# 가상현실론

2021년 1학기 1주차 2 가상현실과 OpenGL 1

# 목차

- 가상현실
- 수업의 중심 내용
- OpenGL 개괄
- OpenGL Programming
- OpenGL 설치와 설정

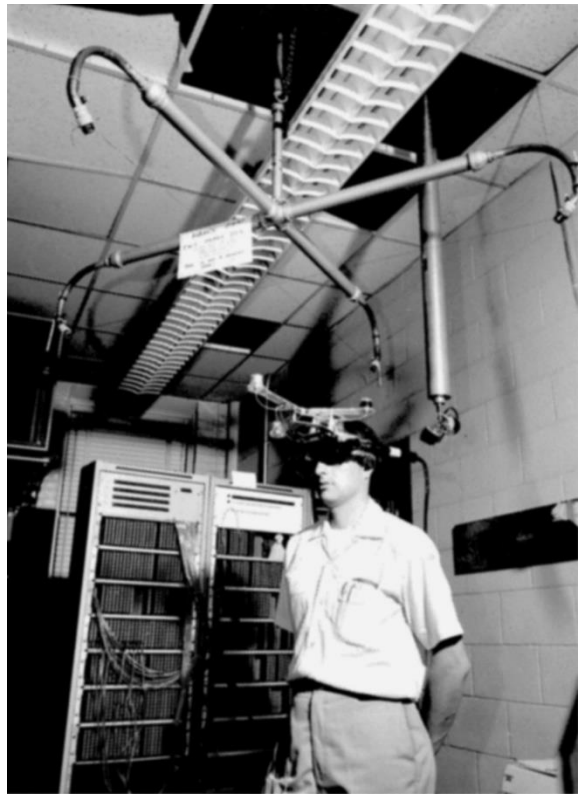
# 가상현실

# 가상현실 (Virtual Reality)

- 인공적인 기술로 만들어낸 실제와 유사하지만 실제가 아닌 어떤 특정한  환경이나 상황 혹은 그 기술 자체.
- 상위 개념: 게임, 시뮬레이터 포함
- 좁은 의미: 사용자가 실제에 근접한 시공간적 체험을 가능케 하는 기술

# 가상 현실의 역사

- 단어로써의 Virtual Reality: Antonin Artaud가 극장을 표현하기 위해 처음 사용
- 최초의 가상현실 시스템: 1968년 Ivan Sutherland가 최초의 HMD (Head Mounted Display) 구현.



# Virtual Reality is “Real” now

- PC mag 선정 2021년 최고의 VR 헤드셋

**Oculus Quest 2**

**HTC Vive Cosmos**

**Sony Playstation VR**

- AR 글래스

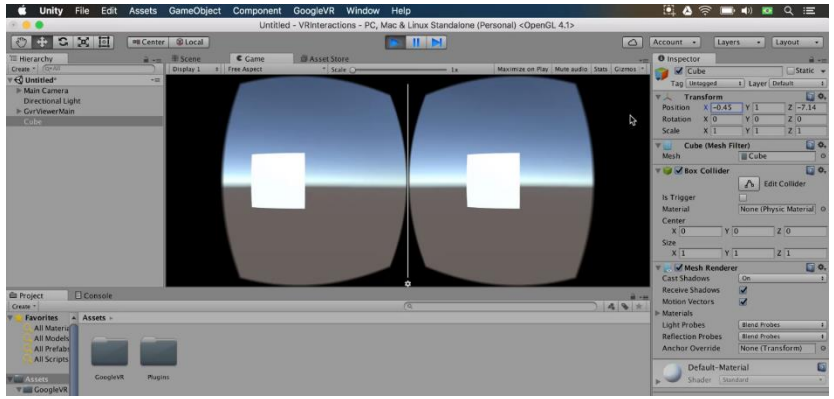
**Microsoft Holo Lens 2**

**Epson Magic Leap**

**Google Glass**

# VR tech. is accesible to developers

- Accessible VR development plamtforms

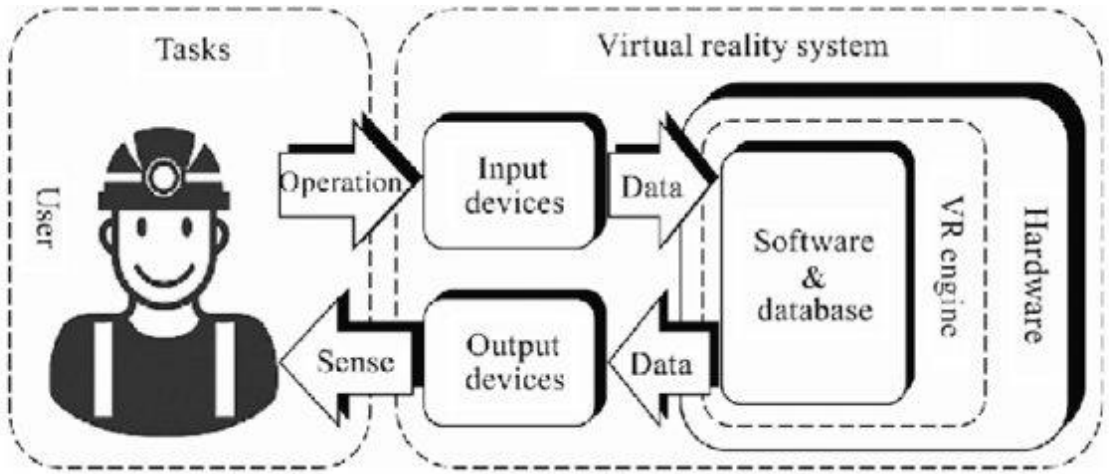



**Unity**

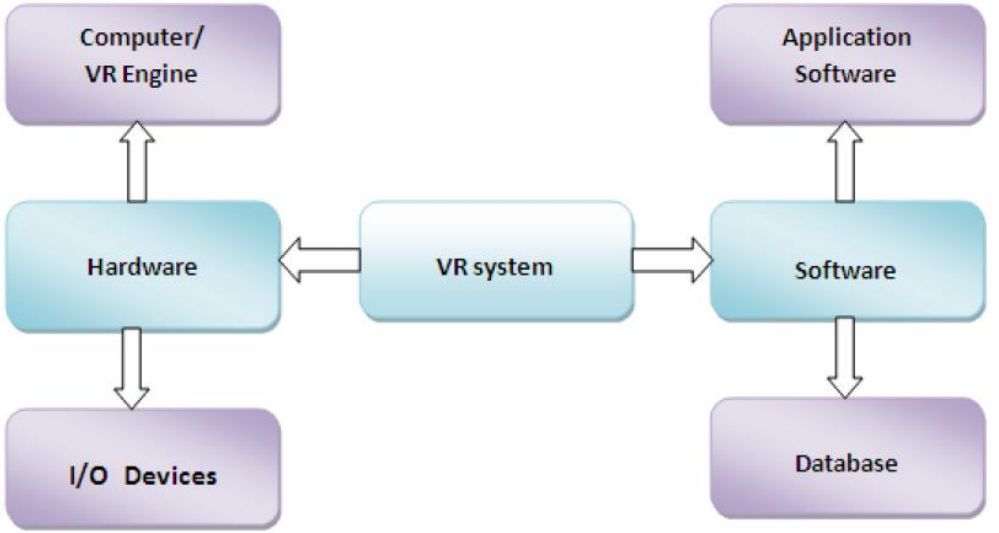

**openVR by Valve**

- Physics Engines are accesible to developers, too
  - ODE
  - MuJoCo
  - Bullet
  - Havoc
  - PhysX

