5장. 오픈지엘 기본틀

- 학습목표
  - 논리적 입력장치를 설정하는 이유와 종류를 이해한다.
  - 세 가지 입력모드의 차이점을 이해한다.
  - GLUT 콜백함수의 종류와 사용법을 이해한다.
  - GL의 화면 좌표계와 GLUT의 화면 좌표계 사이의 차이점을 이해한다.
  - 더블 버퍼링의 필요성에 대해 이해한다.
  - 정점 배열, 디스플레이 리스트의 필요성과 사용법을 이해한다.

5.1 그래픽 입력장치-물리적 입력장치

- 마우스, 조이스틱, 트랙볼, 스페이스 볼
  - 상대입력 / 절대입력

[그림 5-1-4] 마우스 조이스틱 트랙볼 스페이스볼

- 타블렛, 스타일러스 펜
  - 크로스 헤어 커서
  - 디지타이징

[그림 5-6] 타블렛 II

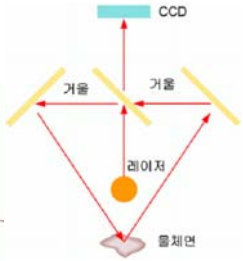
- 터치 패널
  - 광학 패널, 전기 패널

[그림 5-7] 광학 [그림 5-8] 전기

물리적 입력장치

- 3D 스캐너
  - 물체 표면의 X, Y, Z 좌표 인식
  - 레이저
  - 촬상소자(CCD)

[그림 5-9] 3차원 스캔 작업



[그림 5-10] 레이저스캐너

- 버튼 박스와 다이얼
  - 버튼 박스: 매크로 기능
  - 다이얼:
    - 물체에 대한 기하변환
    - 아날로그 방식

[그림 5-11] 버튼

[그림 5-12] 다이얼

논리적 입력장치

- 입력을 논리적으로 취급
  - scanf("%d", &x); 키보드? 버튼박스?
  - 물리적 입력장치가 바뀌어도 프로그램은 동일

- 좌표 입력기(Locator)
  - 절대좌표 또는 상대좌표. 마우스, 키보드의 화살표 키, 트랙 볼
- 연속좌표 입력기(Stroke)
  - 일련의 연속 좌표. 마우스, 태블릿 커서.
- 문자열 입력기(String)
  - 문자열 키보드
- 스칼라 입력기(Valuator)
  - 회전각, 크기조절 비율 등 스칼라 값. 키보드, 마우스, 다이얼
- 메뉴선택 입력기(Choice)
  - 메뉴, 서브메뉴, 메뉴옵션 선택. 마우스, 키보드, 터치 패널, 음성
- 물체선택 입력기(Pick)
  - 물체를 선택. 마우스나 터치 패널

물체 선택

- 마우스 클릭시 어떤 물체를 선택할까?
  - 가까운 물체
  - 맨 윗 layer의 물체
  - 사용자 선택
  - ...
- 그래픽 에디터에서 물체를 선택시 고려할 점은?

5

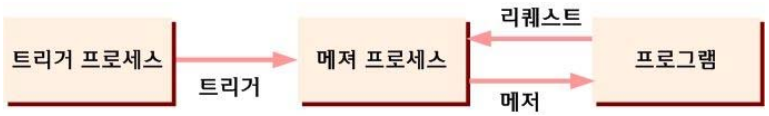
5.2 입력 모드– Measure와 Trigger

- 입력 장치와 응용프로그램간의 상호작용 방법
- 메저와 트리거
  - 메저(Measure): 응용프로그램에게 전달되는 입력 값
    - 항상 시스템 버퍼에 메저 값이 저장되어 있음 (ex 마우스 좌표)
  - 트리거(Trigger): 전달하라는 신호
- Ex.
  - DIR <ENTER>
  - 마우스 좌표와 클릭
  - 선택된 메뉴 아이디와 클릭
- Measure와 Trigger 관계를 기준으로 본 세가지 입력모드
  - Request mode
  - Sample Mode
  - Event Mode
  - 항상 시스템 버퍼에 메저 값이 저장되어 있음.

