

4장. 오픈지엘 API

학습목표

- 표준화의 개념과 필요성을 이해한다.
- **API**의 정의와 필요성을 이해한다.
- 고수준 **API**의 장면묘사 방식을 이해한다.
- 오픈지엘의 설계원리에 반영된 개념을 이해한다.
- 파이프라인 개념, 상태변수 개념을 이해한다.
- 오픈지엘 프로그램 작성을 위한 유틸리티 프로그램 설치방법을 이해한다.

정의

- “주어진 여건에서 최적의 질서를 유지하기 위해, 현존하거나 잠재하는 문제들에 대해, 공유성과 재사용성을 높이기 위한 기반을 확립하는 행위“

ISO/IEC JTC1/SC24, Working Group

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

- 👤 **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.

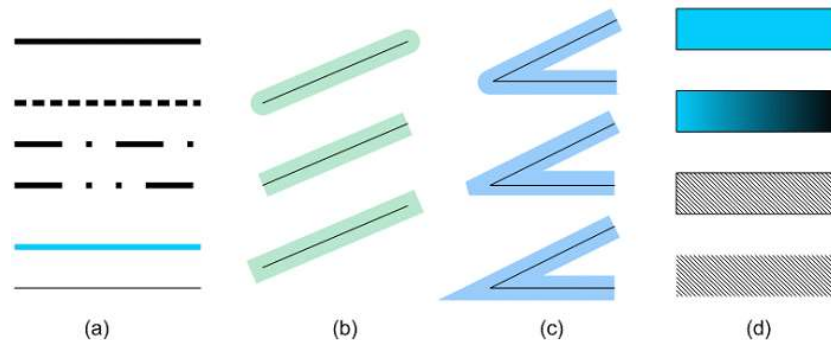
기본요소(Primitives)

- 점(Point), 선(Line), 채움 영역(Fill Area), 꺾은 선(Poly Line), 표시 꺾은선(Poly Marker), 문자(Character)



기본요소 외양

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





GKS(지케이에스, Graphical Kernel System)

- 유럽에 의해 주도. 2차원 위주. 이후 GKS-3D로 발전
- 파일출력
 - 기본요소 수준에서 서술한 가상 레벨(Virtual Level) 저장
 - 기본요소의 위치 좌표, 속성, 가시성, 변환 정보를 저장



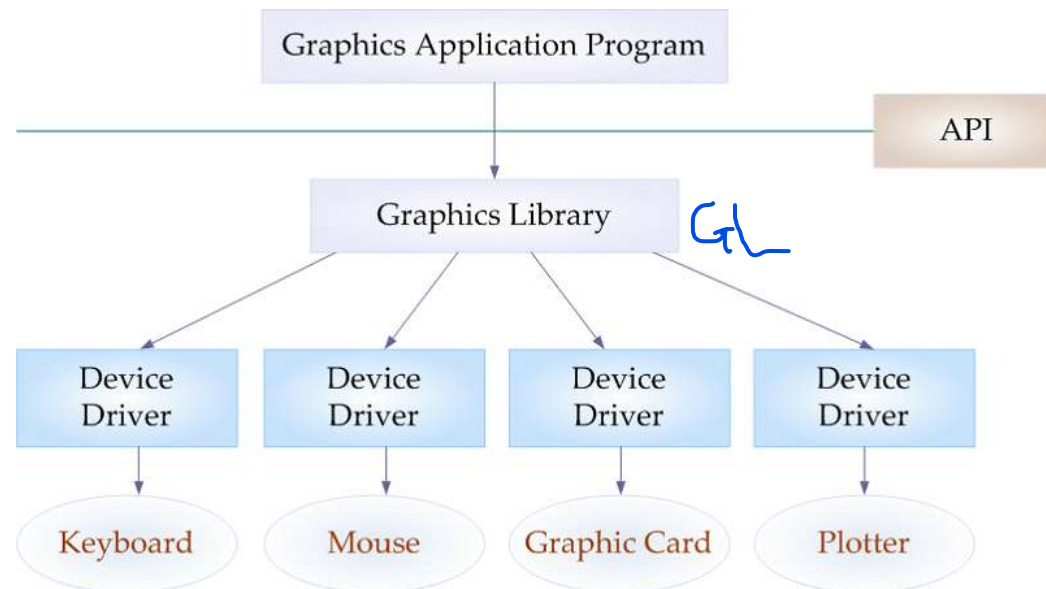
PHIGS(Programmer's Hierarchical Interactive Graphics System)

- 미국에 의해 주도. 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
- 파일출력
 - 기본요소에 관한 정보 + 응용 프로그램 레벨에서 기본요소 사이의 관계
 - CSG의 불리언 연산, 로봇 팔의 객체 계층구조 저장

그래픽 API

응용 프로그램 인터페이스

- 라이브러리
- PHIGS, GKS = 추상적 수준의 API
- API



고수준 그래픽 API

장면묘사 언어(Scene Description Language)

```
Camera {  
  center {0.0 0.0 5.0}  
  direction {0.0 0.0 -1.0}  
}  
Lights {  
  numLights 1  
  DirectionalLight {  
    direction {0.5 0.5 0.5}  
    color {1.0 1.0 1.0}  
  }  
}  
Background {  
  color {1.0 1.0 1.0}  
}  
Group {  
  numObject 2  
  Material {0.0, 0.0, 1.0}  
  Sphere {2.0}  
  Transform {  
    Translate {1.0, 0.0, 0.0}  
    Scale {0.3, 0.3, 0.3}  
    Material {1.0, 0.0, 0.0}  
    Sphere {2.0}  
  }  
}
```

