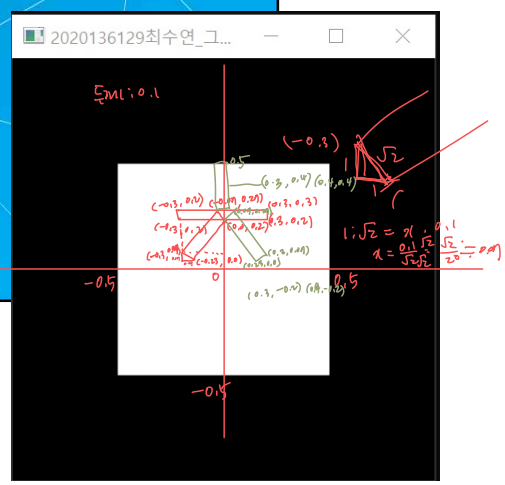
03 CHAPTER

OpenGL 프로그래밍



3장. 학습 내용

- 컴퓨터 그래픽스 표준
- OpenGL이란?
- OpenGL 프로그래밍
- GLUT(GL Utility Toolkit)란?
- GLUT를 이용한 윈도우 프로그래밍

2

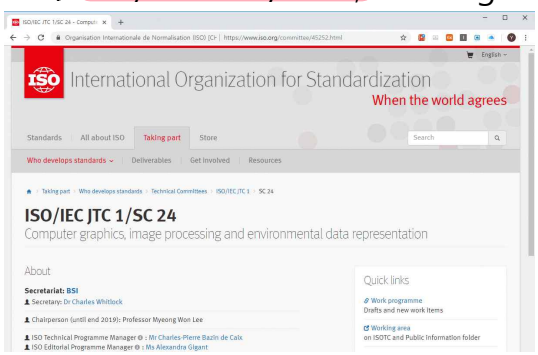
3.1 컴퓨터 그래픽스 표준

- 그래픽스 분야 표준화
- 그래픽 기본요소와 속성
- API
- 그래픽스 함수 분류

3

그래픽스 분야 표준화

- "표준화"의 정의
  - "주어진 여건에서 최적의 질서를 유지하기 위해, 현존하거나 잠재하는 문제들에 대해, 공유성과 재 사용성을 높이기 위한 기반을 확립하는 행위"
- 그래픽 분야: ISO/IEC JTC1/SC24, Working Group



4

ISO/IEC JTC1/SC24, Working Group

- Working Group들
  - "하드웨어 구조(Architecture)"
  - "응용프로그램 인터페이스(API, Application Program Interface)"
  - "메타파일 및 인터페이스(Metafile and Interface)"
  - "언어 수용(Language Binding)"
  - "표준안의 타당성 검증 및 등록(Validation Testing and Registration)"

5

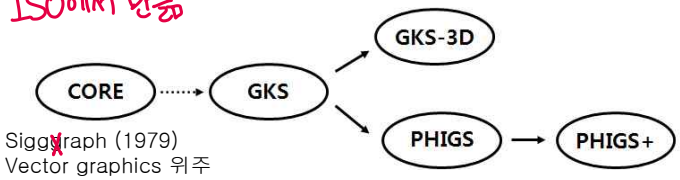
그래픽 분야 표준의 목표

- 주전산기 독립성(Host Machine Independence)
  - 동일한 프로그램을 가지고서 다양한 모든 하드웨어에서 사용할 수 있어야 한다.
- 장비 독립성(Device Independence)
  - 동일 기능을 수행하는 입출력 장비의 종류가 달라도 프로그램 명령은 동일해야 한다.
- 프로그램 언어 독립성(Programming Language Independence)
  - 프로그램 작성에 어떠한 프로그램 언어를 사용해도 된다.
- 운영자 이식성(Operator Portability)
  - 새로운 프로그램 사용법을 누구라도 쉽게 터득할 수 있어야 한다.

6

그래픽 표준의 흐름

- 표준에서 언급하는 내용: 4계층 구조
  - 응용 프로그램 레벨: 추상적인 수준
  - 가상 레벨: 출력할 내용을 기본요소를 이용해 기술
  - 논리적 레벨: 기본요소를 그리기 위한 과정 기술
  - 물리적 레벨: 개별 입출력 장비에 관한 내용
- 표준의 흐름: GKS, PHIGS



Siggraph (1979)
Vector graphics 위주
사장됨

GKS

- GKS (Graphical Kernel System) : 1985
  - 유럽에 의해 주도
  - 2차원 위주
  - 이후 GKS-3D로 발전(1988)
  - 그래픽 파일 출력
    - 기본요소 수준에서 서술한 가상 레벨(Virtual Level) 저장
    - 기본요소의 위치 좌표, 속성, 가시성, 변환 정보를 저장

PHIGS

- Programmer's Hierarchical Interactive Graphics System
  - 미국에 의해 주도(1989)
  - CAD 개념 반영
  - 3차원 모델링(Modeling), 가시화(Viewing) 등에 주안점
  - 상관관계를 포함한 물체의 집합 = 구조체(Structure)
    - 구조체 관통(Traversal)에 의한 드로잉
    - 현 변환 행렬(現, CTM, Current Transformation Matrix) 개념
  - 그래픽 파일 출력
    - 기본요소에 관한 정보+응용 프로그램 레벨에서 이들의 관계
    - CSG의 불리언 연산, 로봇 팔의 객체 계층구조 저장
- PHIGS+
  - realistic 3D rendering

De facto (industry) 표준들

산업계 (ISO에서 하는것X)

- QuickDraw (Apple)
- Xlib (X-window)
- GL (SGI), OpenGL
- DirectDraw, Direct3D (MS)
- Renderman (Pixar)
- Java3D
- Farenheit

Language Binding

- 그래픽 표준
  - 그래픽 명령어에 대한 표준을 정하되 추상적인 수준에서만 정함
- 컴파일러 개발자
  - 정의를 준수하는 범위 내에서 명령어나 문법을 자유롭게 설계할 수 있음
- 예) 꺾은선(Polyline)
  - 표준

    POLYLINE
    Parameters
    In number of points (2, … , n) — n 점의 개수
    In points WC — n x P — P: 점의 (전역) 좌표값
  - Fortran77 명령어

    SUBROUTINE GPL(N, PXA, PYA)

11

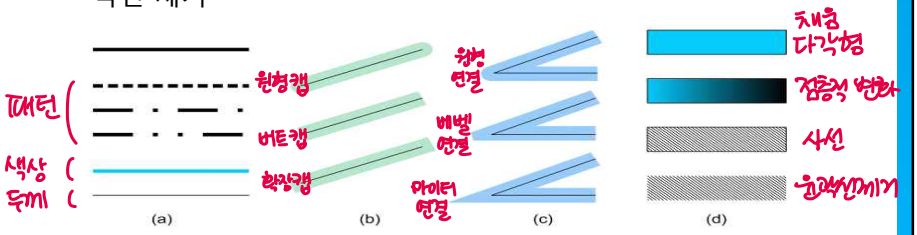
그래픽 기본요소와 속성

- 그래픽 기본 요소들
  - Point
  - Line
  - Polygon
  - Polyline
  - Text
  - ⋮
- 속성들
  - 각 그래픽 기본요소에 대해 어떤 속성들이 가능한가?

