Computer Graphics:
OpenGL 기초

2018년도 2학기

OpenGL 이란

- **OpenGL의 역사**
  - 1982년 Silicon Graphics(SGI)사의 워크스테이션용 그래픽스 라이브러리 아이리스 지엘 (Iris GL) 로 시작하여 1992년 OpenGL 1.0으로 출시된 2D 및 3D 그래픽스 프로그램 개발 업계 표준 API
  - 250개 가량의 함수 호출을 이용하여 단순한 기하도형에서부터 복잡한 삼차원 장면 생성
  - CAD, 가상현실, 정보시각화, 비행 시뮬레이션, 컴퓨터 게임 등의 분야에서 활용되고 있음
  - Direct3D와 함께 산업계에 널리 사용되고 있음
  - 오픈지엘 사용하여 개발된 대표적인 게임: 퀘이크, 둠3 등

  - OpenGL Architecture Review Board (http://www.opengl.org)
    - 다양한 플랫폼에서 작동되도록 GL을 수정하여 OpenGL 제정
    - 1992년 OpenGL 1.0 발표 이후, 2002년 7월 OpenGL 1.4, 2004년 OpenGL 2.0, 2006년에 OpenGL 2.1, 2011년 OpenGL 4.2, 2012년 8월 OpenGL 4.3, 2014년 8월 OpenGL 4.5, 2017년 7월 4.6 출시
    - 비영리 기술 컨소시엄인 Khronos group에서 관리

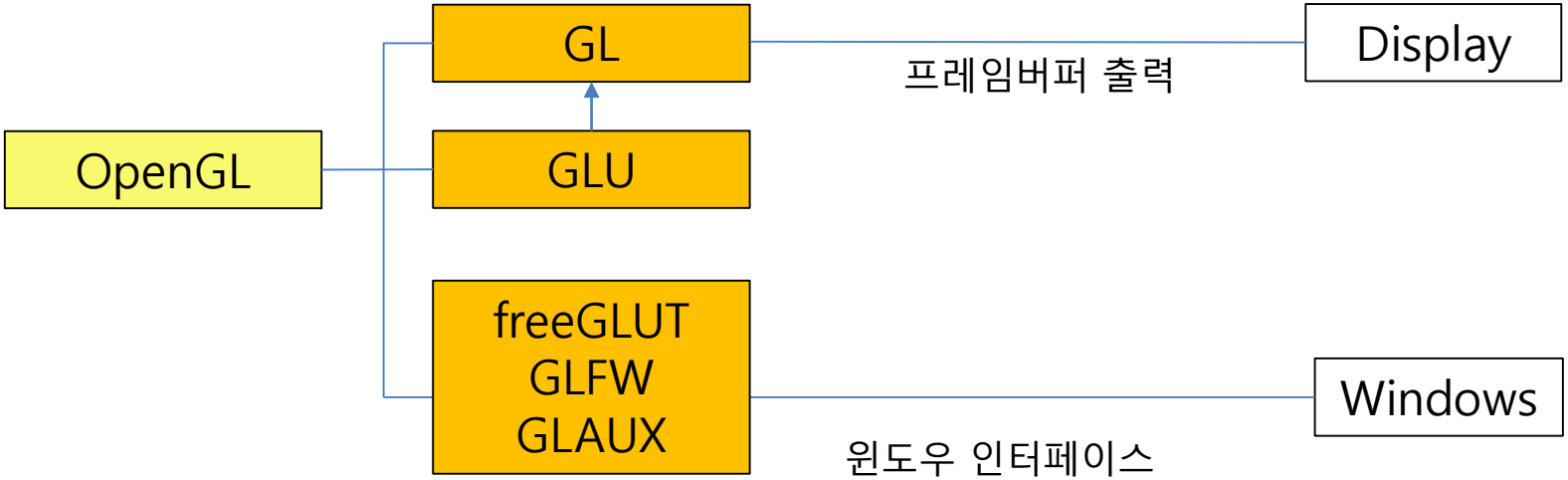
OpenGL 이란

- **OpenGL의 주요한 특징**
  - 그래픽스 하드웨어에 대한 소프트웨어 인터페이스
    - 하드웨어에 독립적인 그래픽스 라이브러리
    - 상위 수준(high-level)의 그래픽스 API로 픽셀 단위가 아니라 객체 단위 프로그래밍 가능 (내부 구현 알고리즘 모르더라도 그래픽스 프로그램을 개발 가능)
  - OpenGL은 플랫폼, 운영체제에 독립적이다.
    - PC 나 워크스테이션 모두에서 가능, 다양한 운영체제 및 호스트 언어를 지원 (C/C++, Fortran, Perl, Java, Visual Basic…)
    - 윈도우 시스템과 관련된 함수는 없다.
    - 윈도우 시스템 관련 함수들은 유틸리티 툴킷에 존재한다.
  - 다양한 그래픽스 기능의 지원으로 응용 소프트웨어 개발이 용이하다.
    - 기본적인 2D 및 3D 그래픽스 함수에서부터 고급 기능까지 지원
    - 셰이딩, 안티알리아싱, 텍스처 매핑, NURBS, 안개, 알파 블렌딩 등
  - 장점:
    - 안정성, 신뢰성 및 이식성, 유연성, 편리성, 문서화

OpenGL Library의 구성

- 라이브러리 구성
  - **GL (openGL Core Library)**: 렌더링 기능을 제공하는 함수 라이브러리
  - **GLU (openGL Utility Library)**: GL 함수로 작성되어 있는 고급 기능을 제공하는 함수들의 집합

  - openGL 지원 라이브러리
    - **GLUT (openGL Utility Toolkit)**: 윈도우를 생성, 사용자와의 상호작용 처리 함수들을 지원
    - **freeGLUT**: GLUT의 대체 라이브러리로 윈도우를 다루는 함수들 지원.
    - **GLAUX:** 오픈지엘 보조 라이브러리
    - **GLFW (GL Frame Work)**: 오픈지엘을 위한 오픈소스 멀티-플랫폼 라이브러리, GLUT와 같은 역할을 하는 라이브러리로 윈도우 생성, 상호작용 처리 함수들을 지원



OpenGL Library의 구성

- 라이브러리 구성
  - **GL library (OpenGL Main Library)**
    - OpenGL의 메인 라이브러리로서 가장 기본이 되는 함수
    - 기본 도형 그리기, 변환, 조명 및 렌더링 등의 함수들
    - 함수들은 모두 **gl-** 이라는 접두어(prefix)가 붙은 이름을 가진다
  - **GLU library (OpenGL Utility Library)**
    - 반복작업을 단순화시키고 개발을 편리하게 해주는 고급 기능의 유틸리티 함수
    - 공통 객체, 곡선, 곡면, 고급 뷰잉, 행렬 연산 등의 함수가 포함된다.
    - 함수들은 모두 **glu-** 라는 접두어가 붙은 이름을 가진다
  - **freeGLUT library (free OpenGL Utility Toolkit Library)**
    - 인터페이스 툴킷 라이브러리로 GLUT 라이브러리가 1998년 이후 버전 업그레이드가 안되고 있어 GLUT를 대체할 목적으로 개발된 오픈 소스 라이브러리
      - GLUT 라이브러리 외에 추가된 API 함수들이 제공됨
      - 윈도우를 띄울 때, 제일 먼저 glutInit(&argc, argv); 함수를 반드시 호출하여 glut를 초기화 한다.
    - 윈도우 생성, 사용자와의 상호 작용 유틸리티 라이브러리 (제한된 GUI 인터페이스만 가능)
    - **glut-** 라는 접두어가 붙은 이름을 가진다

OpenGL Library의 구성

- 라이브러리 구성
  - Static library file (*.lib): 각각 응용 프로그램을 코딩할 때 필요한 라이브러리
  - Dynamic library file (*.dll): 프로그램이 실행될 때 필요한 라이브러리
  - 해당 라이브러리를 저장하여 GL/GLU 및 freeGLUT 사용

Library	LIB file	Header file	DLL file	함수 접두어
opengl library	opengl32.lib	gl.h	opengl32.dll	gl
opengl library	glu32.lib	glu.h	glu32.dll	glu
utility library	glut32.lib	glut.h	glut32.dll	glut
utility library	freeglut.lib	freeglut.h freeglut_ext.h freeglut_std.h	freeglut.dll	glut

OpenGL 기본 데이터 타입

openGL 데이터 형식	표현	C언어 데이터 형식	접미어
GLbyte	8-bit integer	signed char	b
GLshort	16-bit integer	short	s
GLint, GLsizei	32-bit integer	long	i
GLfloat, GLclampf	32-bit floating point	float	f
GLdouble, Glclampd	64-bit floating point	double	d
GLubyte, GLboolean	8-bit unsigned integer	unsigned char	ub
GLushort	16-bit unsigned integer	unsigned short	us
GLuint, GLenum, GLbitfield	32-bit unsigned integer	unsigned long	ui

- **glVertex3fv (const Glfloat *v)**
  - 세 가지 요소의 배열에 대한 포인터, 요소는 x, y, z축에 대한 좌표값

- **함수 형태**
  - **glColor3f** (…): **<접두어><기본명령><접미어>**
    - <접두어>: 함수가 속한 라이브러리
    - <기본 명령>
    - <접미어>: 인자 개수와 데이터 형식, 생략 가능
    - 예) glColor3f, glColor4f, glColor3i…

첫 번째 예제

- **첫번째 예제: 파란색 바탕의 윈도우 띄우기**

```
#include <GL/freeglut.h>
GLvoid drawScene ( GLvoid );
GLvoid reshape (int w, int h);

void main (int argc, char** argv))                          // 윈도우 출력하고 출력함수 설정
{
        // 초기화 함수들
        glutInit (&argc, argv);                             // glut 초기화
        glutInitDisplayMode (GLUT_SINGLE | GLUT_RGBA);      // 디스플레이 모드 설정
        glutInitWindowPosition ( 100, 100 );                // 윈도우의 위치 지정
        glutInitWindowSize ( 250, 250 );                    // 윈도우의 크기 지정
        glutCreateWindow ( "Example1" );                    // 윈도우 생성 (윈도우 이름)
        glutDisplayFunc ( drawScene );                      // 출력 함수의 지정
        glutReshapeFunc (Reshape);                          // 다시 그리기 함수 지정
        glutMainLoop ();                                    // 이벤트 처리 시작
}

GLvoid drawScene ( )                                        // 출력 함수
{
        glClearColor(0.0f, 0.0f, 1.0f, 1.0f);               // 바탕색을 'blue' 로 지정
        glClear( GL_COLOR_BUFFER_BIT );                     // 설정된 색으로 전체를 칠하기
        glFlush();                                          // 화면에 출력하기
}

GLvoid Reshape (int w, int h)                               // 다시 그리기 함수
{
        glViewport (0, 0, w, h);
}
```