카메라 중심을 (0.0 0.0 5.0)에 위치시키되
카메라가 (0.0 0.0 -1.0)을 바라보게

광원의 숫자는 1개
방향성 광원으로 하여
(0.5, 0.5, 0.5) 방향으로 빛을 비추되
백색 빛을 발하는 광원

배경색은 백색

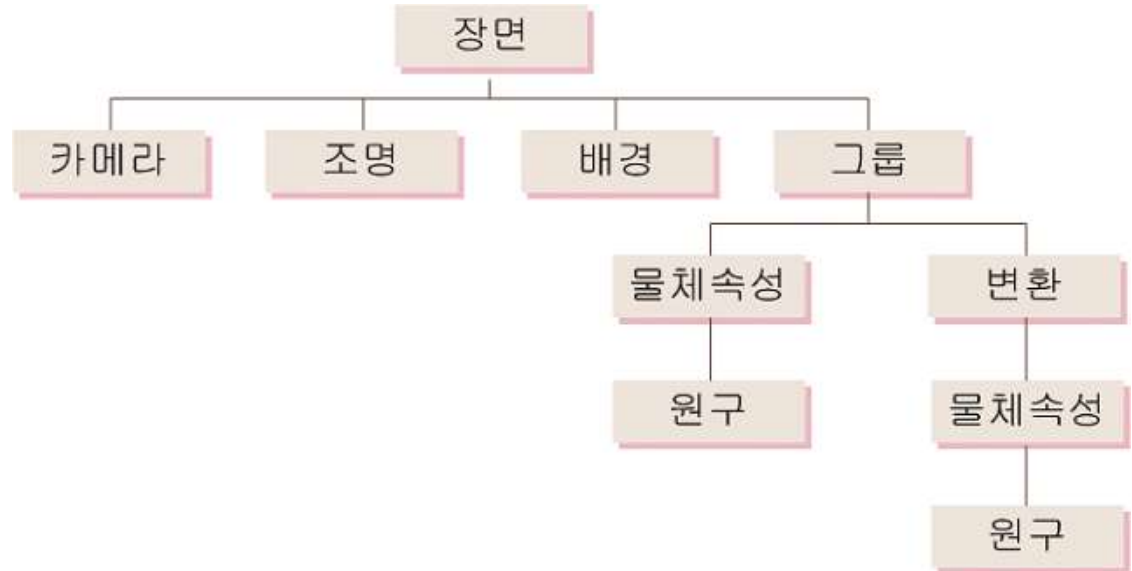
물체 2개로 이뤄진 그룹
첫 물체를 청색으로 하여
반지름 2인 원구를 그림

x축 방향으로 1.0만큼 이동하여
크기를 x, y, z 방향으로 0.3배로 줄여서
둘째 물체를 적색으로 하여
반지름 2인 원구를 그림

고수준 그래픽 API

장면 그래프(Scene Graph)

- 그룹 노드는 네스트 구조
- 트리/그래프 순차 탐색으로 장면을 그려냄



고수준 그래픽 API

- 오픈 인벤터(open Inventor)
- VRML
- Java3D

VRML

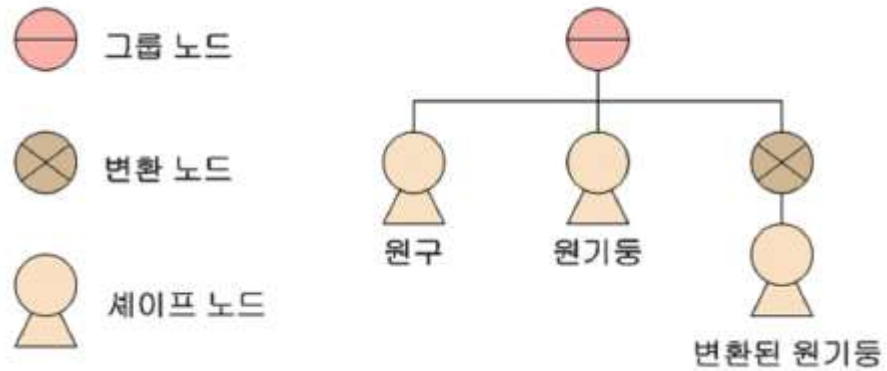
```
#VRML V2.0 utf8
```

```
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 ]
```

