3D 그래픽스 프로그래밍

www.cs.ureginea.ca

2주차 (16.09.09)

□ 3D 그래픽스 프로그래밍

* Graphics (그래픽스) : 그래픽 학문

* Modeling (모델링) : 모델을 만드는 것 / 만들고 옮기는 것 까지가 모델링
  - Mesh : 철사로 이루어진 것 (양파담는 망)
* Animation (애니메이션) : 모델링을 움직인 것
  - keyFrame Animation : 중요한 프레임의 애니메이션
* Rendering (렌더링) : 색을 칠하는 것, 빛을 어떻게 반사하는지 등을 표현하는 것이 렌더링

* 그래픽스 응용

* 오프라인 렌더링 : 고퀄리티 그래픽(시간이 엄청 오래 걸린다)
* 실시간 렌더링 : 게임 등에서 쓰이는 렌더링
* 래스터 그래픽스 : 면 렌더링이 가능 (면의 색들을 한번에 ON)
* 지역 조명 : 지역 내에만 빛을 사용
* 전역 조명 : 전역으로 빛을 사용

□ 응용 함수

* display() 함수

* glPointSize(5); : 점의 사이즈 조절 (glBegin() 전에 써줘야 함)

* main() 함수

* glutidle(display); : idle 상태를 설정할 수 있음

□ OpenGL 기초

OpenGL 기본형태

```
#include whatever you want
callback_fordisplay() {
    for(그려질 모든 객체에 대해){
        변환 설정;
        glBegin(그리기 프리미티브 지정);
          [[정정(vertex) 정보 제공;]]
        glEnd();
    }
    glFlush() 또는 glutSwapBuffers(); //메모리에 있는 것을 디스플레이로 전송
     // buffer : 메모리에 대해 임시저장하는 것
}
void main(int argc, char **argv){ // 콘솔 프로그램에서 입력을 받는 것
    [[윈도우 초기화;]] // GPU에 어떤 메모리를 잡을 것인지 설정
    glMatrixMode(GL_PROJECTION);
    [[투영 행렬 설정;]]
    glMatrixMode(GL_MODELVIEW);
    [[카메라의 위치와 방향 잡기;]]
    [[콜백 함수의 등록;]]
    [[메인 루프로 들어가기;]]
}
```

* OPENGL의 특징 : 운영체제 독립적

* OS dependent : 운영체제 의존적, 종속적
* OS independent : 운영체제 독립적 <= API GLUT lib가 운영체제 독립적으로 띄울 수 있게 해준다
* window의 특징 : loop를 돌게 한다
* event driven : 이벤트를 운전함 (call back)
  - Draw event : 그리는 이벤트 / Idle event : 아무것도 안할 때의 이벤트 / size event : 크기 조정

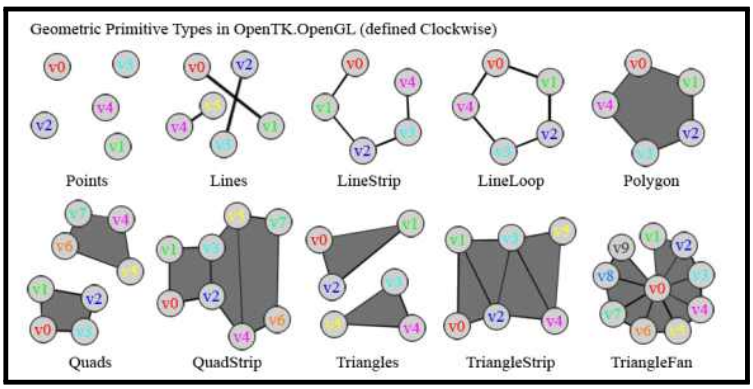
* 프리미티브(primitives/원시적인) : 크기 조정

* 그래픽 하드웨어는 프로그래머가 지정한 프리미티브 설정에 따라 정점의 리스트를 처리
* 프리미티브 사용 방법

프리미티브

```
glBegin (drawing primitive);
    // vertex position, color, normal, etc
    setVertexInfo();
glEnd();
```

- GL_POINT : 입력된 정점을 하나씩 점으로 가시화
- GL_LINES : 입력된 정점을 두 개씩 묶어 선분으로 표현 // 조합이 안되는 것은 버림
- GL_LINE_STRIP : 입력된 정점을 차례대로 연결하여 하나의 폴리라인(polyline)을 구성
- GL_LINE_LOOP : 입력된 정점을 차례로 연결한 뒤에 마지막 점을 시작점으로 연결
- GL_TRIANGLES : 입력된 정점을 세 개씩 묶어 삼각형을 그림
- GL_TRIANGLE_STRIP : 처음 세 개 정점으로 삼각형을 그린 뒤, 정점이 추가될 때마다 삼각형을 직전 두 개 정점과 연결하여 삼각형 추가
- GL_QUADS : 정점 네 개씩을 묶어 사각형 그리기
- GL_QUAD_STRIP : 처음 네 개 정점으로 사각형을 그리고, 이후 두 개씩 묶어 직전 두 개 정점과 함께 사각형 그리기
- GL_POLYGON : 입력된 모든 정점으로 다각형을 그림



* Matrix

- glPushMatrix(); ~ glPopMatrix(); : 한 단락으로 명령을 적용 (진행 시 사용)
- glBegin(GL_POLYGON); ~ glEnd(); : 범위 안의 모든 것에 적용시킨다. (생성 시 사용)

* 오브젝트 변환

* glTranslatef(GLfloat ds, GLfloat dy, GLfloat dz); : dx, dy, dz 만큼 이동하는 함수
* glScalef(GLfloat sx, GLfloat sy, GLfloat sz); : sx, sy, sz배 만큼 확대하는 함수
* glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z); : 정해진 축을 중심으로 회전
  - angle은 라디안 각이 아니라 일반적으로 사용하는 60분법의 도
  - x축을 중심으로 회전할 때는 x를 1, y는 y를 1... 로 하면 됨
  - 2D를 할 때는 z만 1로 하면 됨

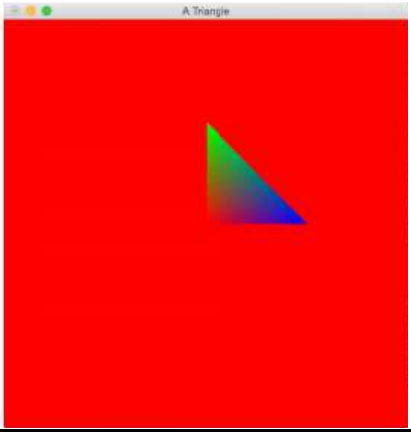
* 기타 Draw함수

* glutWireSphere(0.5, 10, 10); : 스페어 (반지름, 경도, 위도), 선
* glutSolidSphere(0.5, 10, 10); : 스페어 (반지름, 경도, 위도), 면
* glutWireTeapot(0.5, 10, 10); : 주전자 (크기) 선으로 이루어짐
* glutSolidTeapot(0.5, 10, 10); : 주전자 (크기) 면(색)으로 이루어짐
* glLineWidth(1); : 선의 굵기를 설정

소스코드

```
#ifdef WIN32 // 윈도우 일 경우
#include <windows.h>
#include <Gl/gl.h>
#include <Gl/glut.h>
#else // 윈도우가 아닐 경우
#include <OpenGL/OpenGL.h>
#include <GLUT/GLUT.h>
#endif

void myDisplay() { // 오브젝트를 그림
    glClear(GL_COLOR_BUFFER_BIT); // 그림그리기 전에 사용
    glBegin(GL_POLYGON); // 폴리곤을 사용(입력된 모든 정점으로 다각형을 그림)
    glColor3f(1.0, 0.0, 0.0);  // 점~면적 색 지정
    glVertex3f(0.0, 0.0, 0.0); // Vertex3f(세 점)의 값 0,0
    glColor3f(0.0, 0.0, 1.0);
    glVertex3f(0.5, 0.0, 0.0);
    glColor3f(0.0, 1.0, 0.0);
    glVertex3f(0.0, 0.5, 0.0);
    glEnd();
    glFlush(); // 기본 색이 검은 색
}
int main (int argc, char * argv[]) { // 배경을 그림
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGBA); // glut의 RGBA 사용
     // 칼라와 깊이 칼라 버퍼 비트의 설정 하거나 0001 0010 방식으로 확인
    glutInitWindowPosition(0, 0); // glut 윈도우 위치 설정
    glutInitWindowSize(512, 512); // glut 창 사이즈 설정
    glutCreateWindow("12510096 조광민"); // 윈도우 이름 설정
    glClearColor(1.0, 0.0, 0.0, 1.0); // 지울 때 무슨 색으로 지울 것인지 설정
    glutDisplayFunc(myDisplay); // 디스플레이 콜백 등록
    glutMainLoop(); // 이벤트 루프로
    return 0;
}
```

