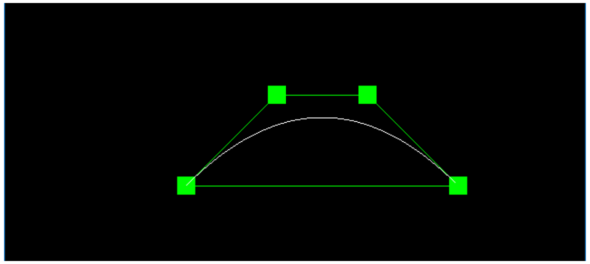
Daily Assignment 24

- Start from the solution of *Daily Assignment 23*, modify this program to draw a **Bezier curve** instead of a Hermite curve

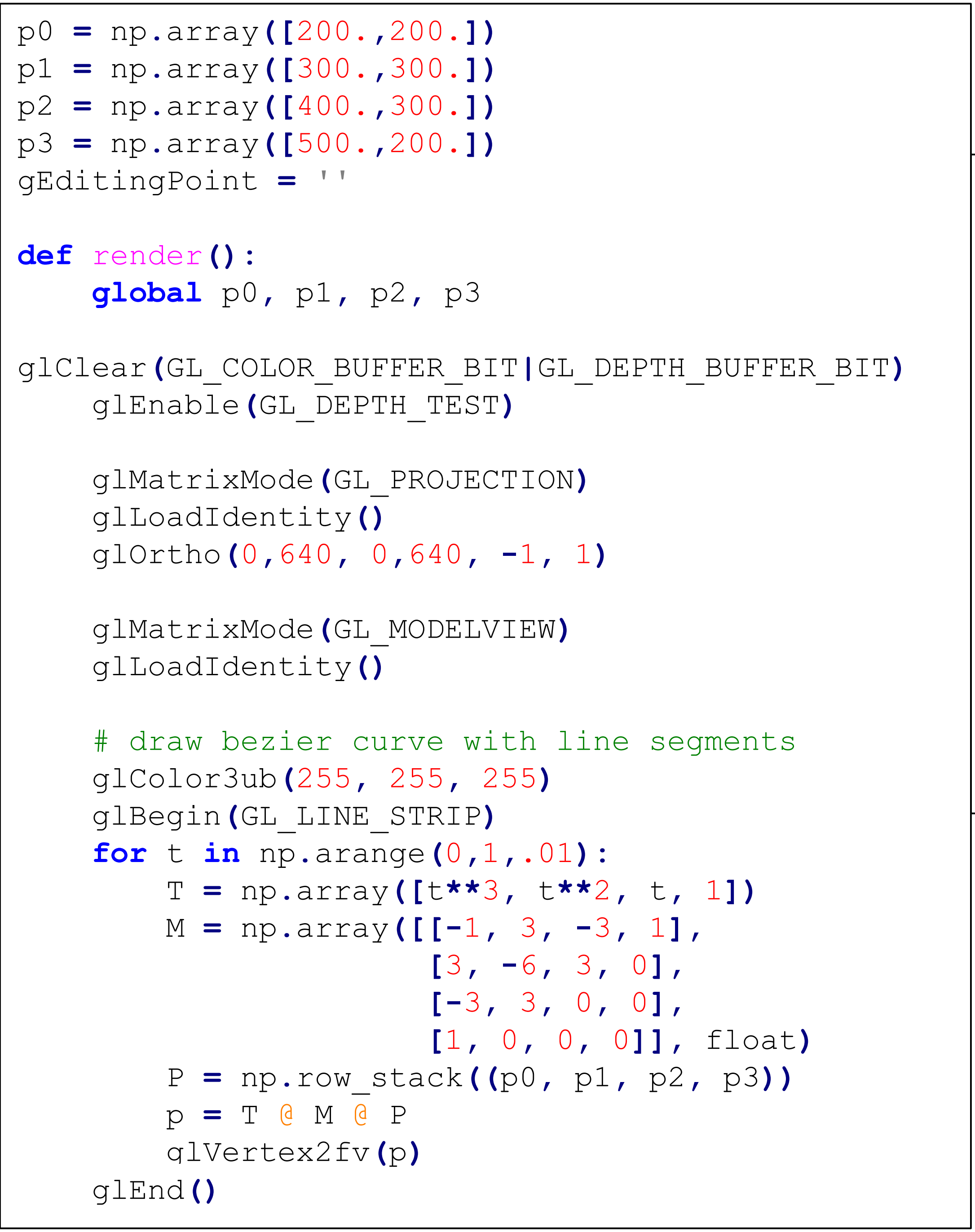
- Control points p0, p1, p2, p3 should be draggable and rendered in green