12

기본요소의 속성들

- 속성 (Attributes)
  - 패턴, 색상, 두께
  - 캡: 원형 (Round Cap), 버트 (Butt Cap), 확장 (Projection Cap)
  - 연결: 원형 (Round Join), 베벨 (Bevel Join), 마이터 (Miter Join)
  - 채움 다각형(Filled Polygon), 점층적 변화(Gradation), 사선 , 윤곽선 제거

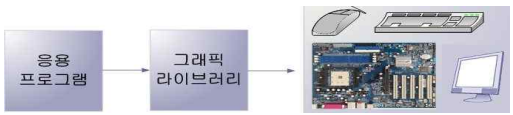


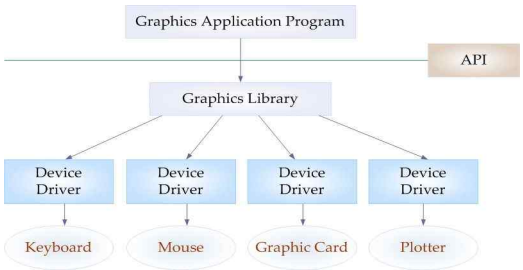
- *Lab: Powerpoint에서 제공하는 primitive와 attribute는?*

13

API

- API: 응용프로그램 인터페이스
  - PHIGS, GKS = 추상적 수준의 API
  - API: 라이브러리 함수 (고수준 API / 저수준 API)

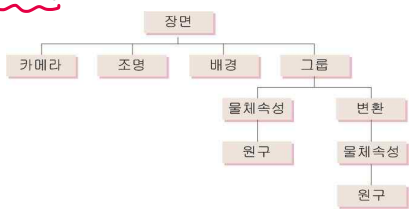




14

고수준 그래픽 API

- 장면 그래프(Scene Graph)
  - 그룹 노드는 nested structure
  - 관통(traversal)에 의해 장면을 그려 냄



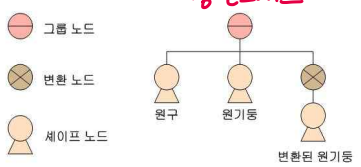
- 고수준 그래픽 API 예
  - VRML(Virtual Reality Modeling Language)
  - Java3D
  - 오픈 인벤터(open Inventor)
- OpenGL ➔ 저수준 그래픽 API

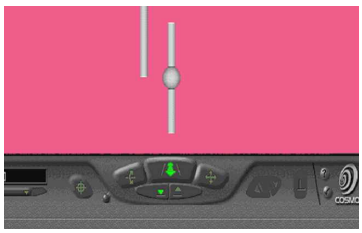
VRML 예

장면그래프

```
#VRML V2.0 utf8
Shape {
    appearance Appearance {
                material Material { }
    }
    geometry Sphere {
                radius 1.2
    }
}
Shape {
    appearance Appearance {
                material Material { }
    }
    geometry Cylinder {
                radius 0.3   height 5.0   top FALSE
    }
}
Transform {
   translation -6.0 2.0 0.0
   children {
      Shape {
          appearance Appearance {
                      material Material { }
          }
          geometry Cylinder {
                      radius 0.3   height 5.0    top FALSE ]
```

장면그래프





저수준 그래픽 API

- 장면을 묘사하는 것이 아니라 구체적 프러시져를 호출
- **OpenGL**, DirectX from Microsoft 등
- 하드웨어와 직접 연관 (하드웨어 성능을 최대한 발휘)
- Inventor, VRML, Java3D 등 고수준 API의 기반
- 드라이버에 비해서는 상대적으로 고수준 함수

17

그래픽스 함수 분류

- 기본요소 함수(primitive functions)
  - graphical primitives --> display
  - primitives: vertex, line, polygon, cylinder, sphere, cone, ...
- 속성 함수(attribute functions)
  - To determine how the primitives are to be displayed.
  - attributes: color, thickness, font, ...
- 관측 함수(viewing functions)
  - To specify the viewing parameters.
  - viewing parameters: window, viewport, clipping functions

    WC ==> NDC, NDC ==> DC
- 변환 함수(transformation functions)
  - geometric transformations within WC and NDC

18

- 입력 함수(input functions)
  - handles the logical input devices
- 제어 함수(control functions)
  - To control displays, input devices, system states,…
  - e.g., close, open, resolution setting, mode setting, …
- Inquiry functions 현재 상태 변수가 뭐니?
  - To find out the properties or the internal states of the graphics system
- Segmentation functions
  - To allow users to form/manipulate identifiable groups of primitives
- Meta-file functions
  - To save and load graphics files (e.g., VRML)

19

3.2 OpenGL이란?

- OpenGL이란?
- OpenGL 파이프라인
- OpenGL 상태변수
- OpenGL 속성할당 방법
- 상태변수 관련 함수
- OpenGL의 여러 모듈들

20

OpenGL이란?

- 1982년 실리콘 그래픽스는 자사의 워크스테이션용 그래픽 하드웨어를 파이프라인화 하여 처리속도를 획기적으로 향상
  - 이를 위해 개발한 라이브러리: GL

- OpenGL
  - SGI에서만 돌아가던 GL을 다양한 플랫폼에도 활용할 수 있도록 일반화 한 것
  - 이식성을 위해 기계에 종속적인 부분을 제거
  - 호환성 좋음: cf. DirectX from Microsoft
  - 저수준 그래픽스 API: 하드웨어와 거의 직접 연관 (하드웨어 성능을 최대한 발휘)
  - Inventor, VRML, Java3D 등 고수준 API의 기반

21

OpenGL 설계 원리

- 범용성(Generality)
  - 워크스테이션, 수퍼 컴퓨터, 개인용 컴퓨터. 운영체제에 무관
- 효율성(Performance)
  - 그래픽 하드웨어의 가속 기능을 최대한 발휘.
  - 회사마다 서로 다른 기능. 공통적인 부분을 찾아내어 그 성능을 극대화
  - 공통부분이 아닌 것에 대해서는 활성화 또는 비활성화 등 기능 모드를 제공.
- 독립성(Orthogonality)
  - 기능 간의 독립성을 최대한 보장.
  - 기능끼리 서로 얽혀 발생하는 오류를 방지.
- 완전성(Completeness)
  - 특정 하드웨어 기능에 대해서는 ARB 확장 형태로 명령어를 제공
  - 다수의 하드웨어가 확장 기능을 지원하면 표준기능으로 변경.
  - 소프트웨어적으로라도 실행할 수 있도록 배려
- 상호 작업성(Interoperability)
  - 그래픽 명령은 A 컴퓨터에서 내리되 실행은 B 컴퓨터에서

[그림 4-16]
네트워크를 사용한 그래픽

22

OpenGL 특징

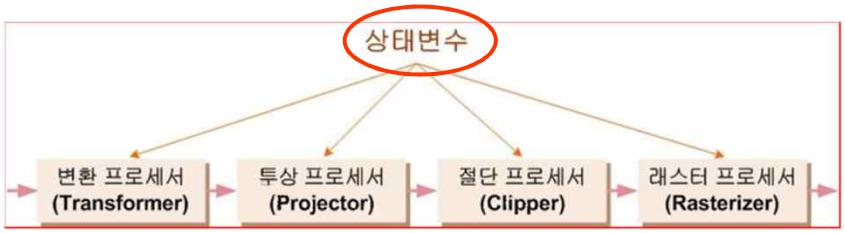
- A software interface to graphics hardware
- A low- level graphics rendering and imaging library
- Only includes operations which can be accelerated
- A layer of abstraction between graphics hardware and an application program
- Window system and OS Independent (use with Unix, Microsoft Windows, IBM OS/ 2, Apple Mac Os)
- 150 distinct commands
- Object specification + image generation
- Client-server model
- 3D rendering
- Simple primitives: points, lines, polygons (pixels, images, bitmaps)

OpenGL 파이프라인

- GPU 설계원리
  - CPU 파이프라인과 유사
  - 분업에 의한 동시처리로 처리속도를 극대화. Ex. 컨베이어 시스템
  - 파이프라인 서브 프로세스는 모두 하드웨어화

OpenGL 상태변수

- OpenGL: 거대한 State Variable Machine
- OpenGL의 역할 ➔ 상태변수 설정
- 파이프라인은 상태변수를 참조해서 자동으로 실행됨

