

# OpenGL 프로그래밍

동명대학교 게임공학과

강영민

# OpenGL 설치

```
$ pip install pyOpenGL
```



OpenGL을 사용할 수 있는 시스템 환경 구축이 완료됨

# OpenGL 윈도우 생성

```
from OpenGL.GL import *
from OpenGL.GLU import *

from PyQt6.QtWidgets import QApplication, QMainWindow, QWidget
from PyQt6.QtOpenGLWidgets import QOpenGLWidget

import sys

class MyGLWindow(QOpenGLWidget):
    def __init__(self, parent=None):
        super(MyGLWindow, self).__init__(parent)

def main(argv = []):
    app = QApplication(argv)
    window = MyGLWindow()
    window.setWindowTitle('Example1')
    window.setFixedSize(600, 600)
    window.show()
    sys.exit(app.exec())

if __name__ == '__main__':
    main(sys.argv)
```



main 함수를 진입점을 사용하는 형태로 코딩

# OpenGL 위젯이 동작하도록 하는 기본 콜백

- 초기화<sup>initialize</sup> – OpenGL 콘텐츠의 초기화 콜백
- 크기 변경<sup>resize</sup> – 윈도우가 생성되거나 크기가 변경될 때 호출되는 콜백
- 디스플레이<sup>display</sup> – 화면에 그림을 그리는 일이 필요할 때 호출되는 콜백

```
class MyGLWindow(QOpenGLWidget) : # QOpenGLWidget 상속

    def __init__(self):
        super().__init__() # 슈퍼클래스 QMainWindow 생성자 실행

        self.setWindowTitle('나의 첫 OpenGL 앱')

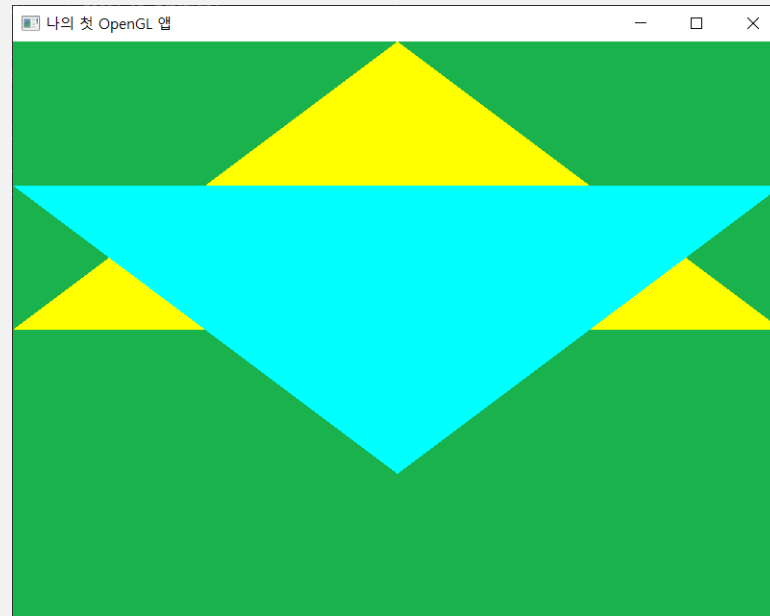
    def initializeGL(self) :
        glClearColor(0.1, 0.7, 0.3, 1.0)

    def resizeGL(self, w: int, h: int) :
        pass

    def paintGL(self):
        pass
```

# 그림 그리기

```
def paintGL(self):  
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)  
  
    glBegin(GL_TRIANGLES)  
    glColor3f(1, 1, 0)  
    glVertex3f(-1, 0, 0)  
    glVertex3f( 1, 0, 0)  
    glVertex3f( 0, 1, 0)  
    glColor3f(0, 1, 1)  
    glVertex3f(-1, 0.5, 0)  
    glVertex3f( 1, 0.5, 0)  
    glVertex3f( 0, -0.5, 0)  
    glEnd()
```



*You are now a graphics programmer!*

# Primitives

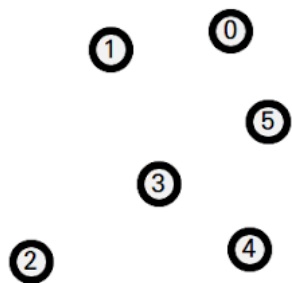
- OpenGL이 제공하는 그리기 기본 요소
  - 정점 데이터를 어떻게 조합할 것인지를 결정