6

리퀘스트 모드

- 프로그램이 실행 중 메저를 요구
  - 트리거가 일어날 때까지 대기상태
  - **Request_Locator(Device_ID, &Measure);**
  - **Device_ID 필드에 의해 물리적 입력장비 제어**



[그림 5-15] 리퀘스트 모드

샘플 모드

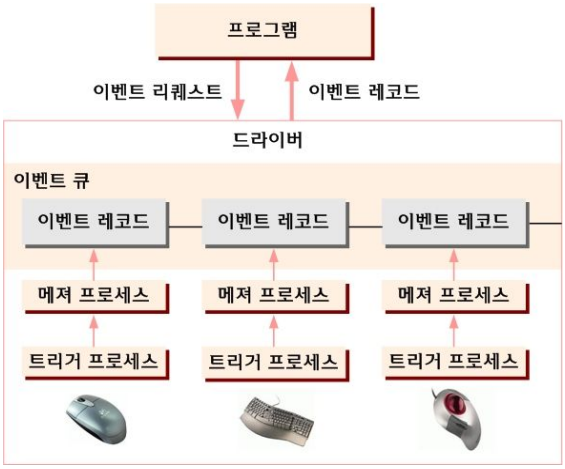
- 직접모드
  - 사용자 트리거가 불필요
  - **sample_Locator(Device_ID, &Measure);**
  - 이미 필요한 메저가 준비된 상태
  - 물체선택 ➔ 회전 메뉴 선택
    - **cf. 회전 메뉴 선택 ➔ 물체선택: 리퀘스트 모드**

[그림 5-16] 샘플 모드

이벤트 모드

- 이벤트 모드
  - 사용자가 입력 선택
  - cf: 리퀘스트/샘플 모드: 프로그램이 주도권
- 이벤트 레코드
  - 이벤트 타입
  - 장치 아이디
  - 메저
- 응용 프로그램은 주기적으로 이벤트 큐를 검사
  - 드라이버에게 이벤트 리퀘스트.
  - 드라이버가 큐 프런트 레코드를 전달
  - 큐가 비어있으면 응용 프로그램은 다른 일을 수행

9



[그림 5-17] 이벤트 모드

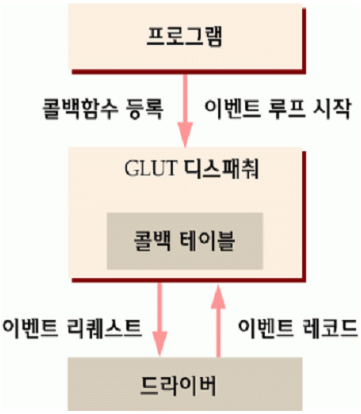
10

콜백함수

- 응용 프로그램 구조

```
Initialize Input Devices;
do
{   if (There Is an Event on the Event Queue)
        switch (Event Type)
        {   case Keyboard Event:
                    Get  Event Record, Run Keyboard Callback
            case Mouse Event:
                Get  Event Record, Run Mouse Callback
                ...
        }
        else Do Background Process
}
while (User Does Not Request Escape);
```


5.3 glut 프로그램 예-지엘의 콜백



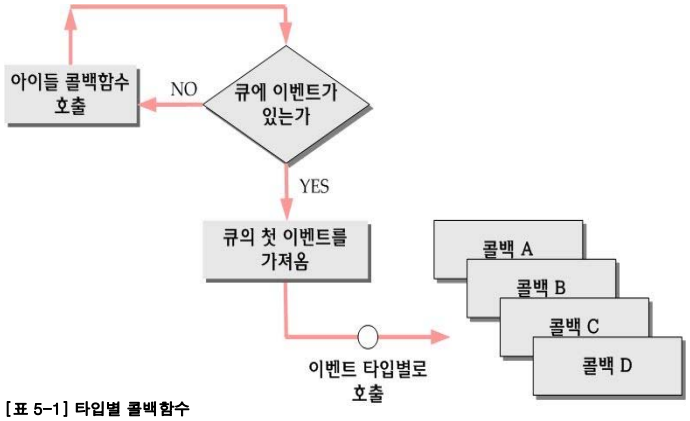
[그림 5-19] 콜백 테이블

이벤트 타입	콜백함수 명
DISPLAY	MyDisplay( )
RESHAPE	MyReshape( )
KEYBOARD	MyKeyboard( )
MOUSE	MyMouse( )
IDLE	MyIdle( )