# 가상 현실의 구성요소



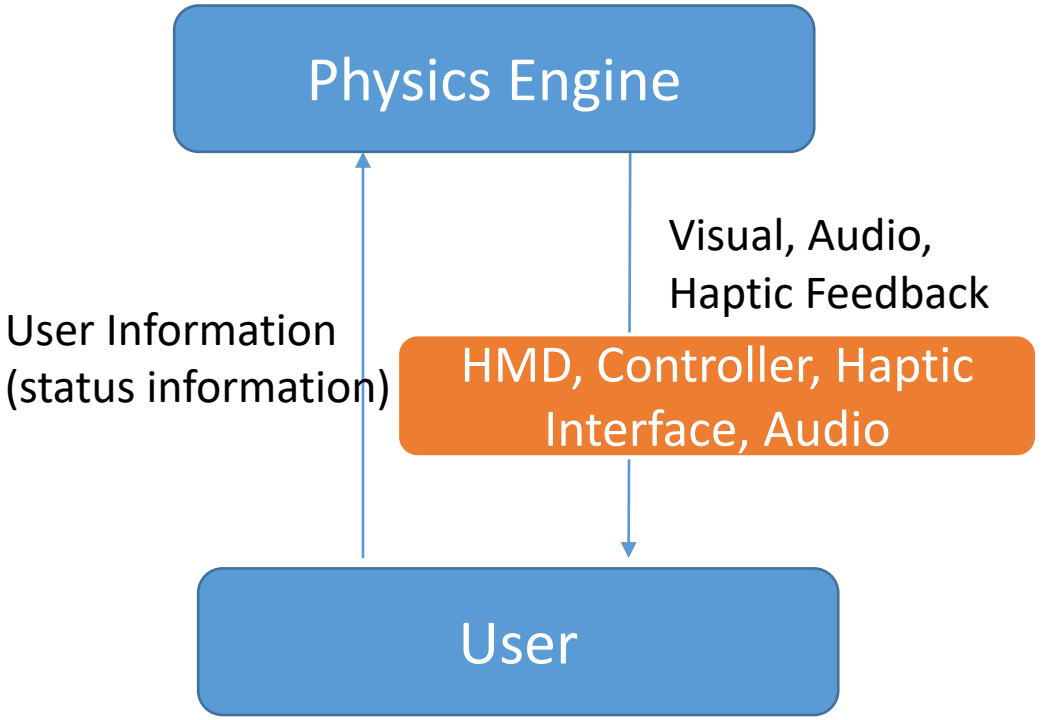

Hui (2017)




Bamodu and Ye (2013)

# 가상현실의 구성요소

Virtual Reality



# 수업의 중심 내용

- 가상 현실 시스템에서 필수적인 물리엔진을 구현하기 위한 이론적 기초
  - OpenGL
  - Collision detection
  - NURBS
  - Kinematics
  - Dynamics Primer
  - Multi-body Dynamics

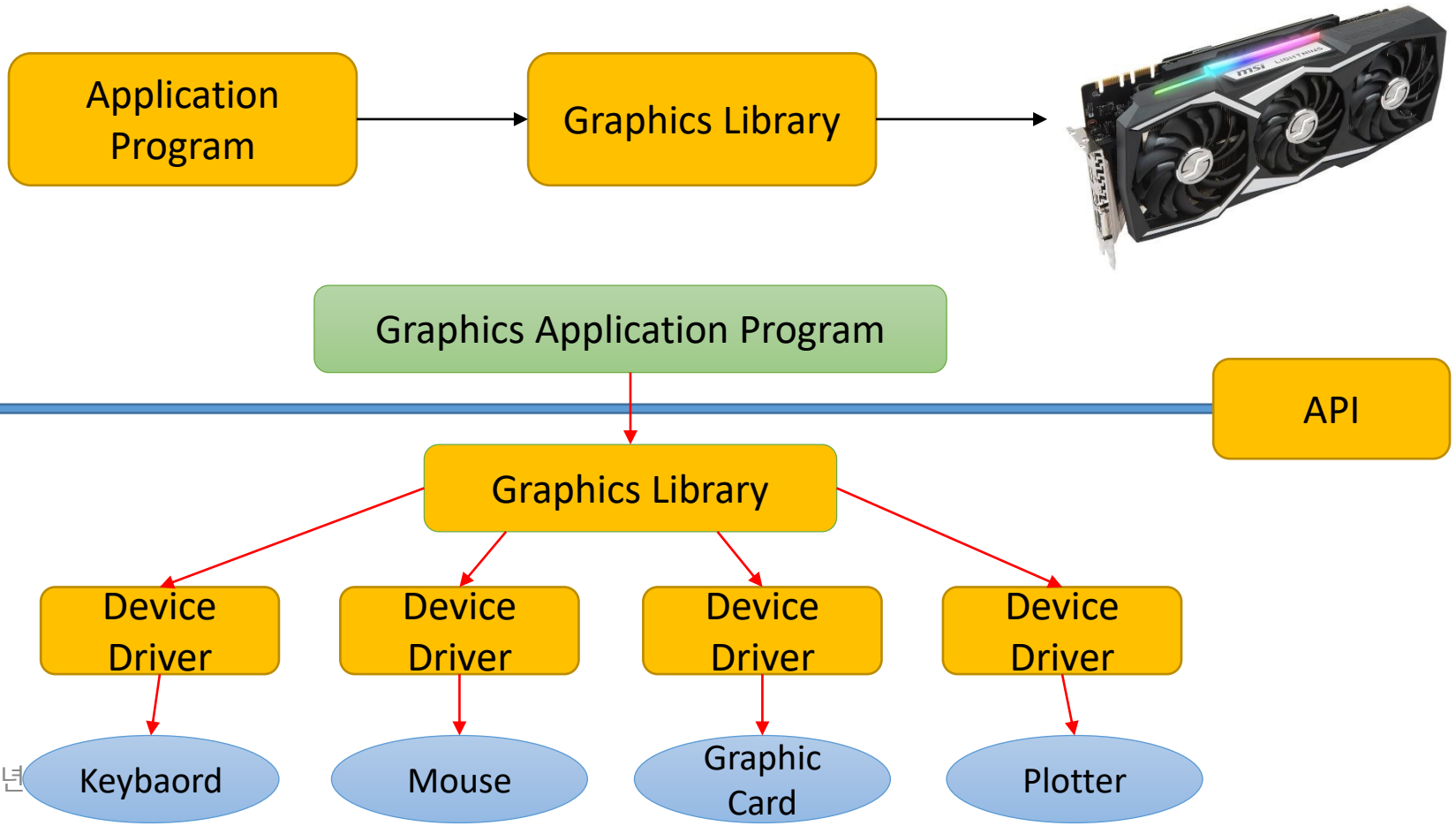
# 굳이 물리엔진 이론을 공부하는 이유?