실행화면



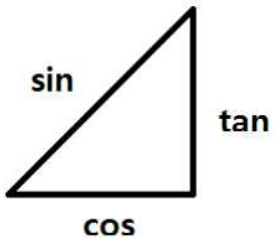
* **법선벡터(nomal)** : 음영을 표시할 때 사용 // 면에 수직인 선 // |a*b| = N 벡터 a와 b를 외적
  - glNomal3f(); : 법선 벡터는 무조건 3f
* **smooth shading** : 삼각형의 점 3개의 법선벡터를 다 다르게 함 // 면과면이 만나는 점의 법선의 중간
* **정점 데이터 설정 방법**
  - 정점 데이터는 위치와 법선벡터, 색 등을 표현
  - 정점의 위치만을 입력한다면 glVertex[dim-type]으로 입력
  - 3차원 정점의 각 성분을 부동소수점 표현으로 넣는다면 glVertex3f(x, y, z)로 입력

정점 데이터

```
float x, y, z;
double dx, dy, dz;
int ix, iy, iz;
float verts = {1.0f, 2.0f, 1.0f};
glBegin(drawing primitive);
    glVertex3f(x, y, z);
    glVertex3d(dx, dy, dz);
    glVertex3i(ix, iy, iz);
    glVertex3fv(verts);
glEnd();
```

* **프리미티브를 이용한 풍경 그리기**
  - **원 그리기** : 반지름이 1인 원 cos(시타) sin(시타)

원 2개 그리기

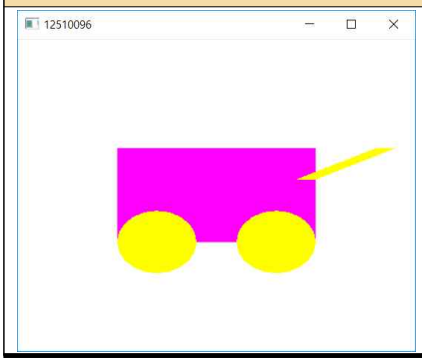



```
/* 원1 */
glBegin(GL_POLYGON);
int nPoints=20;
float radius = 0.1;
glColor3f(1.0, 1.0, 0.0);
float angle = 0.0;
float step=(3.14159*2.0)/n;
// 반복문 내에서 여러 개의 정점 좌표를 계산한 뒤에 지정하는 방식
// 여기서는 원을 이루는 정점들을 계산
while (angle <3.14159*2.0) {
    glVertex2f(radius*cos(angle), radius*sin(angle)+0.75);
    angle += step;
}

/* 원2 */
//원의 중심을 옮기고 반지름을 바꾼 뒤에 다시 그림
glBegin(GL_POLYGON);
n=20;
radius=0.25;
glColor3f(0.0, 1.0, 0.0);
angle = 0.0; step=(3.14159*2.0)/n;
while (angle < 3.14159*2.0){
    glVertex2f(radius*cos(angle)+0.625, radius*sin(anglke)+0.25);
    angle += step;
}
glEnd();
glFlush();
```

2주차 실습1

```
#define GLUT_DISABLE_ATEXIT_HACK

#include <Windows.h>
#include <gl/GL.h>
#include <gl/glut.h>
#include <math.h>

double rotation = 30;

void drawCircle(float setradius, float x, float y){
	glBegin(GL_POLYGON);
	int Points = 20;
	float radius = setradius;
	glColor3f(1.0, 1.0, 0.0);
	float angle = 0.0;
	float step = (3.14159*2.0) / Points;
	while (angle <3.14159*2.0) {
		glVertex3f(radius*cos(angle) + x, radius*sin(angle) + y, 0);
		// cos, sin에 크기 비율을 곱해줌
		angle += step;
	}
	/*
	for(float x=0; x<6.28; x+0.001f){
	glVertex3f(cos(x)*radius, sin(x)*radius, 0);
	}
	*/
	glEnd();
}

void drawRectangle(){
	glBegin(GL_QUADS);

	glColor3f(1, 0, 1);
	glVertex3f(-0.5, -0.3, 0);
	glVertex3f(0.5, -0.3, 0);
	glVertex3f(0.5, 0.3, 0);
	glVertex3f(-0.5, 0.3, 0);

	glColor3f(1, 1, 0);
	glVertex3f(0.4, 0.1, 0);
	glVertex3f(0.5, 0.1, 0);
	glVertex3f(0.9, 0.3, 0);
	glVertex3f(0.8, 0.3, 0);

	glEnd();
}

void myDisplay() {
	glClear(GL_DEPTH_BUFFER_BIT | GL_COLOR_BUFFER_BIT);

	glPushMatrix();
	drawRectangle();
	glPopMatrix();

	glPushMatrix();
	drawCircle(0.2, -0.3, -0.3);
	drawCircle(0.2, 0.3, -0.3);
	glPopMatrix();

	glFlush();
	glutSwapBuffers();
}

int main(int argc, char **argv) {
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
	glutInitWindowPosition(100, 100);
	glutInitWindowSize(500, 500);
	glutCreateWindow("12510096");
	glutDisplayFunc(myDisplay);

	glClearColor(1, 1, 1, 1);

	glutMainLoop();
	return 1;

	return 0;
}
```

실행화면



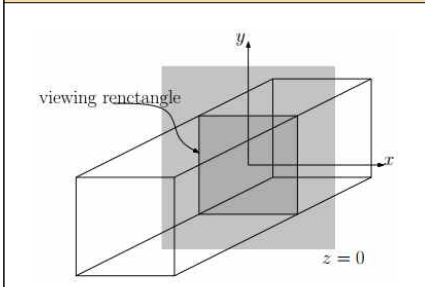
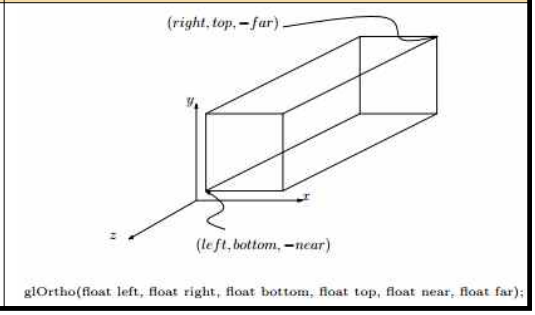
□ 카메라 설정

* 디폴트 카메라 (Default Camera)

* OpenGL의 디폴트 카메라(직교 투영 카메라) : 원점 위치
* z축 음의 방향

* 직교 투영 설정 (Orthographics Projection)

* 원근 투영을 적용하지 않아 카메라에 잡히는 공간은 상자 형태
* 상자 모양의 가시화 공간 내의 객체를 상자를 절단하는 면에 투영
* glOrtho(-1,1, -1,1, -1,1); : 상자의 위치와 길이를 변경하는 함수

디폴트 카메라	직교 투영
	

* 원근 투영(perspective projection) 설정

* 실제 카메라나 우리 눈은 원근이 없는 평행 투영이 불가능
* glFrustum 과 gluPerspective를 이용하여 원근 투영을 설정
* gluPerspective(float fovy, float Aspect, float near, float far);
  - fovy : y축 방향으로의 시야각을 도(degree)로 나타낸 것 // 줌 인, 아웃 할 때 이 값을 변경
  - Aspect : 가시화 볼륨의 종횡비(aspect ratio)
  - near : 카메라에서 상이 맺히는 가까운 평면까지의 거리
  - far : 가시화 공간을 결정하는 평면 중 카메라에서 가장 먼 쪽 평면과 카메라 사이의 거리
* 이 함수는 그리기 동작이 일어날 때마다 불리는 것이 아니라 초기에 한 번, 혹은 렌즈를 바꿀 필요가 있을 때만 불린다.