[표 5-1] 타입별 콜백함수

지엘의 콜백

- 아이들 콜백
  - 큐에 이벤트가 없을 때 실행
  - 정의되어 있지 않으면 운영체제는 다른 일을 수행
  - 드라이버를 통해 주기적으로 이벤트 검사

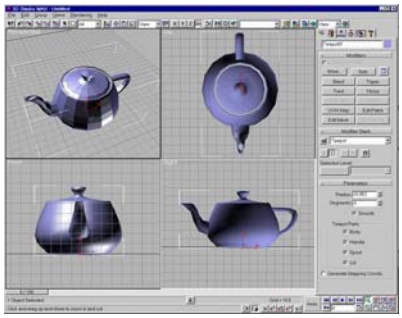


[표 5-1] 타입별 콜백함수

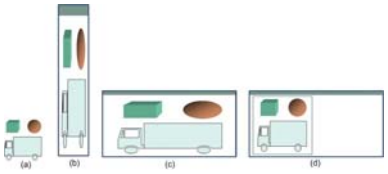

```
#include <glut.h>
#include <gl.h>
#include <glu.h>
void MyDisplay( ){
   glClear(GL_COLOR_BUFFER_BIT);
   glBegin(GL_POLYGON);
            glVertex3f(-0.5, -0.5, 0.0);
            glVertex3f(0.5, -0.5, 0.0);
            glVertex3f(0.5, 0.5, 0.0);
            glVertex3f(-0.5, 0.5, 0.0);
   glEnd( );
   glFlush( );
}
int main( ){
   glutCreateWindow("OpenGL Drawing Example");
   glutDisplayFunc(MyDisplay);
   glutMainLoop( );
   return 0;
}
```

5.4 윈도우와 뷰포트-윈도우와 뷰포트

- 윈도우를 분할
  - 그리기가 뷰포트 내부로 제한됨
- 왜곡
  - 뷰포트 미 설정시 기본값으로 윈도우 = 뷰포트
  - 윈도우 크기조절에 따라 뷰포트 내부 그림도 자동으로 크기조절
  - 별도 뷰포트 설정에 의해 왜곡 방지