- Draw the edges of the control polygon in green as well

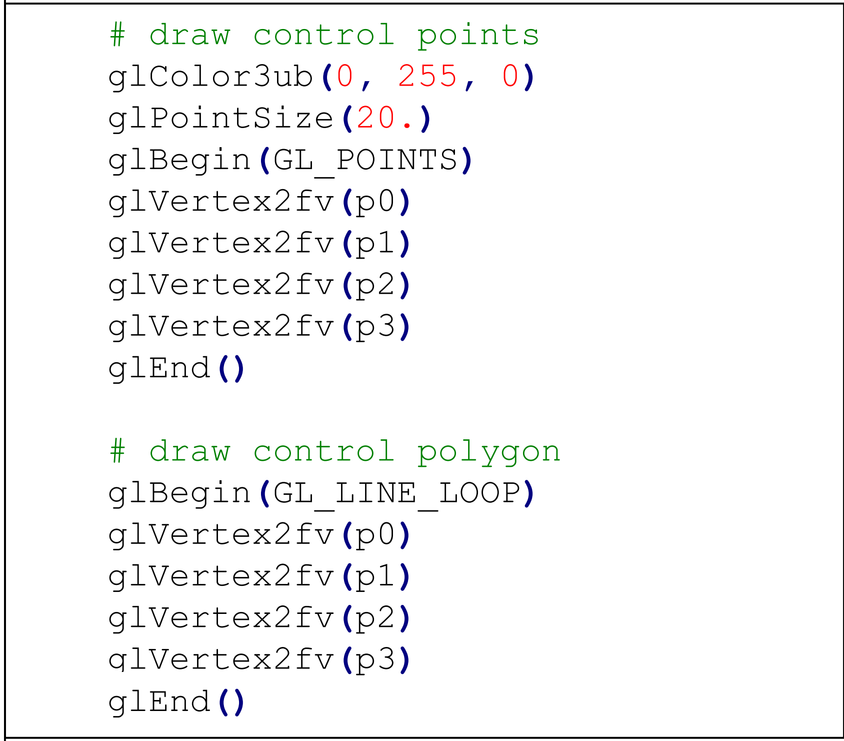


```
# initial values
p0 = np.array([200.,200.])
p1 = np.array([300.,350.])
p2 = np.array([500.,550.])
p3 = np.array([400.,400.])
```

- You can use any method to compute & draw the Bezier curve



```python
p0 = np.array([200.,200.])
p1 = np.array([300.,300.])
p2 = np.array([400.,300.])
p3 = np.array([500.,200.])
gEditingPoint = ''

def render():
    global p0, p1, p2, p3

glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
    glEnable(GL_DEPTH_TEST)

    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glOrtho(0,640, 0,640, -1, 1)

    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

    # draw bezier curve with line segments
    glColor3ub(255, 255, 255)
    glBegin(GL_LINE_STRIP)
    for t in np.arange(0,1,.01):
        T = np.array([t**3, t**2, t, 1])
        M = np.array([[-1, 3, -3, 1],
                      [3, -6, 3, 0],
                      [-3, 3, 0, 0],
                      [1, 0, 0, 0]], float)
        P = np.row_stack((p0, p1, p2, p3))
        p = T @ M @ P
        glVertex2fv(p)
    glEnd()
```



```python
    # draw control points
    glColor3ub(0, 255, 0)
    glPointSize(20.)
    glBegin(GL_POINTS)
    glVertex2fv(p0)
    glVertex2fv(p1)
    glVertex2fv(p2)
    glVertex2fv(p3)
    glEnd()

    # draw control polygon
    glBegin(GL_LINE_LOOP)
    glVertex2fv(p0)
    glVertex2fv(p1)
    glVertex2fv(p2)
    glVertex2fv(p3)
    glEnd()
```

```
def button_callback(window, button, action, mod):
    global p0, p1, p2, p3
    global gEditingPoint
    if button==glfw.MOUSE_BUTTON_LEFT:
        x, y = glfw.get_cursor_pos(window)
        y = 640 - y
        if action==glfw.PRESS:
            if np.abs(x-p0[0])<10 and np.abs(y-p0[1])<10:
                gEditingPoint = 'p0'
            elif np.abs(x-p1[0])<10 and np.abs(y-p1[1])<10:
                gEditingPoint = 'p1'
            elif np.abs(x-p2[0])<10 and np.abs(y-p2[1])<10:
                gEditingPoint = 'p2'
            elif np.abs(x-p3[0])<10 and np.abs(y-p3[1])<10:
                gEditingPoint = 'p3'
        elif action==glfw.RELEASE:
            gEditingPoint = ''

def cursor_callback(window, xpos, ypos):
    global p0, p1, p2, p3
    global gEditingPoint
    ypos = 640 - ypos
    if gEditingPoint=='p0':
        p0[0]=xpos; p0[1]=ypos
    elif gEditingPoint=='p1':
        p1[0]=xpos; p1[1]=ypos
    elif gEditingPoint=='p2':
        p2[0]=xpos; p2[1]=ypos
    elif gEditingPoint=='p3':
        p3[0]=xpos; p3[1]=ypos
```