Chapter 08 - 동명대학교 게임공학과 3D 그래픽스 프로그래밍

계층적 모델링

"소프트웨어 재사용 전에, 우선은 사용부터 가능해야 한다."
"Before software should be reusable, it should be usable"
– 랄프 존슨Ralph Johnson

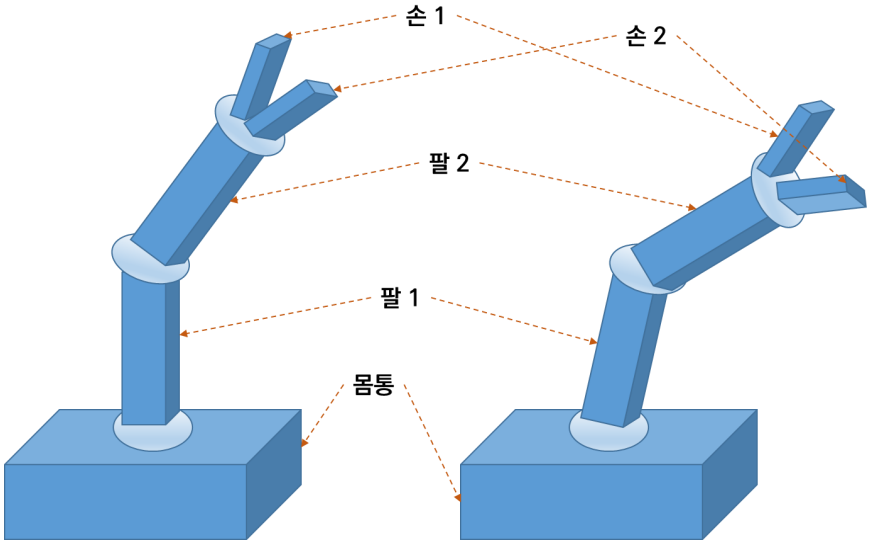
이 장에서 생각할 문제

- ✤ 파일에 담긴 기하객체 정보 읽기
- ✤ 기하객체를 효율적으로 다룰 수 있는 기술
- ✤ 메시 데이터의 가시화를 빠르게 하는 방법

8.1 계층적 모델링이란

컴퓨터 그래픽스에서 다루는 많은 객체들은 구조를 가지고 있다. 이 구조는 인간이나 로봇의 몸이 움직이면 팔이 함께 움직이고, 손을 움직여도 몸은 움직이지 않는 것처럼 각각의 구성 요소가 서로 영향을 미치는 위계가 다를 수 있다. 이러한 현상이 일어나는 이유는 몸에 적용되는 변환transform은 자연스럽게 팔에도 영향을 미치지만, 손에 적용되는 새로운 변환은 몸통에 영향을 미치지 않기 때문이다. 이때 우리는 몸이 팔이나 손의 상위에 있다고 이야기할 수 있다. 이러한 관계를 계층적hierarchical 관계라고 한다.

계층 구조는 트리tree 형태로 표현할 수 있고, 위의 예를 다시 사용하면 몸은 팔이나 손의 상위 노드node가 된다. 계층적 구조를 가진 로봇의 팔을 만들고 제어해 봄으로써 그래픽스에서 계층적 모델링을 어떻게 구현하는지를 살펴보자. 인간형 로봇이 있다고 할 때, 이 로봇은 관절로 연결된 여러 개의 부분 요소로 나뉜다. 이 요소들은 루트root가 있는 트리 구조로 표현할 수 있다. 우선 몸통 위에 굽혀지는 팔이 두 개, 그리고 손이 달린 모델을 만든다고 하자.

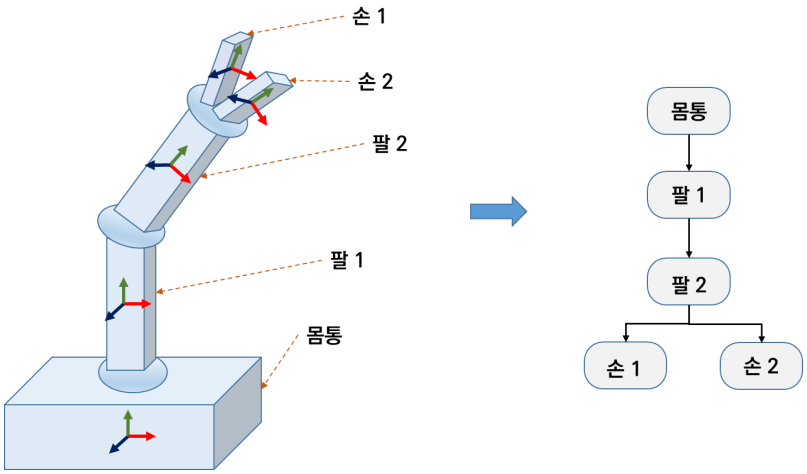


[그림 1] 간단한 로봇 팔 모형

이러한 모델의 움직임을 상상해 보자. 가장 아래 부분에 있는 몸통이 움직이면 다른 모든 요소들이 함께 움직여야 할 것이다. 이것은 팔 1, 팔 2, 그리고 손 1과 손 2를 이루는 요소들이 모두 몸통에 딸린 요소라는 것을 의미한다. 따라서 몸통이 루트 노드인 트리로 이 구조를 표현할 수 있을 것이다. 그런데 나머지 요소들 사이에도 위계를 발견할 수 있을

것이다. 팔 1이 움직이면 팔 2 역시 같이 움직인다. 하지만 팔 2가 움직인다고 해서 팔 1이 영향을 받지는 않는다. 이것은 팔 2가 팔 1의 자식 노드가 된다는 것을 의미한다. 비슷한 관찰을 통해 손 1과 손 2가 팔 2의 자식인 것을 알 수 있다. 그런데 손 1과 손 2는 서로에게 영향을 미치지 않는다. 이것은 손 1과 손 2는 상하관계가 존재하지 않는 관계이며 팔 2의 자식인 형제 관계라는 것을 알 수 있다.

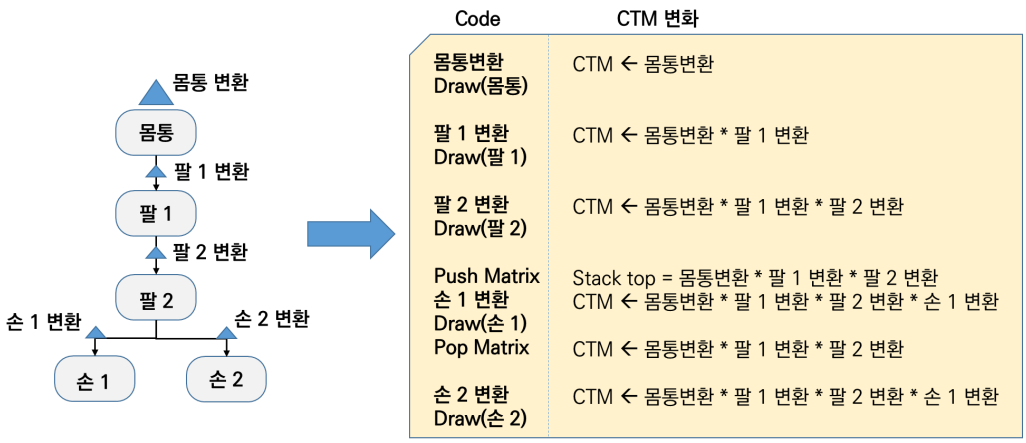
이를 트리 구조로 표현하면 다음과 같다. 로봇 팔을 구성하는 각각의 요소들이 가진 지역좌표계를 로봇 팔 위에 함께 그려 두었다. 앞서 변환을 다루면서 지역좌표계의 이동을 고려하려 변환을 이해하면 코드 상에 나타난 변환의 순서대로 지역좌표계가 계속해서 뒤로 전달되는 것으로 이해하면 된다. 따라서 자식 노드를 만들 때는 부모 노드가 가진 지역좌표계를 옮겨서 자식 노드가 가져야 하는 지역 좌표계로 옮겨 놓으면 된다. 몸통-팔 1-팔 2-손 1까지의 관계는 단순히 부모와 자식 관계가 가지 없이 일렬로 연결된 것과 같다. 따라서 크게 어렵지 않게 구현이 가능하다. 그런데 손 2를 그릴 때는 어떻게 해야 할까?