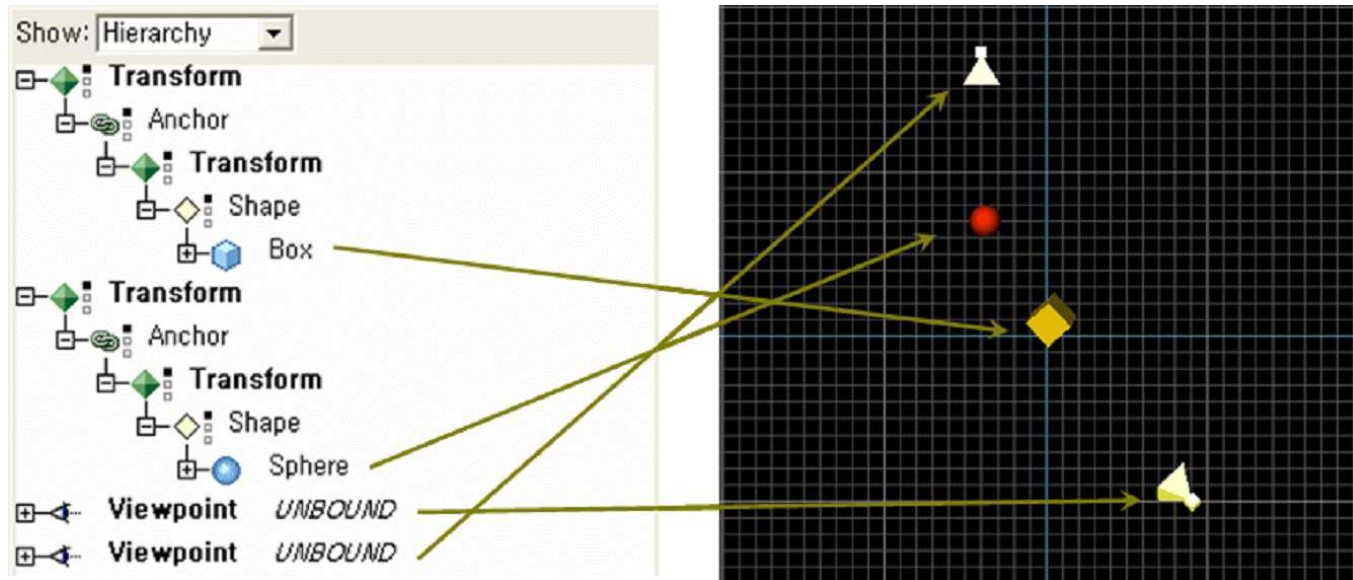


VRML

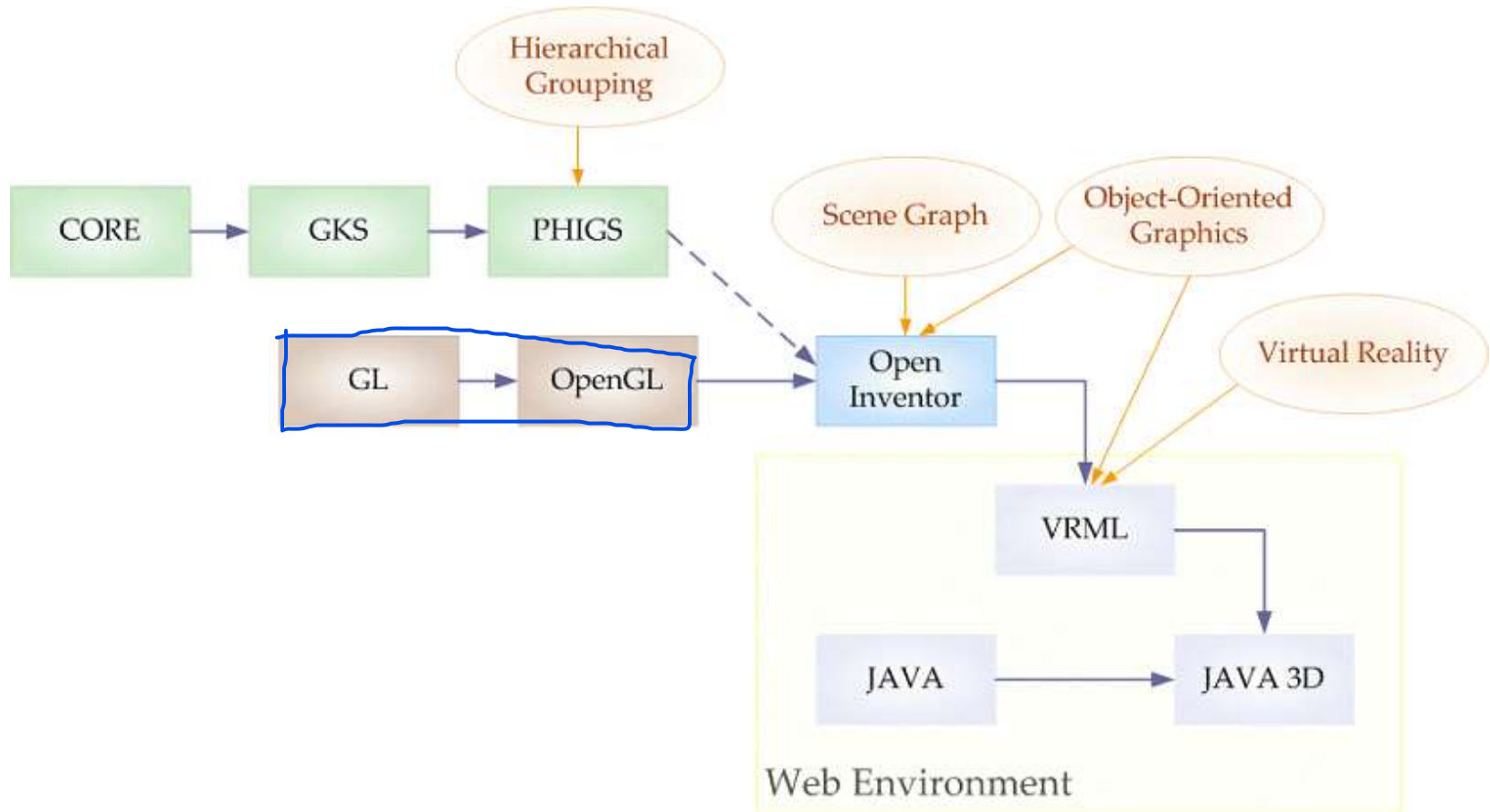
장면 그래프



VRML 저작도구: Cosmo Player



그래픽 API 발전과정



👤 저수준 API

- 장면을 묘사하는 것이 아니라 구체적 프러시저(명령)를 호출
- cf. DirectX from Microsoft: 호환성 결여
- 하드웨어와 거의 직접 연관 (하드웨어 성능을 최대한 발휘)
- Inventor, VRML, Java3D 등 고수준 API의 기반
- 드라이버 소프트웨어에 비해서는 상대적으로 고수준 함수



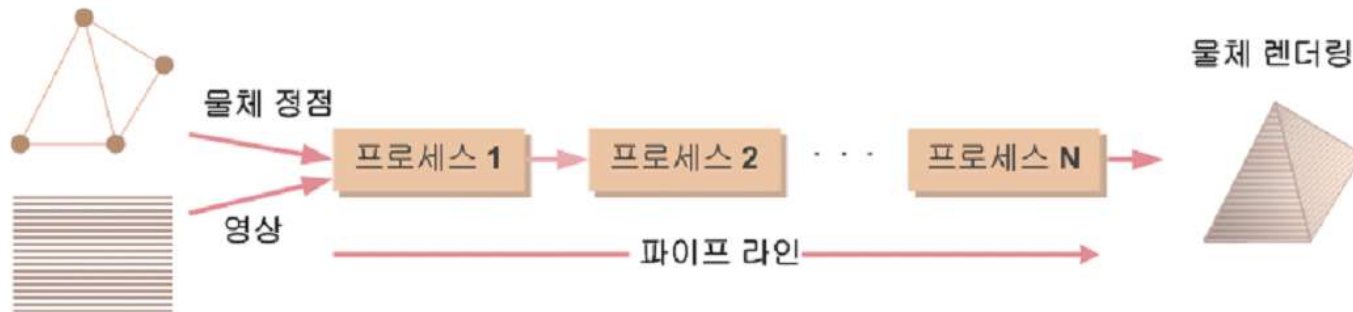
오픈지엘 설계원리

- 👤 **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
 - 클라이언트-서버 모델(**Client-Server Model**)지원.
 - 성능이 낮은 클라이언트 컴퓨터가 고성능 서버를 이용.