프리티미브 사용 방법
glBegin( <b>프리티미브</b> 지정) 정점 정보 제공(glVertex3f 등을 이용) glEnd()

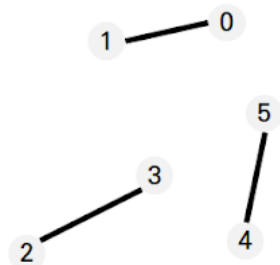
# Primitive의 종류

- GL\_POINTS – 입력된 정점을 하나씩 점으로 가시화
- GL\_LINES – 입력된 정점을 두 개씩 묶어 선분으로 표현
- GL\_LINE\_STRIP – 입력된 정점을 차례로 연결하여 하나의 폴리라인<sup>polyline</sup> 구성
- GL\_LINE\_LOOP – 입력된 정점을 차례로 연결한 뒤에 마지막 점을 시작점으로 연결
- GL\_TRIANGLES – 입력된 정점을 세 개씩 묶어 삼각형을 그림
- GL\_TRIANGLE\_STRIP – 처음 세 정점으로 삼각형 그린 뒤, 정점 추가될 때마다 삼각형을 직전 두 개 정점과 연결
- GL\_TRIANGLE\_FAN – 부채 모양으로 삼각형을 추가해 나감
- GL\_QUADS – 정점 네 개씩을 묶어 사각형 그리기
- GL\_QUAD\_STRIP: 처음 네 개 정점으로 사각형 그리고, 이후 두 개씩 묶어 직전 두 개 정점과 함께 사각형 그리기
- GL\_POLYGON: 입력된 모든 정점으로 다각형을 그림

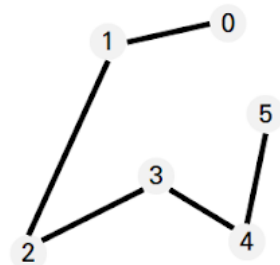
# Primitives



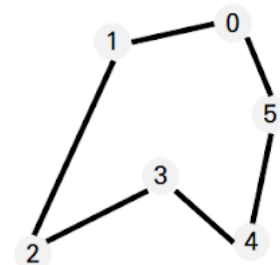
GL\_POINTS



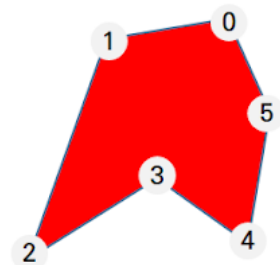
GL\_LINES



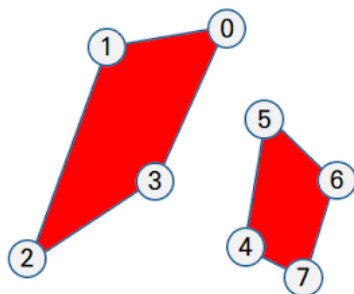
GL\_LINE\_STRIP



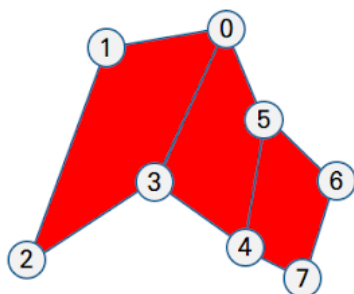
GL\_LINE\_LOOP



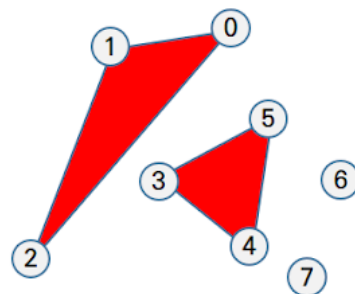
GL\_POLYGON



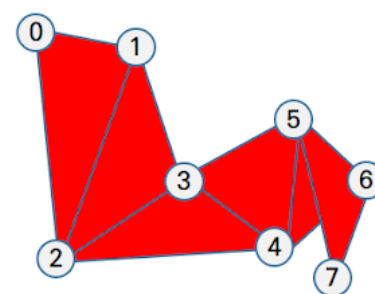
GL\_QUADS



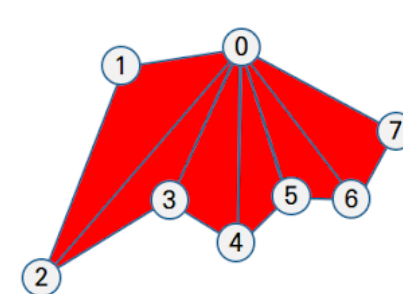
GL\_QUAD\_STRIP



GL\_TRIANGLES



GL\_TRIANGLE\_STRIP



GL\_TRIANGLE\_FAN



# 정점 정보 제공 방법

## 정점 정보 제공 방법

```
ix, iy, iz = -1, 0, 0  
fx, fy, fz = 1.0, 0.0, 0.0  
fverts = [fx-1.0, fy+1.0, fz]
```

```
glBegin(GL_TRIANGLES)  
glVertex3i(ix, iy, iz)  
glVertex3f(fx, fy, fz)  
glVertex3fv(fverts)  
glEnd()
```