- 현재 상용 VR application에서 사용하는 물리 엔진은 완전하지 않음
  - 원래 물리 엔진은 완벽하게 현실을 모사하기 위해 만들어 진 것이 아님. (Resource 부족)
  - 변형하는 표면에 대해서 발전할 여지가 있음. Ex) polygon-to-polygon collision detection
  - Hapic rendering에 대한 고려가 되어있지 않음
- 기술적 희소성
  - 물리엔진을 사용하는 사람은 많지만 알고 사용하는 전문가는 희박
  - 앞에서 거론한 물리 엔진 중에서 국내에서 개발한 건 하나도 없음
- 기술의 확장성
  - 유사 분야에의 확장: 동역학 시뮬레이션, 햅틱스

# OpenGL 개괄

# 그래픽 API

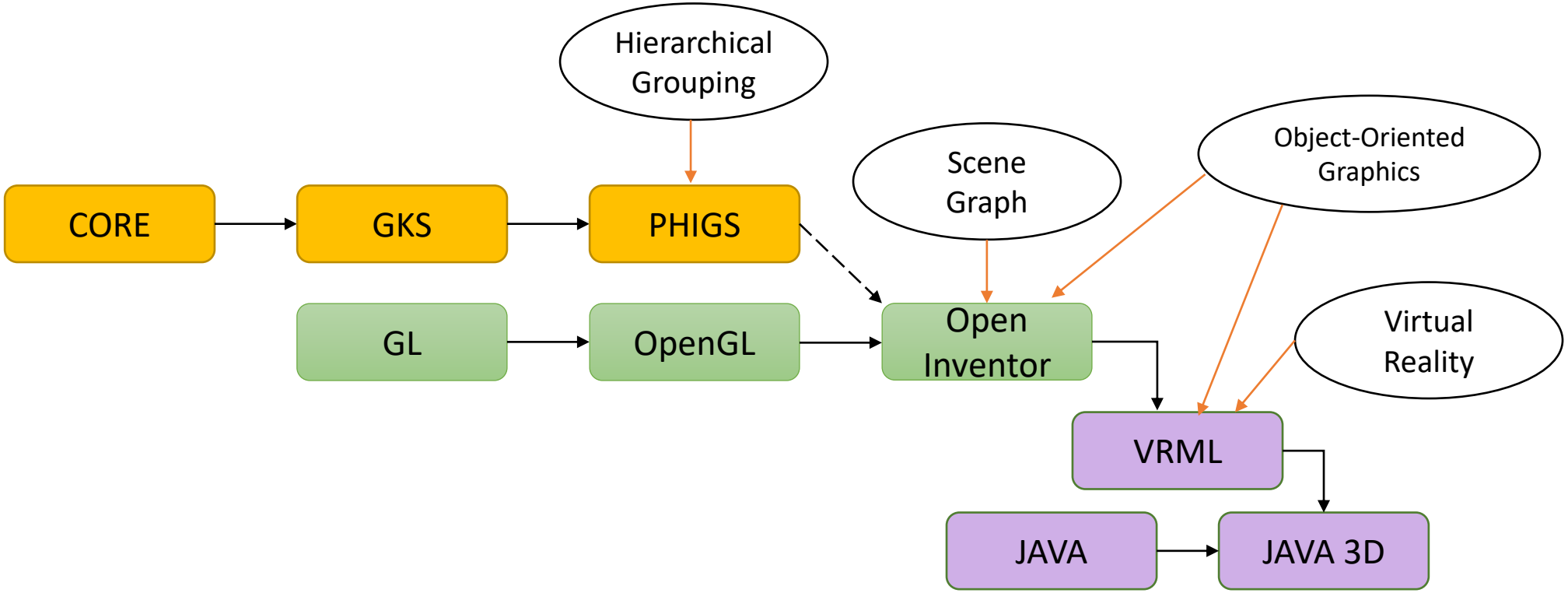
- Graphics API
  - 응용 프로그램 인터페이스 (Application Programming Interface, API): 함수를 이용하여 프로그램을 작성하는 데 직접 활용할 수 있도록 함



# 그래픽 API

- 3차원 그래픽 API
  - High-Level API: 그림을 그리기 위한 실제적인 세부과정을 명세하는 대신, 물체를 정의하고 물체 사이의 관계를 묘사.
    - 예) OpenInventor, VRML, Java3D
  - Low-Level API: 물체를 구성하는 기본 요소의 정의부터 실제 그림을 그리는 세부적인 과정에 이르기까지 일일이 명시.
    - 예) OpenGL, Direct3D

# API 레벨



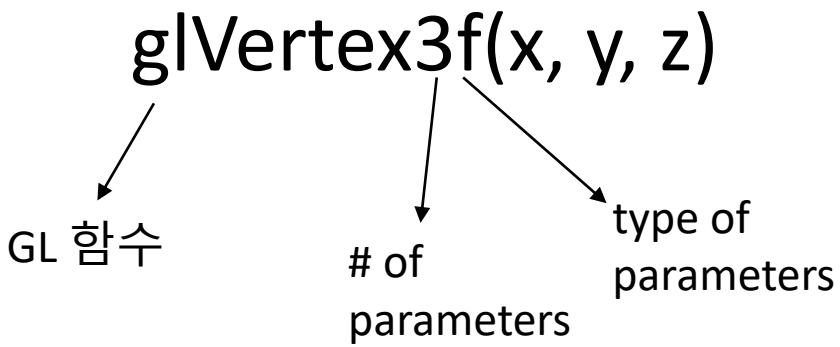

- 표준 API의 발전 과정
  - Silicon Grphics 워크스테이션 API인 GL이 OpenGL로 발전.
  - Open Inventor: OpenGL 기반으로 scene graph와 objected-oriented graphics 도입
  - VRML: Open Inventor 기반으로 가상 현실 개념 도입.

# GL의 설계원리

- C 나 어셈블리 언어로 작성된 200여개의 함수로 구성.
- C/C++, Fortran, Java도 GL 함수 호출 가능
- 범용성: 거의 대부분의 하드웨어에서 실행 가능
- 효율성: 하드웨어가 지원하지 않는 기능을 비활성화 가능
- 독립성: 기능 간의 독립성이 최대한 보장
- 완전성: 하드웨어의 최대성능을 사용할 수 있도록 공통 기능 외에도 특정 하드웨어에 대해 확장 형태 명령어 제공
- 상호작업성: 워크스테이션에 있는 GL 프로그램이 클라이언트가 되고 그래픽 서버가 명령을 서비스 할 수 있음.

