] = dish2[0] - dish1[0];

vectoroneTwo[1] = dish2[1] - dish1[1];

vectoroneTwo[2] = dish2[2] - dish1[2];

}

void VectorCrossProduct(float v1[3], float v2[3], float v1Crossv2[3])//벡터로부터 내적을 구하기위한 함수

{

v1Crossv2[0] = v1[1] \* v2[2] - v1[2] \* v2[1];

v1Crossv2[1] = v1[2] \* v2[0] - v1[0] \* v2[2];

v1Crossv2[2] = v1[0] \* v2[1] - v1[1] \* v2[0];

}

void VectorNormalize(float vector[3], float normalizedVector[3])//단위백터화 시켜주기 위한 작업

{

float temp;

temp = sqrt((vector[0] \* vector[0]) + (vector[1] \* vector[1]) + (vector[2] \* vector[2]));

normalizedVector[0] = vector[0]/temp;

normalizedVector[1] = vector[1]/temp;

normalizedVector[2] = vector[2]/temp;

}

처음에 접시의 벡터를 구할 때 VECTOR 를 만들 생각을 하지 못하고 float형으로 x,y,z를 묶어주기 위해 가져온 메소드들입니다 처음에 이걸로 써서 하다가 x,y,z 를 묶어서 쓰는 VECTOR 가 잇으면 편하겠다 생각해 나중에 VECTOR.h 헤더를 가져왔습니다. 원리는 VECTOR.h 와 동일합니다.

1.테이블 만들기 makeTable()

정육각형 테이블을 만들어주기위해 각각의 좌표를 잡고 TRIANGLE\_FAN 으로 테이블을 그려주었다 처음에는 TRIANBLE\_FAN으로 안그리고 그냥 QUAD\_STRIP으로 그렸더니 옆면에만 색이 입혀지고 상 위에판은 색이 입혀지지않아 어려움을 격었지만 원기둥처럼 TRIANGLE\_FAN으로 그려줘야한다는 것을 알았다 상 다리 또한 좌표를 잡아 정육각형을 그린 뒤 scalef 와 translate를 시켜주었다

2. 테이블 법선잡아주기 MyTableNormal()

테이블의 상판의 벡터는 하나로 테이블 중점좌표에 y값에 +1한 값을 법선으로 넣어주었다

상판의 바닥은 중점의 y 값에 -1한 값을 법선으로 넣어주었다.상판의 옆면은 한점에 대해 a,b,c 를 잡고 a to b, c to b 이런식으로 벡터를 찾아준다음에 cross\_product 로 외적을 하였다.

for(int i=6;i<11;i++){

VECTOR v1(MyTable[i+1],MyTable[i]);

VECTOR v2(MyTable[i-6],MyTable[i]);

Cross\_Product(v1,v2,normal1[j]);//내적하기

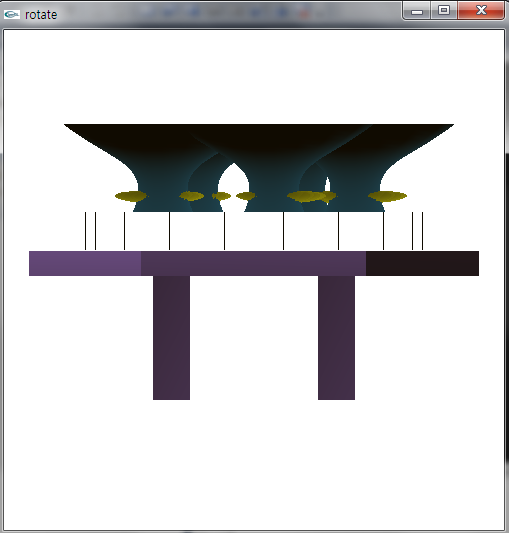
j++;

}

테이블 다리의 벡터도 테이블 옆면의 벡터를 잡아준 원리와 같이 잡아주었다.

3.원판 만들기 cylinder() drawCylinder()

circleData로 원판의 그림을 그려주고 lineData로 옆면을 선으로 그려줍니다 여기서 어려웠던점은 lineData를 이용해 법선을 구해 색을 입혔는데 색이 안입혀 져서 의아해 했는데 그냥 glBegin(GL\_LINE) 을써서 언뜻 보기에는 완성된 원기둥 같지만 사실 옆면은 선만 그려진 것이었습니다 그래서 glEnableClientState(GL\_VERTEX\_ARRAY);glVertexPointer(3, GL\_FLOAT, 0, lineData); 이렇게 설정을 해주고 그렸더니 선이 도형으로 묶이면서 색이 입혀졌습니다.

 문제의 모델이다 옆면이 색이 안들어간다

이그림은 지금의 테이블을 만들기전의 가모델이다.

4.원판 법선잡이주기 MakeTableNormal()

상판과 동일하게 원판의 중점을 중심으로 y 좌표를+1 -1 해서 법선을 잡아주었다

원판의 옆면도 테이블옆면만든것과 동일하게 법선을 넣어주었다.

5.접시 만들어주기 dish() drawDish()

베지의 곡선을 이용하여 접시를 그렸다

-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

p1 p2 P3 기존에 수업시간에 했던 방식이 아닌, 이 순서와 위치대로 점을 넣어주

P0 어 접시모양이 나오게 했다.

아직도 해결이 되지 않은문제:

접시가 부드럽게 그려지지 않았다

처음에 glBegin() glEnd()로 접시를 그렸을때는 접시가 부드럽게 나왔지만 법선을 적용했을 때 적용이 되지 않았다 그래서 쿼드리스트를 잡아주고 drawElement 로 그렸는데 쿼드를 잘못잡았는지 접시의 표면이 살짝 거칠게 나온다. 아직 그 문제는 잡지 못하였다. 접시의 좌표값을 한층에대해 넣어줄때마다 그 값을 출력을 햇는데 38씩 증가를해 한 층엔 점이 0부터 37까지 있고 38부터는 점이 위로올라간다 즉 0,1,38,39 가 한쿼드로 이어진다는 판단하에

쿼드를 for(int i=0;i<723;i++){//접시의 쿼드값 잡아주기

dishQuad[i][0]=i;

dishQuad[i][1]=i+1;

dishQuad[i][2]=i+38;

dishQuad[i][3]=i+39;

로 잡아주었지만 접시 표면이 매끄럽게 나오지 않았다.

6.접시 법선잡아주기 DishNormal()

법선은 각점에대해 외적을 구하고 그 외적의 평균을 구하는 방식으로 법선을 구했다.

처음에는 평균을 구해주지않고 쿼드에서 한점에 대해서만 법선을 구했는데 접시에 법선이 자연스럽게 들어가지 않아 평균값을 구하는 방법으로 고쳤다.

문제의 모델이다 평균값을 구하는 방식이 아닌 그냥 테이블에 법선구하듯이평균을 구하지 않았더니 이렇게 나왔다.

7. 구만들기

구는 openGL 에서 제공하는 glutSolidShpere 을 이용하였다

8.원판 회전시키기

원판은 방향키를 누르면회전이되고 아래로 버튼을 누르면 정지된다.

x축과 z축의 값을 바꾸어 줌으로써 원판이 회전한다

여기서 헷깔렸던거는 방향키도 메인에서 KeyboardFunc 인줄알고 테이블 전체를 회전하는 함수와 원판만 회전시키는 함수를 어떻게 한 콜백함수에서 부르는지 고민을 했는데 방향키는 스페셜Func으로 등록해줄 수 있다는 사실을 알았다.

glutSpecialFunc(myKeyboard);