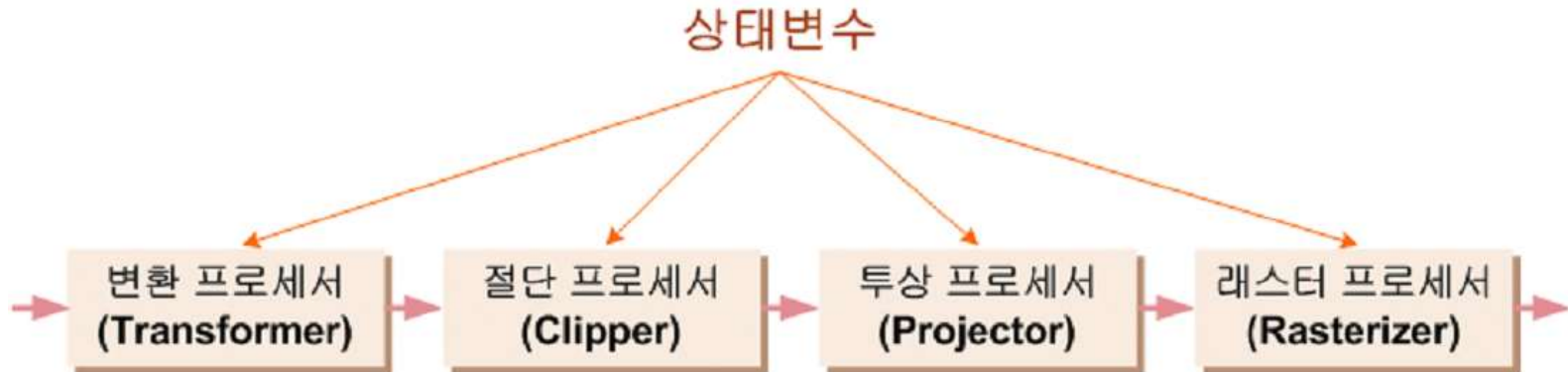
GPU 설계원리

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



상태변수

- 👤 지엘의 역할 = 상태변수 설정
- 👤 파이프라인은 상태변수를 참조해서 자동으로 실행됨



속성할당 방법

👤 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

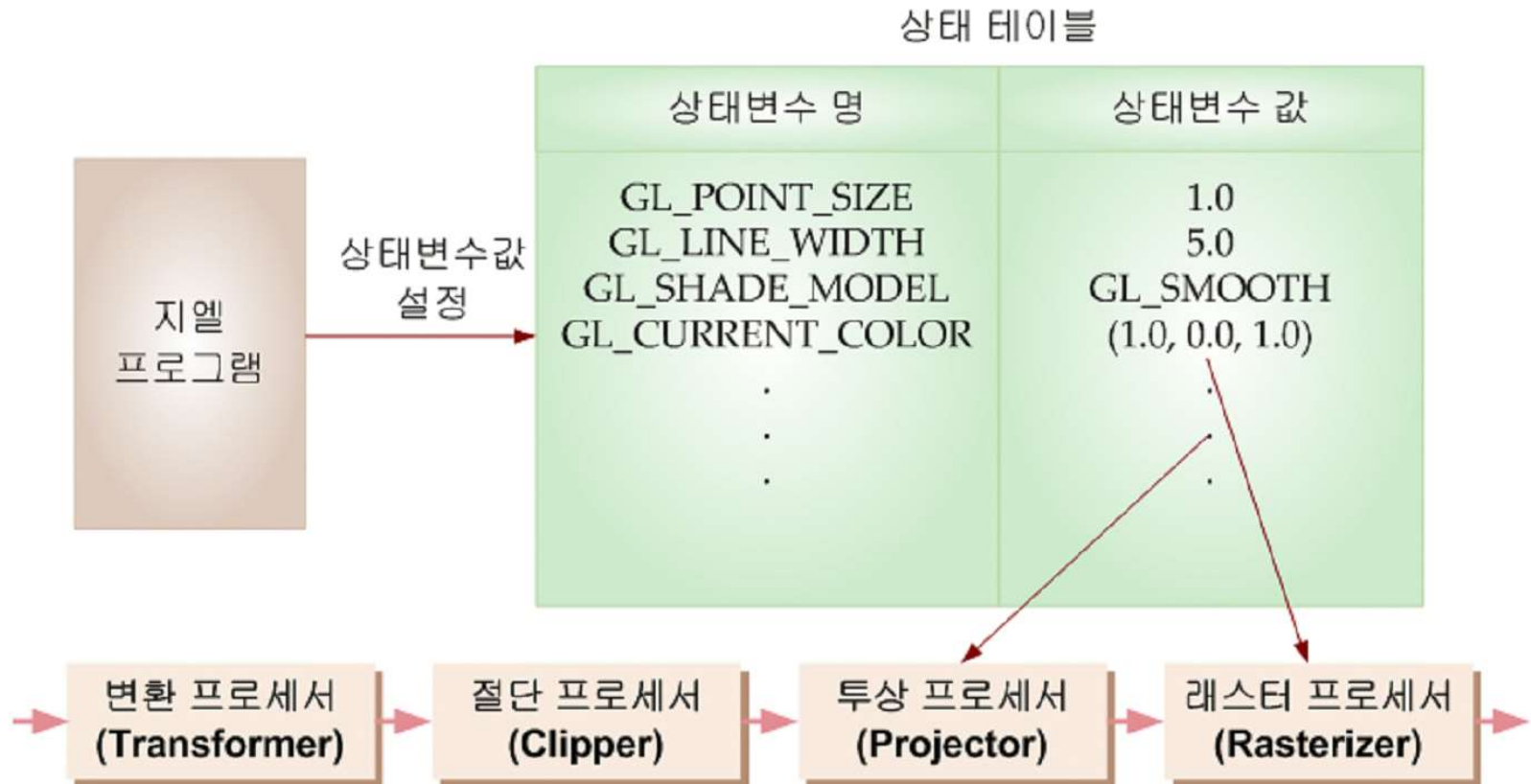
- `setLineStyle(2);`
- `setLineWidth(4);`
- `setLineColor(255, 0, 0);`
- `drawLine((1, 0), (3, 0));`
- `drawLine((3, 0), (2, 5));`
- `drawLine((2, 5), (1, 0));`