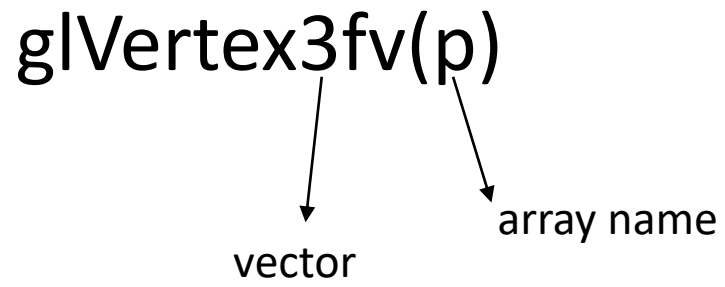
# OpenGL Programming

# Commands



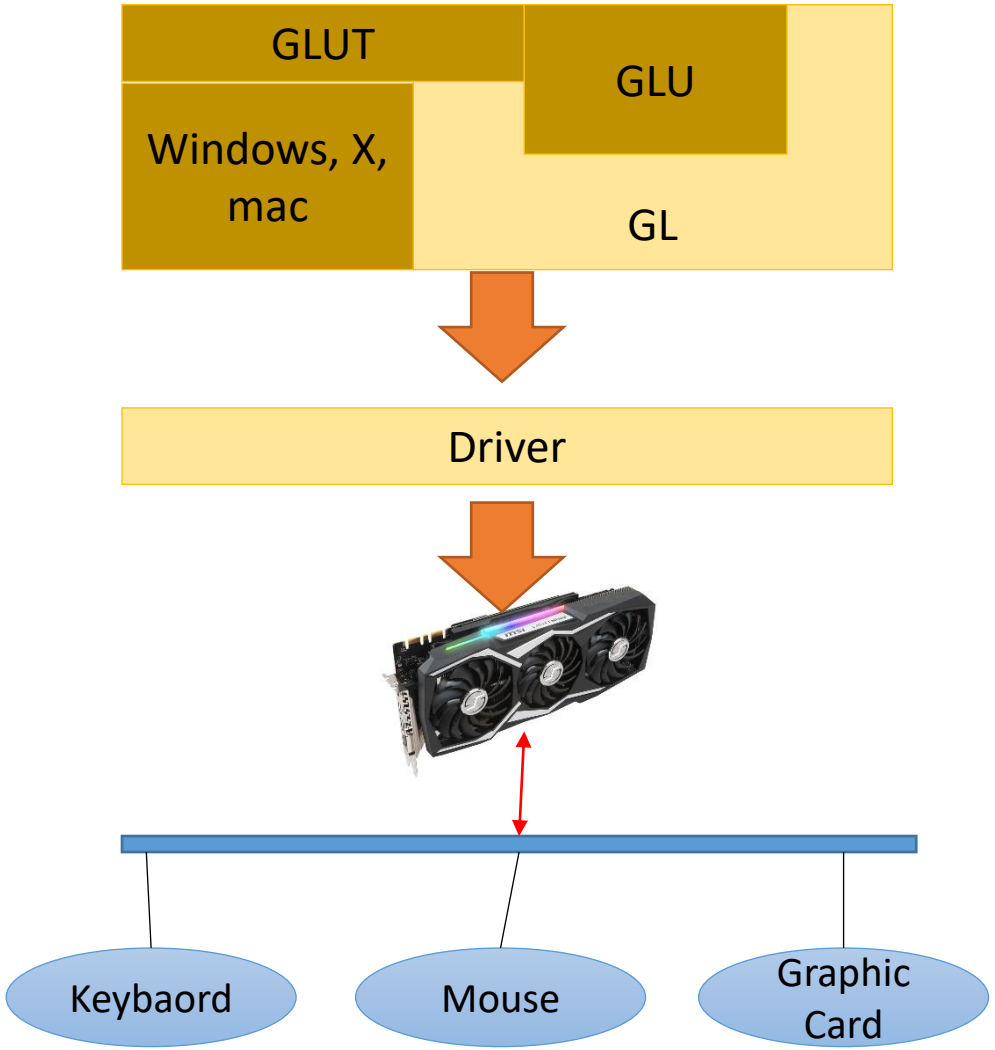

| suffix | Data type | C/C++ type | GL type |
|---|---|---|---|
| f | 32bit floating point | float | Glfloat |
| d | 64bit floating point | double | Gldouble |
| b | 8bit integer | signed char | GLbyte |
| ub | 8bit unsigned integer | unsigned char | GLubyte, GLboolean |
| i | 32bit integer | int or long | GLint |
| ui | 32bit unsigned integer | unsigned long | GLuint, GLenum, GLbitfield |
| s | 16bit integer | short | GLshort |

# Commands




- GL은 객체지향적이지 않다. 객체 지향의 특징 중 하나인 다형성을 지원하지 않기 때문에 일일이 입력 파라미터의 타입을 지정해줘야 한다.

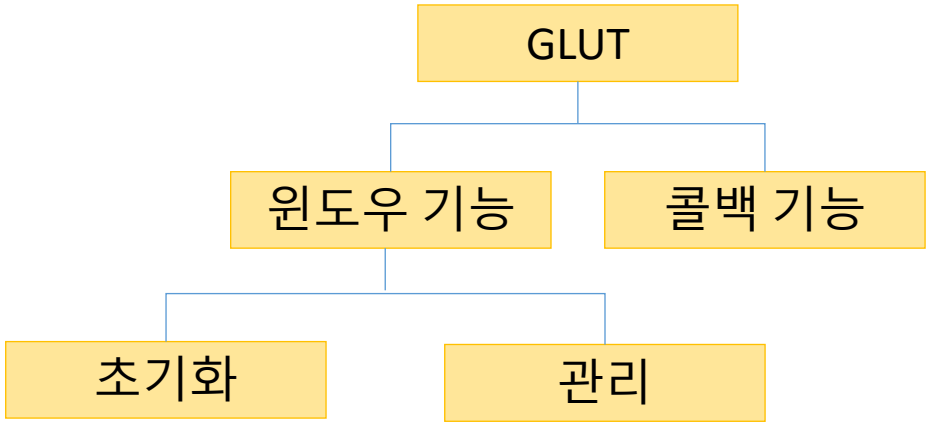
# GLProgram components




- GL: OpenGL Core Liabrary
  렌더링 기능을 제공하는 함수 라이브러리
- GLU: OpenGL Utility Library
  약 50개의 함수로 구성되며 사실상 GL 라이브러리의 도우미 역할을 함. 다각형 분할 (Tesselllation), 투상 (Projection), 2차원 곡면 (Quadratic Surface), NURBS 등 고급기능을 제공. GLU 함수는 결국 GL 함수 호출로 변환.
- GLUT: OpenGL Utility Toolkit
  사용자 입력을 받아들이거나 화면 윈도우를 제어하기 위한 함수로, 윈도우 운영체제가 실행하는 기능들

# GLUT와 윈도우 운영체제

- GLUT는 GL의 일부가 아니가 Mark Kilgard가 GL과는 별도로 작성한 프리웨어. GL과 마찬가지로 GLUT API도 자체 상태 변수및 상태 변수 테이블을 사용.
- GLUT로는 제한적인 GUI 인터페이스(메뉴, 버튼, 스크롤바 등)만이 사용 가능하다.
- OS 기능 중 GL program 실행에 필요한 것은 아래와 같은 윈도우 기능과 콜백 기능



# GLUT와 윈도우 운영체제

| 함수명 | | 기능 설명 |
|---|---|---|
| 윈도우 초기화 | glutInit() | 윈도우 운영체제와 세션 연결 |
| | glutInitWindowPosition() | 윈도우 위치 설정 |
| | glutInitWindowSize() | 윈도우 크기 설정 |
| | glutInitDisplayMode() | 디스플레이 모드 설정 |
| 윈도우 관리 | glutSetWindowTitle() | 윈도우 타이틀 설정 |
| | glutCreateWindow() | 새로운 윈도우 생성 |
| | glutReshapeWindow() | 크기 변경에 따른 윈도우 조정 |
| | glutPostRedisplay() | 현 윈도우가 재생되어야 함을 표시 |
| | glutSwapBuffers() | 현 프레임 버퍼 변경 |

# GLUT와 윈도우 운영체제

- glutInit(int *argcp, char** argv)

  GLUT 라이브러리를 초기화한 후 윈도우 시스템과 세션을 연다.

- glutInitWindowPosition(int x, int y)

  윈도우 좌상단을 화면 좌표 (x, y)에 위치시키는 함수

- glutInitDisplayMode(unsigned int mode)

  생성될 윈도우의 디스플레이 모드를 설정하는 함수. GLUT_RGB, GLUT_INDEX, GLUT_DOUBLE, GLUT_ALPHA, GLUT_DEPTH 등의 상수 값을 할당할 수 있음. 컬러모드는 GLUT_RGB를 사용하면 RGB 모드로, GLUT_INDEX를 사용하면 인덱스 모드로 설정. 프레임 모드는 GLUT_SINGLE을 사용하면 단일 버퍼, GLUT_DOUBLE을 사용하면 더블 버퍼로 설정. Ex)
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);

- glutReshapeWindow(int width, int height)

  프로그램 실행 중 사용자가 윈도우 크기를 재조정했을 때 호출되는 함수. 윈도우 운영체제가 사용자가 변경한 윈도우의 폭과 높이를 파라미터를 통해 GLUT에 전달.