[그림 2] 계층 구조로 표현된 로봇 모델

손 2를 그릴 때는 손 1의 지역좌표계에서 시작하는 것이 아니라 팔 2의 지역좌표계에서 생각하는 것이 쉽다. 따라서 손 1에 적용된 변환을 무효화시키는 것이 필요하다. 이때 우리가 사용할 수 있는 것이 glPushMatrix와 glPopMatrix라는 것을 앞서 살펴 보았다.

위의 로봇 팔을 구현할 때 각각의 노드들을 옮겨 놓기 위해 필요한 변환들을 삼각형으로 표현하고 이름을 그림 3과 같이 붙였다고 하자. 그러면 그림 3의 오른쪽과 같은 의사 코드를 따라 코드를 작성하여 이를 구현할 수 있을 것이고, 이 과정에 CTM이 어떻게 변화하는지는 의사 코드 오른쪽에 있는 CTM 변화를 통해 확인할 수 있을 것이다.



[그림 3] 로봇 팔 모델의 계층 구조와 이를 구현하는 의사 코드pseudocode

우선 지금까지 다루었던 기술들을 이용하여 몸통 모델을 하나 평면 위에 올리고, 이를 키보드를 이용하여 위치를 옮길 수 있도록 제어해 보자. 먼저 평면 그리기, 축 그리기, 한 변의 길이가 1인 정육면체 그리기를 위한 함수를 다음과 같이 구현해 보자. 이는 앞에서도 다루었던 코드들과 유사하다.

```
def drawPlane():
    n, w = 100, 500
    # n: 체스판 한면의 정점수, w: 체스판 한면의 길이

    d = w / (n-1) # 인접한 두 정점 사이의 간격

    #  체스판 그리기
    glColor3f(0.3,0.5,0)
    glBegin(GL_QUADS)
    for i in range(n):
        for j in range(n):
            if (i+j)%2 == 0:
                startX = -w/2 + i*d
                startZ = -w/2 + j*d
                glVertex3f(startX, 0, startZ)
                glVertex3f(startX, 0, startZ+d)
                glVertex3f(startX+d, 0, startZ+d)
                glVertex3f(startX+d, 0, startZ)
    glEnd()

def drawAxes():
    glBegin(GL_LINES)
```

```
    glColor3f(1,0,0)
    glVertex3f(0,0,0)
    glVertex3f(1,0,0)
    glColor3f(0,1,0)
    glVertex3f(0,0,0)
    glVertex3f(0,1,0)
    glColor3f(0,0,1)
    glVertex3f(0,0,0)
    glVertex3f(0,0,1)
    glEnd()

def drawCube():
    v0 = [-0.5, 0.5, 0.5]
    v1 = [ 0.5, 0.5, 0.5]
    v2 = [ 0.5, 0.5,-0.5]
    v3 = [-0.5, 0.5,-0.5]
    v4 = [-0.5,-0.5, 0.5]
    v5 = [ 0.5,-0.5, 0.5]
    v6 = [ 0.5,-0.5,-0.5]
    v7 = [-0.5,-0.5,-0.5]
    glBegin(GL_LINES)
    glVertex3fv(v0); glVertex3fv(v1)
    glVertex3fv(v1); glVertex3fv(v2)
    glVertex3fv(v2); glVertex3fv(v3)
    glVertex3fv(v3); glVertex3fv(v0)
    glVertex3fv(v4); glVertex3fv(v5)
    glVertex3fv(v5); glVertex3fv(v6)
    glVertex3fv(v6); glVertex3fv(v7)
    glVertex3fv(v7); glVertex3fv(v4)
    glVertex3fv(v0); glVertex3fv(v4)
    glVertex3fv(v1); glVertex3fv(v5)
    glVertex3fv(v2); glVertex3fv(v6)
    glVertex3fv(v3); glVertex3fv(v7)
    glEnd()
    drawAxes()
```

다음은 OpenGL 위짓을 구현해 보자. 여기에는 몸통의 위치를 제어하기 위해 base_position이라는 멤버를 관리할 것이다. 그리고 평면 그리기는 많은 정점을 다루기 때문에 OpenGL 초기화 함수 내에서 디스플레이 리스트로 만들어 두고 필요할 때 호출할 것이다. 그러면 다음과 같이 OpenGL 위짓의 생성자, 초기화, 화면 크기 변경 이벤트 처리 함수를 구현할 수 있을 것이다.

```
class MyGLWidget(QOpenGLWidget):
    def __init__(self, parent=None):
```

```
        super().__init__(parent)
        self.base_position = [0.0,0.0]

    def initializeGL(self):
        glClearColor(0.0, 0.0, 0.0, 1.0)
        self.planeList = glGenLists(1)
        glNewList(self.planeList, GL_COMPILE)
        # 그리기 코드
        drawPlane()
        glEndList()

        glEnable(GL_DEPTH_TEST)


    def resizeGL(self, width, height):
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()
        gluPerspective(60, width/height, 0.01, 100)
```

다음으로 그리기 이벤트 처리 메소드를 구현할 것이다. 지금까지의 방식대로 버퍼를 지우고, 카메라 위치를 찹은 뒤에 평면을 그리고 최초의 좌표계를 나타내는 축을 표시할 것이다.

```
class MyGLWidget(QOpenGLWidget):

    ...

    def paintGL(self):
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        glMatrixMode(GL_MODELVIEW)
        glLoadIdentity()
        gluLookAt(0,7,10, 0,0,0, 0,1,0)

        glCallList(self.planeList)
        drawAxes()
```

다음으로는 높이가 1, 전후와 좌우 폭이 2인 몸통을 그리려고 한다. 이를 위해서는 glScalef(1, 2, 1)을 적용하여 큐브를 변형하면 된다. 앞 장에서 크기 변경이 들어간 변환에서는 이를 glPushMatrix와 glPopMatrix를 이용하여 다른 곳에 영향을 미치지 않도록 하는 것이 좋다고 하였다. 이를 그대로 따르도록 하자. 그러면 이 큐브가 평면 위에 놓이도록 하려면 어떻게 해야 할까? 몸통의 높이가 1이므로 중심을 0.5 만큼 들어 올리면 평면에 놓이게 될 것이다. 이것이 기준 위치가 된다. 키보드를 이용하여 몸통의 위치를