윈도우 띄우기

- 윈도우 만들기 함수: freeGLUT 라이브러리 사용
  - glut 초기화
    - void **glutInit** (int *argc, char **argv);
  - 디스플레이 모드 설정
    - void **glutInitDisplayMode** (unsigned int mode);
      - 컬러모델, 윈도우 버퍼 등 초기의 출력 모드를 결정한다.
      - mode:
        - » GLUT_DOUBLE: 더블 버퍼 윈도우
        - » GLUT_SINGLE: 싱글 버퍼 윈도우 (디폴트 모드)
        - » GLUT_RGBA : RGBA 모드 (디폴트 모드)
        - » GLUT_DEPTH: 깊이 버퍼 윈도우
      - 1개 이상의 모드인 경우 | 연산자로 연결한다.
  - 윈도우의 위치 지정
    - void **glutInitWindowPosition** (int x, int y);
      - 스크린에서 윈도우의 좌측 상단 모서리에 해당하는 위치를 지정한다.
    - void **glutPositionWindow** (int x, int y);
      - 윈도우의 위치 변화

윈도우 띄우기

- 윈도우의 크기 지정
  - void **glutInitWindowSize** (int width, int height);
    - 윈도우의 크기를 픽셀단위로 지정한다.
    - width, height: 윈도우의 폭과 높이 픽셀 값
  - void **glutReshapeWindow** (int width, int height);
    - 윈도우의 새로운 넓이와 높이

- 윈도우 생성 및 파괴
  - int **glutCreateWindow** (char *string);
    - 윈도우를 생성한다.
    - string: 윈도우 이름
  - void **glutDestroyWindow** (int win);
    - 윈도우를 파괴한다.
    - win: 윈도우의 번호

- GLUT 이벤트 프로세싱 루프 실행
  - void **glutMainLoop** ():
    - 지금까지 생성한 윈도우들과 여기에 그린 그림들을 화면에 출력한다. 또한, 이벤트 처리가 시작되고 디스플레이 콜백으로 등록된 함수가 호출된다.
    - 마우스, 키보드 등의 콜백 함수들이 호출된다.
    - 메인 함수는 최소한 한번의 glutMainLoop 함수를 호출해야 한다. (대개 메인 함수의 마무리 부분에서 호출한다.)
  - 윈도우 종료: **glutLeaveMainLoop** ()함수 (glut 지원X)

윈도우 띄우기

- Full screen 만들기
  - void **glutFullScreen** (void);
    void **glutLeaveFullScreen** (void);
    void **glutFullScreenToggle** (void);
    - 전체 화면으로 세팅/해제한다.

- 그외 다양한 윈도우 매니저 함수들,
  - void glutPopWindow(), void glutPushWindow()
  - void glutShowWindow(), void glutHideWindow(), void glutIconifyWindow()
  - void glutSetWindowTitle (char *name), void glutSetIconTitle (char *name)
  - void glutSetCursor (int cursor);

기본 함수들

- 몇 가지 기본 함수들
  - void **glClearColor** (GLclampf r, GLclampf g, GLclampf b, GLclampf a)
    - 윈도우를 clear할 때 사용되는 색 지정 (Glclampf는 0.0 ~ 1.0 사이의 float 값)
    - r, g, b: red, green, blue 값
    - a: alpha 값 (1.0값으로 고정)
  - void **glClear** (GLbitfield flag)
    - 특정 버퍼나 혼합된 버퍼의 영역을 glClearColor에서 선택한 값으로 설정한다.
      - 컬러 버퍼: GL_COLOR_BUFFER_BIT
      - 깊이 버퍼: GL_DEPTH_BUFFER_BIT
      - 누적 버퍼: GL_ACCUM_BUFFER_BIT
      - 스텐실 버퍼: GL_STENCIL_BUFFER_BIT
  - void **glColor3f** (GLfloat r, GLfloat g, GLfloat b);
    - 색을 선택하는 함수로써 불투명도는 지정하지 않는다.
      - 데이터 타입(d/f/i/s/ub/ui/us) 과 파라미터의 숫자 (3/4) 설정할 수 있다.
  - void **glFlush** ();
    - 명령어들을 큐에 저장되었다가 한꺼번에 실행되게 한다.
    - 출력 (그리기) 콜백 함수 마지막에 glFlush 함수를 호출하여 (모든 명령어를 실행되게 한다.
    - **디스플레이 모드가 싱글 버퍼**일 때 사용

Callback 함수

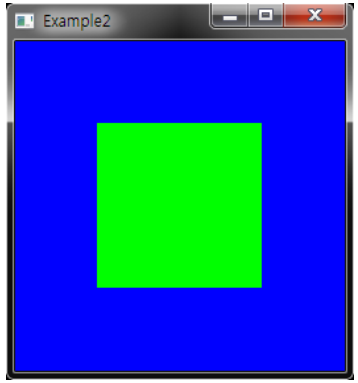
- 콜백 함수
  - 일반 함수 호출은 응용 프로그램이 운영체제에 내장된 함수를 호출하여 원하는 작업을 하는데, 특정 사건이나 메시지가 발생했을 때 거꾸로 운영 체제가 응용 프로그램을 부르는 함수를 콜백함수라고 한다.
  - 함수 이름에 **Func**이란 접미어가 붙는다.

  - 출력 함수 지정
    - void **glutDisplayFunc** (void (*func)(void));
      - 현재 윈도우의 출력 콜백 함수 설정
      - 윈도우의 내용을 다시 출력해야 할 필요가 있을 때마다 이 함수로 등록한 콜백 함수를 호출한다. 장면을 다시 그리는데 필요한 루틴들은 모두 이 함수 안에 넣어둔다.

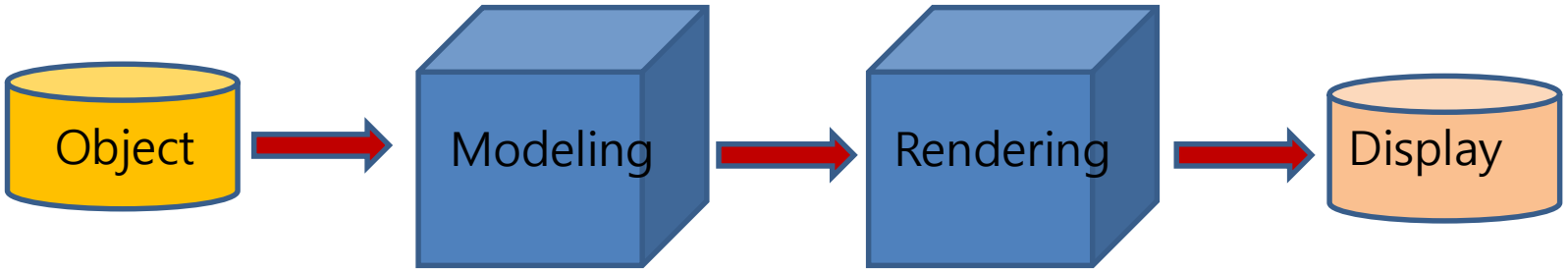
  - 화면 크기 변했을 때 이벤트 처리 함수 지정
    - void **glutReshapeFunc** (void (*func)(int w, int h));
      - 윈도우 크기가 변경될 때 취할 동작을 지정한다.
      - func: 화면 크기가 변했을 때 호출될 콜백 함수 이름
        - » w: 윈도우의 새로운 폭
        - » h: 윈도우의 새로운 높이

샘플 프로그램

```
#include <GL/freeglut.h>
GLvoid drawScene (GLvoid);
GLvoid Reshape (int w, int h);

void main ( int argc, char *argv[] )
{
	glutInit (&argc, argv);
	glutInitDisplayMode ( GLUT_SINGLE | GLUT_RGBA );		// 디스플레이 모드 설정
	glutInitWindowPosition ( 100, 100 );			// 윈도우의 위치지정
	glutInitWindowSize ( 250, 250 );			// 윈도우의 크기 지정
	glutCreateWindow ( "Example2" );			// 윈도우 생성 (윈도우 이름)
	glutDisplayFunc (drawScene);				// 출력 함수의 지정
	glutReshapeFunc (Reshape);
	glutMainLoop();
}

// 윈도우 출력 함수
GLvoid drawScene ( GLvoid )
{
	glClearColor(0.0f, 0.0f, 1.0f, 1.0f);		// 바탕색을 'blue' 로 지정
	glClear( GL_COLOR_BUFFER_BIT );			// 설정된 색으로 전체를 칠하기
	glColor4f(0.0f, 1.0f, 0.0f, 1.0f);		// 그리기 색을 'green' 으로 지정
	glRectf (-0.5f, -0.5f, 0.5f, 0.5f);		// 사각형 그리기
	glFlush(); // 화면에 출력하기
}
GLvoid Reshape (int w, int h)
{
	glViewport(0, 0, w, h);
}
```



