FACULTY OF

COMPUTING SEMESTER

1/20232024

SECJ3553 – GEOMETRIC MODELING

SECTION 01

ASSIGNMENT 3

Bezier/B-Spline Curve

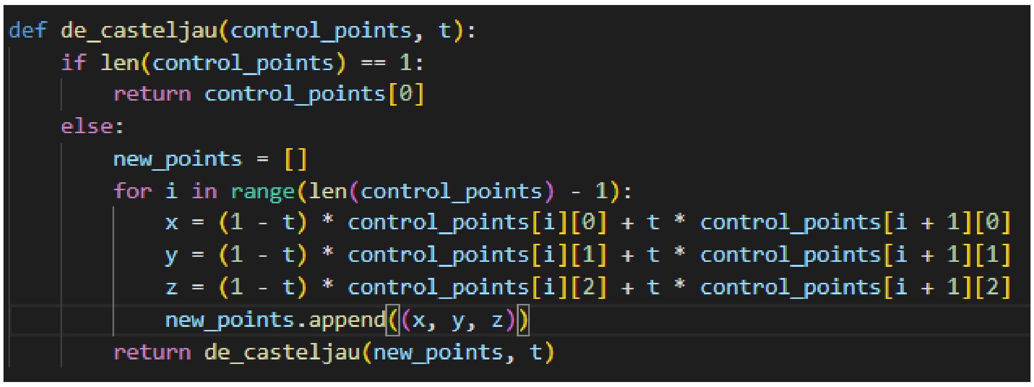
LECTURER: DR. NORHAIDA BTE MOHD SUAIB

NAME	MATRIC NO
GAN HENG LAI	A21EC0176
NG KAI ZHENG	A21EC0101
LEW CHIN HONG	A21EC0044
YEO CHUN TECK	A21EC0148

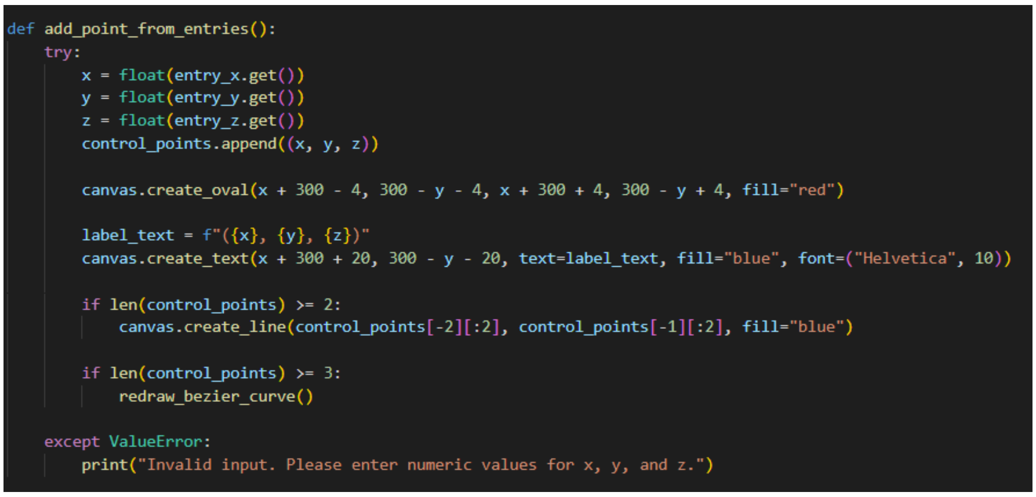
How to control the shape of the generated surface

Firstly, we leverage the OpenCV library to implement an interactive 3D modelling tool. In our proposed solution code, the Bezier 3D surface is generated based on the control points using De Casteljau's algorithm. To control the shape of the generated surface, the user needs to define three control point values by entering the points' x, y and z coordinates value in the provided entry widgets and then clicking the "Add Point" button. This is achieved by the "add_point_from_entries" function with association with the Tkinter library function like "tk.Entry" and "tk.button".

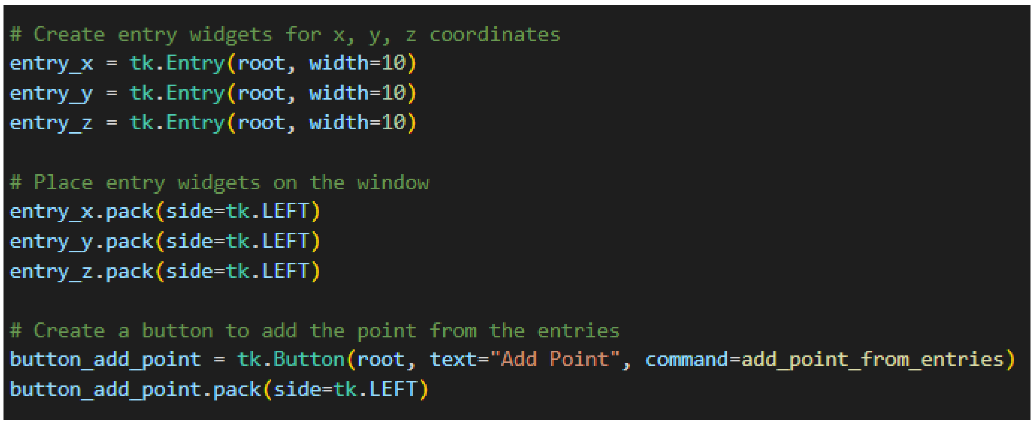
When there are only two control points, only a line will be drawn. When the control points are greater or equal to three, a bezier curve will be drawn by calling the function "redraw_bezier_curve"