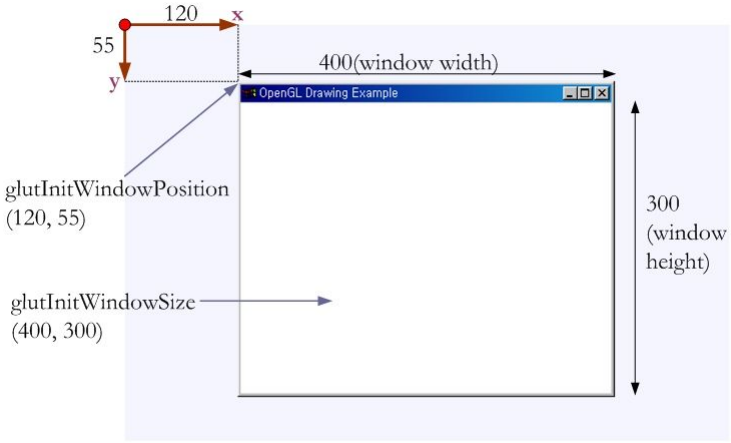


[그림 5-24] 뷰 포트



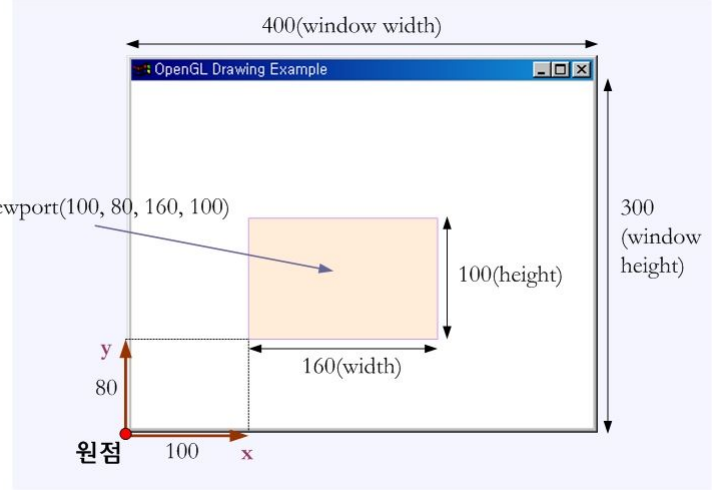
[그림 5-25] 왜곡

GLUT 윈도우 제어



[그림 5-26] GLUT 윈도우 제어명령

GL의 뷰포트 설정



[그림 5-27] 지엘의 뷰포트 설정

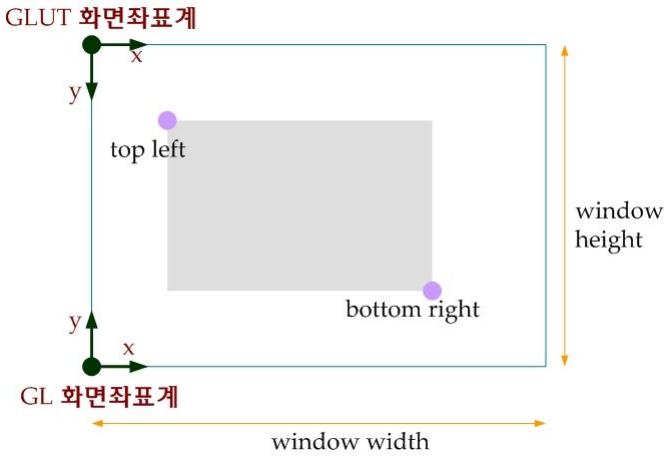

```
void MyDisplay( ){              //디스플레이 콜백함수
    glClear(GL_COLOR_BUFFER_BIT);//GL 상태변수 설정
    glViewport(0, 0, 300, 300);
    glColor3f(1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);                  //입력 기본요소 정의
          glVertex3f(-0.5, -0.5, 0.0);   glVertex3f(0.5, -0.5, 0.0);
          glVertex3f(0.5, 0.5, 0.0);     glVertex3f(-0.5, 0.5, 0.0);
    glEnd( );
    glFlush( );
}
int main(int argc, char** argv) {
    glutInit(&argc,argv);       //GLUT 윈도우 함수
    glutInitDisplayMode(GLUT_RGB);
    glutInitWindowSize(300, 300);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("OpenGL Sample Drawing");
    glClearColor (0.0, 0.0, 0.0, 1.0);    //GL 상태변수 설정
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity( );
    glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);
    glutDisplayFunc(MyDisplay); //GLUT 콜백함수 등록
    glutMainLoop( );            //이벤트 루프 진입
    return 0;
}
```

GLUT 모델링

- **Cube**
  - void glutSolidCube (GLdouble size);
  - void glutWireCube (GLdouble size);
- **Sphere**
  - void glutSolidSphere(GLdouble radius, GLint slices, GLint stacks);
  - void glutWireSphere(GLdouble radius, GLint slices, GLint stacks);
- **Torus**
  - void glutSolidTorus (inR, outR, nSides, rings);
  - void glutWireTorus (inR, outR, nSides, rings);
- **Cone**
  - void glutSolidCone (base, height, slices, stacks);
  - void glutWireCone (base, height, slices, stacks);
- **Tetrahedron : 정 4면체**
- **Icosahedron : 정 20면체**
- **Teapot**