* 행렬 모드(matrix mode)

* OpenGL 행렬의 종류 : 텍스처 행렬, 모델뷰 행렬, 투영행렬
  - GL_MODELVIEW (모델뷰 행렬) : 공간 내에서 가상 객체의 좌표를 변경 (물체의 위치) ,
  - GL_PROJECTION (투영 행렬) : 가상 객체를 투영면에 옮겨 놓음 // GL_PROJECTION
  - GL_TEXTURE
  - 광원(광각) 렌즈 : 원금감이 많다. (풍경)
  - 협각(망원) 렌즈 : 원근감이 없어진다. (인물)

행렬 모드

```
glMatrixMode(GL_PROJECTION); // 렌즈를 설정함(렌즈속성만 바꾸는게 PROJECTION)
glLoadIdentity(); // 항등행렬(1행렬)로 만듬 (아무런 변환이 없게 만듬)
glOrtho(-2, 2, -2, 2, -2, 2); // -1, 1 일 때 보다 크기가 줄어든다.
gluPerspective(60, 1.0, 0.1, 100.0); // 투영의 특성을 변경하는 것이므로 투영 행렬모드

glMatrixMode(GL_MODELVIEW); // 카메라의 위치를 바꿈
glLoadIdentity(); // 항등행렬(1행렬)로 만듬 (아무런 변환이 없게 만듬)
gluLookAt(1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
// 카메라의 위치를 옮기는 것이고, 이는 바꾸어 말해 물체의 위치를 카메라 기준에서 옮기는 것이므로 모델뷰 행렬을 변경
[[draw something]]
```

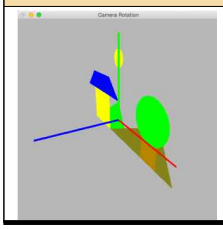
* 카메라 위치, 회전 변경

* gluLookAt() : 카메라 위치 변경
  - gluLookAt(float eye_x, float eye_y, float eye_z, float at_x, float at_y, float at_z, float up_x, float up_y, float up_z);
    - eye : 보는 위치(카메라의 위치)
    - at : 초점(바라볼 물체)의 위치
    - up : 카메라를 돌리는 것 (상향 벡터)
* 카메라 이동의 반대로 물체를 옮겨 놓음
* 카메라 위치가 매 프레임마다 변경될 수 있으므로, 그리기 함수 내에 매번 불림
* gluRotatef() : 물체 회전 (축도 같이 회전함)
  - gluRotatef(theta, 1, 1, 0);

행렬 모드

```
#include <stdiob.h>
#include <stdio.h>
#dsfine GLUT_DISABLE_ATEXIT_HACK
#include <Gl/glut.h>
#include <Gl/gl.h>
#include <Gl/glu.h>

void init (int argc, char **argv){ // 윈도우 생성, 버퍼 설정
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGBA);
    glutInitWindowPosition(0, 0);
    glutInitWindowSize(512, 512);
    glutCreateWindow("12510096 조광민");
    glClearColor(1.0, 1.0, 1.0, 1.0);
     // 카메라 투영 특성 설정(glPerspective 사용), 이 때는 GL_PROJECTION 행렬모드여야 한다.
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60, 1.0, 0.1, 100.0);
}

void drawScene() {
    [[앞서 사용한 코드 ??의 그리기 코드를 넣음. 단, glFlush는 여기서 사용하지 않음]]
}

void drawAxes() { // 3D를 구분하기 위해 3D 줄 사용
    glBegin(GL_LINES);
    glColor3f(1.0, 0.0, 0.0);    glVertex3f(0.0, 0.0, 0.0);    glVertex3f(1.0, 0.0, 0.0);
    glColor3f(0.0, 1.0, 0.0);    glVertex3f(0.0, 0.0, 0.0);    glVertex3f(0.0, 1.0, 0.0);
    glColor3f(0.0, 0.0, 1.0);    glVertex3f(0.0, 0.0, 0.0);    glVertex3f(0.0, 0.0, 1.0);
    glEnd();
}

void display() {
    static float t=0.0;
     // 카메라의 위치와 방향을 설정한다. 이 때는 GL_MODELVIEW 행렬 모드여야 한다.
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt( 2.0*sin(t), 1, 2.0*cos(t),  0, 0, 0,  0, 1, 0);
    t += 0.001;
    drawScene();
    glClear(GL_COLOR_BUFFER_BIT);
    glLineWidth(5);
    drawAxes();
    glLineWidth(1);
    glFlush();
};

int main (int argc, char **argv){
    init(argc, argv);
    glutDisplayFunc(display);
    glutIdleFunc(display);
    glutMainLoop();
}
```

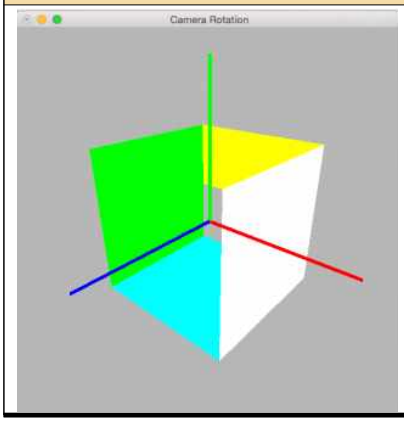
실행 결과



네 개의 면으로 상자 모양 그리기

```
#include <stdiob.h>
#include <stdio.h>
#dsfine GLUT_DISABLE_ATEXIT_HACK
#include <Gl/glut.h>
#include <Gl/gl.h>
#include <Gl/glu.h>

void drawScene() { // drawing code
    glBegin(GL_QUADS);
    // 천장
    glColor3f(1.0, 1.0, 0.0);
    glVertex3f(-0.5, 0.5, -0.5);
    glVertex3f(0.5, 0.5, -0.5);
    glVertex3f(-0.5, 0.5, 0.5);
    glVertex3f(0.5, 0.5, 0.5);

    // 바닥
    glColor3f(0.0, 1.0, 1.0);
    glVertex3f(-0.5, -0.5, -0.5);
    glVertex3f(0.5, -0.5, -0.5);
    glVertex3f(-0.5, -0.5, 0.5);
    glVertex3f(0.5, -0.5, 0.5);

    // 왼쪽 벽
    glColor3f(0.0, 1.0, 0.0);
    glVertex3f(-0.5, 0.5, -0.5);
    glVertex3f(0.5, 0.5, -0.5);
    glVertex3f(-0.5, -0.5, 0.5);
    glVertex3f(0.5, -0.5, 0.5);

    // 오른쪽 벽
    glColor3f(1.0, 1.0, 1.0);
    glVertex3f(-0.5, 0.5, -0.5);
    glVertex3f(0.5, 0.5, -0.5);
    glVertex3f(-0.5, -0.5, 0.5);
    glVertex3f(0.5, -0.5, 0.5);

    glEnd();
}

void drawAxes(){ // 3차원 좌표 선을 생성
    glBegin(GL_LINES);
    glColor3f(1.0, 0.0, 0.0);    glVertex3f(0.0, 0.0, 0.0);    glVertex3f(1.0, 0.0, 0.0);
    glColor3f(0.0, 1.0, 0.0);    glVertex3f(0.0, 0.0, 0.0);    glVertex3f(0.0, 1.0, 0.0);
    glColor3f(0.0, 0.0, 1.0);    glVertex3f(0.0, 0.0, 0.0);    glVertex3f(0.0, 0.0, 1.0);
    glEnd();
}

void draw() {
    drawScene();
    drawAxes();
}

nt main (int argc, char **argv){
    /*init(argc, argv);
    glutDisplayFunc(display);
    glutIdleFunc(display);
    glutMainLoop();*/
}
```

실행 화면



□ 상태(Function)