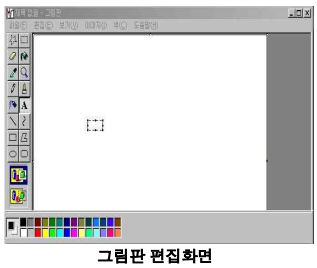


OpenGL 속성할당 방법

- Parameter List v.s. System Table
- 파라미터 리스트
  - drawLine((1, 0), (3, 0), 3, 4, (255, 0, 0));
  - drawLine((3, 0), (2, 5), 3, 4, (255, 0, 0));
  - drawLine((2, 5), (1, 0), 3, 4, (255, 0, 0));

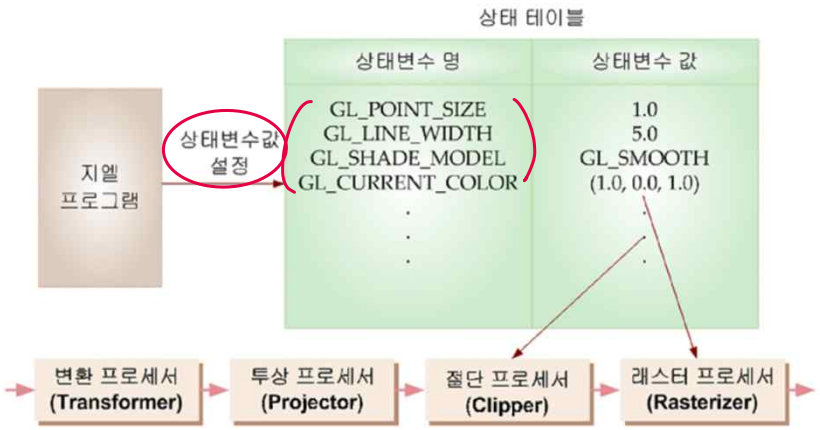
시스템 테이블
  - setLineStyle(2);
  - setLineWidth(4);
  - setLineColor(255, 0, 0);
  - drawLine((1, 0), (3, 0));
  - drawLine((3, 0), (2, 5));
  - drawLine((2, 5), (1, 0));

- ISO ➔ System Table 권장
- "현 상태" 라는 개념
  - Current State



그림판 편집화면

프로그램, 상태변수, 파이프라인



27

상태변수 관련 함수

- 상태변수 설정
  - glColor3f(1.0, 1.0, 1.0); 명령어
  - GL_CURRENT_COLOR 상태변수 값을 (1.0, 1.0, 1.0)으로 설정
  - 다른 명령에 의해 값이 바뀔 때까지 모든 물체를 그릴 때 유효함.
- 상태변수 설정 (동시에 정의됨)
  - glPointSize(0.5);
  - glLineWidth(5);
  - glShadeModel(GL_SMOOTH);
- 상태변수 검색 "C "언어 기반
  - float MyColor[3]; → 임의 배열
  - glGetFloatv(GL_CURRENT_COLOR, MyColor); → 검색 함수
- 기능관련 상태변수
  - glEnable(GL_LIGHTING); 조명 모드를 활성화
  - glDisable(GL_LIGHTING); 조명 모드를 비활성화

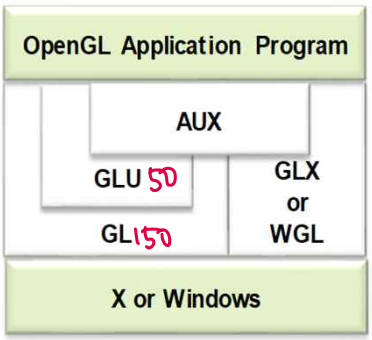
gl… → GL 라이브러리
glu… → GLU 라이브러리
glut… → GLUT라이브러리

28

OpenGL의 여러 모듈들

- OpenGL Core Library: GL
  - OpenGL의 핵심
  - OpenGL을 제어하는 기본적인 함수들의 집합체
  - OpenGL의 렌더링 기능을 제공하는 함수 라이브러리

- **GLU**
- **GLUT**
- **GLAUX**



29

OpenGL Utility Library: GLU

- 50여개의 함수. GL 라이브러리의 도우미
  - 다각형 분할, 투상, 2차원 곡면, 너브스 등
  - GL 사용 시 발생하는 많은 반복 작업 간소화
  - 고급 기능을 제공하는 함수들로 GL 함수로 작성됨
- (예) 구, 실린더, 디스크 등의 Object 생성
  - 정점을 일일이 계산하여 glVertex함수로 하나하나 찍는 방법
  - GLU 함수 호출

```
GLUquadricObj *pObj
pObj = gluNewQuadric();
gluSphere(pObj , 30.0 , 10 , 10);
gluDeleteQuadric(pObj);   // 객체를 삭제
```

30

기타 라이브러리

- **OpenGL Auxiliary: GLAUX**
  - 플랫폼에 독립적인 윈도우 제어기능
  - GLU를 능가하는 물체 생성기능을 가지는 툴킷 라이브러리
  - OpenGL 프로그램을 처음 작성하는 사람이 사용하기 용이함
  - 윈도생성, 입력제어, 3차원 물체 생성, 더블 버퍼링, 텍스처 매핑용 이미지의 로딩 등
- **OpenGL Related Libraries**
  - GLX
    - OpenGL Extension to the X Window System
  - **GLUT**
    - windowing utility library for OpenGL
    - handle keyboard, mouse, and redraw events
  - Open Inventor
    - object-oriented developers toolkit

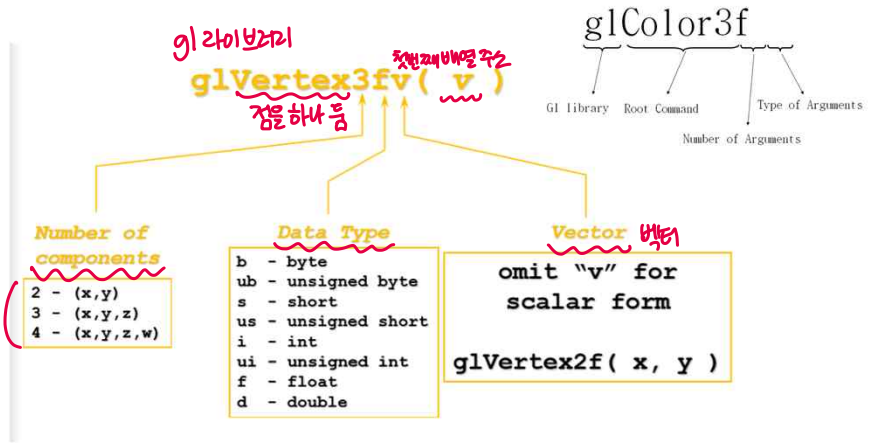
31

3.3 OpenGL 프로그래밍

- OpenGL 함수들의 기본 구조
- OpenGL 그래픽 요소들(primitives)
- 처음 만들어 본 OpenGL 프로그램
- OpenGL은 거대한 상태 변수 기계이다.
- 자신만의 그리기 함수를 만들자.