오픈지엘 그래픽스 파이프라인

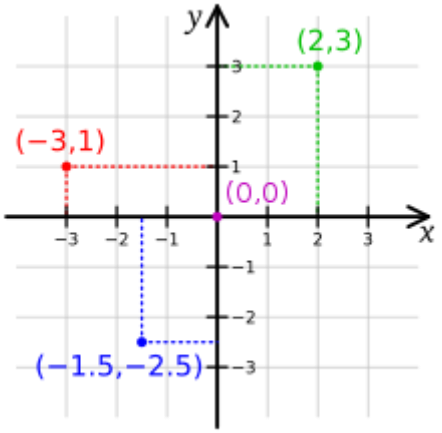


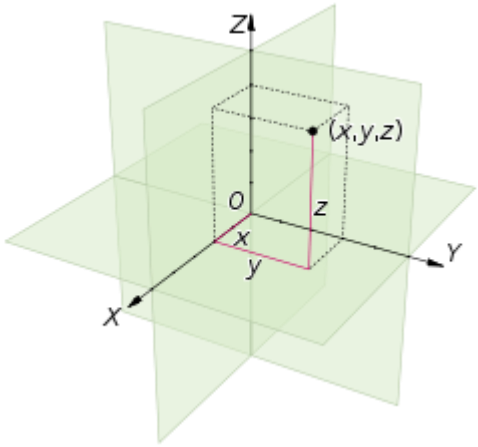
- Wire-frame modeling
- Surface modeling
- Solid modeling

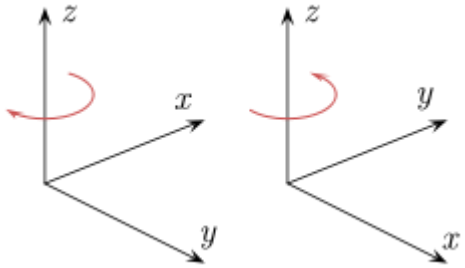
- Hidden surface removal
- Shading
  - Flat
  - Gouraud
  - Phong
- Texture mapping

좌표계 시스템

- 좌표계
  - 평면이나 공간에서 점을 나타내는 모양
  - 2차원 직교 좌표계 시스템 - 3차원 직교 좌표계 시스템



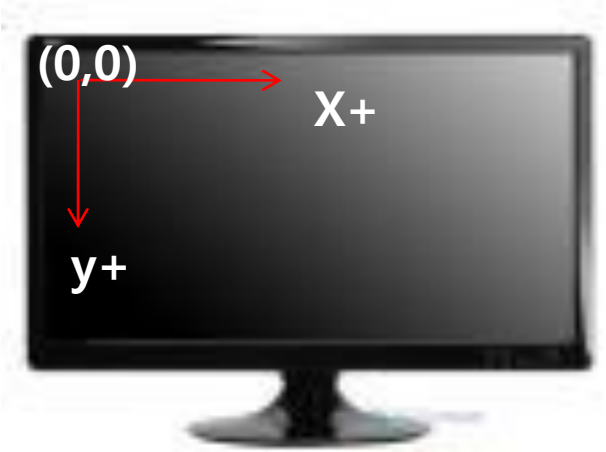




좌측 (왼손 좌표계), 우측 (오른손 좌표계)

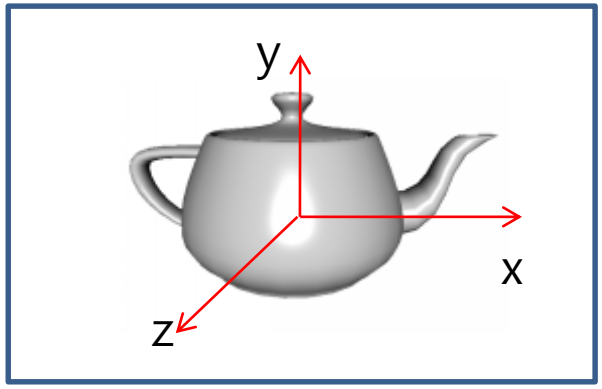
좌표계 시스템

- 윈도우 화면 좌표계 시스템 (Screen Coordinate System)
  - 원점이 화면의 좌측 상단에 위치
  - X축은 오른쪽으로 가면 값이 늘어난다.
  - Y축은 아래쪽으로 가면 값이 늘어난다.



좌표계 시스템

- 오픈지엘 좌표계 시스템
  - 오른손 좌표계
  - 초기에는 **화면 중앙이 원점으로 설정**
  - **x+방향은 오른쪽, y+ 방향은 위쪽, z+ 방향은 화면 밖으로 나오는 쪽**
  - **좌표계의 범위는 각 좌표축이 (-1.0 ~ 1.0)으로 설정되어 있다.**



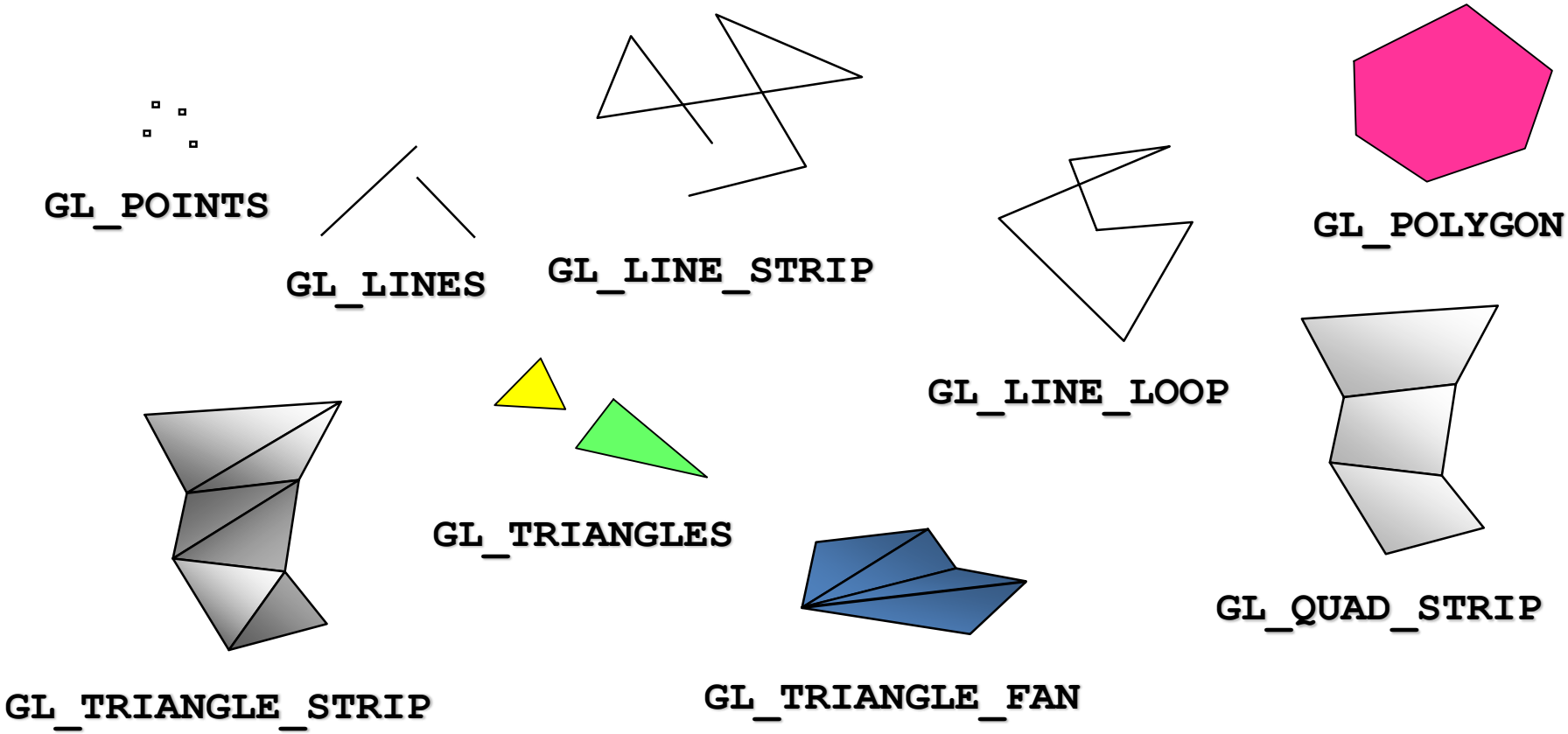
기본적인 그리기

- OpenGL에서 기하 프리미티브들을 표현하기
  - 기본 단위: 정점 (Vertex)
  - 정점 표시하기
    - **glVertex{234}{sifd}[v]** (좌표값…)
      - glVertex2s (2, 3);
      - glVertex3d (0,0, 0.0, 3.14);
      - glVertex4f (2.3, 1.0, -2.2, 2.0);
      - GLdouble dvect[3] = {5.0, 9.0, 10.0};
      - glVertex3dv (dvect);
  - 기본 도형 그리기
    - **glBegin** ()과 **glEnd** ()사이에 점의 좌표들을 **glVertex**…()로 지정
      - **glBegin(GLenum mode)** 함수의 파라미터에 도형의 타입을 설정
        - » mode: GL_POINTS, GL_LINES, GL_TRIANGLES, GL_QUADS, GL_LINE_STRIPE, GL_LINE_LOOP, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_QUAD_STRIP

기본적인 그리기

Value	Description
GL_POINTS	각 Vertex들을 하나의 Point로 표현한다.
GL_LINES	Vertex들을 두 개씩 묶어서 Line을 만든다.
GL_LINE_STRIP	Line들을 한 줄로 연결한다.
GL_LINE_LOOP	마지막 Vertex와 첫 번째 Vertex를 연결하는 Line이 추가된 GL_LINE_STRIP
GL_TRIANGLES	세 개의 Vertex를 묶어서 삼각형을 만든다.
GL_TRIANGLE_STRIP	삼각형들을 길게 연결한다.
GL_TRIANGLE_FAN	삼각형들을 부채 모양으로 연결한다.
GL_QUADS	Vertex들을 네 개씩 묶어서 만든 네 개의 모서리를 가진 Polygon을 만든다.
GL_QUAD_STRIP	사각형들을 길게 연결한다.
GL_POLYGON	단순, 볼록 Polygon

기본적인 그리기


GL_POINTS
GL_LINES
GL_LINE_STRIP
GL_LINE_LOOP
GL_POLYGON
GL_TRIANGLES
GL_QUAD_STRIP
GL_TRIANGLE_STRIP
GL_TRIANGLE_FAN

기본적인 그리기

- 점 그리기
  - **glBegin(GL_POINTS)** 과 **glEnd()** 사이의 glVertex 함수로 정의된 위치에 점이 생성된다.

```
glVertex2f(x, y);
glVertex3f(x, y, z);
glVertex4f(x, y, z, w);
```

    - size의 기본값은 1.0이다.
    - 점의 크기는 0.5부터 10.0까지 지정할 수 있으며, 설정 간격은 0.125 이상이다.