* 메인함수의 func 기능
* glutDisplayFunc(display); : 유저가 기능을 수행하고 있는 상태일 때를 그려줌
* glutIdleFunc(display); : 유저가 아무것도 하지 않은 상태에서 그려줌
* glutReshapeFunc(reshape); : 윈도우의 창 크기가 바뀌어도 모델의 크기는 바뀌지 않음
  - glViewport(0, 0, w, h); : 윈도우 창의 비율을 바꾼 상태의 비율을 맞춰줌

reshape

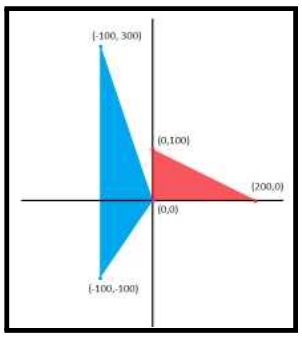
```
void reshape(int w, int h){
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        float asp = float(w)/h;
        glOrtho(-asp*2,asp*2,-1*2,1*2,-2,2);
        glViewport(0, 0, w, h);
}
```

* glutKeyboardFunc(keyboard); : 윈도우의 창 크기가 바뀌어도 모델의 크기는 바뀌지 않음
  - glPostRedisplay(); : 윈도우를 그려야 할 때만 그려줌 (glutIdleFunc를 안써도됨)

6주차 (16.10.07)

□ 변환

* 어파인(affine) 변환 : 직선은 직선으로, 평행선은 평행선으로 유지
  - 이동(translate) : 주어진 변위 벡터만큼 좌표를 동일하게 옮겨 놓는다
  - 회전(rotate) : 2차원에서는 기준점, 3차원에서는 기준축을 중심으로 주어진 각도만큼 돌아간다.
  - 크기변경(scale) : 각 축 방향으로 주어진 비율에 따라 좌표 값이 커지거나 줄어든다.



* 동차 좌표계(Homoheneous coordinate) : 이동과 회전 모두 4x4 행렬의 곱으로 표현 가능
  - n차원의 사영공간을 n+1 차원의 좌표로 나타내는 좌표계
  - 무한의 위치에 있는 점을 유한 좌표로 표현하는 데 적합
  - (x, y, z) => (kx, ky, kz, k) // k가 1일 때, (x, y, z, 1) // k가 2일 때, (2x, 2y, 2z, 2)
  - 사영기하학에서 사용하며, 4차원 좌표에서 3차원으로 한 점을 바라보는 떨어지는 점이기 때문에 (1,1,1,1)과 (2,2,2,2), (3,3,3,3)는 같은 점이 된다. // (x, y, z, 0)는 무한
  * 동차좌표계를 사용하는 이유
    - 3차원 데카르트 좌표를 사용할 경우 이동은 벡터의 덧셈으로 표현되고, 회전은 3 x 3 행렬의 곱으로 표현
    - 이동과 회전이 누적되면 벡터 덧셈과 행렬 곱셈이 연속적 적용됨
    - 동차좌표를 사용하면 이동과 회전 모두 4 x 4 행렬의 곱으로 표현 가능
    - 누적된 이동, 회전 변환을 하나의 행렬로 표현 가능
  - 동차좌표계를 사용한 이동 행렬곱 : $\begin{pmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$

* CTM (Current Transform Matrix)

- 처음 카메라가 만들어진 것 : I
- 카메라를 이동하여 그려줌 : C (glLookat(x,y,z eye, x,y,z at, x,y,z up))
  - 물체 이동 : T
  - 물체 회전 : R
- ex) glTranslatef(1, 0, 0)일 경우 : I*T(1,0,0)
- ex) I(CTM) * R(90도) * T(1,0,0) 일 경우/ y방향의 1위치에 있는 위로보는 주전자가 된다.

* 물체가 Rotatef() 되면 축자체도 전부 회전한다.

* gluRotatef() : 물체 회전 (축도 같이 회전함)

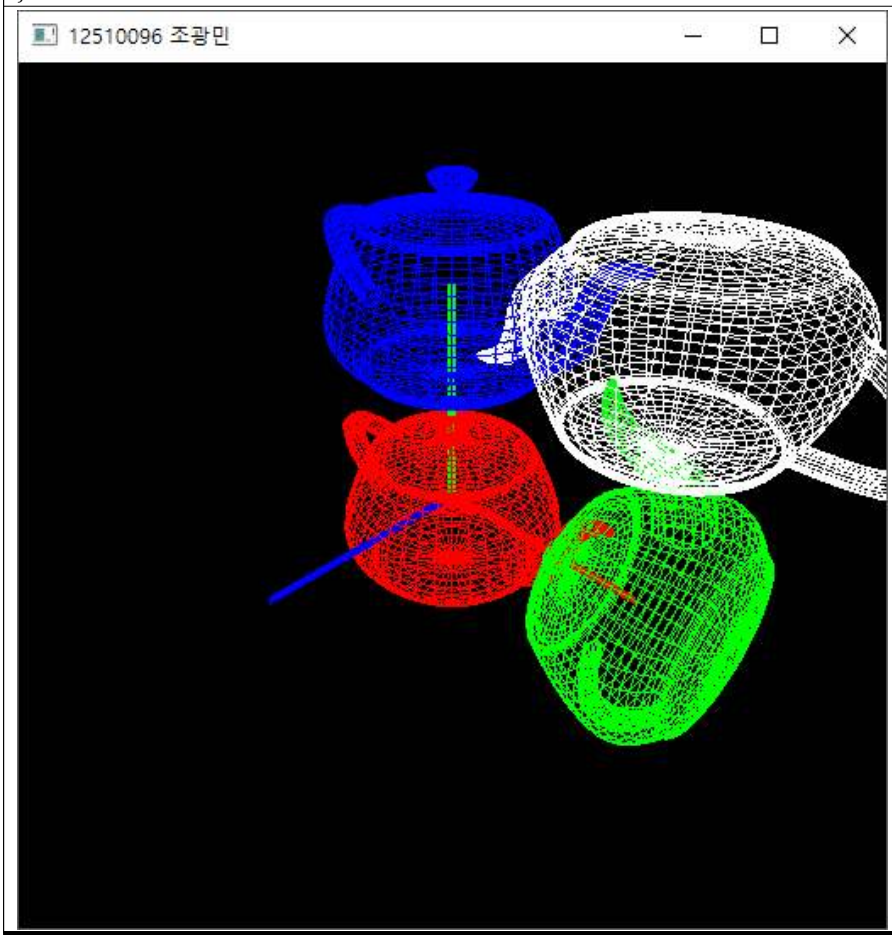
- gluRotatef(theta(각도), 1, 1, 0);

glRotatef() 실습 코드 - 1

```
#define GLUT_DISABLE_ATEXIT_HACK

#include <Windows.h>
#include <gl/GL.h>
#include <gl/glut.h>
#include <math.h>

float range = 1.0;
float aspRatio = 1.0;
float dX = 0.0;

void drawAxis(){
	glBegin(GL_LINES);

	glColor3f(1, 0, 0);
	glVertex3f(0, 0, 0);
	glVertex3f(1, 0, 0);

	glColor3f(0, 1, 0);
	glVertex3f(0, 0, 0);
	glVertex3f(0, 1, 0);

	glColor3f(0, 0, 1);
	glVertex3f(0, 0, 0);
	glVertex3f(0, 0, 1);

	glEnd();
}

void myDisplay() {
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
	gluLookAt(2, 2, 2, 0, 0, 0, 0, 1, 0);

	glClear(GL_DEPTH_BUFFER_BIT | GL_COLOR_BUFFER_BIT);

	glRotated(angle, ax, ay, 0.0);

	glLineWidth(5);
	drawAxis();
	glLineWidth(1);

	glColor3f(1, 0, 0);
	glutWireTeapot(0.5); // 스페어 (반지름, 경도, 위도)

	glColor3f(0, 1, 0);
	glTranslatef(1, 0, 0);
	glRotatef(90, 0, 0, 1);
	glutWireTeapot(0.5); // 스페어 (반지름, 경도, 위도)

	glColor3f(1, 1, 1);
	glTranslatef(1, 0, 0);
	glRotatef(90, 0, 0, 1);
	glutWireTeapot(0.5); // 스페어 (반지름, 경도, 위도)

	glColor3f(0, 0, 1);
	glTranslatef(1, 0, 0);
	glRotatef(180, 0, 0, 1);
	glutWireTeapot(0.5); // 스페어 (반지름, 경도, 위도)


	glutSwapBuffers();
}
```

glRotatef() 실습 코드 - 2

```
void SetCamera() {
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluPerspective(60, aspRatio, 0.1, 1000);
}

void reshape(int w, int h){
	aspRatio = float(w)/h;
	SetCamera();
	glViewport(0, 0, w, h);
}

int main(int argc, char **argv) {
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DOUBLE | GLUT_DEPTH | GLUT_RGBA);
	glutInitWindowPosition(0, 0);
	glutInitWindowSize(512, 512);
	glutCreateWindow("12510096 조¶광쐼´민öÍ");
	glEnable(GL_DEPTH_TEST);

	glClearColor(0.0, 0.0, 0.0, 1.0);

	glutDisplayFunc(myDisplay);
	glutIdleFunc(myDisplay);
	glutReshapeFunc(reshape);

	glutMainLoop();

	return 0;
}
```