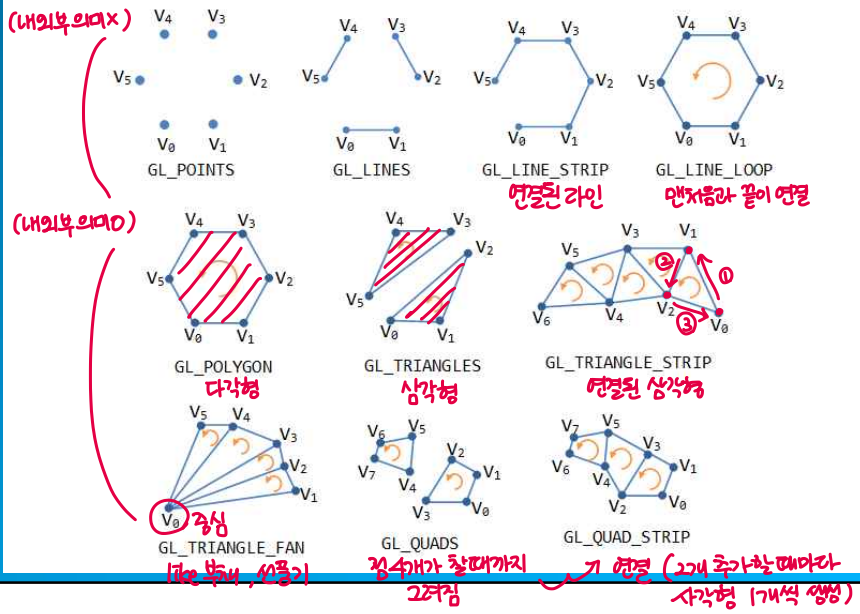
32

OpenGL 함수들의 기본 구조

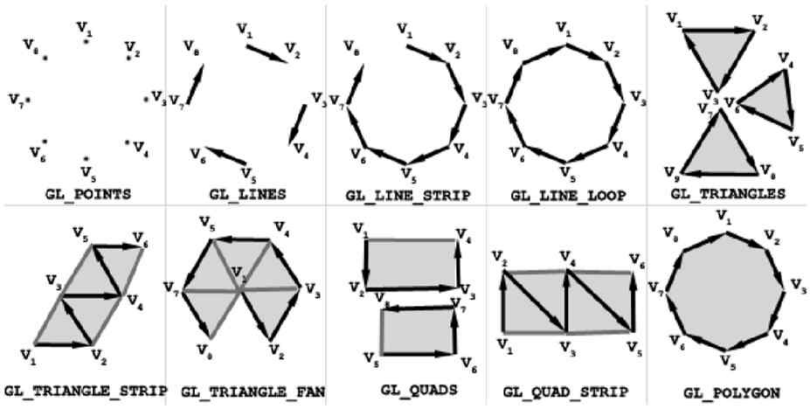


33

OpenGL의 그래픽 요소들(primitives)



34

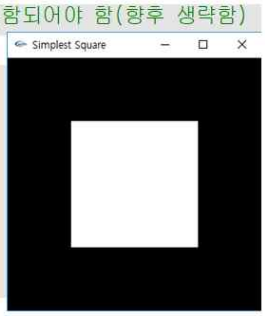


35

처음 만들어 본 opengl 프로그램

프로그램 3.1 처음 만들어 본 opengl 프로그램

```
1. #include <GL/glut.h> // glut.h 포함. 모든 코드에 포함되어야 함(향후 생략함)
2. void display() {
3.     glColor3d(1, 1, 1);
4.     glBegin(GL_POLYGON);
5.             glVertex3f(-0.5, -0.5, 0.0);
6.             glVertex3f(0.5, -0.5, 0.0);
7.             glVertex3f(0.5, 0.5, 0.0);
8.             glVertex3f(-0.5, 0.5, 0.0);
9.     glEnd();
10.    glFlush();
11. }
12. int main(int argc, char** argv) {
13.    glutCreateWindow("Simplest Square");
14.    glutDisplayFunc(display);
15.    glutMainLoop();
16.    return 0;
17. }
```



36

OpenGL은 거대한 상태 변수 기계이다.

- 상태 변수
  - 선택된 그래픽 요소가 어떻게 그려져야 할지를 나타내는 변수
  - Parameter List, System Table
  - ISO ➔ System Table 권장
- 파라미터 리스트 방식

```
drawLine( x1, y1, x2, y2, lineWidth, R, G, B, style );
```

- 시스템 테이블 방식

```
setLineWidth(lineWidth);    // 현재의 선분 두께를 lineWidth로 변경
setColor(R,G,B);            // 현재색상을 (R,G,B)로 변경
setLineStyle(style);        // 현재 선분 스타일을 style로 변경
moveto(x1,x2);              // 현재 점의 위치를 (x1, y1)으로 옮김
lineto(x2,y2);              // 현재 위치에서 (x2,y2)로 선을 그음
```

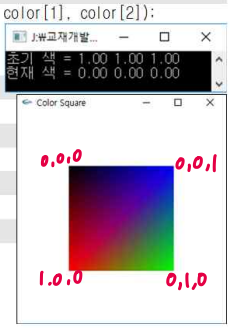

컬러 사각형 그리기

프로그램 3.2 컬러 사각형 그리기 함수

```
1. void display() {
2.     glClearColor(1.0, 1.0, 1.0, 1.0);→배경흰색
3.     glClear(GL_COLOR_BUFFER_BIT);→색 초기화
4.
5.     float color[10];                    상태변수
6.     glGetFloatv(GL_CURRENT_COLOR, color);검색
7.     printf("초기색 = %3.2f %3.2f %3.2f₩n", color[0], color[1], color[2]);
8.
9.     glBegin(GL_POLYGON);
10.        glColor3d(1, 0, 0);
11.        glVertex3d(-0.5, -0.5, 0.0);
12.        glColor3d(0, 1, 0);
13.        glVertex3d(0.5, -0.5, 0.0);
14.        glColor3d(0, 0, 1);
15.        glVertex3d(0.5, 0.5, 0.0);
16.        glColor3d(0, 0, 0);
17.        glVertex3d(-0.5, 0.5, 0.0);
18.    glEnd();
19.    glFlush();→화면출력
20.    glGetFloatv(GL_CURRENT_COLOR, color);
21.    printf("현재색 = %3.2f %3.2f %3.2f₩n₩n", color[0], color[1], color[2]);
22. }
```

검색 (5~6)

검색 (20~21)



OpenGL 상태 변수의 종류

- https://docs.microsoft.com/en-us/windows/desktop/opengl/state-variables
  - Transformation
  - Coloring, Lighting
  - Rasterization
  - Texturing
  - Framebuffer control
  - Pixel, Evaluator, Hint
  - Implementation-dependent
  - Implementation-dependent pixel depth
  - Miscellaneous

39

자신만의 그리기 함수를 만들자

- gl, glu, glut 등과 다른 나만의 함수(이니셜 등 사용)
  - 예) glkColorSquare()

프로그램 3.3 나만의 컬러풀한 사각형 그리기 함수

```
1. void glkColorSquare(double w, bool bFill)
2. {
3.     if( bFill ) glBegin(GL_POLYGON);
4.     else glBegin(GL_LINE_LOOP);
5.
6.     glColor3d(1, 0, 0); glVertex3d(-w / 2, -w / 2, 0.0);
7.     glColor3d(0, 1, 0); glVertex3d( w / 2, -w / 2, 0.0);
8.     glColor3d(0, 0, 1); glVertex3d( w / 2,  w / 2, 0.0);
9.     glColor3d(0, 0, 0); glVertex3d(-w / 2,  w / 2, 0.0);
10.    glEnd();
11. }
```

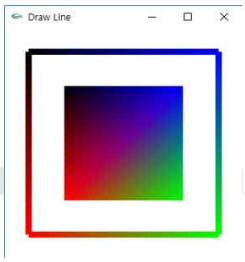
40

나만의 함수 사용 코드

프로그램 3.4 나만의 함수를 이용해 컬러 사각형 그리기

```
1. extern void glkColorSquare(double w, bool bFill=true)
2. void display() {
3.     glClearColor(1.0, 1.0, 1.0, 1.0);
4.     glClear(GL_COLOR_BUFFER_BIT);
5.
6.     glLineWidth(16);    // 선분 두께 속성을 변경
7.     glkColorSquare(1.6, false);   // 나만의 함수 호출 --> 테두리만 그림
8.     glkColorSquare(1);            // 나만의 함수 호출 --> 채움
9.     glFlush();
10. }
11. int main(int argc, char** argv)
12. {
13.     glutCreateWindow("Draw Line");
14.     glutDisplayFunc(display);
15.     glEnable(GL_LINE_SMOOTH);
16.     glutMainLoop();
17.     return 0;
18. }
```

두께 테두리만!