- 선 그리기
  - **glBegin(GL_LINES)** 와 **glEnd()** 사이에 정의된 점들을 연결하여 선을 그린다.

```
glBegin(GL_LINES);                  // 쌍을 이루어 선을 그린다.
glBegin(GL_LINE_STRIP);             // 순서대로 연결하여 선을 그린다.
glBegin(GL_LINE_LOOP);              // 순서대로 연결하며, 마지막 점은
                                    // 첫번째 점과 연결한다.
```

    - 선의 두께를 0.5에서 10.0까지 지정 가능, 설정 간격은 0.125

기본적인 그리기

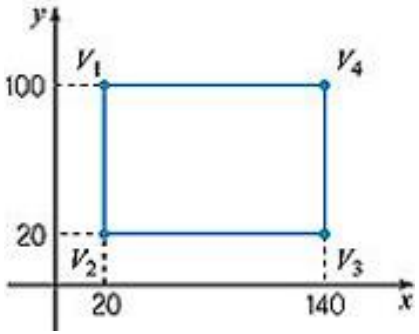
- 삼각형 그리기

```
glBegin (GL_TRIANGLES);
    glVertex3f(x1, y1, z1);        // v1
    glVertex3f(x2, y2, z2);        // v2
    glVertex3f(x3, y3, z3);        // v3
glEnd ();
```

  - 위와 같이 입력하면 v1-v2, v2-v3, v3-v1 의 순서로 선을 그려서 삼각형을 완성
  - 꼭지점을 시계 반대 방향으로 그린 쪽이 앞면

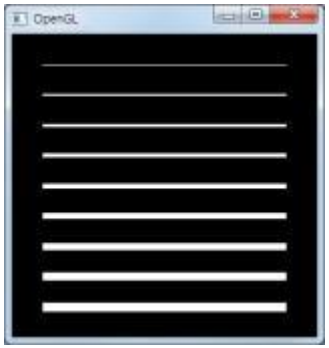
- 도형 그리기

```
glBegin (GL_POLYGON);
    glVertex2i (20, 100);          //v1
    glVertex2i (20, 20);           //v2
    glVertex2i (140, 20);          //v3
    glVertex2i (140, 100);         //v4
glEnd ();
```

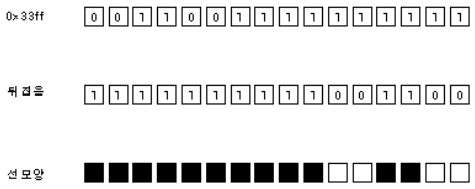


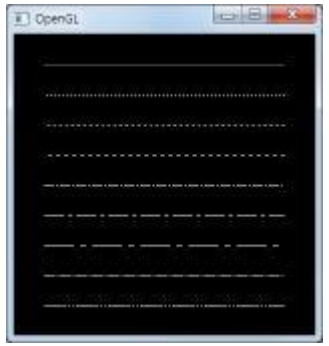
기본적인 그리기

- 도형의 속성 바꾸기
  - 점의 크기: **glPointSize** (GLfloat size);
    - 기본 값은 1.0
    - 0.5부터 10.0까지 지정할 수 있으며, 설정 간격은 0.125 이상
    - glPointSize()함수는 반드시 glBegin() – glEnd() 밖에서 사용한다



  - 선의 굵기: **glLineWidth** (Glfloat width);
    - Size/width는 0.0보다 커야 하며 기본 값은 1.0이다.
  - 색상: **glColor3f** (GLfloat r, GLfloat g, GLfloat b);
    - 선이나 점을 그리기 전에 색상을 설정한다.
  - 선의 종류: **glLineStipple** (Glint factor, GLushort pattern);
    - pattern: 이진수로 표현한 선의 모양. 하위 비트부터 선의 앞쪽 부분의 점 모양을 지정한다.
      - 대응되는 비트가 1인 자리는 점이 찍히고 0인 부분은 찍히지 않는다.
    - factor: 비트 하나가 점 몇 개에 대응될 것인가를 지정
      - 1: 비트 하나가 점 하나에 대응
      - 2: 비트당 2개의 점이 그려져 좀 더 긴 모양을 만들 수 있다.
    - 사용하기 전에 기능 활성화를 해야한다. (glEnable (GL_LINE_STIPPLE);)





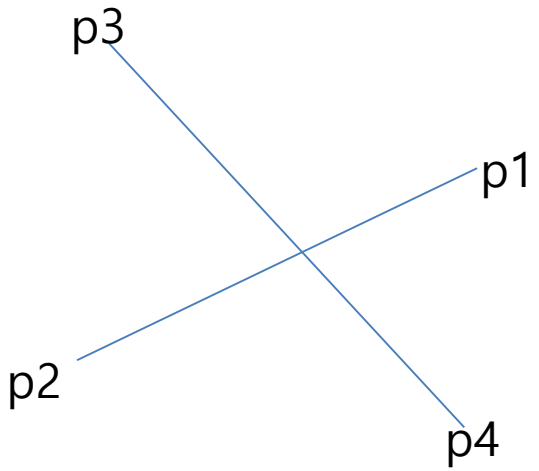
기본적인 그리기: 점 (Point)

- **Point**

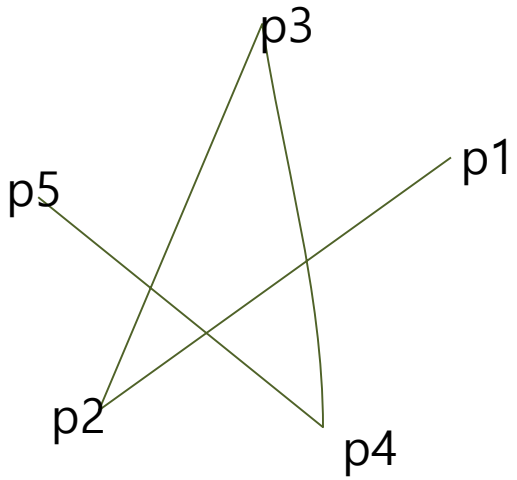
```
GLint p1[2] = {50, 100};
GLint p2[2] = {75, 150};
GLint p3[2] = {100, 100};

glBegin(GL_POINTS);
        glVertex2i (50, 100);
        glVertex2i (75, 150);
        glVertex2i (100, 100);
glEnd ();

glBegin (GL_POINTS);
        glVertex2iv (p1);
        glVertex2iv (p2);
        glVertex2iv (p3);
glEnd ();

glBegin (GL_POINTS);
        glVertex3f (-20.0, -15.5, 14.5);
        glVertex3f (23.4, 19.3, 11.0);
glEnd ();
```

기본적인 그리기: 선 (Line)

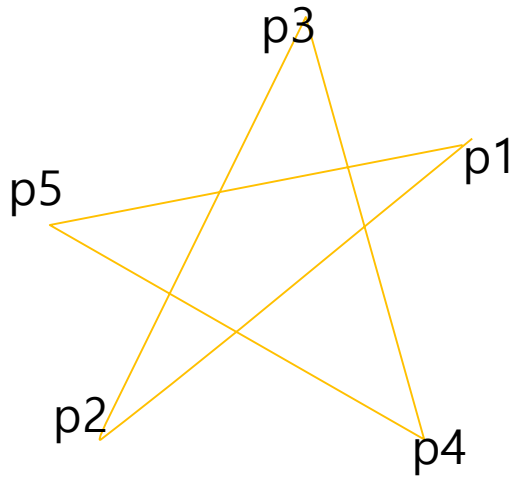
```
glBegin (GL_LINES);
        glVertex2iv (p1);
        glVertex2iv (p2);
        glVertex2iv (p3);
        glVertex2iv (p4);
        glVertex2iv (p5);
glEnd ();
```




```
glBegin (GL_LINE_STRIP);
        glVertex2iv (p1);
        glVertex2iv (p2);
        glVertex2iv (p3);
        glVertex2iv (p4);
        glVertex2iv (p5);
glEnd ();
```

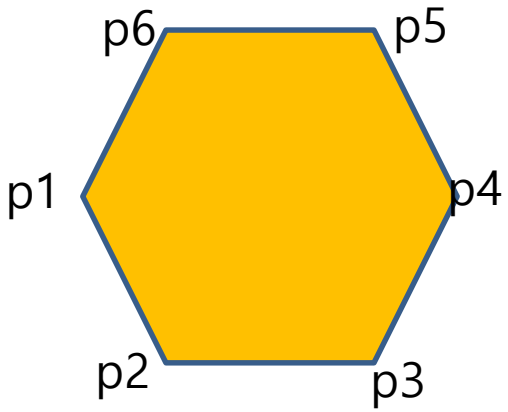



```
glBegin (GL_LINE_LOOP);
        glVertex2iv (p1);
        glVertex2iv (p2);
        glVertex2iv (p3);
        glVertex2iv (p4);
        glVertex2iv (p5);
glEnd ();
```

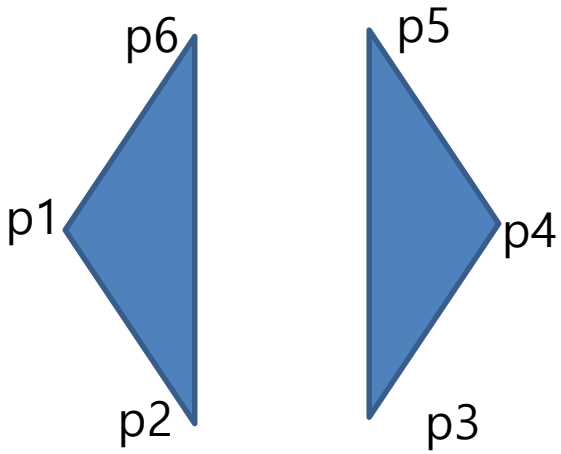


기본적인 그리기: 다각형 (Polygon)

```
glBegin (GL_POLYGON);
        glVertex2iv (p1);
        glVertex2iv (p2);
        glVertex2iv (p3);
        glVertex2iv (p4);
        glVertex2iv (p5);
        glVertex2iv (p6);
glEnd ();
```