제어하려고 하며, 그 위치는 self.base_position에 x와 z 좌표를 저장하여 사용할 것이다. 따라서 몸통의 위치는 glTranslatef를 이용하여 (self.base_position[0], 0, self.base_position[1]) 만큼 이동해야 한다. 이것을 종합하면 몸통의 위치는 다음과 같은 변환 적용을 통해 원하는 곳에 둘 수 있다.

```
    def paintGL(self):
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

        ...

        ###  Base: 전후 좌우로 이동 가능

        # 제어를 통해 옮겨간 위치
        glTranslatef(self.base_position[0], 0, self.base_position[1])

        glTranslatef(0, 0.5, 0) # 몸통을 평면으로 들어올리는 변환
        glPushMatrix()
        glScalef(2, 1, 2)   # 몸통의 크기 변경
        drawAxes()
        glColor3f(1,1,1)
        drawCube()
        glPopMatrix()
```

윈도우에서는 이러한 OpenGL 위짓을 생성하고, 키보드 제어에 따라 OpenGL 위짓이 가지고 있는 base_position을 변경해 주면 된다.

```
class MyWindow(QMainWindow):
    def __init__(self, title=''):
        QMainWindow.__init__(self)
        self.setWindowTitle(title)
        self.glWidget = MyGLWidget()
        self.setCentralWidget(self.glWidget)

    def keyPressEvent(self, e):

        step = 0.1

        if e.key() == Qt.Key.Key_W:
            self.glWidget.base_position[1] -= step
        elif e.key() == Qt.Key.Key_S:
            self.glWidget.base_position[1] += step
```

```
        elif e.key() == Qt.Key.Key_A:
            self.glWidget.base_position[0] -= step
        elif e.key() == Qt.Key.Key_D:
            self.glWidget.base_position[0] += step

        self.glWidget.update()
```

로봇 팔의 몸통을 평면 위에 두고 키보드를 이용하여 위치 제어하기

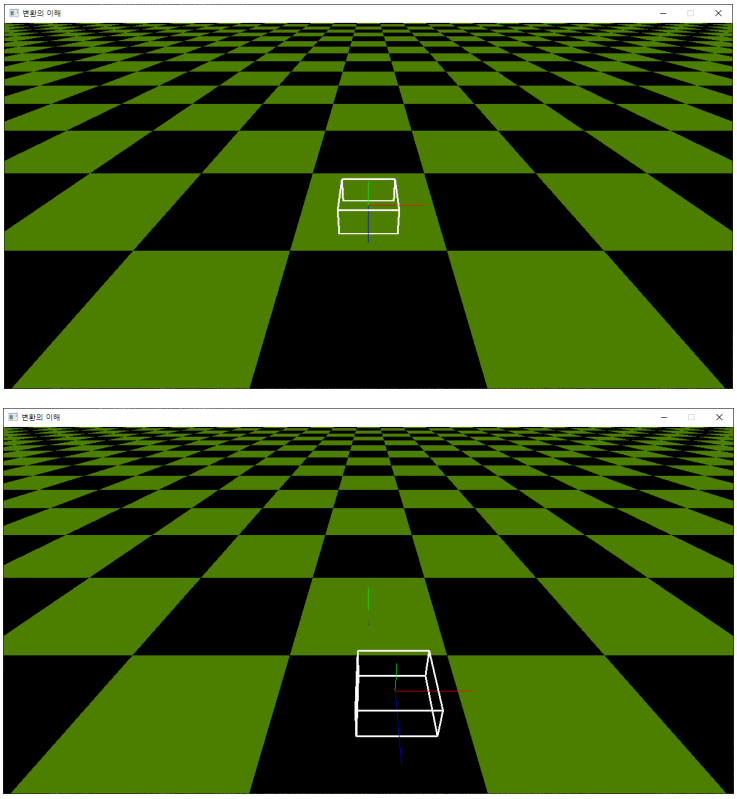
```
from OpenGL.GL import *
from OpenGL.GLU import *
import sys
from PyQt6.QtWidgets import *
from PyQt6.QtOpenGLWidgets import QOpenGLWidget
from PyQt6.QtCore import *

import math
import numpy as np

def drawPlane():
    n, w = 100, 500
    # n: 체스판 한면의 정점수, w: 체스판 한면의 길이

    d = w / (n-1) # 인접한 두 정점 사이의 간격

    #  체스판 그리기
    glColor3f(0.3,0.5,0)
    glBegin(GL_QUADS)
    for i in range(n):
        for j in range(n):
            if (i+j)%2 == 0:
                startX = -w/2 + i*d
                startZ = -w/2 + j*d
                glVertex3f(startX, 0, startZ)
                glVertex3f(startX, 0, startZ+d)
                glVertex3f(startX+d, 0, startZ+d)
                glVertex3f(startX+d, 0, startZ)
    glEnd()

def drawAxes():
    glBegin(GL_LINES)
    glColor3f(1,0,0)
    glVertex3f(0,0,0)
    glVertex3f(1,0,0)
    glColor3f(0,1,0)
```

```
    glVertex3f(0,0,0)
    glVertex3f(0,1,0)
    glColor3f(0,0,1)
    glVertex3f(0,0,0)
    glVertex3f(0,0,1)
    glEnd()

def drawCube():
    v0 = [-0.5, 0.5, 0.5]
    v1 = [ 0.5, 0.5, 0.5]
    v2 = [ 0.5, 0.5,-0.5]
    v3 = [-0.5, 0.5,-0.5]
    v4 = [-0.5,-0.5, 0.5]
    v5 = [ 0.5,-0.5, 0.5]
    v6 = [ 0.5,-0.5,-0.5]
    v7 = [-0.5,-0.5,-0.5]
    glBegin(GL_LINES)
    glVertex3fv(v0); glVertex3fv(v1)
    glVertex3fv(v1); glVertex3fv(v2)
    glVertex3fv(v2); glVertex3fv(v3)
    glVertex3fv(v3); glVertex3fv(v0)
    glVertex3fv(v4); glVertex3fv(v5)
    glVertex3fv(v5); glVertex3fv(v6)
    glVertex3fv(v6); glVertex3fv(v7)
    glVertex3fv(v7); glVertex3fv(v4)
    glVertex3fv(v0); glVertex3fv(v4)
    glVertex3fv(v1); glVertex3fv(v5)
    glVertex3fv(v2); glVertex3fv(v6)
    glVertex3fv(v3); glVertex3fv(v7)
    glEnd()
    drawAxes()


class MyGLWidget(QOpenGLWidget):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.base_position = [0.0,0.0]

    def initializeGL(self):
        glClearColor(0.0, 0.0, 0.0, 1.0)
        self.planeList = glGenLists(1)
        glNewList(self.planeList, GL_COMPILE)
        # 그리기 코드
        drawPlane()
        glEndList()

        glEnable(GL_DEPTH_TEST)
```

```
    def resizeGL(self, width, height):
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()
        gluPerspective(60, width/height, 0.01, 100)

    def paintGL(self):
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        glMatrixMode(GL_MODELVIEW)
        glLoadIdentity()
        gluLookAt(0,7,10, 0,0,0, 0,1,0)

        glCallList(self.planeList)
        drawAxes()

        ###  Base: 전후 좌우로 이동 가능

        # 제어를 통해 옮겨간 위치
        glTranslatef(self.base_position[0], 0, self.base_position[1])

        glTranslatef(0, 0.5, 0) # 몸통을 평면으로 들어올리는 변환
        glPushMatrix()
        glScalef(2, 1, 2)   # 몸통의 크기 변경
        drawAxes()
        glColor3f(1,1,1)
        drawCube()
        glPopMatrix()



class MyWindow(QMainWindow):
    def __init__(self, title=''):
        QMainWindow.__init__(self)
        self.setWindowTitle(title)
        self.glWidget = MyGLWidget()
        self.setCentralWidget(self.glWidget)

    def keyPressEvent(self, e):

        step = 0.1

        if e.key() == Qt.Key.Key_W:
            self.glWidget.base_position[1] -= step
        elif e.key() == Qt.Key.Key_S:
            self.glWidget.base_position[1] += step
        elif e.key() == Qt.Key.Key_A:
            self.glWidget.base_position[0] -= step
        elif e.key() == Qt.Key.Key_D:
```

```
            self.glWidget.base_position[0] += step

        self.glWidget.update()

def main(argv = []):
    app = QApplication(argv)
    window = MyWindow('변환의 이해')
    window.setFixedSize(1200, 600)
    window.show()
    app.exec()

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

이 코드를 실행하면 다음과 같이 W, S, A, D 키를 이용하여 옮기 수 있는 몸통이 나타난다.

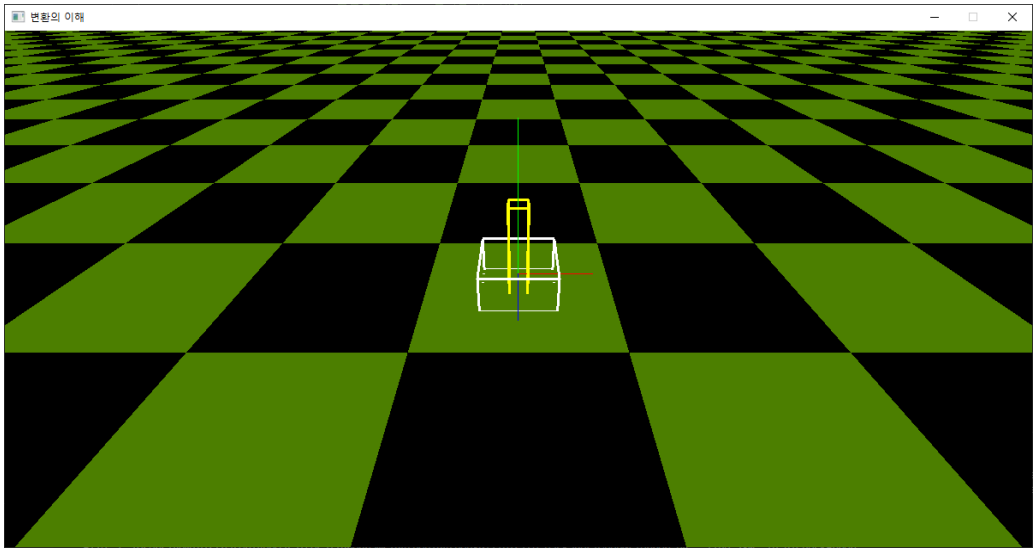


[그림 4] 로봇 팔의 몸통 구현하고 이동하기

이 몸통 위에 높이가 4이고, 앞뒤와 좌우로 폭이 0.5인 팔 1을 구현해 놓아 보자. 이 필은 몸통의 위에서 구부러져야 할 것이다. 이를 위해서는 팔의 중심을 평면 위로 옮긴 뒤에 회전을 시키고, 이후 팔의 아래쪽 끝이 몸통 위에 놓이도록 한다. 이를 구현해 보자. 우선 이 팔은 y 축 기준으로 회전을 하여 방향을 정한 뒤, x 축 기준으로 회전하여 굽힐 수 있도록 하자. 이를 위해 이들 회전의 정도를 저장할 self.arm1Y, self.arm1X라는 멤버 변수를 준비하자. 그리고 우리가 원하는 크기로 큐브를 변형하는 일만 해보자.

