

Chapter 03 - 동명대학교 게임공학과 3D 그래픽스 프로그래밍

프리미티브

“우리가 내다볼 수 있는 거리는 짧지만, 거기에도 해야 할 것들이 아주 많이 보인다.”

- 엘런 튜링, 1950.



독일군의 암호를 해독하기 위해 튜링이 설계한 기계 봄브 Bombe의 일종
Neil Barnwell, <https://500px.com/p/neilbarnwell>, CC BY-SA 3.0

이 장에서 생각할 문제

- ❖ 강의에 대한 이해.
- ❖ 강의의 목표와 평가 방법에 대한 이해.

3.1 프리미티브의 역할

프리미티브 primitive는 OpenGL이 제공하는 그리기 기본 요소인데, OpenGL에 제공되는 정점 데이터들을 어떻게 조합할 것인지를 결정한다. 이 프리미티브를 사용하는 방법은 다음과 같다.

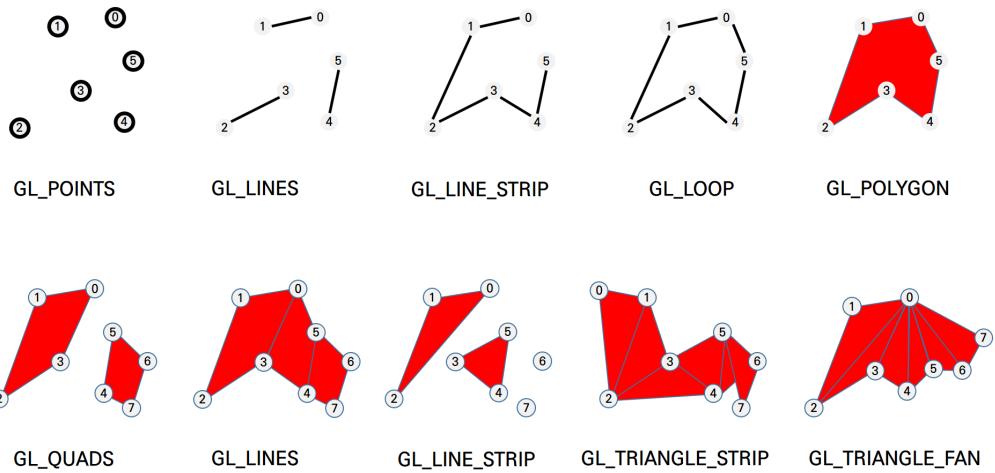
프리미티브 사용 방법

```
glBegin(프리미티브 지정)  
    정점 정보 제공(glVertex3f 등을 이용)  
glEnd()
```

그래픽 하드웨어는 프로그래머가 지정한 프리미티브에 따라 정점의 리스트에서 정점을 몇 개 단위로 끊어서 그릴지 결정한다. 예를 들어 삼각형을 그리기 위해서는 GL_TRIANGLES라는 프리미티브를 사용하면, 입력된 정보를 읽고 정점 3 개씩 묶어 삼각형을 그린다. 프리미티브는 다음과 같은 것들이 존재한다.

- 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 polyline 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 개 정점으로 삼각형을 그린 뒤, 정점이 추가될 때마다 삼각형을 직전 두 개 정점과 연결하여 삼각형 추가
- 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

아래 그림은 정점이 순서를 가지고 제공되었을 때 각각의 프리미티브가 정점을 어떤 방식으로 조합 assembly하여 그림을 그리는지를 보이고 있다.



3.2 정점 정보 제공 방법

프리미티브가 설정되면 이 프리미티브에 따라 그림을 그릴 때에 필요한 정점 정보가 제공되어야 한다. 정점 데이터는 앞으로 여러 종류를 살펴볼 것인데, 대표적인 것은 정점의 위치, 해당 정점에서 면의 법선^{normal vector}, 그리고 색상 등이다. 정점의 위치를 입력하는 OpenGL 명령어는 `glVertex[차원]자료형`이다. 예를 들어 3차원 정점의 각 성분을 부동소수점 표현으로 입력하고 싶다면 `glVertex3f(x, y, z)`와 같이 입력이 가능하다. 따라서 다음과 같은 다양한 방법으로 정점 데이터를 입력할 수 있다.