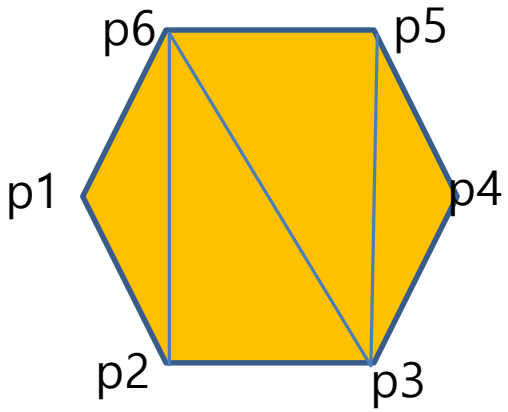



```
glBegin (GL_TRIANGLES);
        glVertex2iv (p1);
        glVertex2iv (p2);
        glVertex2iv (p6);
        glVertex2iv (p3);
        glVertex2iv (p4);
        glVertex2iv (p5);
glEnd ();
```

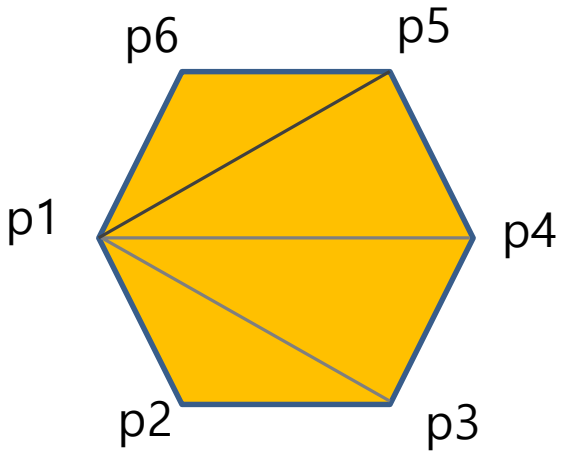


기본적인 그리기: 다각형

```
glBegin (GL_TRIANGLE_STRIP);
        glVertex2iv (p1);
        glVertex2iv (p2);
        glVertex2iv (p6);
        glVertex2iv (p3);
        glVertex2iv (p5);
        glVertex2iv (p4);
glEnd ();
```

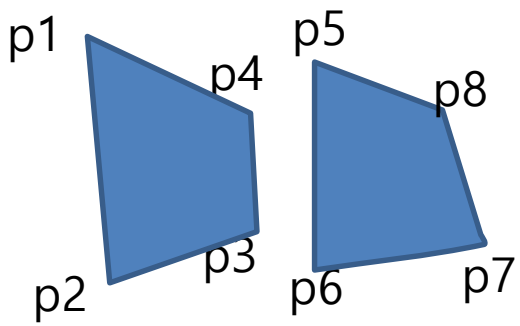



```
glBegin (GL_TRIANGLE_FAN);
        glVertex2iv (p1);
        glVertex2iv (p2);
        glVertex2iv (p3);
        glVertex2iv (p4);
        glVertex2iv (p5);
        glVertex2iv (p6);
glEnd ();
```

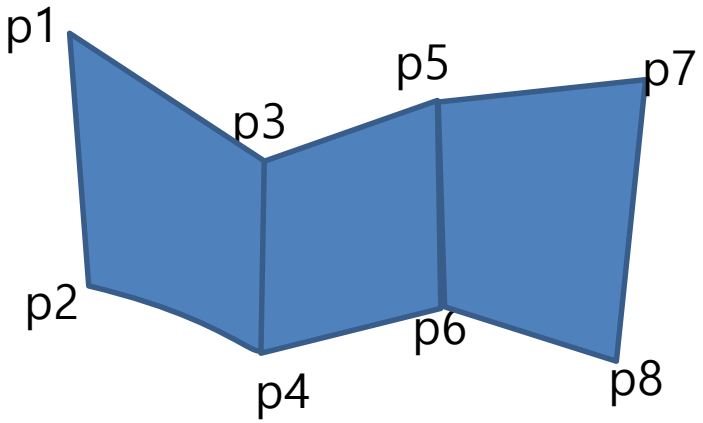


기본적인 그리기: 다각형

```
glBegin (GL_QUADS);
        glVertex2iv (p1);
        glVertex2iv (p2);
        glVertex2iv (p3);
        glVertex2iv (p4);
        glVertex2iv (p5);
        glVertex2iv (p6);
        glVertex2iv (p7);
        glVertex2iv (p8);
glEnd ();
```




```
glBegin (GL_QUAD_STRIP);
        glVertex2iv (p1);
        glVertex2iv (p2);
        glVertex2iv (p3);
        glVertex2iv (p4);
        glVertex2iv (p5);
        glVertex2iv (p6);
        glVertex2iv (p7);
        glVertex2iv (p8);
glEnd ();
```



OpenGL 설치하기

- **Visual Studio 2017에 오픈지엘 설치하기**
  - 기본 GL 라이브러리는 이미 저장
  - 오픈지엘 레퍼런스 사이트: www.opengl.org
  - freeGLUT 라이브러리를 사용하여 윈도우 띄우고 콜백 함수 다룬다.

  - 라이브러리 다운로드: http://www.transmissionzero.co.uk/software/freeglut-devel 에서 Freeglut3.0.0 MSVC Package 다운로드
  - freeGLUT 헤더 파일과 라이브러리 파일을 각각 해당 폴더에 저장
    - header file (include/GL)
      - freeglut.h
      - freeglut_ext.h
      - freeglut_std.h

      → C:₩Program Files (x86)₩Windows Kits₩10₩Include₩10.0.17134.0₩um₩gl
    - library file (lib/)
      - freeglut.lib

      → C:₩Program Files (x86)₩Windows Kits₩10₩Lib₩10.0.17134.0₩um₩x86
    - dll file(bin/)
      - freeglut.dll

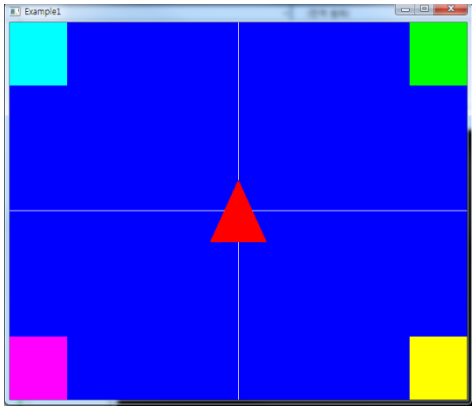
      → C:₩Windows₩SysWOW64

    - freeGLUT 함수 관련 레퍼런스: http://freeglut.sourceforge.net/

- **프로젝트 설정은 콘솔 응용 프로그램으로 설정**

실습 1

- 화면에 윈도우 띄우고 사각형 그리기
  - 위치 (100, 100)에 크기 800 * 600의 윈도우를 띄운다.
  - 아래의 그림과 같은 위치에 5개의 도형을 그린다.
    - 중앙: 삼각형, 좌우상하: 사각형
    - 각각 다른 색으로 설정한다.
    - 화면 중앙에 2개의 직선으로 좌표계를 그린다.



- 사각형 그리기 함수:

  void glRectf (GLfloat x1, GLfloat y1, GLfloat x2, GLfloat y2);
  - x1, y1: 사각형 꼭짓점 중 최소값
  - x2, y2: 사각형 꼭짓점 중 최대값

- **OpenGL에서 윈도우의 기본 좌표계:**
  - **중심: (0, 0)**
  - **각 축의 값 범위: [-1.0 ~ 1.0]**

실습 1

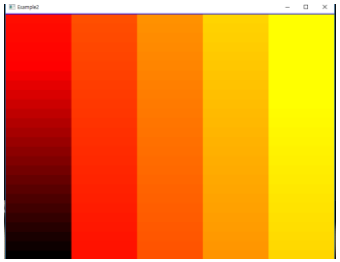
```
#include <GL/freeglut.h>

void main ( int argc, char *argv[] )
{
	//초기화 함수들
	glutInit (&argc, argv);
	glutInitDisplayMode ( GLUT_SINGLE | GLUT_RGBA );		// 디스플레이 모드 설정
	glutInitWindowPosition ( 100, 100 );			// 윈도우의 위치지정
	glutInitWindowSize ( 250, 250 );			// 윈도우의 크기 지정
	glutCreateWindow ( "Example" );				// 윈도우 생성 (윈도우 이름)
	glutDisplayFunc ( drawScene );				// 출력 함수의 지정
	glutReshapeFunc (Reshape);				// 다시 그리기 함수의 지정
	glutMainLoop ();
}

// 윈도우 출력 함수
GLvoid drawScene( GLvoid )
{
	glClearColor (0.0f, 0.0f, 1.0f, 1.0f);			// 바탕색을 'blue' 로 지정
	glClear ( GL_COLOR_BUFFER_BIT );			// 설정된 색으로 전체를 칠하기

	// 이 부분에 색상 지정하고 사각형 그리기
	…

	glFlush(); // 화면에 출력하기
}

GLvoid Reshape (int w, int h)
{
	glViewport(0, 0, w, h);
}
```

실습 2

- 바둑판 모양 그리기
  - 윈도우 초기화 하는 함수를 만든다.
  - Reshape 함수에서 윈도우의 좌표계 범위를 조절하는 함수를 호출한다.
    - void **glOrtho** (GLdouble left, GLdouble right,
      GLdouble bottom, GLdouble top, GLdouble near, GLdouble far);
      - left/right: x의 최소값/최대값
      - bottom/top: y의 최소값/최대값
      - near/far: z의 최소값/최대값

    예) 중앙이 중점이고 크기를 x축은 -400~400, y축은 -300~300, z축은 -1.0~1.0 간격으로 정한다.