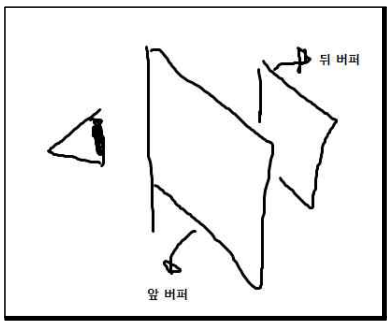
□ 깊이 버퍼와 이중 버퍼

* 깊이 버퍼 사용

* gluInitDisplayMode(GLUT_SINGLE | GLUT_DEPTH | GLUT_RGBA) : 깊이 버퍼를 쉽게 사용할 수 있도록 지원
  - GLUT_DEPTH : 깊이 테스트 작업을 수행하지 않는 것이 디폴트
* glEnable(GLUT_DEPTH_TEST) : 깊이 테스트를 수행 (앞, 뒤에 그려질 것을 설정해줌)
* glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  : 새로운 그리기 전 색상 버퍼를 깨끗이 지우듯, 깊이 버터의 내용도 깨끗하게 지움
* 단계 : main의 glutIUnitDisplayMode에 GLUT_DEPTH 버퍼 선언
  - main에 glEnable(GL_DEPTH_TEST); 로 깊이를 존재하게 함
  - Display에 GL_DEPTH_BUFFER_BIT 로 윈도우를 지워줌

* 깊이 버퍼와 이중 버퍼 사용

* 단일 버퍼 환경은 애니메이션 등이 있을 때 깜빡임 발생
* glubInitDisplayMode(GLUT_DOUBLE | GLUT_DEPTH | GLUT_RGBA);
  : 앞 버퍼(front buffer)와 뒷 버퍼(back burffer)의 이중 구조
* glutSwapBuffers(); : 디스플레이 장치로 프레임 버퍼를 보내는 것은 glFlush가 아니라 glutSwapBuffer를 이용
  - 앞쪽 버퍼는 손을 대지 않고 뒤쪽 버퍼에만 손을 대기 때문에 자연스럽게 애니메이션을 보여줌



네 개의 면으로 상자 모양 그리기

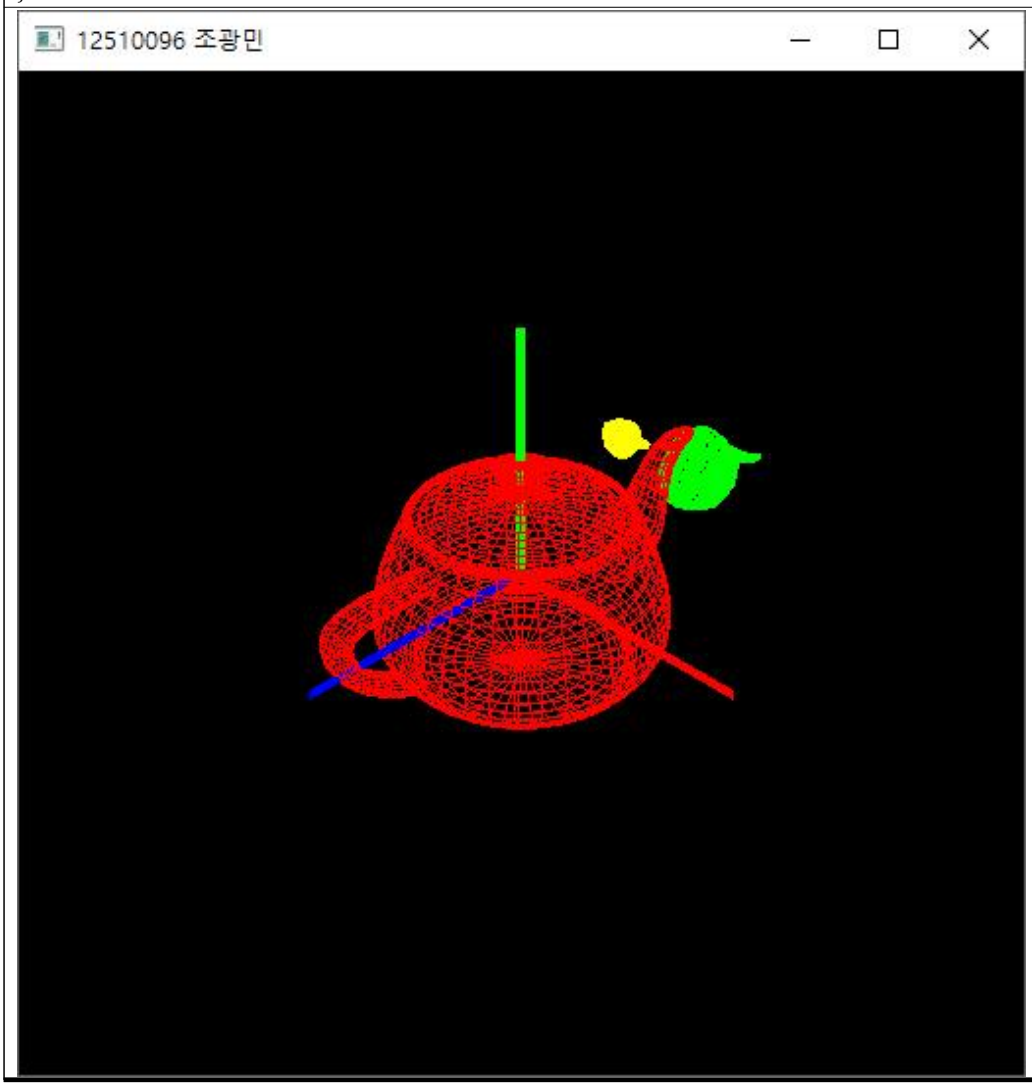
```
#include <stdiob.h>
#include <stdio.h>
#dsfine GLUT_DISABLE_ATEXIT_HACK
#include <Gl/glut.h>
#include <Gl/gl.h>
#include <Gl/glu.h>

void init(int argc, char **argv) { // 윈도우 생성, 버퍼 설정
	glutInit(&argc, argv);

	// 이중 버퍼링, RGBA 색상 버퍼와 함께, 깊이 버퍼를 준비하도록 함
	glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
	glutInitWindowPosition(0, 0);
	glutInitWindowSize(512, 512);
	glutCreateWindow("DEPTH BUFFER");
	glClearColor(0.7, 0.7, 0.7, 1.0);

	//깊이 버퍼 검사를 활성화
	glEnable(GL_DEPTH_TEST);

	// 카메라 투영 특성 설정 (glPerspective 사용), 이 때는 GL_PROJECTION 행렬 모드여야 함
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluPerspective(60, 1.0, 0.1, 100.0);
}

void drawScene(){


}

void drawAxes(){ // 3차원 좌표 선을 생성
	glBegin(GL_LINES);
	glColor3f(1.0, 0.0, 0.0);	glVertex3f(0.0, 0.0, 0.0);	glVertex3f(1.0, 0.0, 0.0);
	glColor3f(0.0, 1.0, 0.0);	glVertex3f(0.0, 0.0, 0.0);	glVertex3f(0.0, 1.0, 0.0);
	glColor3f(0.0, 0.0, 1.0);	glVertex3f(0.0, 0.0, 0.0);	glVertex3f(0.0, 0.0, 1.0);
	glEnd();
}

void draw() {
}
```