```
    def __init__(self, parent=None):
        super().__init__(parent)
        self.base_position = [0.0,0.0]
        ###  팔 1: y축 기준 회전과, x 축 기준 회전이 가능
        self.arm1Y = 30
        self.arm1X = 15

    def paintGL(self):
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

        ...
        ###  Base: 전후 좌우로 이동 가능
        # 제어를 통해 옮겨간 위치
        glTranslatef(self.base_position[0], 0, self.base_position[1])

        glTranslatef(0, 0.5, 0) # 몸통을 평면으로 들어올리는 변환
        glPushMatrix()
        glScalef(2, 1, 2)   # 몸통의 크기 변경
        drawAxes()
        glColor3f(1,1,1)
        drawCube()
        glPopMatrix()

        ###  팔 1: y축 기준 회전과, x 축 기준 회전이 가능
        glPushMatrix()
        glScalef(0.5, 4, 0.5)   # 팔 1의 크기 변경
        drawAxes()
        glColor3f(1,1,0)
        drawCube()
        glPopMatrix()
```

이를 적용하면 그림 5와 같이 회전은 적용되지 않고 크기만 우리가 원하는 방식대로 변형된 팔 1을 얻게 될 것이다.

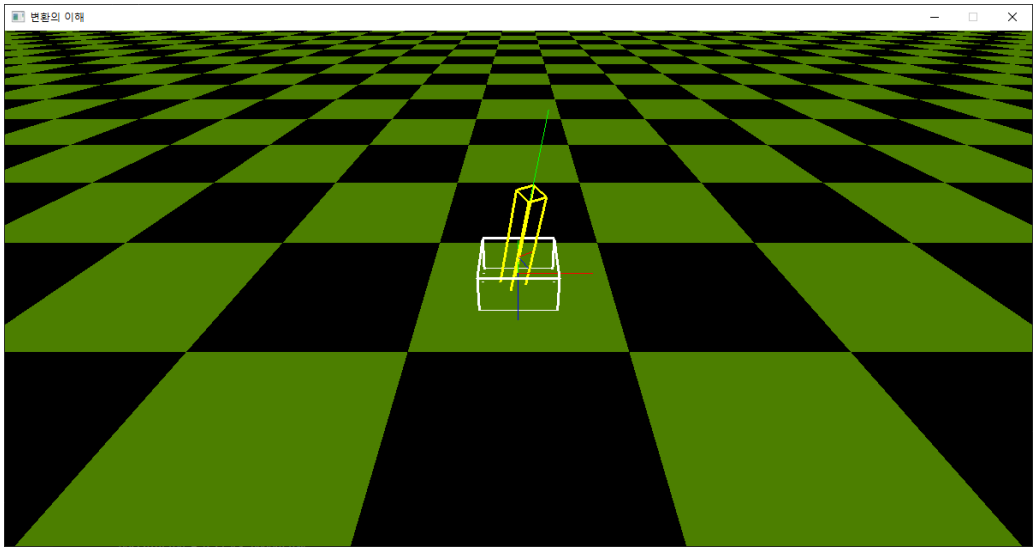


[그림 5] 원하는 크기로 변형된 팔 1

이것의 회전은 이 객체의 지역좌표계를 중심으로 이루어지기 때문에, 지역좌표계를 먼저 관절의 위치로 옮겨 놓는 일이 필요하다. 이를 위해서는 몸통의 높이인 1의 반인 0.5 만큼 팔 1을 y 축 방향으로 올려 놓아야 한다. 그리고 y축 회전과 x축 회전을 차례로 적용해 보자.