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()

```

모두 3차원 좌표를 사용하여 차원을 표시하는 숫자가 3으로 되어 있다. 정수 좌표를 제공하기 위해 glVertex3i를 사용했고, 부동소수점으로 표시된 좌표를 다루기 위해서 glVertex3f를 사용한 것을 확인할 수 있을 것이다. 마지막으로 사용한 함수에는 v가 접미사로 붙어 있는데, 이것은 벡터vector를 의미하며, 정점 정보가 배열이나 리스트와 같이 뮤음으로 제공될 때 사용할 수 있다. 3fv라는 접미사는 결국 3차원 데이터를 표현하는 부동소수점 데이터 3 개가 뮤음으로 제공된다는 의미이며, 여기서는 파이썬의 리스트를 이용하여 좌표값이 전달되었다.

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()

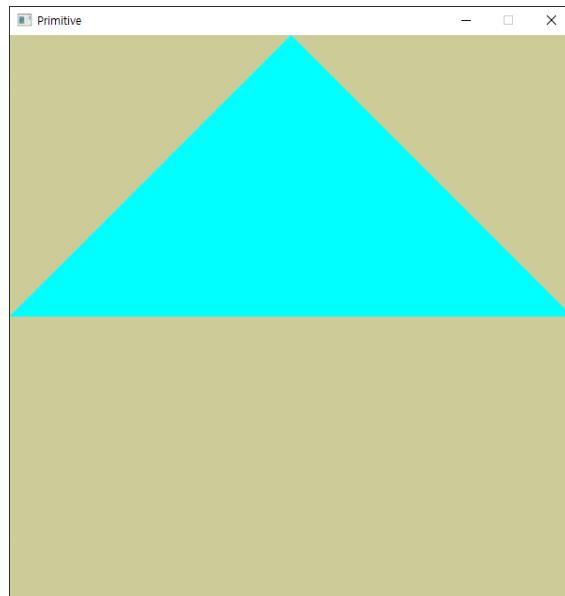
# 그려진 프레임버퍼를 화면으로 송출
glFlush()

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)

```

이러한 방식으로 그리기 코드를 구현하여 실행하면 다음 그림과 같이 세 개의 정점을 가진 삼각형이 그려진다. 프리미티브로는 GL_TRIANGLES가 사용되었고, 이 프리미티브는 세 개의 정점을 모아 하나의 다각형을 그린다. 제공된 정점이 3 개이므로 이들을 이용하여 삼각형을 그리게 된다.



입력 즉시 볼 수 있기 때문에 입문자도 간편하게 프로그램의 실행을 살펴볼 수 있다.

3.3 정점 정보 제공 및 OpenGL 위치과 염계

임의의 정점을 입력하고, 이 정점을 다양한 프리미티브로 그림을 그려볼 수 있도록 해 보자.
우선은 정점을 입력받는 프로그램을 작성해 보자.

정점을 입력받기 위한 사용자 인터페이스 구현

```
from PyQt5.QtWidgets import QApplication, QMainWindow, QVBoxLayout,
QHBoxLayout, QWidget
from PyQt5.QtWidgets import QGroupBox, QPushButton
from PyQt5.QtCore import *
from PyQt5.QtGui import QPainter, QPen
import sys
import numpy as np

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

```

class MyWindow(QMainWindow):

    def __init__(self, title=''):
        QMainWindow.__init__(self) # QMainWindow 슈퍼 클래스의 초기화
        self.setWindowTitle(title)

    ### GUI 설정

    central_widget = QWidget()
    self.setCentralWidget(central_widget)

    gui_layout = QHBoxLayout() # CentralWidget의 수평 나열 레이아웃

    # 배치될 것들 - 정점 입력을 받기 위한 위치
    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 위치를 만들고, pointInput이라는 멤버로 관리하자.
    self.pointInput = Drawer(parent=self)
    gui_layout.addWidget(self.pointInput)

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

    #### 정점 입력을 위해 사용되는 위치으로 QPainter를 활용한다.

class Drawer(QWidget):
    def __init__(self, parent=None):
        QWidget.__init__(self, parent)
        self.parent = parent
        # QPainter 멤버 준비
        self.painter = QPainter()

```

```

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

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

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

    self.painter.setPen(QPen(Qt.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()