glRotatef() 실습 코드2 - 1 (행성 자전, 공전)

```
#define GLUT_DISABLE_ATEXIT_HACK

#include <Windows.h>
#include <gl/GL.h>
#include <gl/glut.h>
#include <math.h>

float range = 1.0;
float aspRatio = 1.0;
float dX = 0.0;
float rt = 0, grt=0;

void drawAxis(){
	glBegin(GL_LINES);

	glColor3f(1, 0, 0);
	glVertex3f(0, 0, 0);
	glVertex3f(1, 0, 0);

	glColor3f(0, 1, 0);
	glVertex3f(0, 0, 0);
	glVertex3f(0, 1, 0);

	glColor3f(0, 0, 1);
	glVertex3f(0, 0, 0);
	glVertex3f(0, 0, 1);

	glEnd();
}

void myDisplay() {
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
	gluLookAt(2, 2, 2, 0, 0, 0, 0, 1, 0);

	glClear(GL_DEPTH_BUFFER_BIT | GL_COLOR_BUFFER_BIT);

	glLineWidth(5);
	drawAxis();
	glLineWidth(1);

	glPushMatrix();
	glColor3f(1, 0, 0);
	glRotatef(rt, 0, 1, 0);
	glutWireSphere(0.6, 20, 20); // 스쮳¬페쵚어úi (반öY지o름A×, 경튜¡도伊ì, 위§도伊ì)
	glPopMatrix();

	rt += 1;
	grt += 2;
	if (rt >= 360){
		rt = 0;
	}
	if (grt >= 360){
		grt = 0;
	}


	// 수ùo성彼¬
	glPushMatrix();
	glColor3f(0, 0, 1);
	glRotatef(rt, 0, 1, 0);
	glTranslatef(1.2, 0, 0);
	glRotatef(rt, 1, 0, 0);
	glutWireSphere(0.2, 10, 10); // 스쮳¬페쵚어úi (반öY지o름A×, 경튜¡도伊ì, 위§도伊ì)

	glRotatef(-rt, 1, 0, 0);
	glColor3f(1, 1, 1);
	glRotatef(rt, 0, 1, 0);
	glTranslatef(0.5, 0, 0);
	glRotatef(rt, 1, 0, 0);
	glutWireSphere(0.1, 10, 10); // 스쮳¬페쵚어úi (반öY지o름A×, 경튜¡도伊ì, 위§도伊ì)
	glPopMatrix();

	// 금쬢성彼¬
	glPushMatrix();
	glColor3f(1, 1, 0);
	glRotatef(grt, 0, 1, 0);
	glTranslatef(2, 0, 0);
	glRotatef(grt*2, 1, 0, 0);
	glutWireSphere(0.4, 10, 10); // 스쮳¬페쵚어úi (반öY지o름A×, 경튜¡도伊ì, 위§도伊ì)

	glRotatef(-grt, 1, 0, 0);
	glColor3f(1, 1, 1);
	glRotatef(grt, 0, 1, 0);
	glTranslatef(0.5, 0, 0);
	glRotatef(grt, 0, 1, 0);
	glRotatef(grt, 0, 0, 1);
	glutWireSphere(0.1, 10, 10); // 스쮳¬페쵚어úi (반öY지o름A×, 경튜¡도伊ì, 위§도伊ì)
	glPopMatrix();

	glutSwapBuffers();
}
```

glRotatef() 실습 코드2 - 2 (행성 자전, 공전)

```
void SetCamera() {
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	//glOrtho(-aspRatio * range + dX, aspRatio * range + dX, -range, range, -2, 2);
	gluPerspective(60, aspRatio, 0.1, 1000);
}

void reshape(int w, int h){
	aspRatio = float(w)/h;
	SetCamera();
	glViewport(0, 0, w, h);
}

int main(int argc, char **argv) {
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DOUBLE | GLUT_DEPTH | GLUT_RGBA);
	glutInitWindowPosition(0, 0);
	glutInitWindowSize(512, 512);
	glutCreateWindow("12510096 조¶광쐼´민öÍ");
	glEnable(GL_DEPTH_TEST);

	glClearColor(0.0, 0.0, 0.0, 1.0);

	glutDisplayFunc(myDisplay);
	glutIdleFunc(myDisplay);
	glutReshapeFunc(reshape);

	glutMainLoop();

	return 0;
}
```



□ display() 함수 추가 코드

- glBegin(프리미티브) ~ glEnd() : 그림을 그리는 것
- glFushMatrix() : 이전 변환행렬을 저장 (push ~ pop 내부에 사용한 함수는 push~pop 내부에서만 적용)
- glPopMatrix() : Push로 저장된 행렬 값을 가져옴
- glRotate*(), glTranslate*(), glScale*() 등의 함수의 기능을 활성화하기 위해 위의 두 함수를 지정

display() 함수 추가 코드

```
// global variables
GLfloat   xrot, yrot, zrot;
GLfloat   red=1.0, green=1.0, blue=1.0, alpha=1.0;

void myDisplay() {
        char info[128];

        glClear(GL_DEPTH_BUFFER_BIT | GL_COLOR_BUFFER_BIT);

        /* Translate & rotate the image */
        glPushMatrix();
        glTranslatef(1.0, 1.0, 0.0); // move rotation axis to triangle center
        glRotatef(xrot, 1.0f, 0.0f, 0.0f); // rotate the image
        glRotatef(yrot, 1.0f, 0.0f, 0.0f); // rotate the image
        glRotatef(zrot, 1.0f, 0.0f, 0.0f); // rotate the image
        glTranslatef(-1.0, -1.0, 0.0); // restore axis origin

        /* Draw triangle */
        glColor4f(red, green, blue, alpha); // color set as RGBA

        glBegin(GL_TRIANGLES);
        glVertex3f(0.0f, 0.0f, 0.0f);
        glVertex3f(2.0f, 1.0f, 0.0f);
        glVertex3f(1.0f, 2.0f, 0.0f);
        glEnd();

        glPopMatrix(); // restore the coord. matrix

        sprintf(info, "x=%.1f, y=%.1f, z=%.1f, mag=%.2f", xrot, yrot, zrot, magfac);
        glutSetWindowTitle(info);

        glFlush();
}
```

* display() 함수 설명

- glutSetWindowTitle() : 'g' key를 누르고, PgUp key로 삼각형을 회전시키고, Window Title창에 x,y,z축에 대한 회전각이 나타남
- 방향키 사용으로 회전, +, - 로 이미지의 크기를 확대 및 축소
- glRotate*(angle, x, y, z); : 좌표계의 축에 따라 주어진 각만큼 이미지를 회전시킴
- glutGetModifiers() : callback 함수에서 SHIFT, CTRL, ALT를 동시에 눌러 이벤트를 Handle할 경우 GLUT_ACTIVE_SHIFT, GLUT_ACTIVE_CTRL, GLUT_ACTIVE_ALT mode를 연계해서 사용, Alt+r, Alt+b를 이용하여 색상 변화
- 3개의 키 사용 코드

Special key 3개의 키 사용

```
mod = glutGetModifiers();
if (mod &&(GLUT_ACTIVE_CTRL | GLUT_ACTIVE_ALT)) {
```

* Advanced Keyboard 기능

- 화살표 같은 key를 계속 눌러 자동 반복 실행을 할 때, delay현상의 문제점을 해결해주는 함수

glutSetRepeat (전역 기반으로 작용)

```
int  glutSetRepeat( int repeatMode );
```

Parameters:

- GLUT_KEY_REPEAT_OFF - 자동반복 Mode 기능 비활성
- GLUT_KEY_REPEAT_ON - 자동반복 Mode 기능 활성
- GLUT_KEY_REPEAT_DEFAUT - default 상태로 자동반복 Mode 를 Reset