19

5.5 콜백 프로그래밍–GL의 화면좌표, GLUT 화면좌표



[그림 5-45] GLUT, GL의 화면좌표

20

Reshape Callback

- glOrtho()
- Code5-5: 194~195쪽

21

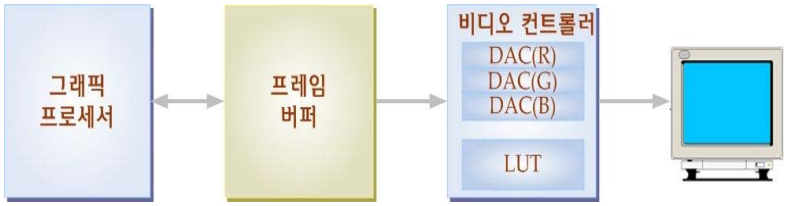
Keyboard / Mouse / Menu Callback

- glutKeyboardFunc()
  - 표5-3: 특수키에 대한 GLUT정의
  - 코드5-6: 키보드 콜백 (198~199쪽)
- glutMouseFunc()
- glutMotionFunc()
- glutPassiveMotionFunc()
- glutEntryFunc()
  - 코드5-7: 마우스 콜백 (201~202쪽)
- 메뉴 콜백
  - 코드5-8: 메뉴 콜백 (205~206쪽)
- 계단식 메뉴
  - 코드5-9: 계단식 메뉴 (207~208쪽)
- Idle Callback :코드5-10
- Timer Callback: 코드5-11, 12

22

5.6 애니메이션과 더블 버퍼링–프레임 버퍼

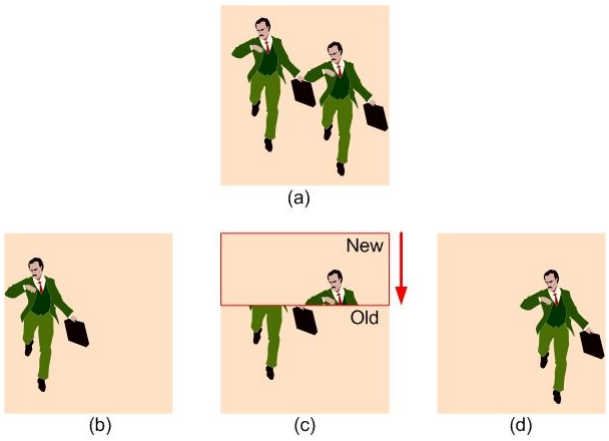
- 2중 포트 구조
  - **Read Port, Write Port**
  - 버퍼를 읽어서 화면에 뿌리는 것은 거의 동시
  - 버퍼에 기록하는 것은 상대적으로 느림
  - 애니메이션에서 문제가 됨



[그림 5-59] Dual-Port 프레임 버퍼

23

애니메이션 시의 문제

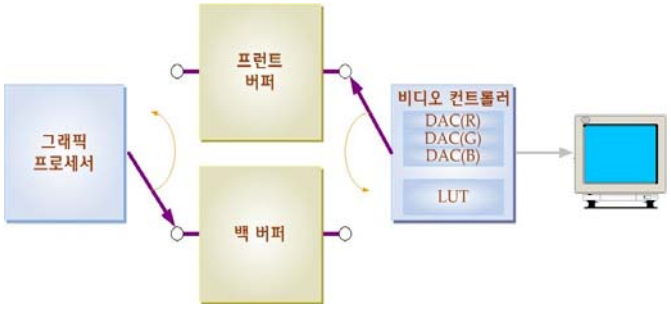


[그림 5-60] 프레임 버퍼 쓰기와 읽기

24

더블 버퍼링

- 프런트 버퍼와 백 버퍼
  - **glutInitDisplayMode(GLUT_DOUBLE);**
  - **glutSwapBuffers();**