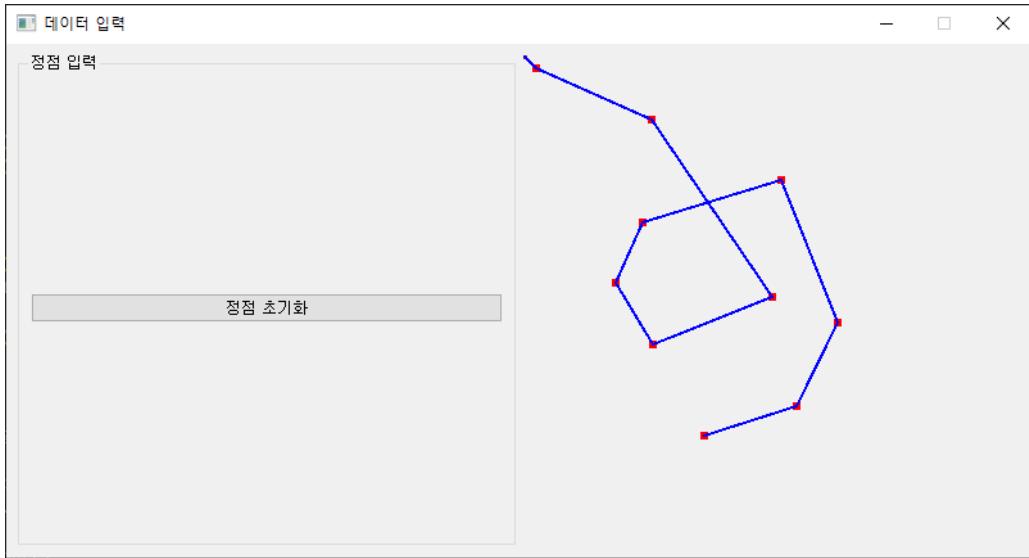
# 정점을 입력하는 방법은 마우스 이벤트 발생시에 좌표를 읽어
# 이 정점을 표현하는 2차원 정보를 리스트로 만들어 정점 리스트 POINTS에 추가
def mousePressEvent(self, event):
    POINTS.append([event.x(), event.y()])
    print(event.x(), event.y())
    self.update()

def main(argv=[]):
    app = QApplication(argv)
    window = MyWindow('데이터 입력')
    window.setFixedSize(800, 400)
    window.show()
    sys.exit(app.exec_())

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

```

이 프로그램을 실행하면 아래와 같이 우측에 마우스 클릭을 하면 해당 위치의 좌표에 정점 데이터가 추가되고 입력된 데이터들을 화면에 보여주는 프로그램이 동작할 것이다.



다음으로 우리가 할 일은 이 윈도에 OpenGL을 이용하여 그림을 그리는 위젯을 추가하고, 이 OpenGL 위젯 안에 입력된 정점을 이용하여 그림을 그리는 것이다.

정점을 입력받기 위한 사용자 인터페이스 구현

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

#####
#####

#### 추가로 필요한 패키지
from PyQt5.QtWidgets import QOpenGLWidget, QComboBox
#####

from PyQt5.QtWidgets import QApplication, QMainWindow, QVBoxLayout,
QHBoxLayout, QWidget
from PyQt5.QtWidgets import QGroupBox, QPushButton
from PyQt5.QtCore import *
from PyQt5.QtGui import QPainter, QPen
import sys
import numpy as np

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

```

##### 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
#####

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

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

    def initializeGL(self):
        # ^{\it \color{gray} OpenGL 그리기를 수행하기 전에 각종 상태값을 초기화}^
        glClearColor(0.8, 0.8, 0.6, 1.0)
        glPointSize(4)
        glLineWidth(2)
        glEnable(GL_BLEND)

    def resizeGL(self, width, height):
        # ^{\it \color{gray} 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()

        # ^{\it \color{gray} 색과 프리미티브를 이용한 객체 그리기}^
        glColor3f(1, 1, 0)
        glBegin(GL_POINTS)
        for i in range(len(POINTS)):
            glVertex2fv(POINTS[i])
        glEnd()

        glBegin(PRIMITIVE_VALUES[selected])
        nPoints = len(POINTS)
        for i in range(nPoints):
            glVertex2fv(POINTS[i])
        glEnd()

```

```

# ^{\it \color{gray}} 그려진 프레임버퍼를 화면으로 송출}^
glFlush()
#####
# ^{\it \color{gray}} OpenGL Widget 추가
self.glWidget = MyGLWidget() # ^{\it \color{gray}} OpenGL Widget}^
gui_layout.addWidget(self.glWidget)
#####

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

# 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)

#####
# ^{\it \color{gray}} primitive 선택 기능 추가
primitive_selection = QComboBox()
for i in range(len(PRIMITIVES)):
    primitive_selection.addItem(PRIMITIVES[i])

# ^{\it \color{gray}} 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)
#####
## 정점을 입력받기 위한 위젯을 만들고, pointInput이라는 멤버로 관리하자.
self.pointInput = Drawer(parent=self)
gui_layout.addWidget(self.pointInput)

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

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

#####
## 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()

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

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

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

        self.painter.setPen(QPen(Qt.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()

```

```
# 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.x(), event.y()])
    print(event.x(), event.y())
    #####
    self.parent.g1Widget.update()
    #####
    self.update()

def main(argv=[]):
    app = QApplication(argv)
    window = MyWindow('데이터 입력')
    window.setFixedSize(800, 400)
    window.show()
    sys.exit(app.exec_())

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