```
        ###  팔 1: y축 기준 회전과, x 축 기준 회전이 가능
        glTranslatef(0, 0.5, 0) # 몸통 높이의 반 이동
        glRotatef(self.arm1Y, 0, 1, 0)
        glRotatef(self.arm1X, 1, 0, 0)
        glPushMatrix()
        glScalef(0.5, 4, 0.5)   # 팔 1의 크기 변경
        drawAxes()
        glColor3f(1,1,0)
        drawCube()
        glPopMatrix()
```

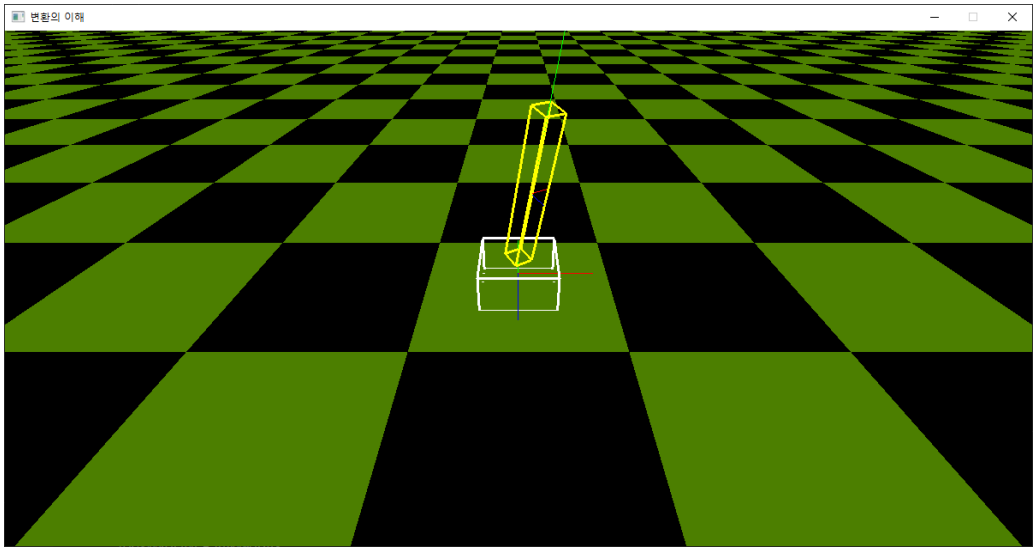
팔 1의 회전 중심이 몸통의 윗쪽 면에 놓이고, 회전이 이루어진 것을 확인할 수 있을 것이다.



[그림 6] 팔 1의 중심이 관절 위치로 옮겨지고 회전이 적용된 결과 1

팔 1의 회전 중심이 몸통의 윗쪽 면에 놓이고, 회전이 이루어진 것을 확인할 수 있을 것이다. 다음으로 이 팔 1을 y 축 방향으로 더 끌어올려 아래쪽 면이 몸통 위의 관절 위치까지 와야 할 것이다. 이때 올라와야 하는 정도는 팔 1의 높이인 4의 반 만큼이 될 것이다. 즉 y 축으로 2 만큼 올리면 된다.

```
        ###  팔 1: y축 기준 회전과, x 축 기준 회전이 가능
        glTranslatef(0, 0.5, 0) # 몸통 높이의 반 이동
        glRotatef(self.arm1Y, 0, 1, 0)
        glRotatef(self.arm1X, 1, 0, 0)
        glTranslatef(0, 2, 0) # 팔 1의 높이 반 이동
        glPushMatrix()
        glScalef(0.5, 4, 0.5)   # 팔 1의 크기 변경
        drawAxes()
        glColor3f(1,1,0)
        drawCube()
        glPopMatrix()
```



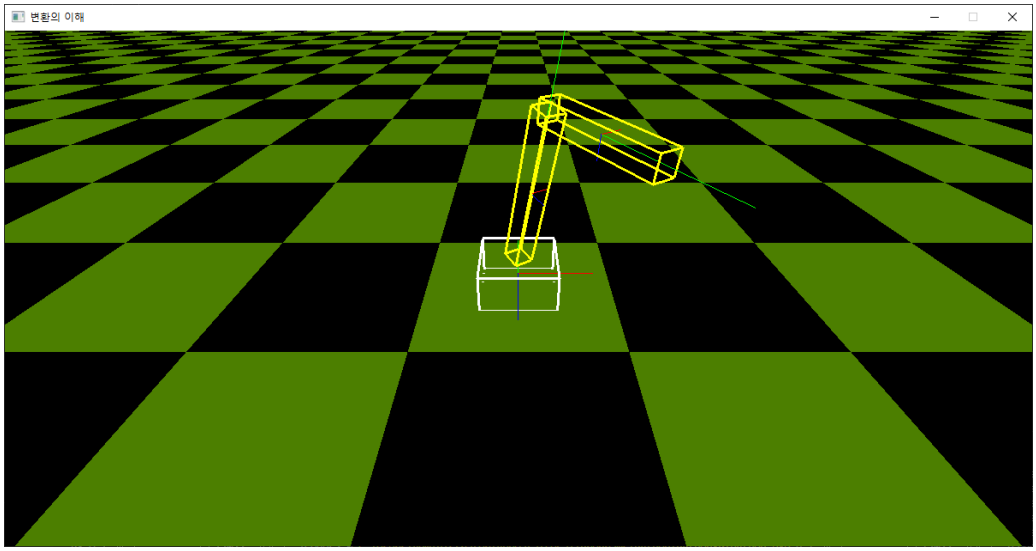
[그림 7] 팔 1을 들어올려 아래쪽 끝이 관절 위치에 놓이게 조정한 결과

다음으로 팔 2를 구현해 보자. 팔 2는 팔 1의 자식이므로 이를 기준으로 생각하면 된다. 우선 회전은 x축 기준으로만 이루어진다고 하자. 이때 회전의 정도는 self.arm2X로 관리하자. 팔 2를 팔 1 위에 다는 것은 팔 1을 몸통 위에 다는 것과 동일하다. 따라서 다음과 같이 구현할 수 있을 것이다.

```
    def __init__(self, parent=None):
        super().__init__(parent)
        self.base_position = [0.0,0.0]
        ###  팔 1: y축 기준 회전과, x 축 기준 회전이 가능
        self.arm1Y = 30
        self.arm1X = 15
        self.arm2X = 80

    def paintGL(self):
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

        ...
        ###  Base: 전후 좌우로 이동 가능
        # 제어를 통해 옮겨간 위치
        glTranslatef(self.base_position[0], 0, self.base_position[1])

        glTranslatef(0, 0.5, 0) # 몸통을 평면으로 들어올리는 변환
        glPushMatrix()
        glScalef(2, 1, 2)   # 몸통의 크기 변경
        drawAxes()
```

```
        glColor3f(1,1,1)
        drawCube()
        glPopMatrix()

        ###  팔 1: y축 기준 회전과, x 축 기준 회전이 가능
        glTranslatef(0, 0.5, 0) # 몸통 높이의 반 이동
        glRotatef(self.arm1Y, 0, 1, 0)
        glRotatef(self.arm1X, 1, 0, 0)
        glTranslatef(0, 2, 0) # 팔 1의 높이 반 이동
        glPushMatrix()
        glScalef(0.5, 4, 0.5)   # 팔 1의 크기 변경
        drawAxes()
        glColor3f(1,1,0)
        drawCube()
        glPopMatrix()