### 주요 정점 정보

- 대표적인 것은 정점의 위치, 해당 정점에서 면의 **법선**normal vector, 그리고 색상 등
- 위의 예는 정점의 위치만 제공

# 간단한 그리기 연습

Qt의 OpenGL 위젯인 QOpenGLWidget을 이용하여 프리미티브와 정점 정보 사용하기

```
from OpenGL.GL import *
from OpenGL.GLU import *

import sys

from PyQt6.QtWidgets import QApplication, QMainWindow, QWidget
from PyQt6.QtOpenGLWidgets import QOpenGLWidget

class MyGLWindow(QOpenGLWidget):

    def __init__(self, parent=None):
        super(MyGLWindow, self).__init__(parent)

    def initializeGL(self):
        # OpenGL 그리기를 수행하기 전에 각종 상태값을 초기화
        glClearColor(0.8, 0.8, 0.6, 1.0)

    def resizeGL(self, width, height):
        # 카메라의 투영 특성을 여기서 설정
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()

    def paintGL(self):
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        glMatrixMode(GL_MODELVIEW)
        glLoadIdentity()
```

```
glColor3f(0, 1, 1)
```

```
ix, iy, iz = -1, 0, 0
fx, fy, fz = 1.0, 0.0, 0.0
fverts = [fx-1.0, fy+1.0, fz]
```

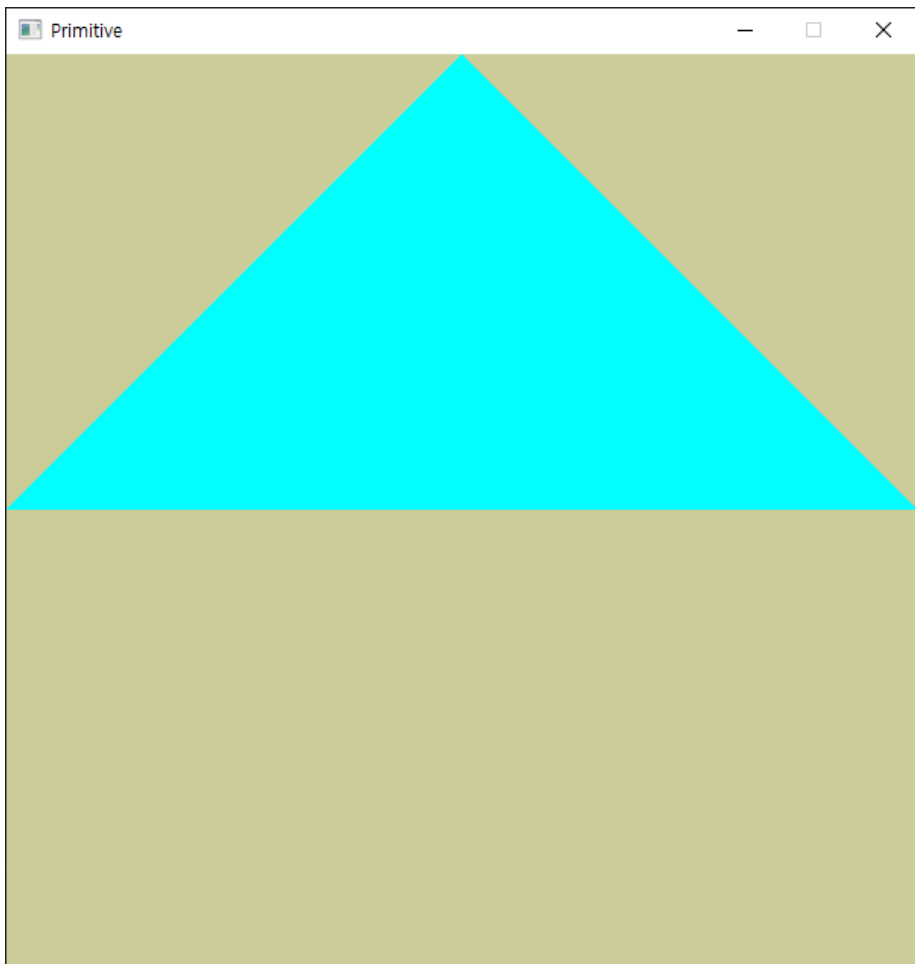
```
glBegin(GL_TRIANGLES)
glVertex3i(ix, iy, iz)
glVertex3f(fx, fy, fz)
glVertex3fv(fverts)
glEnd()
```

```
# 그려진 프레임버퍼를 화면으로 송출
glFlush()
```

```
def main(argv = []):
    app = QApplication(argv)
    window = MyGLWindow()
    window.setWindowTitle('Primitive')
    window.setFixedSize(600, 600)
    window.show()
    sys.exit(app.exec())
```

```
if __name__ == '__main__':
    main(sys.argv)
```