41

3.4 GLUT(GL Utility Toolkit) 란?

- GLUT의 윈도우 초기화와 관리
- GLUT 메뉴 만들기
- GLUT 콜백함수 등록
- GLUT 3D 모델
- 실습: GLUT 설치

42

GLUT

- GLUT는 OpenGL의 일부가 아니라 프리웨어(Freeware)
- 공식 홈페이지: http://www.opengl.org/resources/libraries/glut/
- 특징
  - OpenGL 렌더링을 위한 다양한 윈도우 관리 기능 제공
  - 콜백 처리를 이용한 ~~통한~~ 이벤트 기반 윈도우 프로그래밍 지원
  - 다양한 팝업 메뉴 지원
  - 구(Sphere), 정육면체(cube)등 다양한 모델링 함수 제공
  - API V.3: https://www.opengl.org/resources/libraries/glut/spec3/spec3.html

GLUT의 윈도우 초기화와 관리

윈도우 초기화	glutInit	GLUT 초기화
	glutInitWindowPosition, glutInitWindowSize	윈도우 위치와 크기 설정
	glutInitDisplayMode	디스플레이 모드
윈도우 관리	glutCreateWindow	윈도우 만들기
	glutCreateSubWindow	서브 윈도우 만들기
	glutSetWindow, glutGetWindow	현재 윈도우 설정, 반환
	glutDestroyWindow	윈도우 해제
	glutPostRedisplay	현재윈도우를 다시 그리도록 함
	glutSwapBuffers	버퍼 교환(더블 버퍼 사용시)
	glutPositionWindow glutReshapeWindow	현재 윈도우의 위치와 크기 변경
	glutFullScreen	전체화면으로
	glutPopWindow, glutPushWindow	현재 윈도우의 순서 변경
	glutShowWindow, glutHideWindow, glutIconifyWindow	현재 화면을 보이거나, 안보이거나, 아이콘화 함
	glutSetWindowTitle glutSetIconTitle	윈도우와 아이콘의 타이틀 변경
	glutSetCursor	현재 윈도우의 커서 영상 변경

GLUT 메뉴 만들기

```
glutCreateMenu,
glutDestroyMenu,
glutSetMenu,
glutGetMenu
glutAddMenuEntry,
glutAddSubMenu,
glutChangeToMenuEntry
glutChangeToSubMenu,
glutRemoveMenuItem,
glutAttachMenu,
glutDetachMenu
```

45

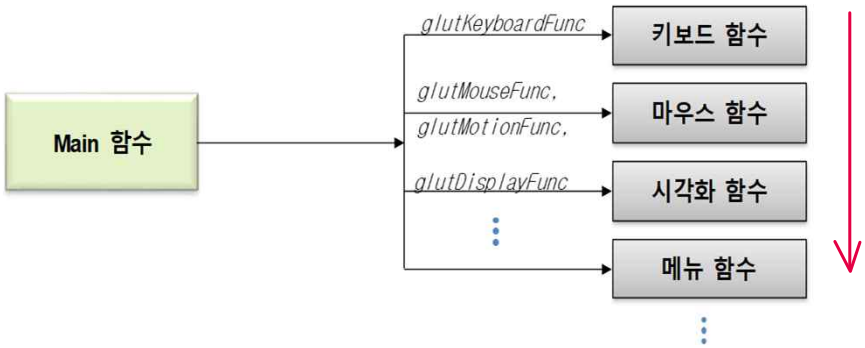
GLUT 콜백함수 등록

glutDisplayFunc	디스플레이 콜백(화면을 다시 그려야 할 때)
glutOverlayDisplayFunc	오버레이 디스플레이 콜백
glutReshapeFunc	화면의 크기변환 시 호출되는 함수
glutKeyboardFunc glutSpecialFunc	키보드 입력시 호출되는 함수 특수키 입력시 호출되는 함수
glutMouseFunc, glutMotionFunc, glutPassiveMotionFunc (마우스휠X 움직임)	마우스 클릭이나 움직임 발생 시 호출되는 함수
glutVisibilityFunc	화면의 visibility가 변경되면 호출되는 함수
glutEntryFunc	마우스가 화면 진입/진출시 호출되는 함수
glutSpaceballMotionFunc glutSpaceballRotateFunc glutSpaceballButtonFunc	스페이스볼의 움직임이나 회전, 버튼 이벤트 발생 시 호출되는 함수
glutButtonBoxFunc glutDialsFunc	dial과 button box 버튼 관련 처리함수 dial과 button box 다이얼 관련 처리함수
glutTabletMotionFunc glutTabletButtonFunc	태블릿 관련 모션과 키보드 관련 함수
glutMenuStatusFunc	메뉴 상태와 관련된 콜백 함수
glutIdleFunc, glutTimerFunc	글로벌 아이들과 타이머 관련 콜백

46

이벤트 처리

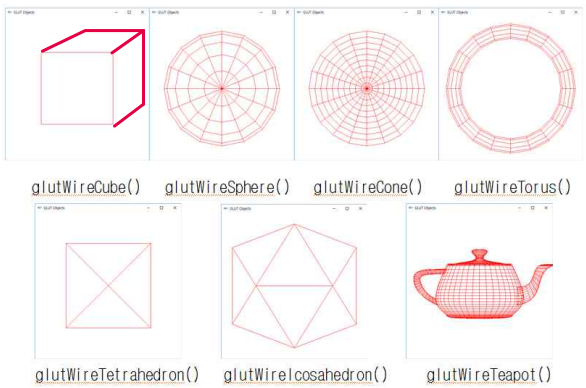
- glutMainLoop();
  - 반복적으로 프로그램의 이벤트 큐를 확인하고, 이벤트가 들어온 순서대로 콜백함수를 호출하여 처리



GLUT 3D Model

```
Basic Shapes: Cube, Sphere, Cone, Torus, Tetrahedron
    예) glutWireCube( ), glutSolidCube( )
Advanced Shapes: Octahedron, Dodecahedron, Icosahedron, Teapot
```

- 프로그램 3.5



실습3.1 : GLUT 설치

- GLUT는 직접 설치해야 함
  - Visual Studio를 설치하면 GL, GLU는 자동으로 설치되어 있음.
  - OpenGL 홈페이지(http://www.opengl.org/)부터 탐색해 볼 것
  - 탐색하다가 **freeglut 3.0.0 MSVC Package** 를 다운로드
  - 이제 설치하면 됨. (How???)
  - 동영상 확인

- GLUT Source File도 있음
  - 화려한(?) Windows Api 기술이 들어 있음
  - 인터넷에서 찾아볼 것. **glut-3.7.6-src.zip**
  - 여러 가지 test / sample / demo 프로그램들 테스트
  - Sample 폴더의 파일들을 컴파일해 볼 것

- C:₩Program Files (x86)₩Windows Kits₩10₩Include₩10.0.17763.0₩um₩gl
- C:₩Program Files (x86)₩Windows Kits₩10₩Lib₩10.0.17763.0₩um₩x86
- C:₩Windows₩SysWOW64