👤 “현 상태” 라는 개념

- Current State
- 한번 파란색이 선택되면, 이후에 그리는 모든 도형이 파란색

지엘 프로그램, 상태변수, 파이프라인



상태변수 = 클래스 멤버 변수

상태변수 예

상태변수 설정

- `glColor3f(1.0, 1.0, 1.0);`
- `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

- `glPointSize(0.5);`
- `glLineWidth(5);`
- `glShadeModel(GL_SMOOTH);`

상태변수 검색

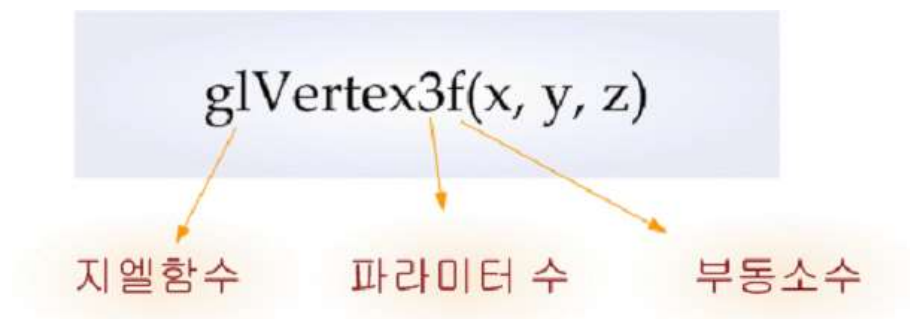
- `float MyColor[3];` 임의 배열
- `glGetFloatv(GL_CURRENT_COLOR, MyColor);` 검색 함수

기능관련 상태변수

- `glEnable(GL_LIGHTING);` 조명 모드를 활성화
- `glDisable(GL_LIGHTING);` 조명 모드를 비활성화

지엘 명령어 구조

👤 정점 정의



접미사	데이터 타입	C/C++ 타입명	지엘 타입명
f	32-bit floating point	float	GLfloat
d	64-bit floating point	double	GLdouble
b	8-bit integer	signed char	GLbyte
ub	8-bit unsigned integer	unsigned char	GLubyte, GLboolean
i	32-bit integer	int or long	GLint
ui	32-bit unsigned integer	unsigned long	GLuint, GLenum, GLbitfield
s	16-bit integer	short	GLshort

👤 float: C/C++ 타입, GLfloat: GL 타입

👤 벡터 타입



👤 지엘은 API

- 명령어가 아니라 함수명. 그러나 혼용

👤 지엘은 비 객체지향적

- 처리속도를 향상
- 함수 오버로딩이 불가능
- 3차원 정점이라면 `glVertex3f()`, 2차원 정점이라면 `glVertex2f()`

지엘 프로그램 구성요소

지엘 라이브러리(GL: OpenGL Core Library) 순수 그래픽

- 렌더링 기능을 제공하는 함수 라이브러리

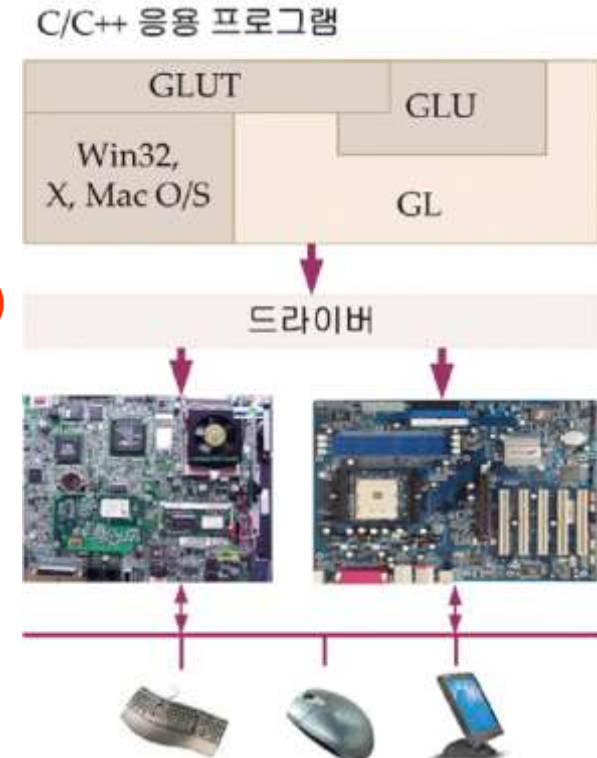
지엘 유틸리티 라이브러리(GLU: OpenGL Utility Library) 수학 함수

- 50여개의 함수. GL 라이브러리의 도우미
- 다각형 분할, 투상, 2차원 곡면, 너브스등 고급기능을 제공하는 함수
- GL 함수로 작성

지엘 유틸리티 툴 킷(GLUT: OpenGL Utility Toolkit)

- 사용자 입력을 받아들이거나 화면 윈도우를 제어하기 위한 함수
- 윈도우 운영체제 기능과의 인터페이스

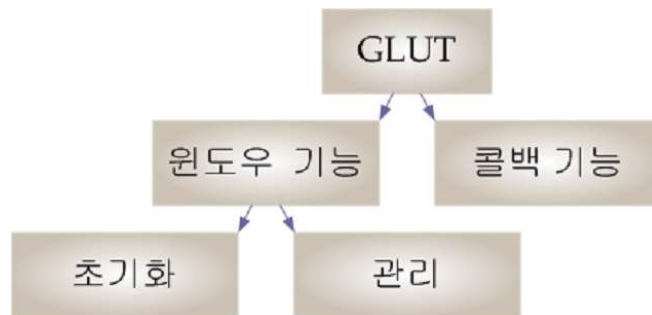
플랫폼을 신경쓰지 않는다.