# GLUT와 윈도우 운영체제

- glutPostRedisplay()

  현재의 윈도우를 재생하도록 요구. 내용이 변경되었으므로 가능한 가장 빠른 시간 내에 다시 렌더링해달라는 뜻을 게시(post)하는 함수

- glutSwapBuffers()

  더블 버퍼링에서는 프론트 버퍼의 내용이 화면에 뿌려지는 동안 새로운 내용이 백 버퍼에 쓰여진다. 백 버퍼에 기록이 완료되는 순간 이번에는 백 버퍼가 프런트 버퍼가 되어 그 내용이 화면에 뿌려짐. 백 버퍼와 프런트 버퍼를 교대(swap)하기 위한 함수.

# OpenGL 설치와 설정

# GLUT 설치


nate robins

main OpenGL chronicles allies

*GLUT for Win32*

The OpenGL Utility Toolkit (GLUT), originally written by Mark Kilgard, ported to Win32 (Windows 95,98,Me,NT,2000,XP) by Nate Robins. For more information check out the **README-win32.txt** provided in the distribution for installation instructions, and Win32 specific information..

**The latest version of the library is 3.7.6 (Nov 8, 2001)!**

glut-3.7.6-bin.zip (117 KB)
GLUT for Win32 dll, lib and header file (everything you need to get started programming with GLUT).

glut-3.7.6-src.zip (4.76 MB)
GLUT source code distribution (including a whole slew of great example programs + data).

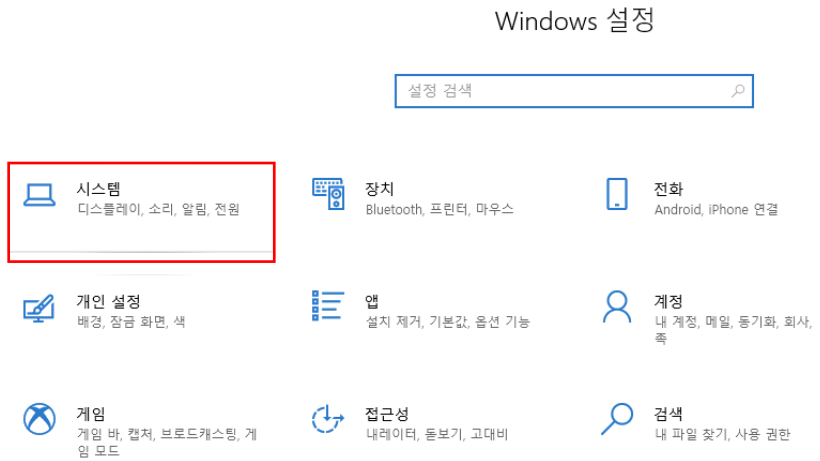
Documentation for the GLUT API is available in HTML, Postscript and PDF formats. For general information about GLUT, see opengl.org's GLUT page or opengl.org's GLUT FAQ. or sgi's GLUT ftp.

nate@pobox.com

November 2001
© Nate Robins

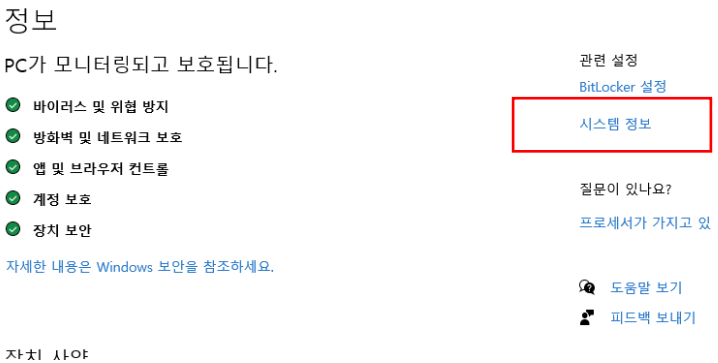
- 윈도우 기준으로 openGL과 GLU는 기본 설치되어있으나 GLUT은 설치되어있지 않음.
- https://user.xmission.com/~nate/glut.html 에서 glut 파일을 다운받아서 설치함. glut32.dll을 windows/syswow64로 이동