# 결과



정점 정보 제공시 사용한 함수

- 정수 좌표를 제공하기 위해 glVertex3i
- 부동소수점으로 표시된 좌표 glVertex3f
- v가 접미사로 붙은 함수
  - 이것은 벡터vector를 의미
  - 정점 정보가 배열이나 리스트와 같은 묶음
  - 3fv = 3차원 데이터, 부동소수점, 묶음

# Project 1

## 점 정보를 제공하여 각종 프리미티브로 그리기

정점 입력을 처리하기 위한 기본 프로그램 작성

- 마우스 클릭 위치에 정점 그리고 데이터 추가하기
  - Qt의 QPainter와 QPen으로 그리기

```
from PyQt6.QtGui import QPainter, QPen
```

입력한 위치를 기준으로 포인트 좌표 누적

```
# 정점 데이터 (2차원 정점을 리스트로 표현하고, 이들의 리스트로 복수의 정점 관리)  
POINTS = [[0, 0], [10, 10], [100, 50]]
```

# Project 1

## 점 정보를 제공하여 각종 프리미티브로 그리기

윈도를 두 영역으로 나누어

- 한 쪽은 정점을 입력받는 위젯
- 다른 쪽
  - 정점을 초기화하는 버튼
  - OpenGL 프리미티브 선택 콤보박스

```
### GUI 설정
central_widget = QWidget()
self.setCentralWidget(central_widget)

gui_layout = QHBoxLayout() # CentralWidget의 수평 나열 레이아웃

# 배치될 것들 - 정점 입력을 받기 위한 위젯
central_widget.setLayout(gui_layout)

self.controlGroup = QGroupBox('정점 입력')
gui_layout.addWidget(self.controlGroup)

control_layout = QVBoxLayout()
self.controlGroup.setLayout(control_layout)

## 정점을 초기화하는 버튼을 추가. 버튼을 누르면 정점이 사라진다.
## 이 버튼을 누르면 resetPoints라는 이 메소드가 호출된다.
reset_button = QPushButton('정점 초기화', self)
reset_button.clicked.connect(self.resetPoints)

control_layout.addWidget(reset_button)

## 정점을 입력받기 위한 위젯을 만들고, pointInput이라는 멤버로 관리하자.
self.pointInput = Drawer(parent=self)
gui_layout.addWidget(self.pointInput)
```

# Project 1

## 점 정보를 제공하여 각종 프리미티브로 그리기

### 정점 초기화

```
# 정점 초기화 버튼을 눌렀을 때 호출되는 메소드
def resetPoints(self, btn):
    global POINTS
    POINTS = []
    self.pointInput.update()
```

### QPainter 클래스를 상속해 Drawer 클래스 만들기

```
#### 정점 입력을 위해 사용되는 위젯으로 QPainter를 활용한다.
class Drawer(QWidget):
    def __init__(self, parent=None):
        QWidget.__init__(self, parent)
        self.parent = parent
        # QPainter 멤버 준비
        self.painter = QPainter()
```