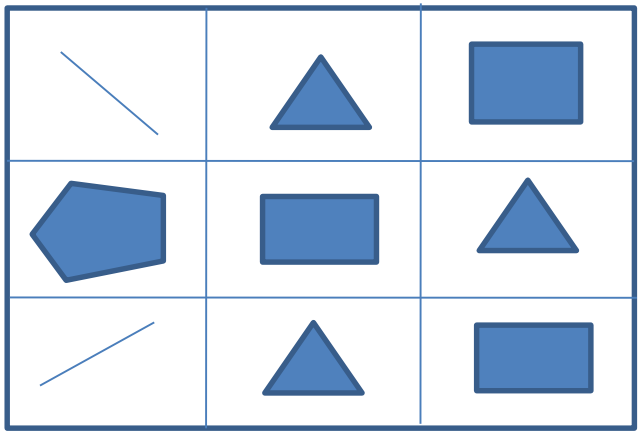
```
GLvoid Reshape (int w, int h)
{
        glViewport(0, 0, w, h);
        glOrtho (-400.0, 400.0, -300.0, 300.0, -1.0, 1.0);
}
```

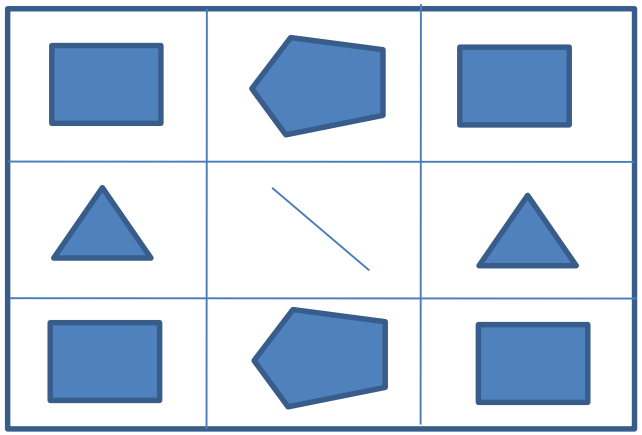


  - 가로와 세로를 특정 개수로 나눠 특정 색에서 점차적으로 변화되는 형태로 색을 배정한다.
  - 특정 개수와 색은 랜덤하게 설정한다.
    - 실행 할 때마다 개수와 색은 다르게 나타난다.

실습 3

- 임의의 도형 그리기
  - 화면의 가로와 세로를 각각 3등분 한다.
  - 화면의 첫 칸에 아래의 도형 중 한 개의 도형을 그린다.
    - 도형: 선, 삼각형, 사각형, 오각형
  - 다음 칸부터 도형의 꼭지점이 한 개씩 늘어나게 그리고, 오각형 후에는 역순으로 다시 그린다.
    - 선 → 삼각형 → 사각형 → 오각형 → 사각형 → …
  - 시작되는 도형은 임의로 선택된다.





콜백 함수

- freeGLUT callback 함수
  - **콜백 함수**: 운영체계가 호출할 어플리케이션의 함수를 지정해 특정한 사건 또는 메시지가 발생했을 때 호출되도록 지정할 수 있는데, 이런 함수를 콜백 함수라고 함.
  - **Display callback**: 처음 윈도우를 열 때, 윈도우의 위치를 이동시킬 때, 윈도우의 크기를 변경할 때, 뒤의 윈도우가 활성화되어 앞으로 나타날 때, glutPostRedisplay 함수에 의해 출력할 때 호출되는 출력 콜백 함수
  - **Reshape callback**: 처음 윈도우를 열 때, 윈도우의 위치를 이동시킬 때, 윈도우의 크기를 변경할 때,
  - **Keyboard, Mouse callback**: 키보드, 마우스 관련 입력 이벤트가 발생할 때
  - **Menu callback**: 팝업 형태의 메뉴를 호출할 때,
  - **Idle callback**: event loop을 돌 때, 큐에 이벤트가 들어있지 않을 때 호출되는 콜백 함수
  - **Timer callback**: 시간 제어 콜백 함수

콜백 함수

Function Name	Description
glutDisplayFunc()	현재 Window를 위한 Display Callback을 설정한다.
glutReshapeFunc()	현재 Window를 위한 Reshape Callback을 설정한다.
glutKeyboardFunc()	현재 Window를 위한 Keyboard Callback을 설정한다.
glutSpecialFunc()	현재 Window를 위한 Special Keyboard Callback을 설정한다.
glutMouseFunc()	현재 Window를 위한 Mouse Callback을 설정한다.
glutMotionFunc()	현재 Window를 위한 각각의 Motion Callback을 설정한다.
glutPassiveMotionFunc()	현재 Window를 위한 각각의 Passive Motion Callback을 설정한다.
glutEntryFunc()	현재 Window를 위한 Mouse Enter/Leave Callback을 설정한다.
glutMouseWheelFunc()	현재 Window를 위한 Mouse Wheel Callback을 설정한다.
glutCreateMenu()	새로운 Pop-up 메뉴를 생성한다.
glutSetMenu()	현재 메뉴를 설정한다.
glutAddMenuEntry()	현재 메뉴의 하단에 메뉴 항목을 추가한다.
glutAttachMenu()	현재 메뉴의 식별자로 현재 Window를 위한 Mouse Button을 부착한다.
glutAddSubMenu()	현재 메뉴의 하단에 서브 메뉴 Trigger를 추가한다.
glutIdleFunc()	현재 Window를 위한 Idle Callback을 설정한다.
glutTimerFunc()	Milliseconds의 지정된 숫자로 Trigger되는 Timer Callback을 설정한다.

입력 컨트롤

- 키보드 입력
  - void **glutKeyboardFunc (void (*func)(unsigned char key, int x, int y)):** 키보드와 인자로 지정한 루틴을 연결하여 키를 누를 때 호출되도록 설정한다.
    - 키보드 입력이 일어날 때마다 ASCII 코드값이 설정된다.
      - key: 입력 키보드,
      - x, y: 키보드 입력할 때의 마우스의 위치
    - 사용 예:

```
void main ( int argc, char *argv[] )
{
        glutInit (&argc, argv);
        glutInitDisplayMode ( GLUT_SINGLE | GLUT_RGBA );        // 디스플레이 모드 설정
        glutInitWindowSize ( 250, 250 );                        // 윈도우의 크기 지정
        glutCreateWindow ( "Input Control" );                   // 윈도우 생성
        glutDisplayFunc ( drawScene );                          // 출력 함수의 지정
        glutKeyboardFunc ( Keyboard );                          // 키보드 입력 콜백 함수
        glutMainLoop ();
}
void Keyboard ( unsigned char key, int x, int y)
{
        switch (key) {
           case 't':      ...; break;
           case 'r':      ...; break;
           }
}
```

입력 컨트롤

- ASCII 가 아닌 특수 키인 경우에는: **glutSpecialFunc** 함수를 사용한다.
  - Key: GLUT_KEY_F1, GLUT_KEY_F2, GLUT_KEY_F3, … GLUT_KEY_F12, GLUT_KEY_LEFT…
  - **void glutSpecialFunc (void (*func)(int key, int x, int y));**
    Key: GLUT_KEY_F1 ~ GLUT_KEY_F12,
    GLUT_KEY_LEFT, GLUT_KEY_RIGHT, GLUT_KEY_UP, GLUT_KEY_DOWN,
    GLUT_KEY_HOME, GLUT_KEY_END, GLUT_KEY_INSERT,
    GLUT_KEY_PAGE_UP, GLUT_KEY_PAGE_DOWN

- 사용 예:

```
void main ( int argc, char *argv[] )
{
    glutInit (&argc, argv);
    glutInitDisplayMode ( GLUT_SINGLE | GLUT_RGBA );      // 디스플레이 모드 설정
    glutInitWindowSize ( 250, 250 );                      // 윈도우의 크기 지정
    glutCreateWindow ( "Input Control" );                 // 윈도우 생성 (윈도우 이름)
    glutDisplayFunc ( drawScene );                        // 출력 함수의 지정
    glutSpecialFunc (SpecialKeyboard);
    glutMainLoop ();
}

void SpecialKeyboard (int key, int x, int y)
{
    if (key == GLUT_KEY_F1)
        …;
}
```

입력 컨트롤

  - Escape, backspace, and delete 키는 ASCII 문자로 인식됨

- **CTRL, ALT, SHIFT 함수 체크하기:**
  - **int glutGetModifiers (void);**
  - GLUT_ACTIVE_CTRL, GLUT_ACTIVE_ALT, GLUT_ACTIVE_SHIFT 값을 리턴

- **키보드를 뗐을 때 호출되는 콜백 함수 설정**
  - void glutKeyboardUpFunc (void (*func) (unsigned char key, int x, int y));

입력 컨트롤

- 마우스 입력
  - **void glutMouseFunc (void (*func)(int button, int state, int x, int y)):** 마우스 버튼과 인자로 지정한 루틴을 연결하여 호출되도록 한다.
    - button (버튼 파라미터): GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, GLUT_RIGHT_BUTTON
    - state (상태 파라미터): GLUT_UP, GLUT_DOWN
    - x, y: 윈도우에서 마우스의 위치

    - 사용 예:

```
void main ( int argc, char *argv[] )
{
        glutInit (&argc, argv);
        glutInitDisplayMode ( GLUT_SINGLE | GLUT_RGBA );          // 디스플레이 모드 설
        glutInitWindowSize ( 250, 250 );                          // 윈도우의 크기 지정
        glutCreateWindow ( "Input Control" );                     // 윈도우 생성 (윈도우 이름)
        glutDisplayFunc ( drawScene );                            // 출력 함수의 지정
        glutMouseFunc (Mouse);
        glutMainLoop ();
}

void Mouse (int button, int state, int x, int y)
{
        if ( button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
            …;
}
```

입력 컨트롤

- 마우스 이동 입력
  - void **glutMotionFunc** (void (*func)(int x, int y)): 마우스 버튼을 누른 채 마우스를 움직일 때 호출될 콜백 함수를 등록한다.
  - void **glutPassiveMotionFunc** (void (*func)(int x, int y)): 마우스 버튼을 누르지 않은 채 마우스를 움직일 때 호출될 함수 등록
    - x, y: 마우스의 위치

- 사용 예:

```
bool left_button;

void main ( int argc, char *argv[] )
{
        glutInit (&argc, argv);
        glutInitDisplayMode ( GLUT_SINGLE | GLUT_RGBA );
        glutCreateWindow ( "Input Control" );
        glutDisplayFunc ( drawScene );
        glutMouseFunc ( Mouse );
        glutMotionFunc ( Motion );
        glutMainLoop ();
}

void Mouse (int button, int state, int x, int y)
{
        if (button == GLUT_LEFT_BUTTON )        left_button = true;
}

void Motion (int x, int y)
{
        if ( left_button == true ) {        …        }
}
```

타이머 함수

- 애니메이션 구현을 위한 타이머 설정 함수
  - void **glutTimerFunc** (unsigned int msecs, (*func)(int value), int value);
    - 타임 아웃이 발생할 경우 호출될 콜백 함수를 등록한다.
    - msecs: 콜백 함수를 호출하기 전까지 기다릴 시간 (밀리세컨 단위)
    - func: 호출할 함수의 이름
    - value: 콜백 함수로 전달할 값
    - 사용 예:

```
void main ( int argc, char *argv[] )
{
        glutInit (&argc, argv);
        glutInitDisplayMode ( GLUT_SINGLE | GLUT_RGBA );
        glutCreateWindow ( "Input Control" );
        glutDisplayFunc ( drawScene );
        glutTimerFunc (100, TimerFunction, 1);          // 타이머 함수 설정
        glutMainLoop ();
}


void Timerfunction (int value)
{
        …
        glutPostRedisplay ();                           // 화면 재 출력
        glutTimerFunc (100, TimerFunction, 1);          // 타이머함수 재 설정

}
```

    - 이 함수는 한 번만 실행되므로 지속적인 애니메이션을 위해서는 타이머 함수 내에 타이머를 다시 호출해야한다.