        ###  팔 2: x 축 기준 회전이 가능
        glTranslatef(0, 2, 0) # 팔 1 높이의 반 이동
        glRotatef(self.arm2X, 1, 0, 0)
        glTranslatef(0, 1.5, 0) # 팔 2의 높이 반 이동
        glPushMatrix()
        glScalef(0.5, 3, 0.5)   # 팔 2의 크기 변경
        drawAxes()
        glColor3f(1,1,0)
        drawCube()
        glPopMatrix()
```

팔 2의 크기는 높이가 3이 되도록 하였다. 따라서 팔 1의 높이 4의 반인 2 만큼 올린 뒤에 회전을 적용하고, 팔 2 자신의 높이 3의 반인 1.5를 높이면 될 것이다. 이를 실행하면 다음과 같이 팔 2가 팔 1에 관절로 연결된 형태를 얻게 된다.



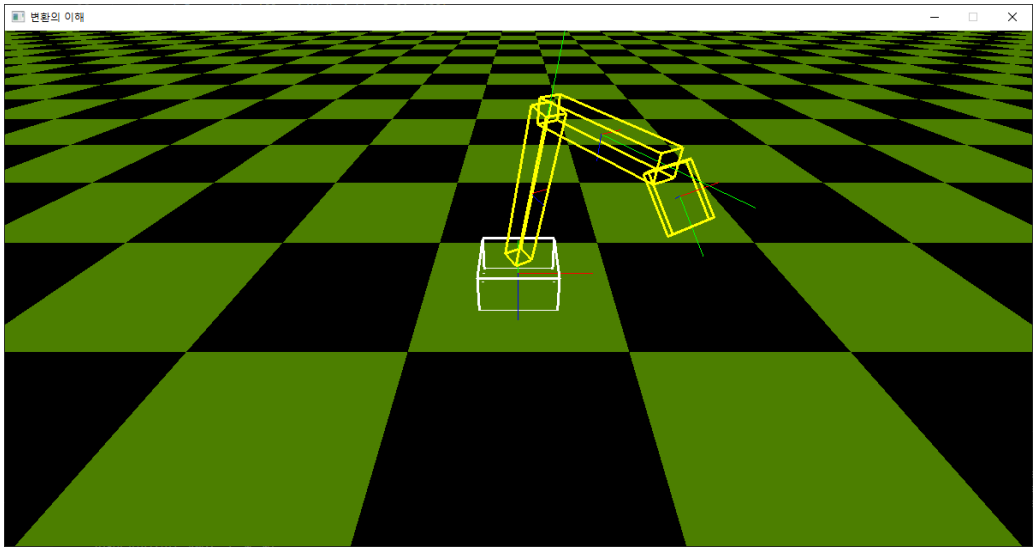
[그림 8] 팔 2를 추가한 결과

다음으로 손 1을 달 것이다. 이는 지금까지 해 온 방식과 동일하다. 손 1의 크기는 x 축 방향으로 1, y축 높이는 1, z 축 두께는 0.1 정도로 두자. 그리고 손이 벌어진 정도를 self.handAngle로 관리하면 다음과 같이 구현할 수 있을 것이다.

```
    def __init__(self, parent=None):
        super().__init__(parent)
        self.base_position = [0.0,0.0]
        ###  팔 1: y축 기준 회전과, x 축 기준 회전이 가능
        self.arm1Y = 30
        self.arm1X = 15
        self.arm2X = 80
        self.handAngle = 45

    def paintGL(self):

        ...

        ###  손 1: x 축 기준 회전이 가능
        glTranslatef(0, 1.5, 0) # 팔 2 높이의 반 이동
        glRotatef(self.handAngle, 1, 0, 0)
        glTranslatef(0, 0.5, 0) # 손 1 높이의 반 이동
        glPushMatrix()
        glScalef(1, 1, 0.1)   # 손 1의 크기 변경
        drawAxes()
```

```
        glColor3f(1,1,0)
        drawCube()
        glPopMatrix()
```

그러면 다음과 같이 손 1을 그릴 수 있다.



[그림 9] 손 1을 추가한 결과

그런데 손 2를 그릴 때에는 손 1이 그려진 위치에서 생각하면 안 된다. 손 1을 그리기 이전이 팔 2가 손 2의 부모이기 때문이다. 따라서 손 1에 적용된 변환을 모두 없애 버려야 한다. 이를 위해서 손 1을 그리기 전에 glPushMatrix를 하고, 다 그린 뒤에는 glPopMatrix를 하여 팔 2의 변환 상태로 돌아가야 할 것이다. 따라서 코드를 수정하면 다음과 같다.

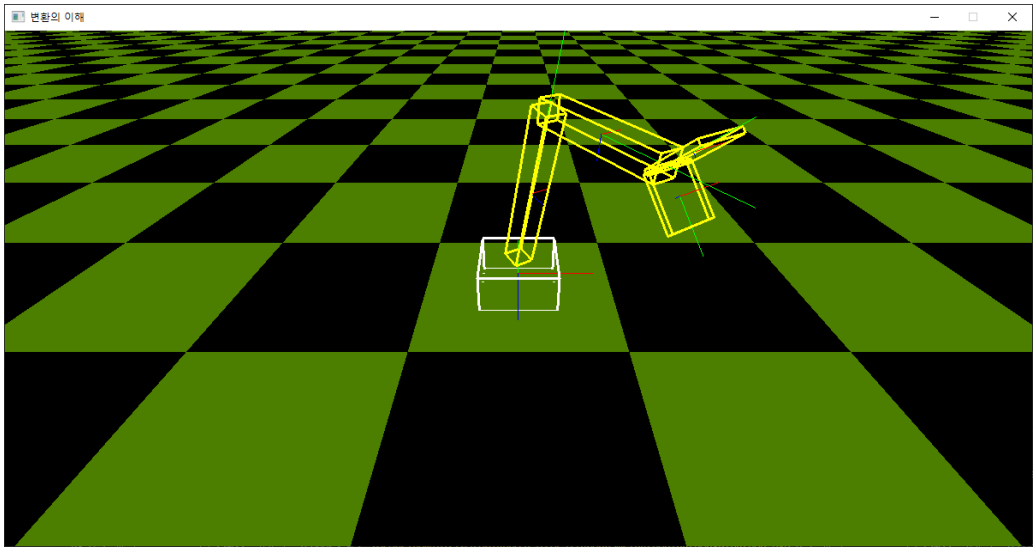
```
    def paintGL(self):
        ...

        ###  손 1: x 축 기준 회전이 가능
        glPushMatrix() ############## 손 1 변환 이전 상태 기록
        glTranslatef(0, 1.5, 0) # 팔 2 높이의 반 이동
        glRotatef(self.handAngle, 1, 0, 0)
        glTranslatef(0, 0.5, 0) # 손 1 높이의 반 이동
        glPushMatrix()
```

```
        glScalef(1, 1, 0.1)   # 손 1의 크기 변경
        drawAxes()
        glColor3f(1,1,0)
        drawCube()
        glPopMatrix()
        glPopMatrix() ################## 손 1 변환 무효화
```

그리고 손 2를 self.handAngle의 반대로 회전시켜 만들어 보자.

```
    def paintGL(self):
        ...