Drawer 클래스 객체가 반응하는 이벤트

- 마우스를 이용하여 점의 위치를 입력하는 이벤트
- 그리기 동작을 수행해야 하는 이벤트

# Project 1

## 점 정보를 제공하여 각종 프리미티브로 그리기

### QPainter 클래스

```
# 정점을 입력하는 방법은 마우스 이벤트 발생시에 좌표를 읽어
# 이 정점을 표현하는 2차원 정보를 리스트로 만들어 정점 리스트 POINTS에 추가
def mousePressEvent(self, event):
    POINTS.append([event.position().x(), event.position().y()])
    print(event.position().x(), event.position().y())
    self.update()
```

```
# QPainter를 이용하여 입력된 정점을 출력한다.
def paintEvent(self, event):
    global POINTS

    self.painter.begin(self)
    self.painter.setPen(QPen(Qt.GlobalColor.red, 6))

    for i in range(len(POINTS)):
        self.painter.drawPoint(POINTS[i][0], POINTS[i][1])

    self.painter.setPen(QPen(Qt.GlobalColor.blue, 2))
    for i in range(len(POINTS) - 1):
        self.painter.drawLine(POINTS[i][0], POINTS[i][1],
                               POINTS[i + 1][0], POINTS[i + 1][1])

    self.painter.end()
```