화면 출력

- **void glutReshapeFunc (void (*func)(int w, int h)): 윈도우 크기가 변경될 때 취할 동작을 지정한다.**
  - 윈도우 크기 변경
  - 콜백함수는 새롭게 변한 윈도우의 폭(w)과 높이(h)를 파라미터로 받는다.

- **void glutPostRedisplay (void): 현재 윈도우를 refresh하게 한다.**
  - 출력 자료가 변경된 후 화면 다시 그리기를 할 때 불러준다.
  - Refresh 되기 전에 여러 번 호출해도 단 한번만 refresh한다.

- **void glutIdleFunc (void (*func))**
  - 이벤트가 없을 경우에 호출되는 함수
  - 애니메이션 효과를 줄 수 있다.

- **void glutCloseFunc (void (*func) (void));**
  - 함수를 종료한다.

메뉴 만들기

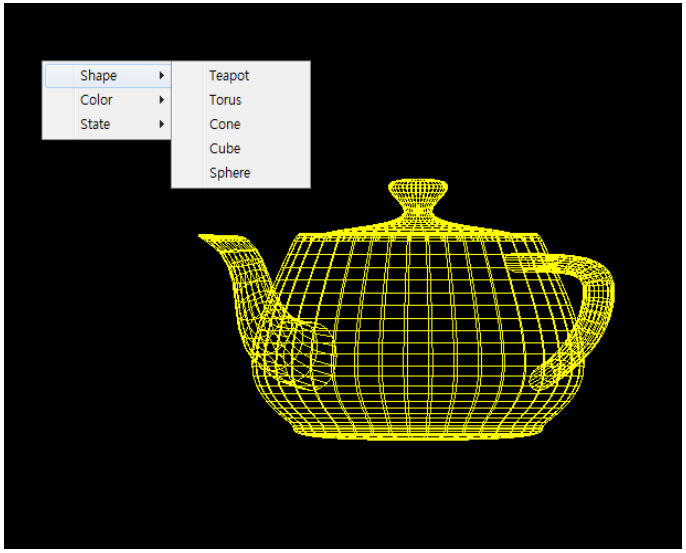
- 팝업 메뉴 만들기
  - 팝업 메뉴를 만든다.
    - int **glutCreateMenu** (void (*func)(int value));
      - 리턴값: 유일한 정수타입의 메뉴 구별자 (1부터 시작한다)
  - 마우스 버튼에 메뉴 삽입하기
    - void **glutAttachMenu** (int button);
    - void **glutDetachMeni** (int button);
      - Button: 버튼 (GLUT_LEFT_BUTTON / GLUT_MIDDLE_BUTTON / GLUT_RIGHT_BOTTON)
  - 메뉴 항목 추가하기
    - void **glutAddMenuEntry** (char *name, int value);
      - Name: 메뉴 엔트리의 이름
      - Value: 메뉴가 선택되면 메뉴의 콜백 함수에 리턴할 값
  - 메뉴의 서브 메뉴 추가하기
    - void **glutAddSubMenu** (char *name, int menu);
      - Name: 서브 메뉴의 이름
      - Menu: 메뉴의 구별자
  - 메뉴 없애기
    - void **glutDestroyMenu** (int menu);

메뉴 만들기

– 사용 예:

```
void main ( int argc, char *argv[] )
{
        int SubMenu1, SubMenu2, SubMenu3;
        int MainMenu;

        glutInit (&argc, argv);

        SubMenu1 = glutCreateMenu (MenuFunc);
            glutAddMenuEntry (“Teapot”, 1);
            glutAddMenuEntry (“Torus”, 2);
            glutAddMenuEntry (“Cone”, 3);
            glutAddMenuEntry (“Cube”, 4);
            glutAddMenuEntry (“Sphere”, 5);

        SubMenu2 = glutCreateMenu (MenuFunc);
            glutAddMenuEntry (“Red”, 11);
            glutAddMenuEntry (“Green”, 22);

        SubMenu3= glutCreateMenu (MenuFunc);
            glutAddMenuEntry (“Wire”, 111);
            glutAddMenuEntry (“Solid”, 222);

        MainMenu = glutCreateMenu (MenuFunc);
            glutAddSubMenu (“Shape”, SubMenu1);
            glutAddsubMenu (“Color”, SubMenu2);
            glutAddsubMenu (“State”, SubMenu3);

        glutAttachMenu (GLUT_RIGHT_BUTTON);
}
```

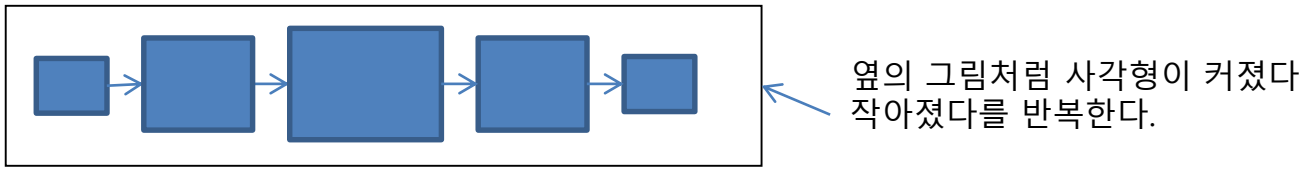
```
void MenuFunc (int button)
{
    switch (button) {
        case 1: …;
            break;
        case 2: …;
            break;
        case 11: ...;
            break;
    }
    glutPostRedisplay ();
}
```



실습 4

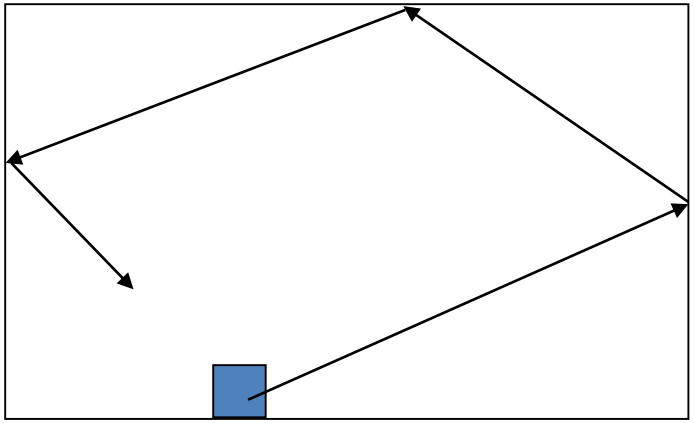
- 사각형 변경 애니메이션 만들기
  - 윈도우의 크기를 800*600으로 만든다.
    - 윈도우의 좌표계값을 glOrtho 함수를 이용하여 800*600으로 조절한다.

      glOrtho (0.0, 800.0, 0.0, 600.0, -1.0, 1.0);
  - 마우스를 클릭하면 그 위치에 임의의 크기의 사각형을 그린다.
    - 최대 10개까지 그린다.
  - 10개가 넘어가면 처음에 그린 사각형이 지워지고 마우스 클릭 위치에 새 사각형이 그려진다. (최대 10개의 사각형 그리기)
  - 사각형에 애니메이션을 추가한다.
    - glutTimerFunc 사용하여 애니메이션을 진행한다.
    - 색상 변경: 사각형의 색상이 변경된다.
    - 크기 변경: 사각형의 크기가 커졌다 작아졌다를 반복한다 (최소 3단계).



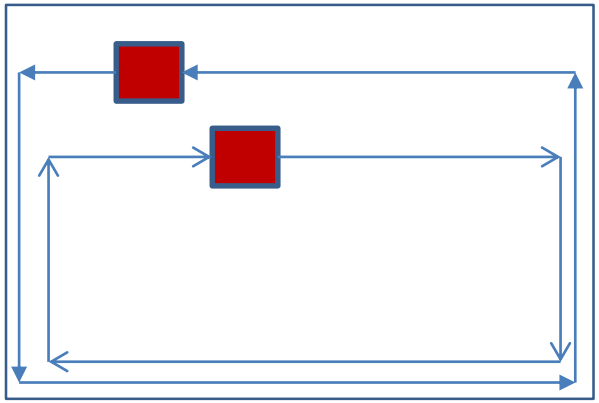
실습 5

- **Bouncing Rectangle (사각형 튀기기)**
  - 실습 4에서 구현한 사각형을 이용한다.
    - 메뉴 또는 키보드 명령어를 넣어서 사각형 또는 삼각형을 선택하도록 한다.
  - 마우스 버튼을 눌러 사각형을 그리면 그 사각형이 그 자리에서부터 움직인다. 최대 10개의 사각형이 동시에 이동한다. 각각 다른 방향 또는 속도로 이동하도록 한다.
  - 키보드를 올려 전체 속도를 올리거나 줄이거나 한다.
  - glutTimerFunc 사용하여 자리를 이동한다