GLUT

- 👤 GL은 운영체제, 하드웨어에 무관한 그래픽 명령어 집합
 - 윈도우 창을 생성하거나, 키보드 마우스 입력을 받는 기능이 없음
- 👤 윈도우 기능: 프로그램 실행에 필요한 창(Window)을 관리
- 👤 콜백 기능: 프로그램 실행 중 발생하는 사용자 입력을 처리
- 👤 창 생성과 입력 관리는 별도의 도구를 사용해야 함
 - GLUT: 전통적(1990년대)방식. 현재 업그레이드 중지됨. 간단함
 - **FreeGLUT**: GLUT와 명령어 호환되며 최신. 64bit지원
 - GLFW: 새로운 방식으로, 더 다양한 기능 사용 가능. 설치 복잡
 - SFML, SDL: opengl 외에도 다양한 기능 포함
- 👤 참고: 최신 opengl 함수를 사용하는 방법
 - GLEW(OpenGL Extension Wrangler - 말몰이꾼 - Library)
 - gl3w
 - GLAD (GLFW 설치시 포함됨)

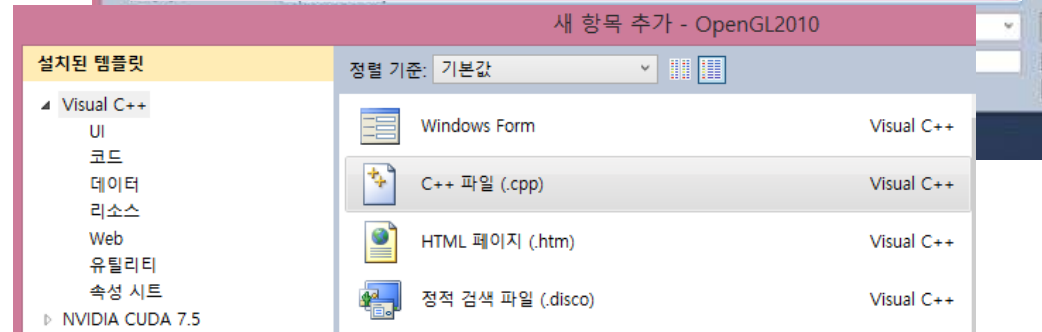
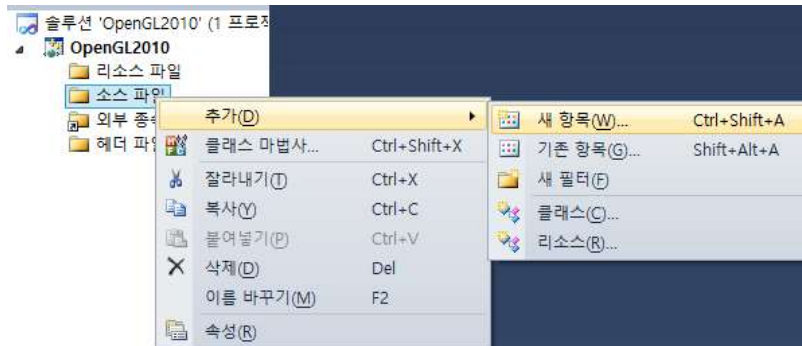
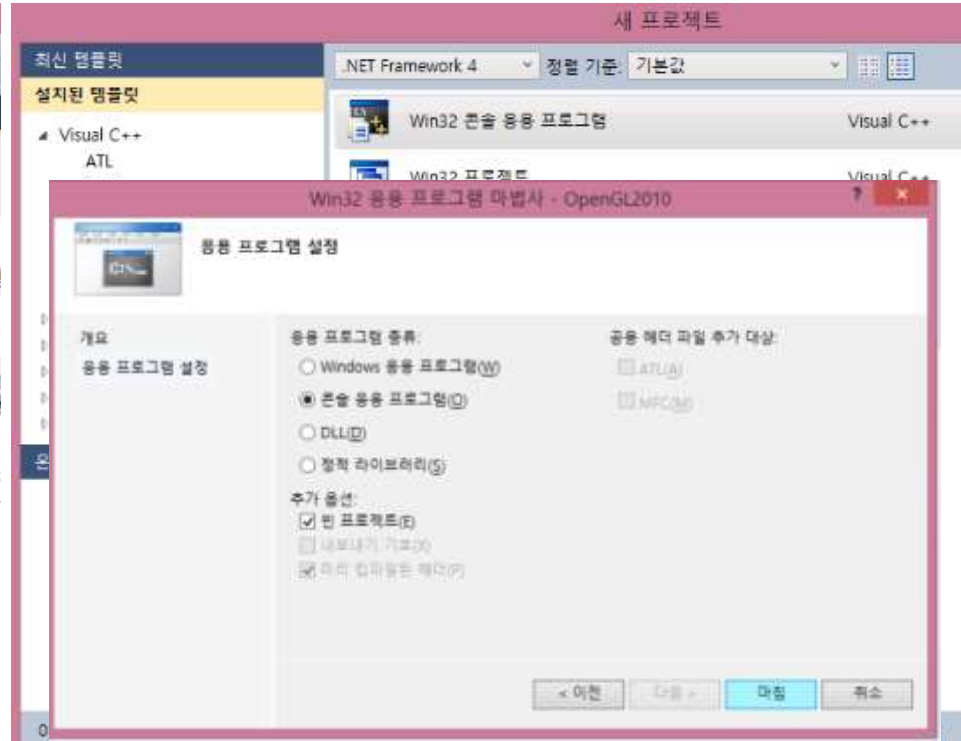
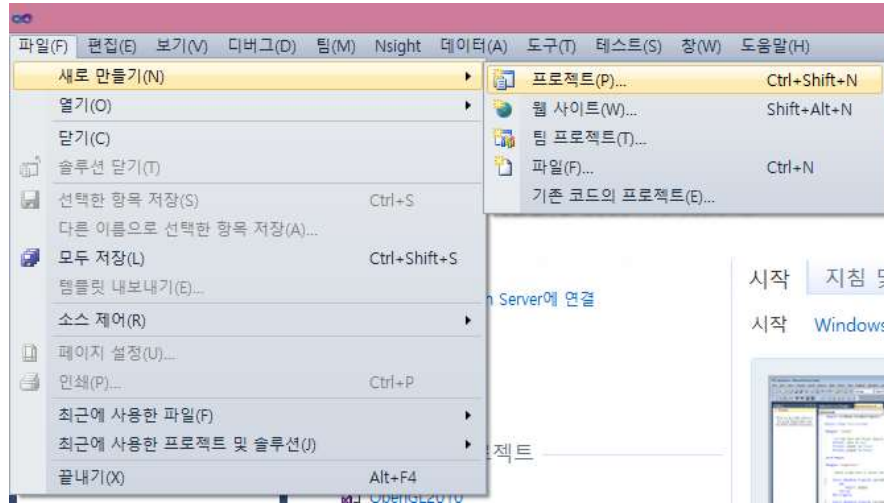
GLUT / FreeGLUT