        ###  손 1: x 축 기준 회전이 가능
        glPushMatrix() ############## 손 1 변환 이전 상태 기록
        glTranslatef(0, 1.5, 0) # 팔 2 높이의 반 이동
        glRotatef(self.handAngle, 1, 0, 0)
        glTranslatef(0, 0.5, 0) # 손 1 높이의 반 이동
        glPushMatrix()
        glScalef(1, 1, 0.1)   # 손 1의 크기 변경
        drawAxes()
        glColor3f(1,1,0)
        drawCube()
        glPopMatrix()
        glPopMatrix() ################## 손 1 변환 무효화

        ###  손 2: x 축 기준 회전이 가능
        glPushMatrix() ############## 손 2 변환 이전 상태 기록
        glTranslatef(0, 1.5, 0) # 팔 2 높이의 반 이동
        glRotatef(-self.handAngle, 1, 0, 0)
        glTranslatef(0, 0.5, 0) # 손 1 높이의 반 이동
        glPushMatrix()
        glScalef(1, 1, 0.1)   # 손 1의 크기 변경
        drawAxes()
        glColor3f(1,1,0)
        drawCube()
        glPopMatrix()
        glPopMatrix() ################## 손 2 변환 무효화
```

이를 실행하면 그림 10과 같이 손 2가 제대로 그려질 것이다.



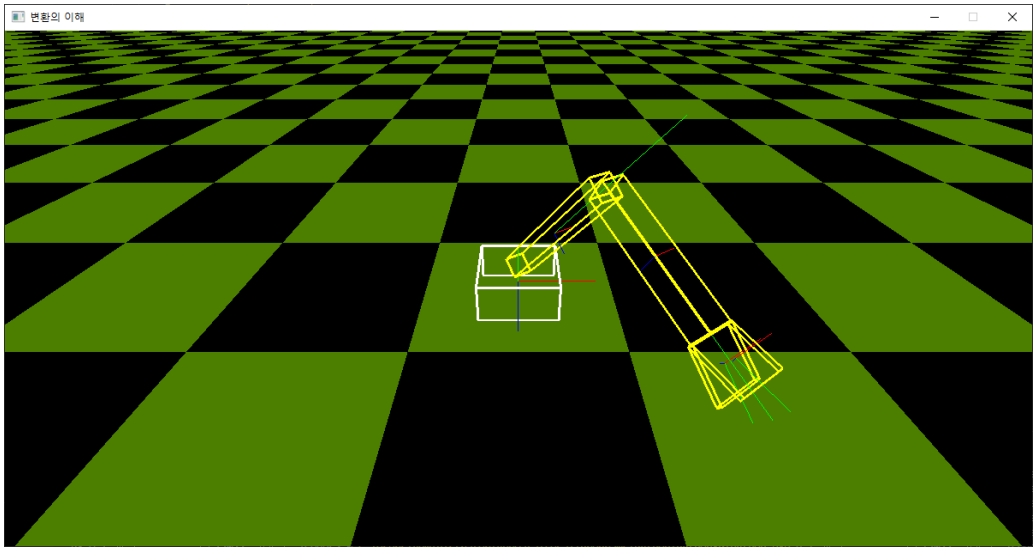
[그림 10] 로봇 팔의 완성

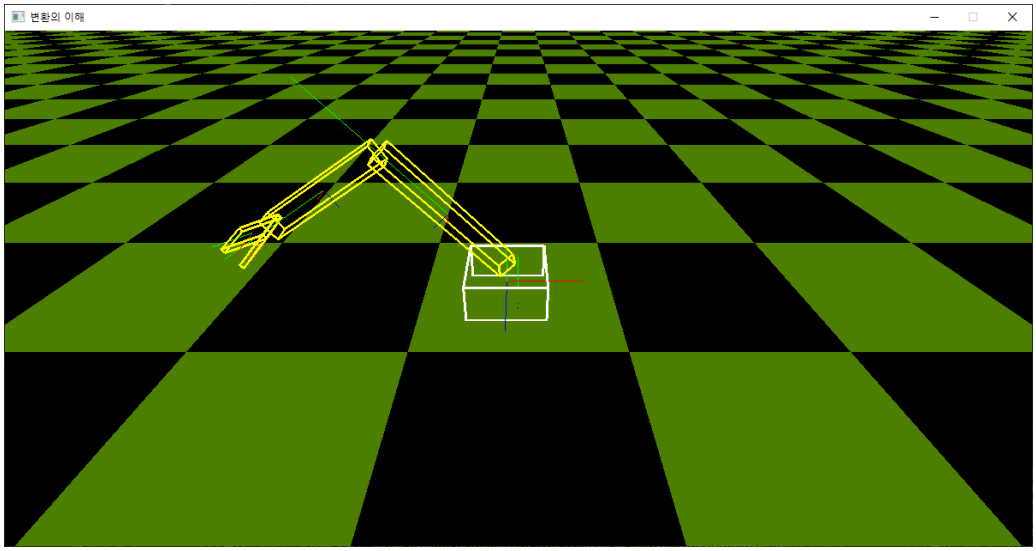
이제 우리는 Q, E 키를 이용하여 로봇 팔이 향하는 방향을 제어하고, 1, 2 키를 이용하여 팔 1을 들어 올리거나 내릴 수 있게 하고, 3, 4 키로 팔 2를 올리고 내리도록 하자. 그리고 Z, X 키를 이용하여 손을 오므리고 펼치도록 만들어 보자. 이것은 우리가 만들어 놓은 회전각 멤버 변수만 변경하면 된다. 이를 제어하는 코드는 윈도 위짓의 키보드 이벤트 처리 메소드를 다음과 같이 변경하면 된다.

```
class MyWindow(QMainWindow):
    def __init__(self, title=''):
        QMainWindow.__init__(self)
        self.setWindowTitle(title)
        self.glWidget = MyGLWidget()
        self.setCentralWidget(self.glWidget)

    def keyPressEvent(self, e):

        step = 0.1
        angle_step = 1

        if e.key() == Qt.Key.Key_W:
            self.glWidget.base_position[1] -= step
        elif e.key() == Qt.Key.Key_S:
            self.glWidget.base_position[1] += step
        elif e.key() == Qt.Key.Key_A:
```

```
            self.glWidget.base_position[0] -= step
        elif e.key() == Qt.Key.Key_D:
            self.glWidget.base_position[0] += step

        elif e.key() == Qt.Key.Key_Q:
            self.glWidget.arm1Y -= angle_step
        elif e.key() == Qt.Key.Key_E:
            self.glWidget.arm1Y += angle_step
        elif e.key() == Qt.Key.Key_1:
            self.glWidget.arm1X -= angle_step
        elif e.key() == Qt.Key.Key_2:
            self.glWidget.arm1X += angle_step
        elif e.key() == Qt.Key.Key_3:
            self.glWidget.arm2X -= angle_step
        elif e.key() == Qt.Key.Key_4:
            self.glWidget.arm2X += angle_step
        elif e.key() == Qt.Key.Key_Z:
            self.glWidget.handAngle -= angle_step
        elif e.key() == Qt.Key.Key_X:
            self.glWidget.handAngle += angle_step

        self.glWidget.update()
```

이를 추가하여 실행하면 다음과 같이 로봇 팔을 제어할 수 있을 것이다.





[그림 11] 다양한 방식으로 제어된 로봇 팔의 모습

완성된 로봇팔 구현

```
from OpenGL.GL import *
from OpenGL.GLU import *
import sys
from PyQt6.QtWidgets import *
from PyQt6.QtOpenGLWidgets import QOpenGLWidget
from PyQt6.QtCore import *

import math
import numpy as np

def drawPlane():
    n, w = 100, 500
    # n: 체스판 한면의 정점수, w: 체스판 한면의 길이

    d = w / (n-1) # 인접한 두 정점 사이의 간격

    #  체스판 그리기
    glColor3f(0.3,0.5,0)
    glBegin(GL_QUADS)
    for i in range(n):
        for j in range(n):
            if (i+j)%2 == 0:
```