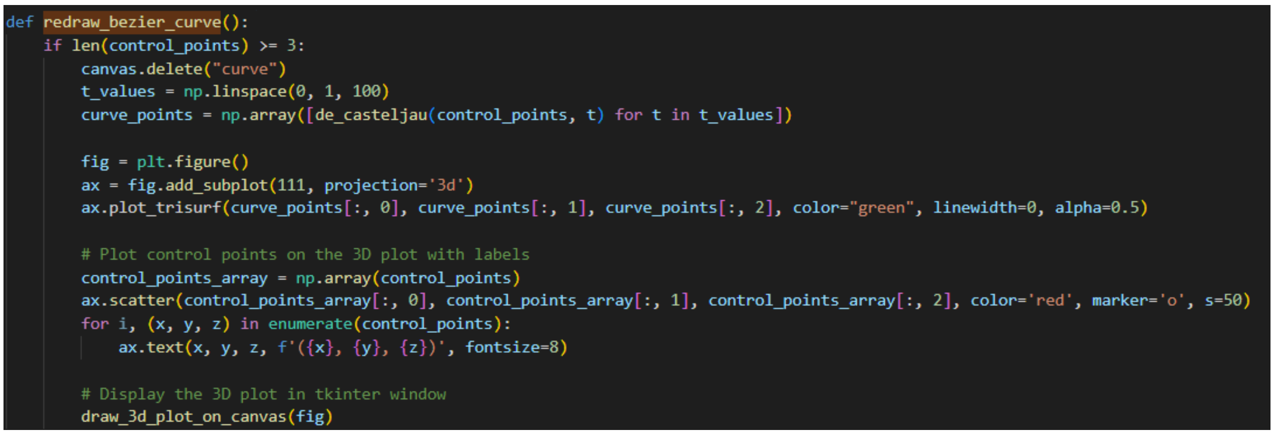
```python
def de_casteljau(control_points, t):
    if len(control_points) == 1:
        return control_points[0]
    else:
        new_points = []
        for i in range(len(control_points) - 1):
            x = (1 - t) * control_points[i][0] + t * control_points[i + 1][0]
            y = (1 - t) * control_points[i][1] + t * control_points[i + 1][1]
            z = (1 - t) * control_points[i][2] + t * control_points[i + 1][2]
            new_points.append((x, y, z))
        return de_casteljau(new_points, t)
```



```python
def add_point_from_entries():
    try:
        x = float(entry_x.get())
        y = float(entry_y.get())
        z = float(entry_z.get())
        control_points.append((x, y, z))

        canvas.create_oval(x + 300 - 4, 300 - y - 4, x + 300 + 4, 300 - y + 4, fill="red")

        label_text = f"({x}, {y}, {z})"
        canvas.create_text(x + 300 + 20, 300 - y - 20, text=label_text, fill="blue", font=("Helvetica", 10))

        if len(control_points) >= 2:
            canvas.create_line(control_points[-2][:2], control_points[-1][:2], fill="blue")

        if len(control_points) >= 3:
            redraw_bezier_curve()

    except ValueError:
        print("Invalid input. Please enter numeric values for x, y, and z.")
```