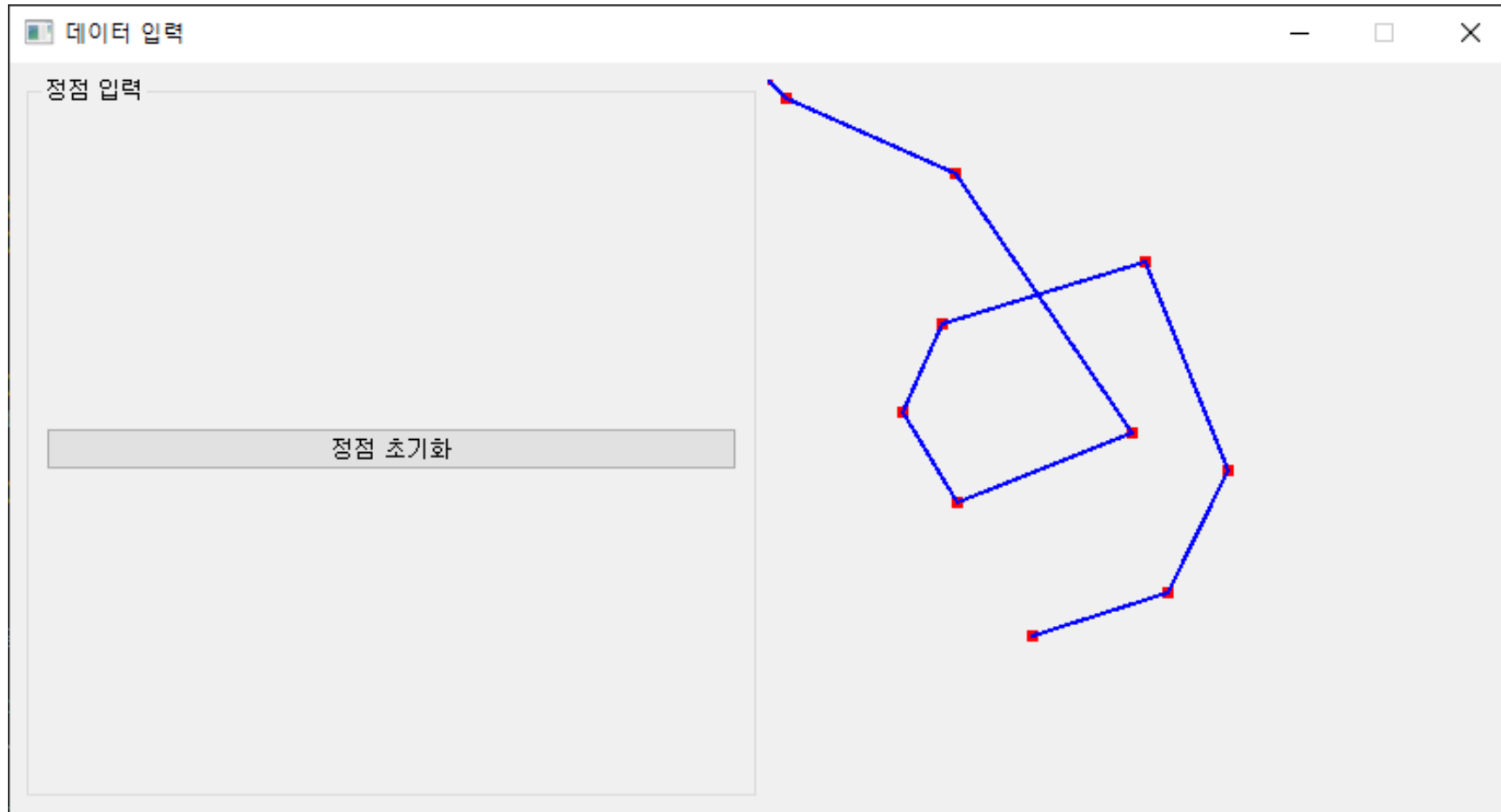
Drawer 클래스 객체가 반응하는 이벤트

- 마우스를 이용하여 점의 위치를 입력하는 이벤트
- 그리기 동작을 수행해야 하는 이벤트

# Project 1

## 점 정보를 제공하여 각종 프리미티브로 그리기

정점 입력을 받는 코드 완성  
교재 3장 참조





# Project 1

## 점 정보를 제공하여 각종 프리미티브로 그리기

### OpenGL 위젯과 연결

```
### OpenGL #####  
from OpenGL.GL import *  
from OpenGL.GLU import *  
  
#### 추가로 필요한 패키지  
from PyQt6.QtWidgets import QOpenGLWidget, QComboBox
```

```
##### 프리미터 선택을 위한 데이터  
PRIMITIVES = ['GL_POINTS', 'GL_LINES', 'GL_LINE_STRIP', 'GL_LINE_LOOP',  
              'GL_TRIANGLES', 'GL_TRIANGLE_STRIP', 'GL_TRIANGLE_FAN',  
              'GL_QUADS', 'GL_QUAD_STRIP', 'GL_POLYGON']  
  
PRIMITIVE_VALUES = [GL_POINTS, GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP,  
                    GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN,  
                    GL_QUADS, GL_QUAD_STRIP, GL_POLYGON]  
  
selected = 0
```

- 콤보 박스를 이용하여 OpenGL 프리미티브를 선택
- 선택된 프리미티브로 정점을 그리기
  - 콤보 박스에 표현될 문자열의 리스트: PRIMITIVES
  - 선택된 프리미티브에 해당하는 OpenGL 프리미티브
    - PRIMITIVE\_VALUES
  - 선택된 프리미티브의 인덱스<sup>index</sup>는 selected

# Project 1

## 점 정보를 제공하여 각종 프리미티브로 그리기

윈도에 부착된 OpenGL 위젯 준비

```
##### 정점 정보 그리기
class MyGLWidget(QOpenGLWidget):

    def __init__(self, parent=None):
        super(MyGLWidget, self).__init__(parent)

    def initializeGL(self):
        # OpenGL 그리기를 수행하기 전에 각종 상태값을 초기화
        glClearColor(0.8, 0.8, 0.6, 1.0)
        glPointSize(4)
        glLineWidth(2)

    def resizeGL(self, width, height):
        # 카메라의 투영 특성을 여기서 설정
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()
        glOrtho(0, 240, 380, 0, -1, 1)
```

```
def paintGL(self):
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

    # 프리미티브를 이용한 객체 그리기
    glBegin(PRIMITIVE_VALUES[selected])
    nPoints = len(POINTS)
    for i in range(nPoints):
        glVertex2fv(POINTS[i])
    glEnd()

    # 그려진 프레임버퍼를 화면으로 송출
    glFlush()
```

# Project 1

## 점 정보를 제공하여 각종 프리미티브로 그리기

OpenGL 위젯을 만들어 윈도우에 추가

```
##### OpenGL Widget 추가
self.glWidget = MyGLWidget() # OpenGL Widget
gui_layout.addWidget(self.glWidget)
```

콤보 박스를 이용하여 프리미티브 선택가능하게 함


```
##### 프리미티브 선택 기능 추가
primitive_selection = QComboBox()
for i in range(len(PRIMITIVES)):
    primitive_selection.addItem(PRIMITIVES[i])

# ComboBox에 기능 연결
primitive_selection.currentIndexChanged.connect(self.selectPrimitive)

reset_button = QPushButton('reset vertices', self)
reset_button.clicked.connect(self.resetPoints)

control_layout.addWidget(primitive_selection)
control_layout.addWidget(reset_button)
```

```
##### Primitive 선택
def selectPrimitive(self, text):
    global selected
    selected = int(text)
    self.glWidget.update()
```



# Project 1

## 점 정보를 제공하여 각종 프리미티브로 그리기

마우스 입력이 있을 때 OepGL 위젯 변경되게 함

```
def mousePressEvent(self, event):  
    POINTS.append([event.position.x(), event.position.y()])  
    print(event.position().x(), event.position().y())  
    #####  
    self.parent.glWidget.update()  
    #####
```