```
                startX = -w/2 + i*d
                startZ = -w/2 + j*d
                glVertex3f(startX, 0, startZ)
                glVertex3f(startX, 0, startZ+d)
                glVertex3f(startX+d, 0, startZ+d)
                glVertex3f(startX+d, 0, startZ)
    glEnd()

def drawAxes():
    glBegin(GL_LINES)
    glColor3f(1,0,0)
    glVertex3f(0,0,0)
    glVertex3f(1,0,0)
    glColor3f(0,1,0)
    glVertex3f(0,0,0)
    glVertex3f(0,1,0)
    glColor3f(0,0,1)
    glVertex3f(0,0,0)
    glVertex3f(0,0,1)
    glEnd()

def drawCube():
    v0 = [-0.5, 0.5, 0.5]
    v1 = [ 0.5, 0.5, 0.5]
    v2 = [ 0.5, 0.5,-0.5]
    v3 = [-0.5, 0.5,-0.5]
    v4 = [-0.5,-0.5, 0.5]
    v5 = [ 0.5,-0.5, 0.5]
    v6 = [ 0.5,-0.5,-0.5]
    v7 = [-0.5,-0.5,-0.5]
    glLineWidth(3)
    glBegin(GL_LINES)
    glVertex3fv(v0); glVertex3fv(v1)
    glVertex3fv(v1); glVertex3fv(v2)
    glVertex3fv(v2); glVertex3fv(v3)
    glVertex3fv(v3); glVertex3fv(v0)
    glVertex3fv(v4); glVertex3fv(v5)
    glVertex3fv(v5); glVertex3fv(v6)
    glVertex3fv(v6); glVertex3fv(v7)
    glVertex3fv(v7); glVertex3fv(v4)
    glVertex3fv(v0); glVertex3fv(v4)
    glVertex3fv(v1); glVertex3fv(v5)
    glVertex3fv(v2); glVertex3fv(v6)
    glVertex3fv(v3); glVertex3fv(v7)
    glEnd()
    glLineWidth(1)
    drawAxes()
```

```
class MyGLWidget(QOpenGLWidget):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.base_position = [0.0,0.0]
        ###  팔 1: y축 기준 회전과, x 축 기준 회전이 가능
        self.arm1Y = 35
        self.arm1X = 15
        self.arm2X = 80
        self.handAngle = 45

    def initializeGL(self):
        glClearColor(0.0, 0.0, 0.0, 1.0)
        self.planeList = glGenLists(1)
        glNewList(self.planeList, GL_COMPILE)
        # 그리기 코드
        drawPlane()
        glEndList()

        glEnable(GL_DEPTH_TEST)


    def resizeGL(self, width, height):
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()
        gluPerspective(60, width/height, 0.01, 100)

    def paintGL(self):
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        glMatrixMode(GL_MODELVIEW)
        glLoadIdentity()
        gluLookAt(0,7,10, 0,0,0, 0,1,0)

        glCallList(self.planeList)
        drawAxes()

        ###  Base: 전후 좌우로 이동 가능

        # 제어를 통해 옮겨간 위치
        glTranslatef(self.base_position[0], 0, self.base_position[1])

        glTranslatef(0, 0.5, 0) # 몸통을 평면으로 들어올리는 변환
        glPushMatrix()
        glScalef(2, 1, 2)   # 몸통의 크기 변경
        drawAxes()
        glColor3f(1,1,1)
        drawCube()
        glPopMatrix()
```

```
    ###  팔 1: y축 기준 회전과, x 축 기준 회전이 가능
    glTranslatef(0, 0.5, 0) # 몸통 높이의 반 이동
    glRotatef(self.arm1Y, 0, 1, 0)
    glRotatef(self.arm1X, 1, 0, 0)
    glTranslatef(0, 2, 0) # 팔 1의 높이 반 이동
    glPushMatrix()
    glScalef(0.5, 4, 0.5)   # 팔 1의 크기 변경
    drawAxes()
    glColor3f(1,1,0)
    drawCube()
    glPopMatrix()

    ###  팔 2: x 축 기준 회전이 가능
    glTranslatef(0, 2, 0) # 팔 1 높이의 반 이동
    glRotatef(self.arm2X, 1, 0, 0)
    glTranslatef(0, 1.5, 0) # 팔 2의 높이 반 이동
    glPushMatrix()
    glScalef(0.5, 3, 0.5)   # 팔 2의 크기 변경
    drawAxes()
    glColor3f(1,1,0)
    drawCube()
    glPopMatrix()

    ###  손 1: x 축 기준 회전이 가능
    glPushMatrix() ############## 손 1 변환 이전 상태 기록
    glTranslatef(0, 1.5, 0) # 팔 2 높이의 반 이동
    glRotatef(self.handAngle, 1, 0, 0)
    glTranslatef(0, 0.5, 0) # 손 1 높이의 반 이동
    glPushMatrix()
    glScalef(1, 1, 0.1)   # 손 1의 크기 변경
    drawAxes()
    glColor3f(1,1,0)
    drawCube()
    glPopMatrix()
    glPopMatrix() ################## 손 1 변환 무효화

    ###  손 2: x 축 기준 회전이 가능
    glPushMatrix() ############## 손 2 변환 이전 상태 기록
    glTranslatef(0, 1.5, 0) # 팔 2 높이의 반 이동
    glRotatef(-self.handAngle, 1, 0, 0)
    glTranslatef(0, 0.5, 0) # 손 1 높이의 반 이동
    glPushMatrix()
    glScalef(1, 1, 0.1)   # 손 1의 크기 변경
    drawAxes()
    glColor3f(1,1,0)
    drawCube()
    glPopMatrix()
```

```
        glPopMatrix() ################## 손 2 변환 무효화

class MyWindow(QMainWindow):
    def __init__(self, title=''):
        QMainWindow.__init__(self)
        self.setWindowTitle(title)
        self.glWidget = MyGLWidget()
        self.setCentralWidget(self.glWidget)

    def keyPressEvent(self, e):

        step = 0.1
        angle_step = 1

        if e.key() == Qt.Key.Key_W:
            self.glWidget.base_position[1] -= step
        elif e.key() == Qt.Key.Key_S:
            self.glWidget.base_position[1] += step
        elif e.key() == Qt.Key.Key_A:
            self.glWidget.base_position[0] -= step
        elif e.key() == Qt.Key.Key_D:
            self.glWidget.base_position[0] += step

        elif e.key() == Qt.Key.Key_Q:
            self.glWidget.arm1Y -= angle_step
        elif e.key() == Qt.Key.Key_E:
            self.glWidget.arm1Y += angle_step
        elif e.key() == Qt.Key.Key_1:
            self.glWidget.arm1X -= angle_step
        elif e.key() == Qt.Key.Key_2:
            self.glWidget.arm1X += angle_step
        elif e.key() == Qt.Key.Key_3:
            self.glWidget.arm2X -= angle_step
        elif e.key() == Qt.Key.Key_4:
            self.glWidget.arm2X += angle_step
        elif e.key() == Qt.Key.Key_Z:
            self.glWidget.handAngle -= angle_step
        elif e.key() == Qt.Key.Key_X:
            self.glWidget.handAngle += angle_step

        self.glWidget.update()

def main(argv = []):
    app = QApplication(argv)
    window = MyWindow('변환의 이해')
    window.setFixedSize(1200, 600)
    window.show()
    app.exec()
```

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