	함수명	기능 설명
원도우 초기화	<u>glutInit()</u>	원도우 운영체제와 세션 연결
	<u>glutInitWindowPosition()</u>	원도우 위치 설정
	<u>glutInitWindowSize()</u>	원도우 크기 설정
	<u>glutInitDisplayMode()</u>	디스플레이 모드 설정
원도우 관리	<u>glutSetWindowTitle()</u>	원도우 타이틀 설정
	<u>glutCreateWindow()</u>	새로운 원도우 생성
	<u>glutReshapeWindow()</u>	크기 변경에 따른 원도우 조정
	<u>glutPostRedisplay()</u>	현 원도우가 재생되어야 함을 표시
	<u>glutSwapBuffers()</u>	현 프레임 버퍼 변경

프로젝트 생성

VS 콘솔응용프로그램, 빈 프로젝트 생성, cpp 파일 추가



4-7 사각형 그리기, 컴파일 시도

👤 4-7 사각형 그리기

```
#include <gl/glut.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(int a, char **pp) { // 강의교재와 freeglut가 다른 부분  
    glutInit(&a, pp); // 이 부분만 다르고 나머지 동일함  
    glutCreateWindow("OpenGL Drawing Example");  
    glutDisplayFunc(MyDisplay);  
    glutMainLoop();  
    return 0;  
}
```

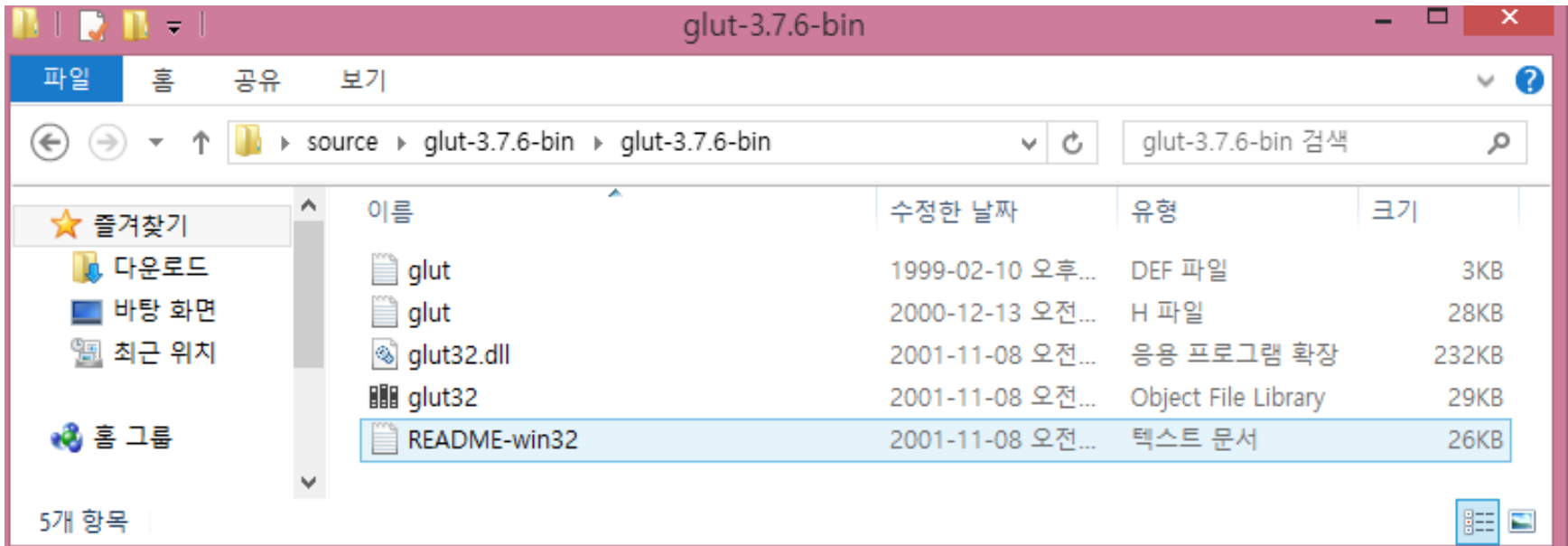
[참고 – 권장하지 않음] GLUT 설치 (2010, 13, 15, 17)

👤 glut3.7.6 압축파일

- glut.h : include 파일 (컴파일시 필요)
- glut32.lib : 라이브러리 파일 (링크시 필요)
- glut32.dll : 동적연결 라이브러리파일 (실행시 필요)

👤 Visual studio 버전에 따라 복사위치가 다름

- glut32.dll은 현재 작업폴더에 복사하는 것을 추천함



(Free)GLUT 설치 폴더(VS 13, 15, 17, 19)