# 환경변수에 OpenGL 라이브러리 경로 추가

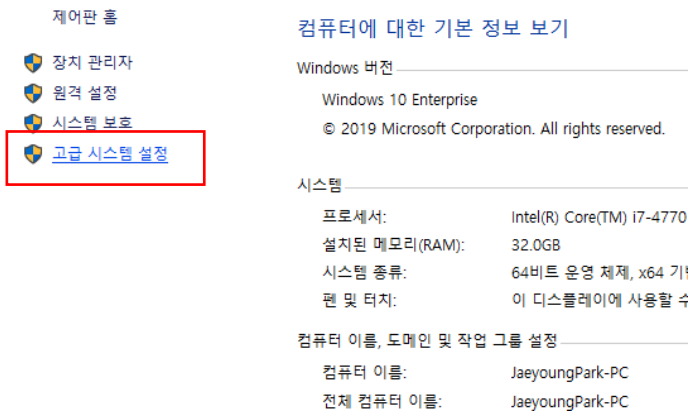



<Windows 설정에서 '시스템'>


<시스템에서 '정보' 선택>

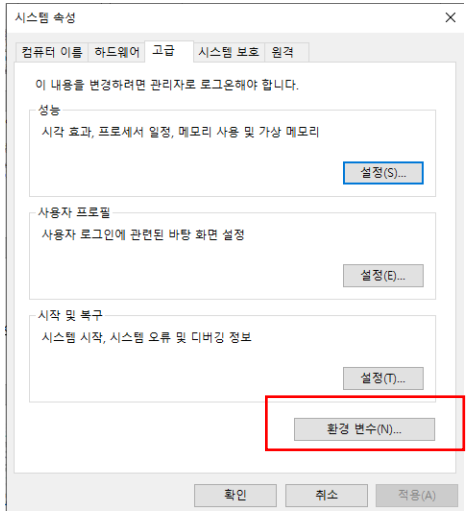



<정보에서 '시스템 정보'>

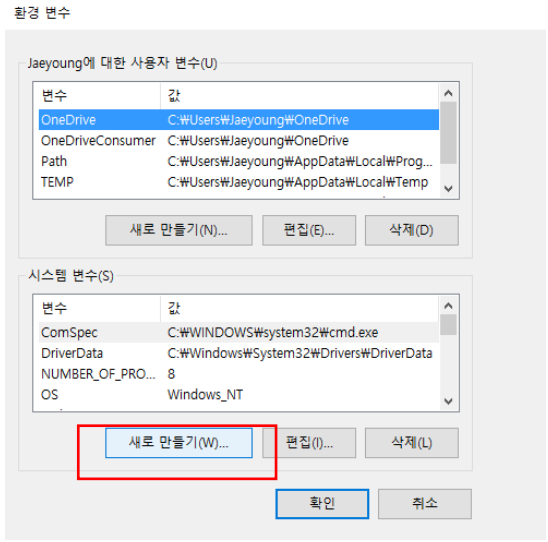



<'고급 시스템 설정'>

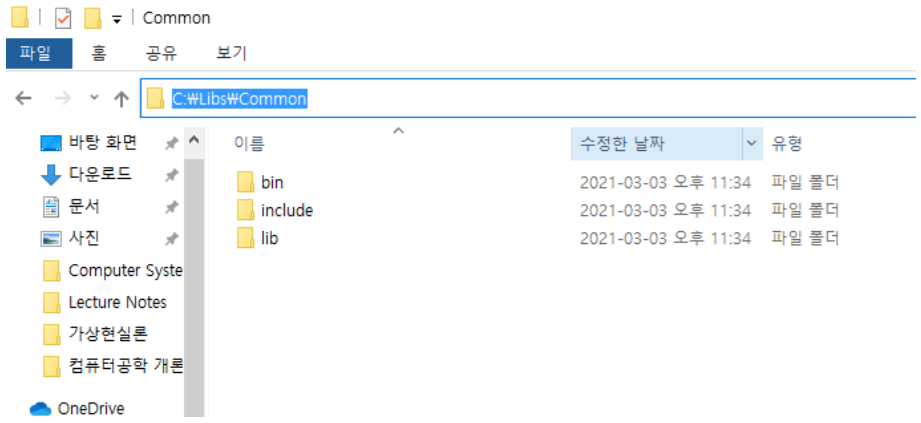
# 환경변수에 OpenGL 라이브러리 경로 추가



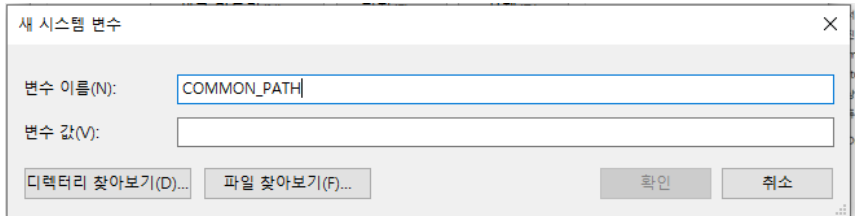

<'환경변수'>




<'새로만들기'>

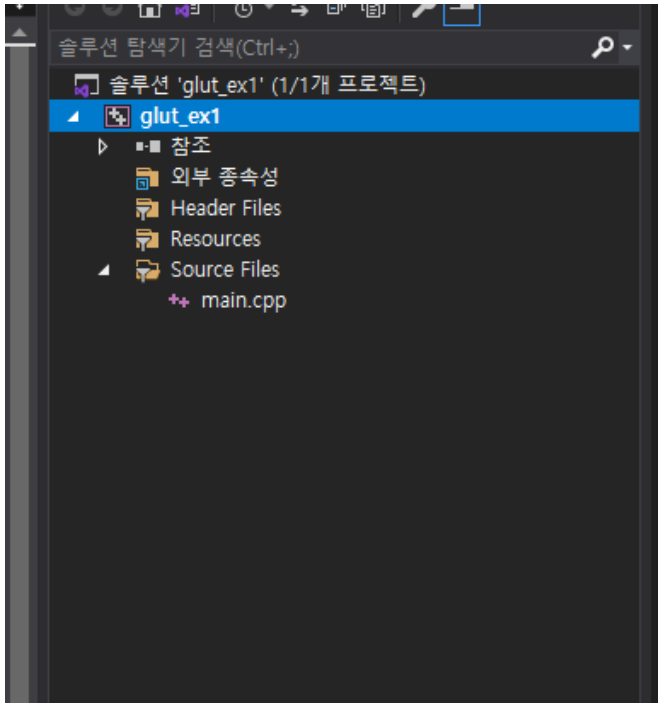



<header와 lib 폴더가 있는 경로 복사>

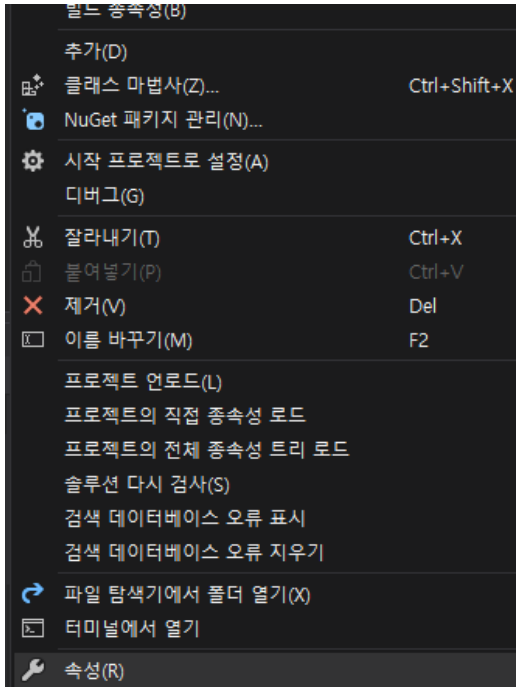



<새 시스템 변수에 변수 COMMON_PATH 설정하고 변수값에 복사한 경로 붙여넣기>

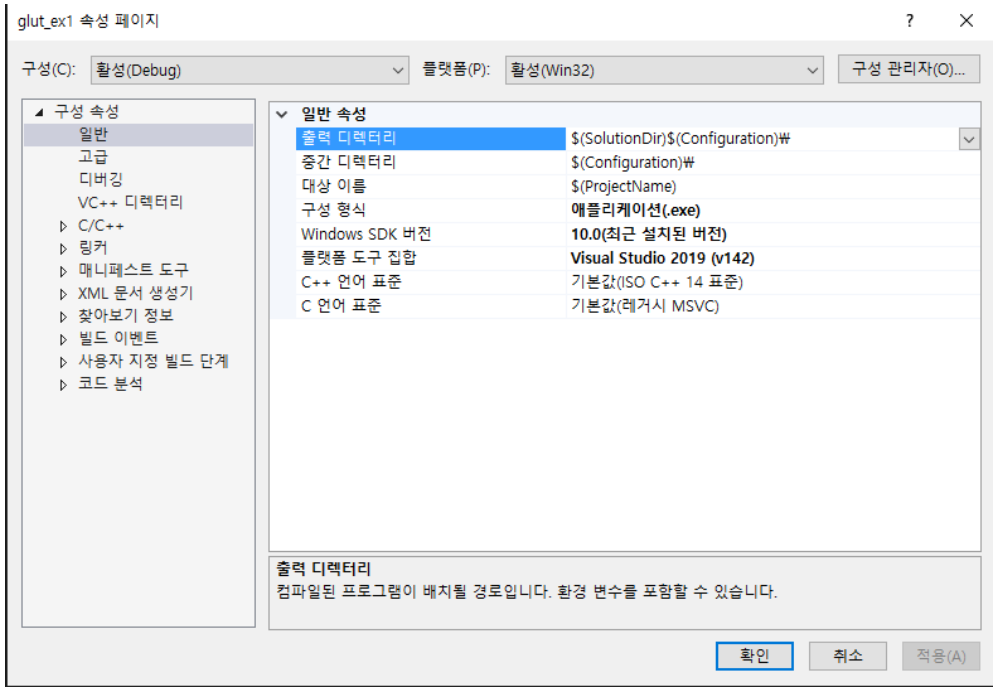
# Visual Studio OpenGL 프로그램 설정




<프로젝트 만들고
오른쪽 마우스 클릭>

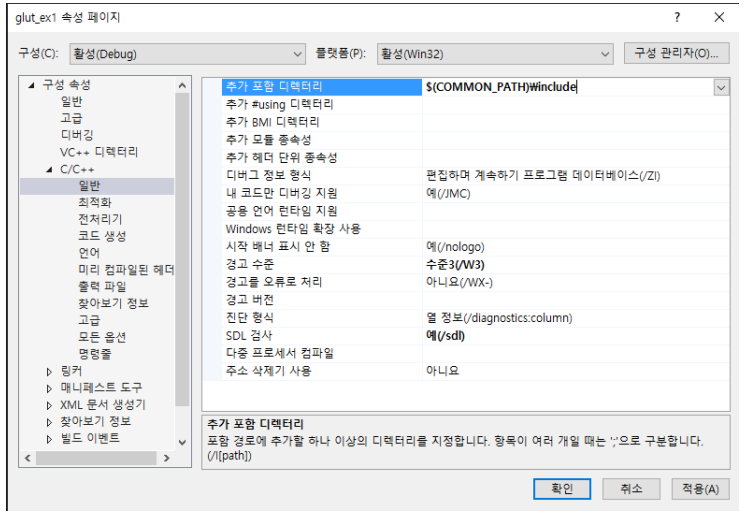



<맨 아래 속성 클릭>

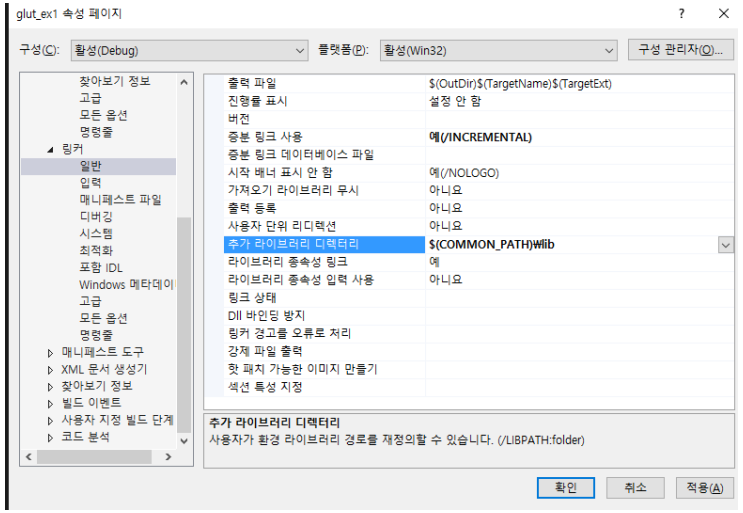



<속성페이지>

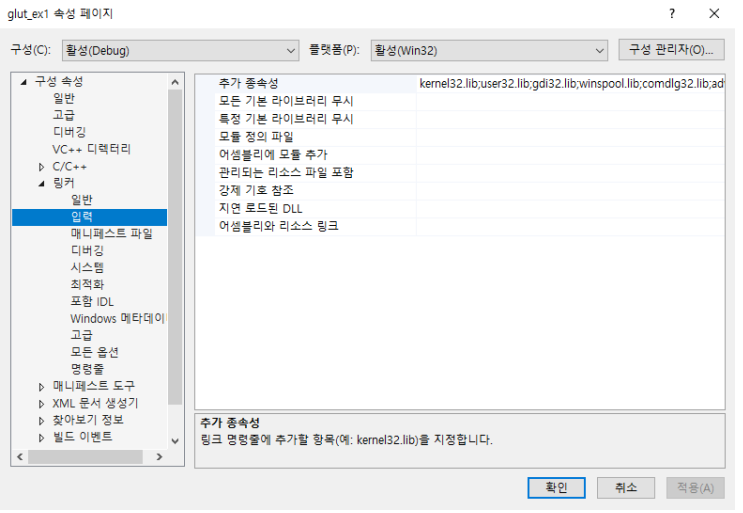