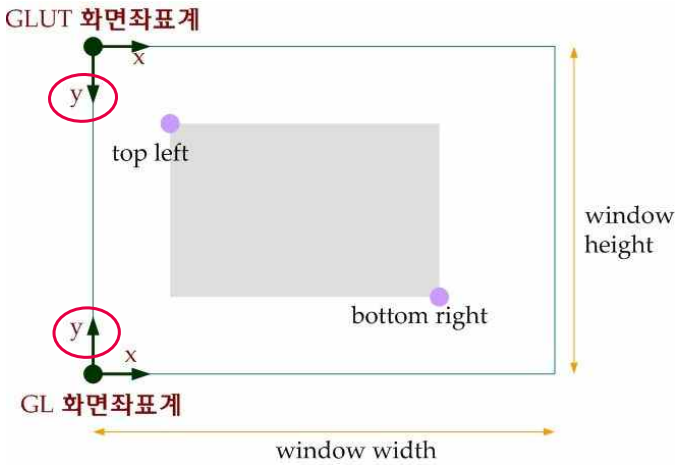
49

3.5 GLUT를 이용한 윈도우 프로그래밍

- GLUT 윈도우 제어
- 윈도우와 뷰포트
- GLUT 키보드, Reshape, 마우스 콜백 구현
- GLUT 메뉴 예제

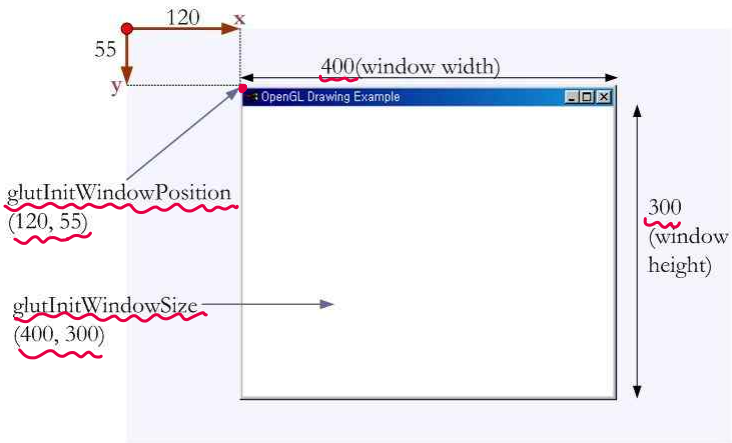
50

화면 좌표계: GLUT / GL



51

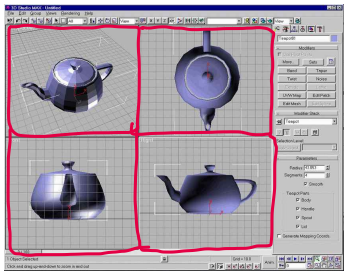
GLUT 윈도우 위치와 크기



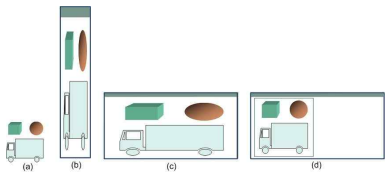
52

윈도우와 Viewport

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

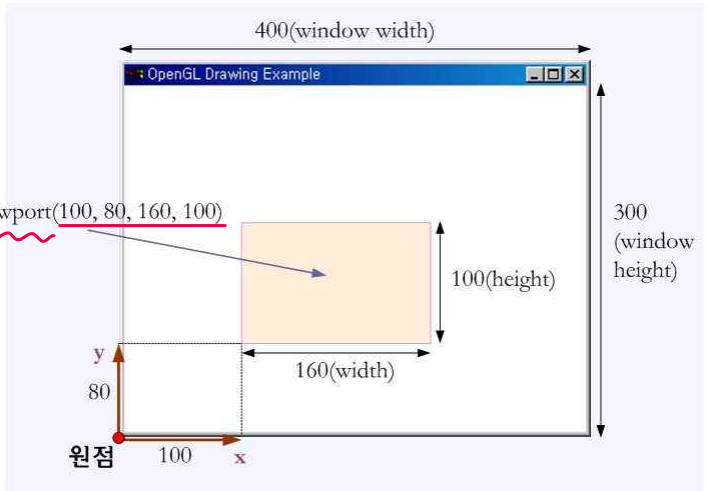


[그림 5-24] 뷰 포트



[그림 5-25] 왜곡

glViewport()

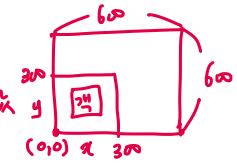


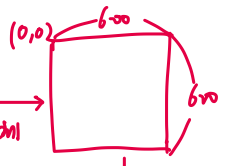
[그림 5-27] 지엘의 뷰포트 설정

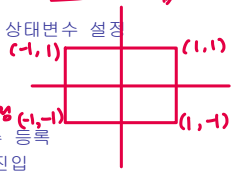
```
float MyColor[10];
glGetFloatv(GL_CURRENT_COLOR, MyColor);
printf("color = %.2f %.2f %.2f \n", MyColor[0], MyColor[1], MyColor[2]);

glBegin(GL_POLYGON);
    glColor3f(1.0, 0, 0);
    glVertex3f(-0.5, -0.5, 0.0);

    glColor3f(0.0, 1.0, 0);
    glVertex3f(0.5, -0.5, 0.0);

    glColor3f(1.0, 1.0, 1.0);
    glVertex3f(0.5, 0.5, 0.0);

    glColor3f(0.0, 0.0, 1.0);
    glVertex3f(-0.5, 0.5, 0.0);
glEnd();
```

```
void MyDisplay( ){
    glClear(GL_COLOR_BUFFER_BIT);
    glViewport(0, 0, 300, 300);
    glColor3f(1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
        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(600, 600);
    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;
}
```

glVertex3f(0.5, -0.5, 0.0);
glVertex3f(-0.5, 0.5, 0.0);







화면 갱신

- glutDisplayFunc(*MyDisplay*);
  - 화면을 다시 그려야 할 때 호출되는 함수

- Quiz: 언제 화면을 다시 그리는가? 테스트 할 것

화면 크기 변경

- glutReshapeFunc(*MyReshape*);
  - 화면의 크기가 변경되었을 때 (화면크기를 바꿀때마다 처음설정한 화면대로 동적으로 비율에 맞게 바뀜)
  - Code 05_05
    - glOrtho()
      ↳화면이 나타내는 3차원 공간의 크기를 바꿔줌

```
void MyReshape(int NewWidth, int NewHeight) {
    printf("화면 크기 = %d x %d\n", NewWidth, NewHeight);
    glViewport(0, 0, NewWidth, NewHeight);
    GLfloat WidthFactor = (GLfloat) NewWidth / (GLfloat) 300;
    GLfloat HeightFactor = (GLfloat) NewHeight / (GLfloat) 300;
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-1.0 * WidthFactor, 1.0 * WidthFactor,
        -1.0 * HeightFactor, 1.0 * HeightFactor, -1.0, 1.0);
}
```

57

키보드 입력

- glutKeyboardFunc(*MyKeyboard*);
  - 일반 문자 키 입력: Code 05_06

```
void MyKeyboard(unsigned char KeyPressed, int X, int Y) {
    switch (KeyPressed){
    case 'Q':
        exit(0); break;
```

(아스키, 마우스위치)

- glutSpecialFunc(*MyKeySpecial*);
  - 특수 키 입력: ex)화살표키

```
void MyKeySpecial(int key, int X, int Y) {
    switch (key){
    case GLUT_KEY_DOWN: …
```

(아스키)

58

마우스 이벤트

- glutMouseFunc(*Myfn*)
  - 클릭
  - Code 05_07
- glutMotionFunc(*Myfn*);
  - 클릭 된 상태에서 이동
- glutPassiveMotionFunc(*Myfn*);
  - 클릭 안 된 상태에서 이동
- glutEntryFunc() 윈도우창 나갈때/들어올때 이벤트 발생가능

glutPostRedisplay() ⇒ 화면을 다시 그려달라고 요청함
Event Queue에 들어감
(MyDisplay() 같은것 쓰면X ⇒ 성능↓)
ex) 마우스를 움직일때 마우스 따라 선이 Rubber-band됨