👤 2017, 2019

- **.h** 파일들: C:\Program Files (x86)\Windows Kits\10\Include\10.0.17763.0\um\gl
- **.lib** 파일: C:\Program Files (x86)\Windows Kits\10\Lib\10.0.17763.0\um\x86
- **.dll** 파일: 현재 VS 작업 폴더에 복사
- VS 버전에 따라 10.0.17763.0 값이 달라지며 가장 큰 숫자를 선택

👤 2013, 2015

- **.h** 파일들 : C:\Program Files (x86)\Windows Kits\8.1\Include\um\gl
- **.lib** 파일 : C:\Program Files (x86)\Windows Kits\8.1\Lib\winv6.3\um\x86
- **.dll** 파일: 현재 VS 작업 폴더에 복사

FreeGLUT 구성

include

- **GL**

- freegult.h, freegult_ext.h, freegult_std.h, gult.h

lib

- freegult.lib

- **x64**

- freegult.lib


bin

- freegult.dll

- **x64**

- freegult.dll

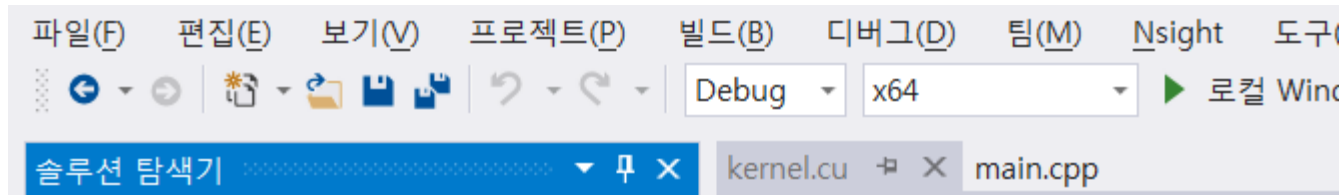
 파일들 복사 위치는 앞페이지 참고

 Visual studio 설정이 x64(64bit 컴파일)인 경우

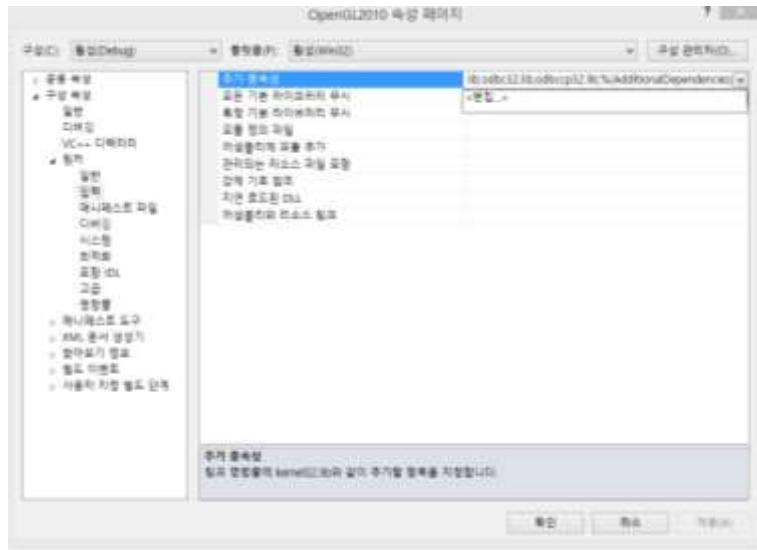
- 압축 파일 내의 x64폴더에서 파일 선택
- 복사할 폴더는 freeglut.lib 만 제외하고 앞페이지와 동일함
- freeglut.lib: C:\Program Files (x86)\Windows Kits\10\Lib\10.0.17763.0\um**x64**

[참고] Visual Studio x64

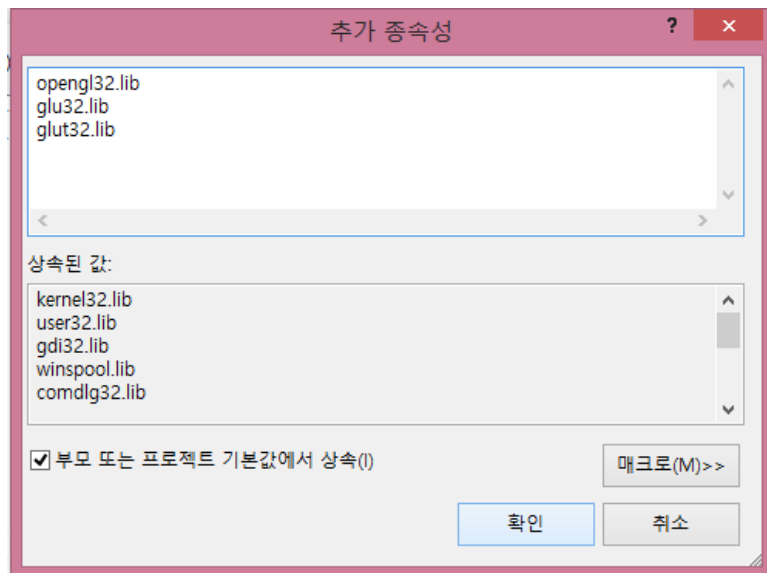
- 기존 32bit 모드는 4GB의 메모리만 사용 가능
 - x86
- 64bit 주소를 사용하는 모드를 x64라고 한다.
 - 포인터 변수의 크기가 4byte가 아닌 8byte가 된다.
 - `char *p; // 8 byte 크기`
 - 4GB 보다 큰 메모리도 할당(malloc, new)할 수 있다.



[참고] 추가 실행 환경 설정 (VS 구버전)



구성 속성 - 링커 - 입력 -
추가 종속성 - ▼버튼 - 편집



opengl32.lib
glu32.lib
glut32.lib

입력 후 확인

[참고] VS 설치폴더가 다른 이유

- 👤 Include
 - \$(WindowsSDK_IncludePath)
- 👤 Lib
 - \$(WindowsSDK_LibraryPath_x86)