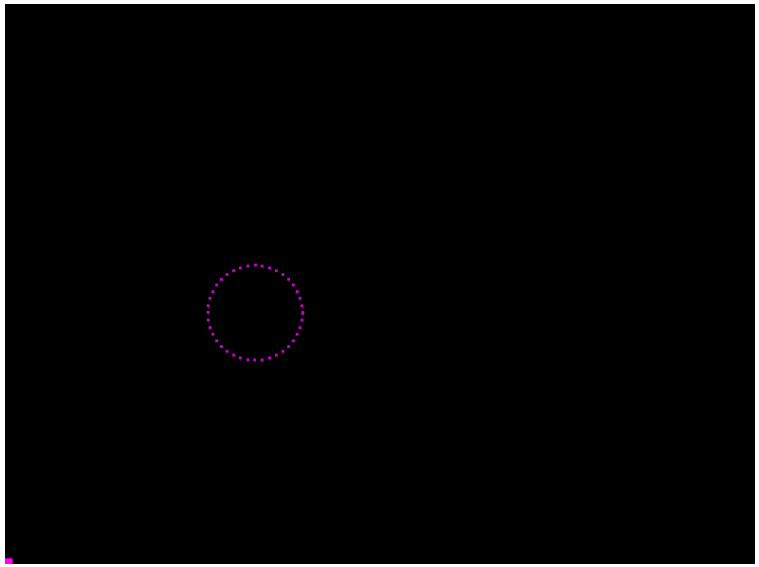
실습 6

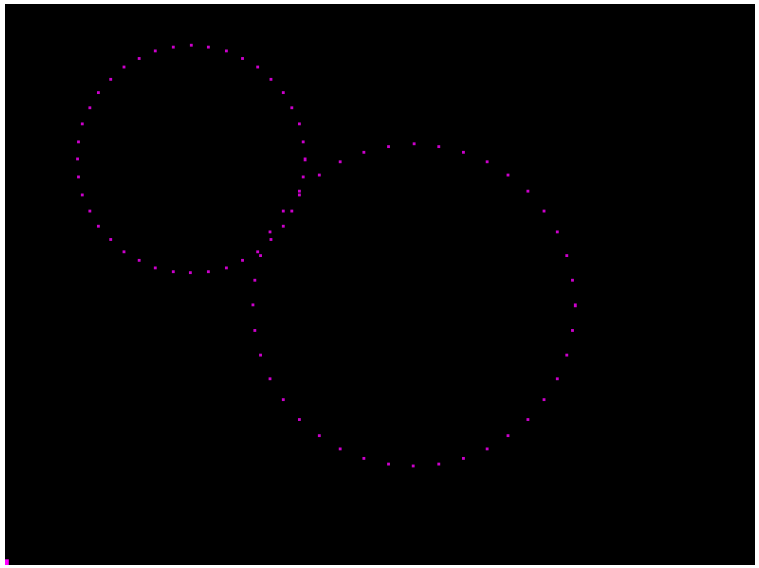
- 마우스 명령 수행하기
  - 실습 5에서 구현한 사각형을 이용한다.
    - 마우스를 클릭하여 사각형을 그리고 크기가 변경되는 애니메이션과 튕기기 애니메이션이 진행된다.
    - 최대 9개의 사각형이 그려진다.
  - 키보드로 사각형이 그린 순서를 클릭하면 (1~9), 튕기던 해당 사각형이 현재 위치에서 화면을 시계방향 또는 반시계 방향으로 한 바퀴 이동한다. 0번을 누르면 모든 사각형이 시계/반시계 방향으로 화면을 한 바퀴 이동하고 제자리에 오면 다시 튕긴다.
  - 키보드 's'를 누르면 이동 애니메이션이 멈춘다.
  - 키보드 'p'를 누르면 모든 사각형이 튕기기 시작한다.



실습 7

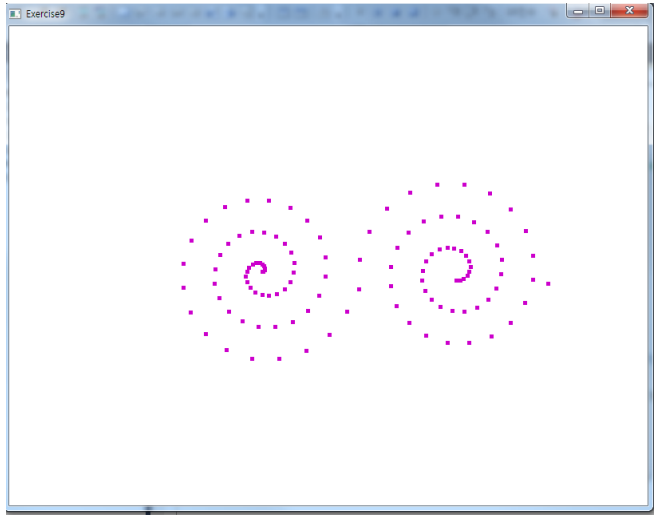
- 원형 애니메이션 만들기
  - 마우스를 클릭하면 그 위치를 중심으로 원형으로 도형이 그려지고, 점점 밖으로 퍼져나가고, 특정 반지름이 되면 다시 시작 점부터 밖으로 퍼져나가는 애니메이션이 진행된다. 1개 이상의 원을 그릴 수 있도록 한다.
  - 여러 개의 원 중 임의의 원은 계속 밖으로 퍼져나가며 사라지도록 한다. 이때는 점의 색이 변한다.





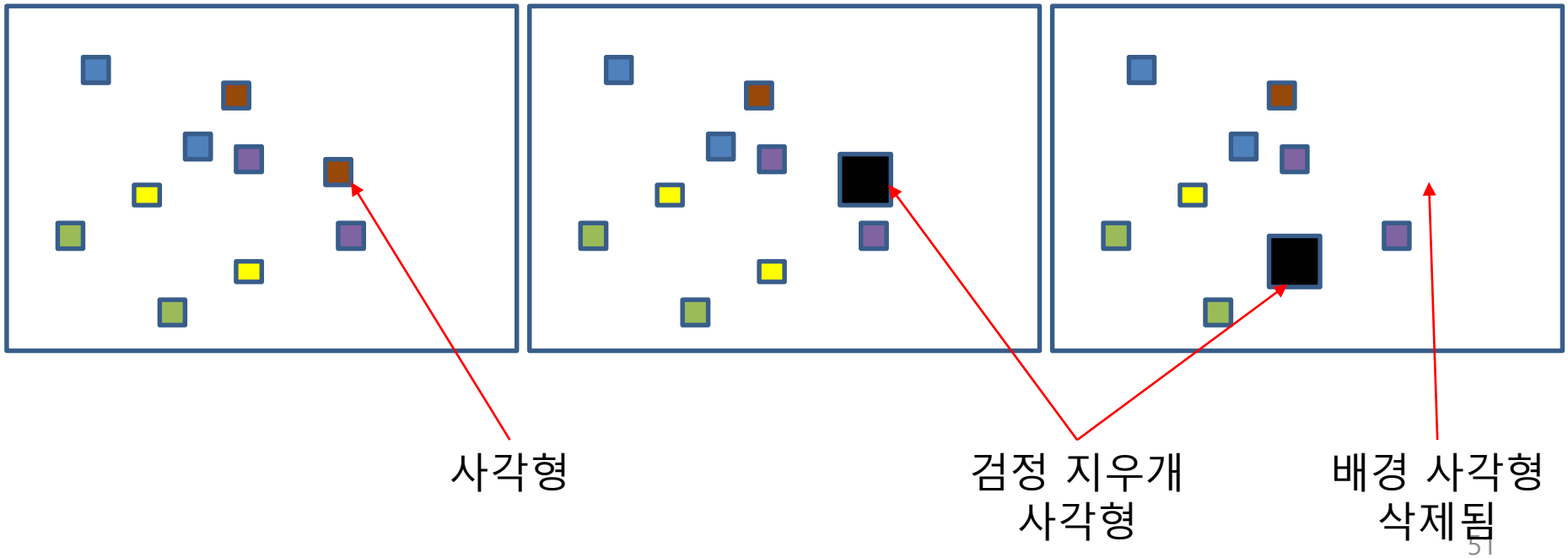
실습 8

- **실습 7 변형하기**
  - 원형 대신 회오리 방향으로 원이 그려진다.
    - 회오리 방향은 시계방향 또는 반시계 방향이 임의로 선택되어 그려진다. 2~3 바퀴 회오리를 그린 후 끝나는 지점에서 밖에서 안으로 다시 2~3바퀴 회오리를 그린다.
    - 점 또는 선으로 그린다.
    - 점이 순서대로 그려진다(애니메이션)
    - 한 개 이상, 최대 10개까지 동시에 그릴 수 있게 한다.



실습 9

- 화면 지우기
  - 윈도우를 띄우고 화면에 같은 크기의 작은 사각형을 다양한 색으로 임의의 위치에 100개 그린다.
  - 왼쪽 마우스 버튼을 누르면 화면의 사각형의 2배의 크기의 사각형이 그려지고 마우스를 누른채로 이동시키면 사각형이 위치를 이동한다.
  - 왼쪽 마우스 버튼의 사각형과 부딪친 배경 사각형은 사라진다.
  - 왼쪽 마우스 버튼을 떼면 지우개 사각형은 사라진다.



실습 10

- 이동하고 변형하기
  - 실습 6과 9를 융합하고 변형한다.
  - 융합:
    - 화면에 30개의 작은 사각형이 임의의 위치에 그려지고 (실습 9) 그려진 위치에서 튕겨진다 (실습 6).
    - 왼쪽 마우스 버튼을 누르면 지우개(?) 사각형이 만들어지고 (실습 9) 마우스를 누른채로 이동시킨다 (실습 9).
  - 변형:
    - 지우개 사각형과 기존의 튕겨지던 사각형이 충돌하면 기존의 사각형이 삼각형으로 바뀌어 지면서 잠시 크기가 커지고 색이 바뀌면서 튕겨진다.
    - 특정 시간 이후에는 기존의 크기로 바뀌고, 다시 특정 시간 이후에는 원래의 사각형으로 바뀐다.
    - 사각형에서 삼각형으로 바뀔 때 중간 과정을 (사다리꼴) 그린다.
    - 삼각형일 때는 지우개 사각형과 충돌체크를 하지 않는다.