메뉴 만들기

- glutCreateMenu(MyMainMenu); 메뉴생성
- glutAddMenuEntry("Draw Sphere", 1);
- glutAddMenuEntry("Exit", 3);) 메뉴항목
- glutAttachMenu(GLUT_RIGHT_BUTTON); 우클릭시 메뉴창 열기
  - Code 05_08

```
void MyMainMenu(int entryID) {
    if (entryID == 1)
        lsSphere = true;
    else if (entryID == 2)
        lsSphere = false;
    else if (entryID == 3)
        exit(0);
    glutPostRedisplay();
}
```

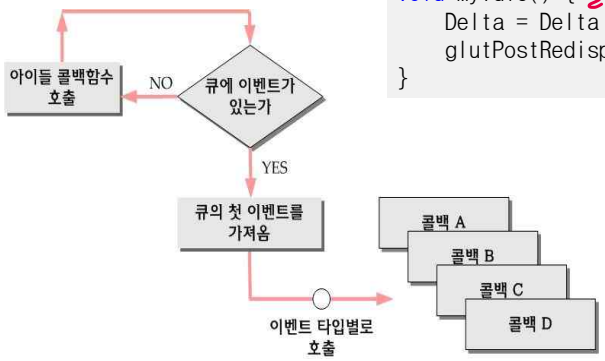
매우 중요

- 계단식 메뉴
  - Code 05_09 "MySubMenuID" 확인

Idle 콜백

- 큐에 이벤트가 없을 때 실행
- 정의되어 있지 않으면 운영체제는 다른 일을 수행
- 드라이버를 통해 주기적으로 이벤트 검사
- Code 05_10

```
void MyIdle() {
    Delta = Delta + 0.0001;
    glutPostRedisplay();
}
```



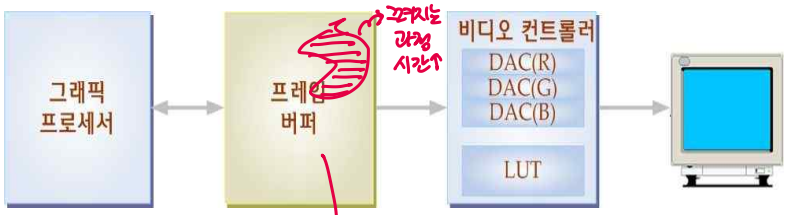
Timer 콜백

- glutTimerFunc(500, MyTimer, 1);
  - 시간을 지정
  - 다시 호출해야 함
- Code 05_11,12

```
void MyTimer(int Value) {
    Delta = Delta + 0.01;
    glutPostRedisplay();
    glutTimerFunc(500, MyTimer, 1);
}
```

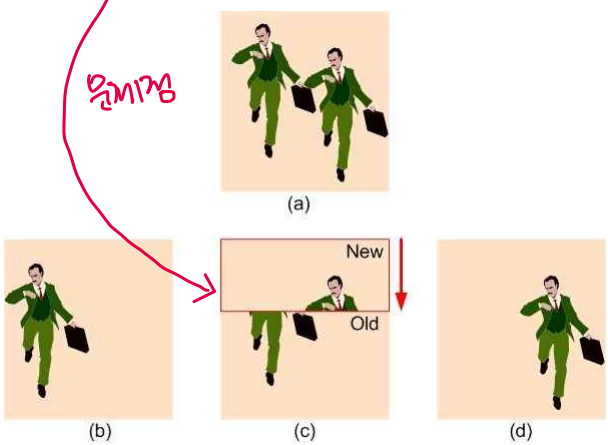
애니메이션과 더블 버퍼링

- 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

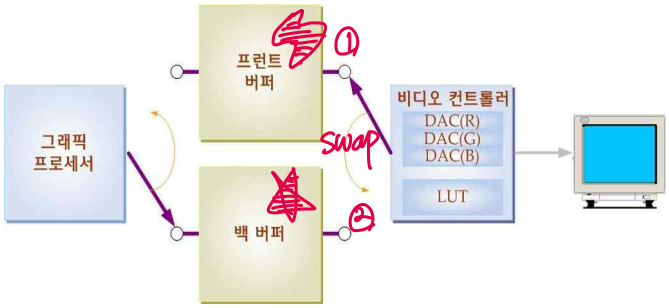
애니메이션 문제



[그림 5-60] 프레임 버퍼 쓰기와 읽기

더블 버퍼링 위 문제점 해결

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



[그림 5-61] 더블 버퍼링

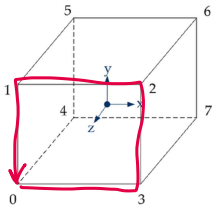
정점 배열-육면체 그리기

- 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)

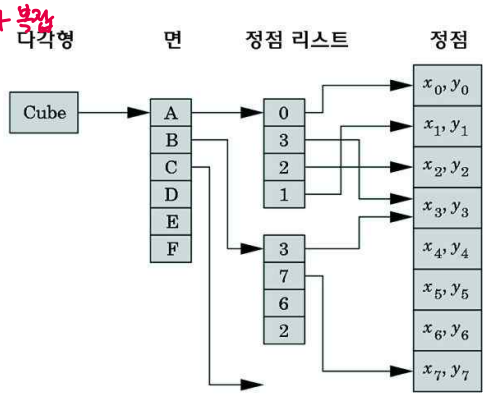
```
glBegin(GL_POLYGON);
    glColor3fv(MyColors[0]); glVertex3fv(MyVertices[0]);
    glColor3fv(MyColors[3]); glVertex3fv(MyVertices[3]);
    glColor3fv(MyColors[2]); glVertex3fv(MyVertices[2]);
    glColor3fv(MyColors[1]); glVertex3fv(MyVertices[1]);
glEnd();
```