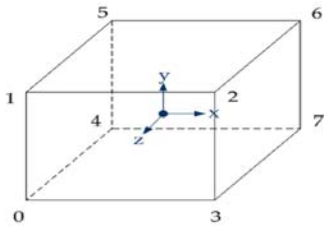


[그림 5-61] 더블 버퍼링

5.7 정점배열-육면체 그리기

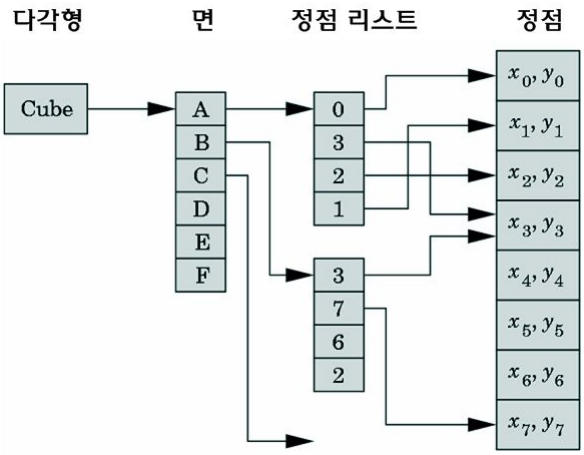
- **GLfloat MyVertices[8][3] = {{-0.25, -0.25, 0.25}, {-0.25, 0.25, 0.25}, {0.25, 0.25, 0.25}, {0.25, -0.25, 0.25}, {-0.25, -0.25, -0.25}, {-0.25, 0.25, -0.25}, {0.25, 0.25, -0.25}, {0.25, -0.25, -0.25}};**
- **GLfloat MyColors[8][3]={{0.2, 0.2, 0.2}, {1.0, 0.0, 0.0}, {1.0, 1.0, 0.0}, {0.0, 1.0, 0.0}, {0.0, 0.0, 1.0}, {1.0, 0.0, 1.0}, {1.0, 1.0, 1.0}, {0.0, 1.0, 1.0}};**
- **정점 0, 3, 2, 1으로 구성된 면 (반시계 방향으로 명시)**
  - **glBegin(GL_POLYGON);**

```
glColor3fv(MyColors[0]); glVertex3fv(MyVertices[0]);
glColor3fv(MyColors[3]); glVertex3fv(MyVertices[3]);
glColor3fv(MyColors[2]); glVertex3fv(MyVertices[2]);
glColor3fv(MyColors[1]); glVertex3fv(MyVertices[1]);
glEnd();
```



[그림 5-62] 육면체

계층구조적 표현



[그림 5-64] 물체의 계층적 자료구조

정점 배열

```
GLfloat MyVertices[8][3] = {{-0.25,-0.25,0.25}, {-0.25,0.25,0.25}, {0.25,0.25,0.25},
    {0.25,-0.25,0.25}, {-0.25,-0.25,-0.25}, {-0.25,0.25,-0.25}, {0.25,0.25,-0.25},
    {0.25,-0.25,-0.25}};
GLfloat MyColors[8][3]={{0.2,0.2,0.2}, {1.0,0.0,0.0}, {1.0, 1.0, 0.0}, {0.0,1.0,0.0},
    {0.0,0.0,1.0}, {1.0,0.0,1.0}, {1.0,1.0,1.0}, {0.0,1.0,1.0}};
GLubyte MyVertexList[24]={0,3,2,1, 2,3,7,6, 0,4,7,3, 1,2,6,5, 4,5,6,7,
    0,1,5,4};

void MyDisplay( ){
   …
   glEnableClientState(GL_COLOR_ARRAY);
   glEnableClientState(GL_VERTEX_ARRAY);
   glColorPointer(3, GL_FLOAT, 0, MyColors);
   glVertexPointer(3, GL_FLOAT, 0, MyVertices);
   …
   for(GLint i = 0; i < 6; i++)
      glDrawElements(GL_POLYGON, 4, GL_UNSIGNED_BYTE,
                     &MyVertexList[4*i]);
   …
  }
```

5.8 디스플레이 리스트-지엘의 실행모드

- 직접 모드(Immediate Mode)
  - 화면 렌더링과 동시에 물체 정보를 모두 파기
  - 다시 그리려면 전체 코드를 다시 실행
- 보류모드(Retained Mode)
  - 이미 정의된 물체를 컴파일 된 형태로 재사용
- 디스플레이 리스트
  - 기본요소(Primitives), 상태변수(State Variable), 영상(Image)
  - 이동, 회전, 조명 작업과 관련된 모든 명령
  - 반복적으로 실행되어야 할 요소를 디스플레이 리스트 내부에 포함
  - 프로그램 속도 향상에 필수적임

실습 문제 3 : glut 다양한 기능 활용하기

- 교재 5장의 본문내의 코드들을 활용하여 다양한 gl및 glut 기능 활용하기
- 윈도우 및 Viewport 생성 : 윈도우를 2개의 viewport로 분할함
- 좌측 뷰포트: 각종 콜백 사용하기
  - 키보드:
    - q:프로그램 종료
    - 1~7: glut 객체 선택(메뉴와 연동)
  - 메뉴: 좌측 Viewport에 그려질 glut 객체를 선택
  - Reshape callback: glOrtho() 를 활용할 것
- 우측 뷰포트: Teapot을 그리고 회전시킴
  - Timer callback을 이용
  - glRotate()
- 정점배열, 디스플레이리스트 활용하기