- Application으로부터 하나가 아닌 모든 Window의 반복 기능에 영향을 준다. 따라서 이 함수를 자동 mode의 비활성화로 사용할 때는 Application을 끝내기 전 default 상태로 복원하는 것이 편리

glutIgnoreKeyRepeat

```
int glutIgnoreKeyRepeat( int repeatMode );
```

Parameters:

- repeatMode - 0 이면 자동반복 mode 활성, 0 이 아니면 자동반복 mode 비활성

- key 반복이 일어날 때의 callback 받는 것을 비활성화 시켜, 다른 Application에 영향을 주지 않고 안전하게 key 누름을 무시해야 할 때 사용

키 입력

```
void glutKeyboardUpFunc( void (*func)(unsigned char key,int x,int y) );
void glutSpecialUpFunc( void (*func)(int key,int x, int y) );
```

Parameters:

- func - callback 함수 이름

- key 반복이 일어날 때 callback 받는 것을 멈출 것인데, 만약 key를 누르고 있는 동안에만 어떤 Action이 실행되기를 원한다면, 그 key가 누르기를 해지하는 때를 알아야함.
- GLUT는 key가 해지되었을 때를 위한 두 개의 register callback 기능을 제공함

□ OPENGL API □

□ Keyboard Envent : 키보드 이벤트는 2개의 callback 함수가 있다.

* glutKeyboardFunc(keyboard); // 일반 키보드처리 (main에 작성)
  - void keyboard(unsigned char key, int x, int y){ switch(){ } glutPostRedisplay(); }

* glutKeyboardUpFunc(keyboard); // 키보드를 뗐을 때 이벤트 발생
  - void keyboard(unsigned char key, int x, int y){ switch(){ } glutPostRedisplay(); }

* glutSpecialFunc(special); // 특수 키보드 처리 (main에 작성)
  - void special(int key, int x, int y){ switch(){ } glutPostRedisplay(); }

* glutSpecialUpFunc(special); // 특수 키보드 처리 (main에 작성)
  - void special(int key, int x, int y){ switch(){ } glutPostRedisplay(); }

  * glutPostRedisplay(); : 윈도우를 그려야 할 때만 그려줌 (glutIdleFunc()를 안 써도 됨)

* **keyboard()의 switch (key) { case : }에 사용할 입력 문자**
  - **'a' ~ 'z' : 일반 키 입력**
  - **GLUT_ACTIVE_ALT, GLUT_ACTIVE_CTRL : 특수 키를 누른 상태**
* **special()의 switch (key) case : 에 사용할 입력 문자**
  - **GLUT_KEY_F1 : GLUT_KEY_특수키 입력**

□ Mouse Envent : 마우 이벤트는 3개의 callback 함수가 있다.

* glutMouseFunc(mouse); // 마우스 클릭 시 발생되는 이벤트
  - void mouse(int button, int direction, int x, int y){ switch(){ } glutPostRedisplay(); }

* glutMotionFunc(motion); // 마우스 클릭 후 이동 시 발생되는 이벤트
  - void motion(int x, int y){ 예제 참조 }

* glutPassiveMotionFunc(mouse); // 마우스 클릭을 하지 않고 이동 시 발생되는 이벤트

* glutMouseWheelFunc(mousewheel); // 마우스 휠을 사용한 이벤트
  - void mousewheel(int wheel, int direction, int x, int y){ 예제 참조 }

* **mouse()의 switch (key) { case : }에 사용할 입력 문자**
  - **GLUT_DOWN : 마우스를 클릭 시**

□ Keyboard Envent 예제

keyboard() 함수 - 일반 워드

```
void  keyboard(unsigned char key, int x, int y)
{
        int    mod;

        switch (key) {
        case 'r':
                red = 1.0f; green = 0.0f; blue = 0.0f;
                mod = glutGetModifiers();
                if (mod && GLUT_ACTIVE_ALT) { // 알트키를 누른상태로 키 입력
                        red = 0.5f;
                }
                break;
        case 'g':
                red = 0.0f; green = 1.0f; blue = 0.0f;
                mod = glutGetModifiers();
                if (mod && GLUT_ACTIVE_ALT) {
                        green = 0.5f;
                }
                break;
        case 'b':
                red = 0.0f; green = 0.0f; blue = 1.0f;
                mod = glutGetModifiers();
                if (mod && GLUT_ACTIVE_ALT) {
                        blue = 0.5f;
                }
                break;
        case 27:
                exit(0);
                break;
        default:
                red = 1.0f; green = 1.0f; blue = 1.0f; alpha = 1.0f;
                break;
        }
        glutPostRedisplay();
}
```

special() 함수 - 특수 키

```
void  special(int key, int x, int y)
{
        switch (key) {
                //  select image view mode when reshape the window
        case GLUT_KEY_F1: // both object shape & size is not changed
                viewmode = 1; // Ortho view mode
                break;
        case GLUT_KEY_F2: // both object shape & size is chaned
                viewmode = 2; // Ortho view mode
                break;

                //  spin key for image rotation
        case GLUT_KEY_UP:
                xrot -= 2.0f;
                if (xrot < -360.0f) xrot += 360.0f;
                break;
        case GLUT_KEY_DOWN:
                xrot += 2.0f;
                if (xrot > +360.0f) xrot -= 360.0f;
                break;
        case GLUT_KEY_PAGE_DOWN:
                zrot -= 2.0f;
                if (zrot < -360.0f) zrot += 360.0f;
                break;
        case GLUT_KEY_PAGE_UP:
                zrot += 2.0f;
                if (zrot > +360.0f) zrot -= 360.0f;
                break;
        case '+':
                magfac += 0.02;
                break;
        case '-':
                magfac -= 0.02;
                break;

        case GLUT_KEY_F10:
                glutFullScreen();
                break;
        case GLUT_KEY_F9:
                glutReshapeWindow(wwidth, wheight);
                glutPositionWindow(100, 100);
                break;
        default:
                break;
        }
        glutPostRedisplay();
}
```

□ Mouse Envent 예제

mouse() 함수 - 마우스 클릭

```
void mouse(int button, int direction, int x, int y)
{
	switch (direction){
	case GLUT_DOWN:
		// 마우스 버튼을 누른 위치를 기록
		cx = x;
		cy = y;
		// 표시하고 있는 물체의 회전 각을 기록
		ca = angle;
		break;
	default:
		break;
	}
}
```

motion() 함수 - 마우스 클릭 후 이동 시

```
void motion(int x, int y){
	double dx, dy, a;

	// 마우스 포인터 위치의 끌기 시작 위치에서의 변위
	dx = (x - cx) * sx;
	dy = (y - cy) * sy;

	// 마우스 포인터 위치의 끌기 시작 위치에서의 거리
	a = sqrt(dx * dx + dy * dy);

	if (a != 0.0){
		// 거리를 각도로 환산하여 드래그 시작시의 회전 각에 가산
		angle = fmod(ca + SCALE * a, 360.0);

		// 마우스 포인터의 변위에서 회전축 벡터를 요청
		ax = dy / a;
		ay = dx / a;
		az = 0.0;

		// 도형의 재 묘화 (윈도우를 그려야 할 때만 그려줌)
		glutPostRedisplay();
	}
}
```

mousewheel() 함수 - 마우스 휠

```
void mousewheel( int wheel, int direction, int x, int y)
{
	switch (direction){
	case GLUT_DOWN:
		// 마우스 버튼을 누른 위치를 기록
		cx = x;
		cy = y;
		// 표시하고 있는 물체의 회전 각을 기록
		ca = angle;
		break;
	default:
		break;
	}
}
```

□ 조명

* 용어 설명

* Material : 어떤 메시가 빛을 받는다고 할 때, 마테리얼에서해당 광원으로부터 빛을 받으면 어떻게 발산할 것인지를 설정해준다. // 물체가 빛을 받으면 그 빛을 반사하고 반사한 빛이

우리 눈에 인지된다. 그리고 반사되는 정도는 물체 마다 다 다르다 (ex) 고무, 거울, 나무 등)
3D에서는 이런 정도를 Ambient, Diffuse, Specular 라는 속성으로 이 재질을 표현한다.