반시계방향



- 육면체의 계층구조적 표현

생각보다 복잡



[그림 5-64] 물체의 계층적 자료구조

정점 배열

Color와 Vertex를 미리 배열에 넣어둠

- `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]);
    …
}
```

정점 배열을 사용하겠다

시작

사이즈

사각형

Integer도 가능

각각의 시작 주소

- Code 05_13

GL의 실행모드

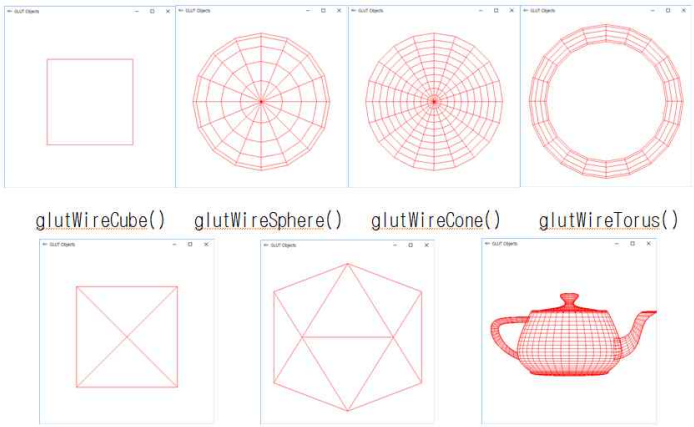
- 직접 모드(Immediate Mode)
  - 화면 렌더링과 동시에 물체 정보를 모두 파기
  - 다시 그리려면 전체 코드를 다시 실행
- 보류 모드(Retained Mode)
  - 이미 정의된 물체를 컴파일 된 형태로 재사용
- 디스플레이 리스트
  - 기본요소(Primitives), 상태변수(State Variable), 영상(Image)
  - 이동, 회전, 조명 작업과 관련된 모든 명령
  - 반복적으로 실행되어야 할 요소를 디스플레이 리스트 내부에 포함
  - 프로그램 속도 향상에 필수적임
- Code 05_14

3장 연습문제

- [Lab 2-1] 사각형과 원을 그리는 과정을 프로그램 2.1과 같이 유사 코드로 구현해 보라. 선분과 유사한 방법으로 구현할 수 있을 것이다.
- [Lab 2-2] 선이나 사각형, 원 등 다양한 그래픽 객체를 그리는 편집기를 구현하려면 현재 모드가 어떤 객체를 그리는 모드인지가 결정되어 있어야 한다. 이를 결정하는 방법에는 어떤 것이 있는지를 MS 사의 파워포인트 프로그램을 통해 알아보라.
- [Lab 2-3] 다각형을 그리기 위한 인터페이스를 설계해 보라. 다각형을 위해서는 다수의 좌표가 입력될 수 있어야 하며, 좌표의 입력이 끝난 것을 알릴 수 있어야 할 것이다.
- [Lab 2-4] 화면에 그려진 그래픽 객체를 이동하거나 회전 또는 신축(크기변환)하기 위한 인터페이스를 설계해 보라.

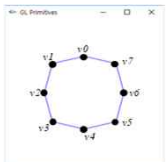
실습3.2 : GLUT 객체 그리기

- 1초마다 다음과 같이 GLUT 3D 객체를 순서대로 출력하는 프로그램을 구현하라.

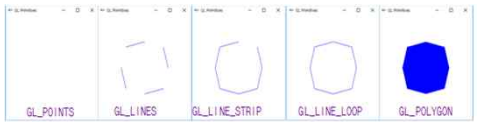


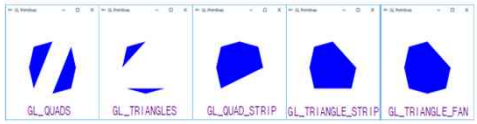
실습3.3 : OpenGL의 그래픽 요소 화면 출력

- 다양한 그래픽 요소들을 다음과 같이 8개의 정점들을 이용해 화면에 출력해 본다.
  - 타이머 콜백을 이용해 1초마다 그래픽 요소가 변경하여 출력되도록 한다.
  - 다음 그림과 같이 순서대로 출력되도록 한다. 단, GL_POINT부터 GL_TRIANGLES 까지는 정점을 v0부터 순서대로 출력하면 되고, 나머지는 정점 입력 순서를 조정해야 한다.



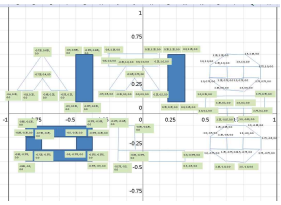

```
float Vtx[8][3] = { {0,0.5,0}, {-0.4,0.4,0},
{-0.5,0,0},    {-0.4,-0.4,0},    {0,-0.5,0},
{0.4,-0.4,0}, {0.5,0,0}, { 0.4,0.4,0}};
```



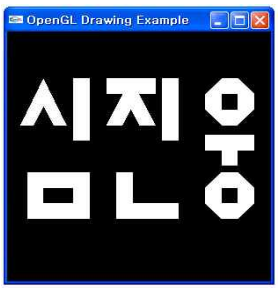


실습3.4 : 자신의 이름을 그리는 함수

- 이름을 도형으로 그려본 후, 각 정점의 좌표를 계산



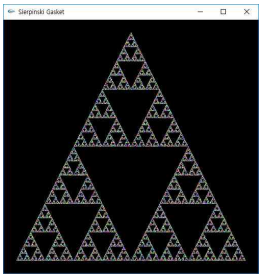
- 계산된 좌표를 이용해 자신의 이름을 출력하는 함수를 구현한다. 이때 다각형(GL_POLYGON)으로 그리는 경우 조심해야 할 부분이 있다. 다각형의 형태가 볼록(convex)해야 할 것이다. 오목(concave)한 부분이 있는 다각형은 왜 원하는 대로 출력되지 않는지 생각해 보라. 실행 결과의 예는 다음과 같다.



73

실습3.5 : 재미있는 그림을 그리는 프로그램

- Sierpinski Gasket
  ① 삼각형 내부에서 무작위로 한 개의 시작점을 잡는다.
  ② 무작위로 세 개의 정점 중 한 개를 선택한다.
  ③ 시작점과 무작위로 선택된 정점의 중간점을 잡는다.
  ④ 그 위치에 작은 원 같은 표지를 함으로써 새로운 점을 표시.
  ⑤ 이 새로운 점을 시작점으로 대치
  ⑥ 단계 2로 돌아간다. (이 과정을 주어진 횟수만큼 반복)



74

실습3.6 : GLUT 객체 그리기 Version 2

- 실습 3.2에 다양한 윈도우 이벤트 처리 기능 추가
- 키보드 이벤트
  - 'a'키를 입력하면 자동 갱신 모드를 토글(toggle)한다. 즉, 이전에 이 모드가 활성화 되었다면 비활성화시키고, 아니면 활성화시킨다. 자동갱신모드가 비활성화되면 더 이상 타이머 콜백이 자동으로 호출되지 않는다.
  - '0'~'6'사이의 키를 입력하면 그 번호에 해당하는 객체를 출력하도록 한다. 이 경우 자동 갱신 모드는 해제된다.
  - 'a'키를 입력하면 자동 갱신 모드를 토글(toggle)한다. 즉, 이전에 ON이라면 OFF되고, OFF면 ON으로 변경된다.
  - 'i'키를 입력하면 객체의 회전이 초기화된다. 즉 모든 객체가 그림 3.6과 같이 출력된다.
  - 'q' 키를 입력하면 프로그램을 종료한다.

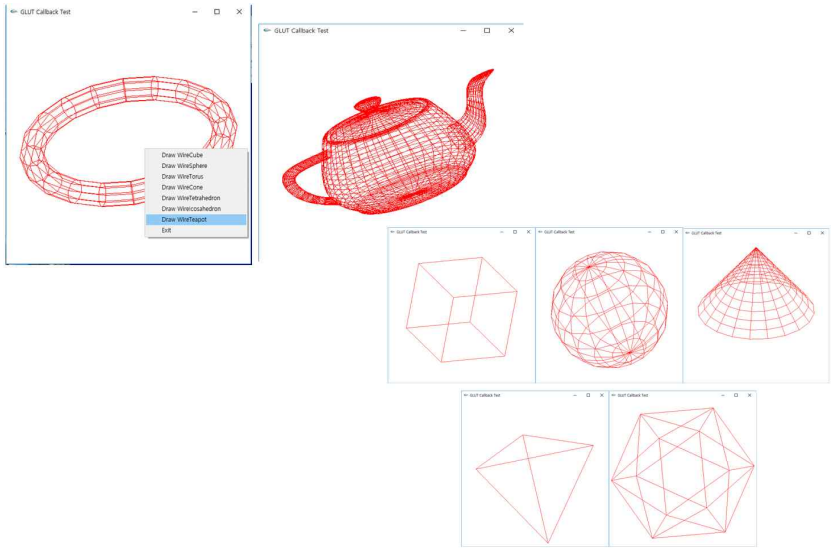
75

실습3.6 :

- Reshape 이벤트
  - 윈도우 크기를 변경하더라도 출력되는 객체의 가로세로 비율은 동일하게 유지되도록 처리한다.
- 마우스 이벤트
  - 마우스의 왼쪽 버튼을 눌러서 드래그하면 그려지는 물체가 회전한다. 마우스가 좌우 또는 상하로 움직이는 정도에 따라 물체는 y축 또는 x축을 따라 회전한다. 이를 위해 마우스 클릭과 마우스 모션의 두 콜백함수를 사용해야 한다.
- 메뉴 처리
  - 팝업 메뉴를 만든다. 메뉴에는 각 GLUT 객체를 출력한다는 메시지가 나타난다. 각 메뉴를 누를 때 마다 해당 기능이 실행된다.

76

실습3.6 :



77

감사합니다!

78