# Visual Studio OpenGL 프로그램 설정




<C/C++→ 일반 → 추가포함 디렉터리에 $(COMMON_PATH)\include 추가>

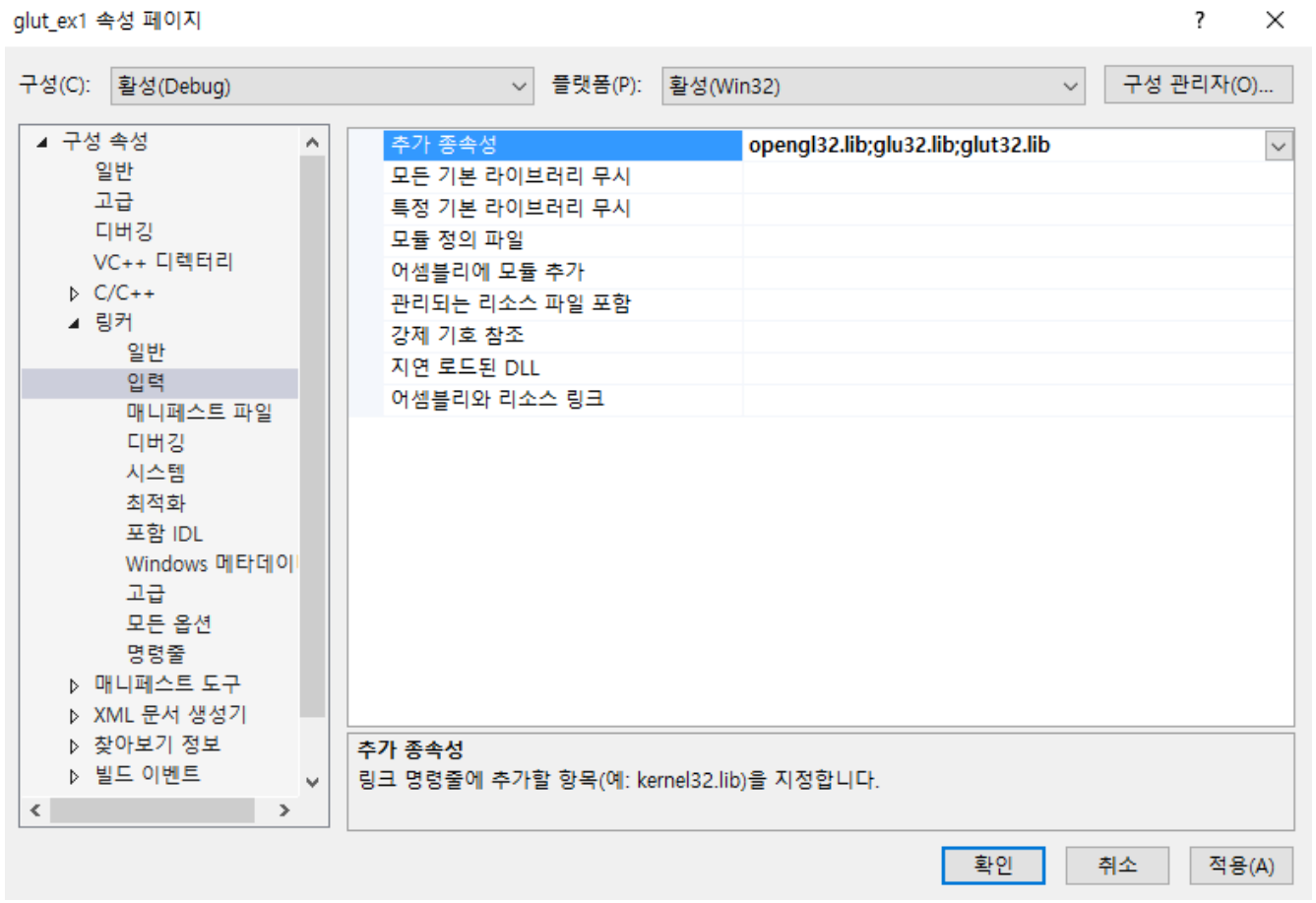



<링커→ 일반 → 추가 라이브러리 디렉터리에 $(COMMON_PATH)\lib 추가>




<링커→ 입력 → 추가종속성 창 클릭>

# Visual Studio OpenGL 프로그램 설정




<opengl32.lib;glu32.lib;glut32.lib 추가>

# OpenGL 프로그램 예제

main.cpp

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

void display() {
	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(display);
	glutMainLoop();
	return 0;
}
```

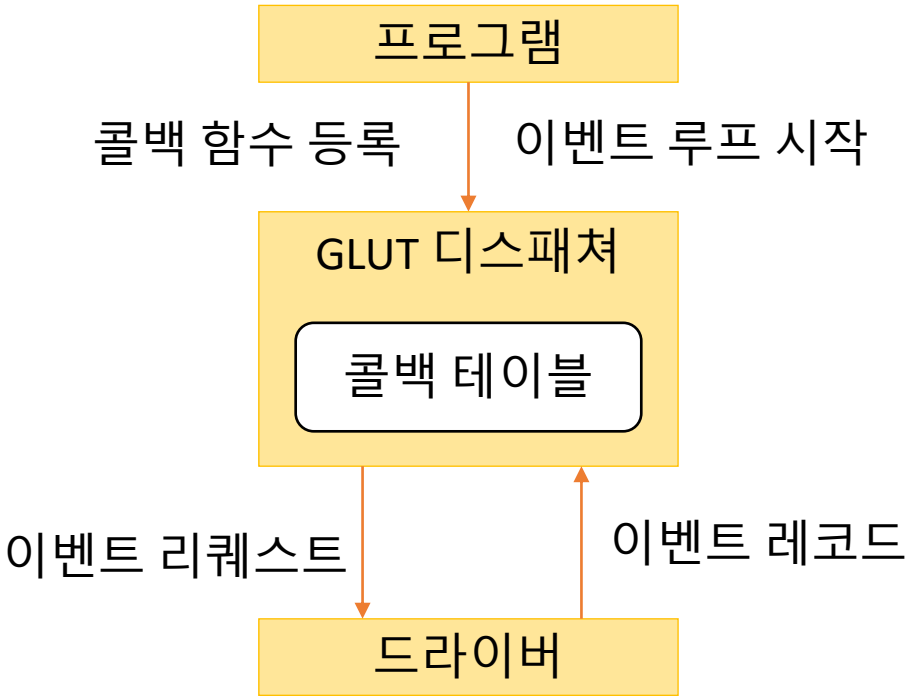
드라이버에 명령어를 쌓아놓지 않고
지금까지 쌓인 명령어 전부 실행

새로운 윈도우 생성

display()를 디스플레이 이벤트에 대한 콜백 함수로 등록

이벤트 루프로 진입

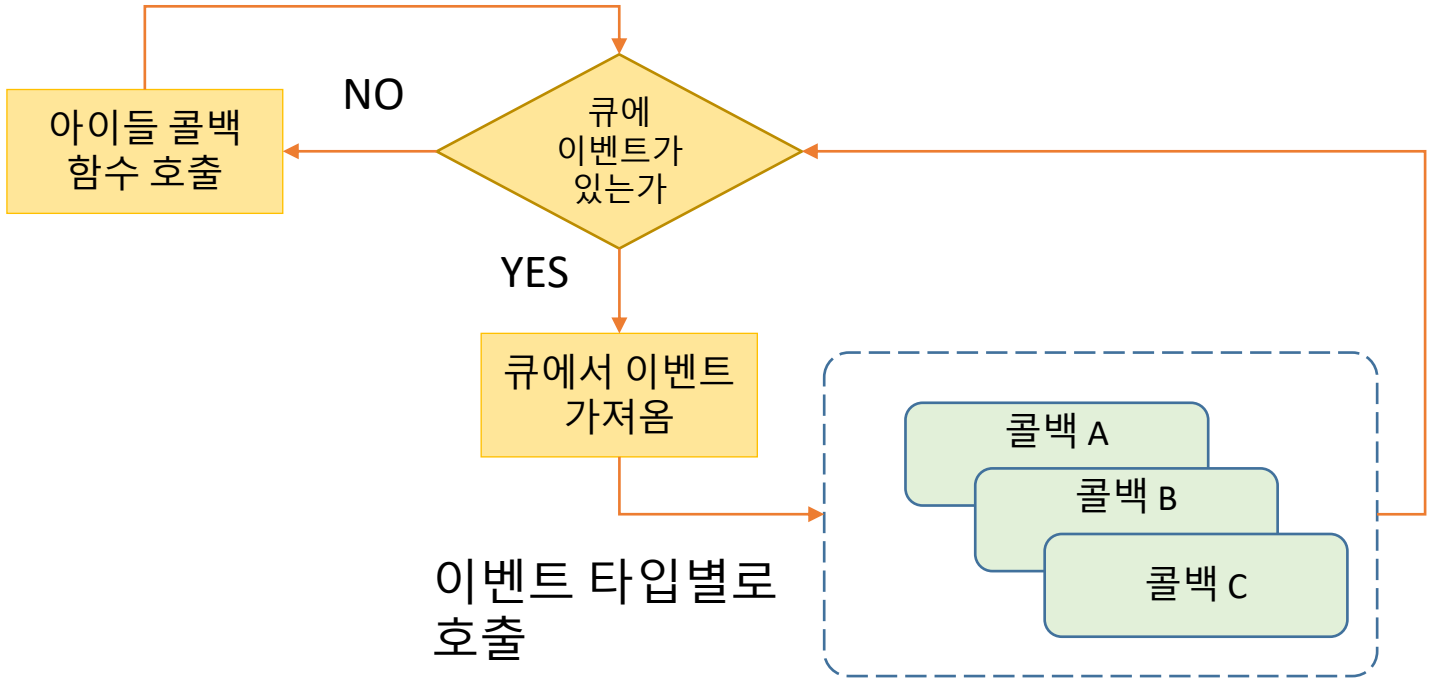
# 입력 callback과 GLUT




- Glut는 GL 프로그램과 드라이버 사이에서 인터페이스 역할을 한다. (프로그래머가 직접 핸들링 하지 않도록)

| 이벤트 타입 | 콜백함수 (ex) |
|---|---|
| DISPLAY | display() |
| RESHAPE | reshape() |