* Diffuse : 표면이 반사하는 난반사광
* Ambient : 표면이 반사하는 환경광
* Specular : 표면이 반사하는 정반사광
* Emissive : 자체 발광
* Power : 발광 정도

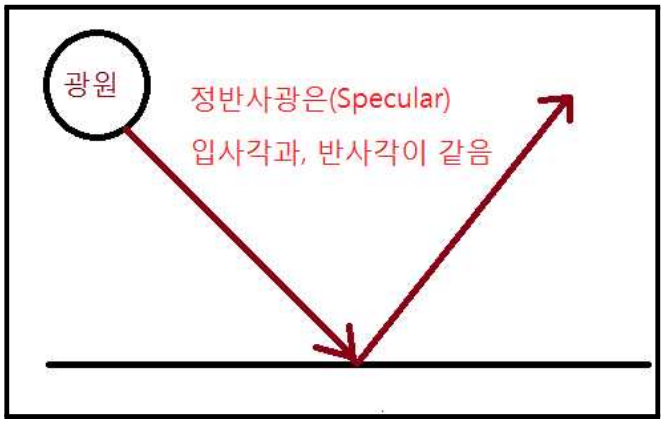
* 다이렉트X의 개념 - 출처 : http://blog.naver.com/znfgkro1/220185340157
  * D3DLIGHTTYPE type : 광원 종류
  * D3DCOLORVALUE Diffuse : 표면이 반사하는 난반사광 색상
  * D3DCOLORVALUE Ambient : 표면이 반사하는 환경광 색상
  * D3DCOLORVALUE Specualr : 표면이 반사하는 정반사광 색상
  * D3DVECTOR Position : 광원의 위치 지정 벡터 (방향성광원은 상관없음- 거리에 따라 빛의 세기가 변하지 않음)
  * D3DVECTOR Direction : 빛이 향하는 방향 지정 (점광원은 필요 없음 - 주변으로 다 비치기 때문)
  * float Range; : 빛이 소멸되는 거리 (가로등 하나가 모든 곳을 다 비추진 않음)
  * float Falloff; : (스포트에서만 사용) 밝은 원과 주변 어두운 원의 밝기 차이
  * float Attenuation0; : 상수 감소
  * float Attenuation1; : 선형 감소
  * float Attenuation2; : 이차 감소
  * float Theta; : 스포트 광원에서 가장 밝은 부분 비추는 각도
  * float Phi; : 스포트 광원에서 전체를 비추는 각도
  * 화면에 표현되는 색상 : 빛 * 물체의 재질 * 텍스쳐(존재할 시)

* 조명 - 조명과 재질 적용할 데이터
  - 조명의 위치는 동차좌표(4x4 행렬)로 표현 // 누적된 이동, 회전 변환을 하나의 행렬로 표현 가능
  - 마지막 성분이 1이면 점광원이고, 0이면 방향광원(directional light source)
    - 점광원 : 한 점에서 시작해서 퍼져나감
    - 방향광원(집중광원) : 해당 지역만 밝게 비춰줌

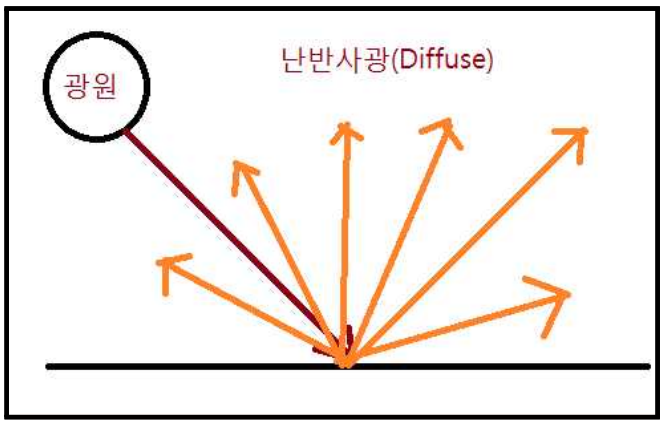
조명 - 전역 변수

```
//재질의 정반사, 난반사, 주변광, 반질거림 특성으로 사용될 데이터
GLfloat matSpec[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat matDiff[] = { 1.0, 1.0, 0.0, 1.0 };
GLfloat matAmbi[] = { 0.5, 0.1, 0.1, 1.0 };
GLfloat matShin[] = { 127.0 };

// 광원의 정반사, 난반사, 주변광 특성으로 사용될 데이터
//GLfloat light[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat lit_specular[] = { 1.0, 1.0f, 1.0f, 1.0f };
GLfloat lit_diffuse[] = { 0.0, 1.0f, 1.0f, 1.0f };
GLfloat lit_ambient[] = { 0.5, 1.0f, 1.0f, 1.0f };

// 광원의 위치로 사용될 데이터
GLfloat lightPos[] = { 1.0, 1.0, 1.0, 1.0 };//{ -1.0, 1.0, 1.0, 1.0 };
```

* 반사광 : 어떤 광원에 의해 어떤 오브젝트가 빛을 받았을 때, 그 빛을 반사시키는 빛
  - 광원에 의해 입사광이 있으면, 반사광이 있다.
  * 반사광의 종류
    - 정반사광(Specular) : 어떤 광원(Spot, Dir, Point)에 의해 어떤 오브젝트가 빛을 받았을 때, 그 빛을 한 방향으로 반사시킴(하이라이트 표현)



- 난반사광(Diffuse) : 어떤 광원(Spot, Dir, Point)에 의해 어떤 오브젝트가 빛을 받았을 때, 그 빛을 여러 방향으로 고르게 반사되는 빛



* 실제 조명과 재질에 적용
  * glLight* : 조명의 특성을 설정하는 함수
  * glMaterial* : 재질의 특성을 설정하는 함수

조명 - LightSet() 함수와 LightPosition() 함수

```
// 조명과 재질의 특성을 준비된 데이터로 설정하는 함수
void LightSet() {
        glMaterialfv(GL_FRONT, GL_SPECULAR, matSpec);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, matDiff);
        glMaterialfv(GL_FRONT, GL_AMBIENT, matAmbi);
        glMaterialfv(GL_FRONT, GL_SHININESS, matShin);

        glLightfv(GL_LIGHT0, GL_SPECULAR, lit_specular);
        glLightfv(GL_LIGHT0, GL_DIFFUSE, lit_diffuse);
        glLightfv(GL_LIGHT0, GL_AMBIENT, lit_ambient);

        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
}

// 조명의 위치를 설정하는 함수
void LightPosition() {
        glLightfv(GL_LIGHT0, GL_POSITION, lightPos);
}
```

* 조명 - 렌더링
  * LightSet(); : 조명의 특성과 재질의 특성을 설정
  * LightPositioning(); : 점광원의 위치를 설정

조명 - 렌더링

```
// 메인에 추가
void GLInit() {
        glClearColor(0.0, 0.0, 0.0, 0.0);

        // 조명의 특성과 재질의 특성을 설정
        LightSet();
        glEnable(GL_DEPTH_TEST);
}

// 디스플레이
void display() {
        [[GL_MODELVIEW 모드 설정]]
        gluLookAt (0, 1, 2, 0, 0, 0, 0, 1, 0);

        // 점광원의 위치를 설정
        LightPositioning();
        gluSolidTeapot(0.5);
        gluSwapBuffers();
}
```

(a) 주변광 포함

(b) 주변광 제외

* 조명 - 반질거림 조정

* **glMaterialf(GL_FRONT, GL_SHININESS, 4.0f + 123.0f * (i / 4.0));**
  - **재질의 반질거림을 변경하여 설정**

조명 - 렌더링

```
for (int i=0; i<5; i++){
        // 재질의 반질거림을 변경하여 설정하고 주전자를 그림
        glMaterialf(GL_FRONT, GL_SHININESS, 4.0f+123.0f*(i/4.0));
        for (int j=0; j<5; j++){
                glPushMatrix();
                glTranslated(float (i)+0.5, float (j)+0.5, 0.5);
                glRotated(45, 1, 1, 1);
                glutSolidTeapot(0.4);
                glPopMatrix();
        }
}
```