| KEYBOARD | keyboard() |
| MOUSE | mouse() |
| IDLE | idle() |

# 입력 callback과 glut




- 만약 큐에 이벤트가 있다면 첫번째 이벤트를 가져와서 해당 타입에 맞는 콜백 함수를 호출하고, 다시 루프로 들어가 다음 이벤트가 있는지 검사.
- 큐에 이벤트가 없으면 idle callback function 호출. 일반적으로 반복적인 작업이나 지속적인 업데이트가 필요할 때 (glutPostRedisplay 등) 사용.

# 입력 callback과 glut

| 이벤트 타입 | 콜백 함수 등록 명령(ex) | 콜백 함수 프로토타입 |
|---|---|---|
| DISPLAY | glutDisplayFunc(display) | void display(); |
| MOUSE | glutMouseFunc(mouse) | void mouse(int button, int x, int y); |
| KEYBOARD | glutKeyboardFunc(keyboard) | void keyboard(char key, int x, int y); |
| RESHAPE | glutReshapeFunc(reshape) | void reshape(int width, int height); |
| IDLE | glutIdleFunc(idle) | void idle(); |

- MOUSE: 마우스 버튼을 누르거나 뗄 때 트리거
- KEYBOARD: 키보드 입력시 트리거
- RESHAPE: 윈도우 크기를 바꿀 때 트리거
- IDLE: 다른 이벤트가 없을 때 반복적으로 트리거