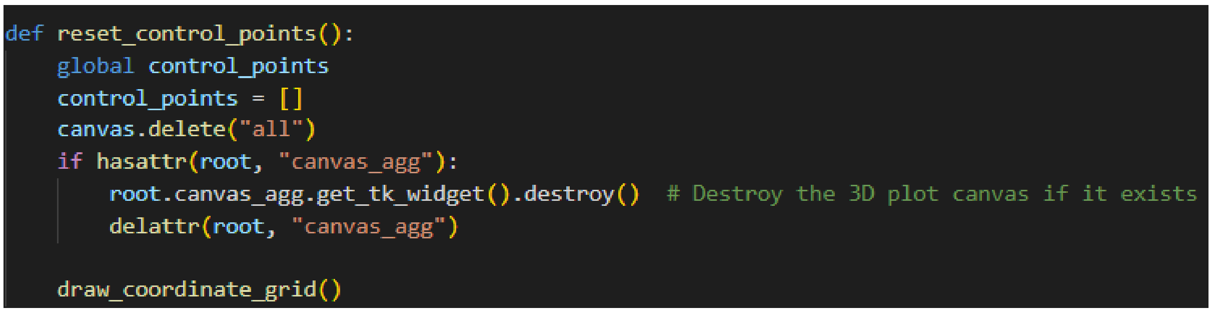
```python
# Create entry widgets for x, y, z coordinates
entry_x = tk.Entry(root, width=10)
entry_y = tk.Entry(root, width=10)
entry_z = tk.Entry(root, width=10)

# Place entry widgets on the window
entry_x.pack(side=tk.LEFT)
entry_y.pack(side=tk.LEFT)
entry_z.pack(side=tk.LEFT)

# Create a button to add the point from the entries
button_add_point = tk.Button(root, text="Add Point", command=add_point_from_entries)
button_add_point.pack(side=tk.LEFT)
```



```python
def redraw_bezier_curve():
    if len(control_points) >= 3:
        canvas.delete("curve")
        t_values = np.linspace(0, 1, 100)
        curve_points = np.array([de_casteljau(control_points, t) for t in t_values])

        fig = plt.figure()
        ax = fig.add_subplot(111, projection='3d')
        ax.plot_trisurf(curve_points[:, 0], curve_points[:, 1], curve_points[:, 2], color="green", linewidth=0, alpha=0.5)

        # Plot control points on the 3D plot with labels
        control_points_array = np.array(control_points)
        ax.scatter(control_points_array[:, 0], control_points_array[:, 1], control_points_array[:, 2], color='red', marker='o', s=50)
        for i, (x, y, z) in enumerate(control_points):
            ax.text(x, y, z, f'({x}, {y}, {z})', fontsize=8)

        # Display the 3D plot in tkinter window
        draw_3d_plot_on_canvas(fig)
```

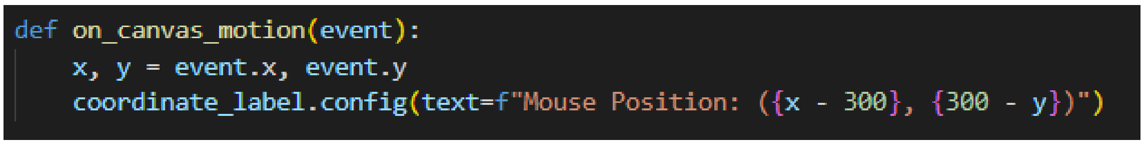
Propose suitable controls and interactions so that users can control/modify the surface

In our proposed solution, we prepared another function "reset_control_points" to allow the user to reset the control points and clear the canvas. As a result, all existing control points and the associated 3D plot canvas will be removed.

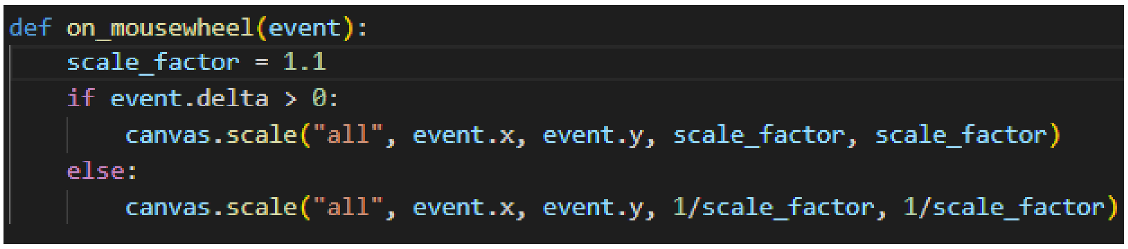


```python
def reset_control_points():
    global control_points
    control_points = []
    canvas.delete("all")
    if hasattr(root, "canvas_agg"):
        root.canvas_agg.get_tk_widget().destroy()  # Destroy the 3D plot canvas if it exists
        delattr(root, "canvas_agg")

    draw_coordinate_grid()
```

Also, we have prepared mouse interaction fo the users. The users can interact with the canva using the mouse. Firstly, the label at the bottom of the window displays the real-time coordinates of the mouse pointer as it moves over the canvas. Besides, the left mouse button enables dragging and moving(rotating) the 3D view canvas, while the middle mouse button facilitates dragging and moving the surface of the curve itself as well as the mouse wheel can be used to zoom in and out on the canvas.



```python
def on_canvas_motion(event):
    x, y = event.x, event.y
    coordinate_label.config(text=f"Mouse Position: ({x - 300}, {300 - y})")
```



```python
def on_mousewheel(event):
    scale_factor = 1.1
    if event.delta > 0:
        canvas.scale("all", event.x, event.y, scale_factor, scale_factor)
    else:
        canvas.scale("all", event.x, event.y, 1/scale_factor, 1/scale_factor)
```

Propose a creative product/usage of your codes/extension of your codes

We plan to enhance the proposed code into an interactive 3D modelling tool that allows users to easily build and visualize bespoke 3D forms with Bezier curves. To achieve that, we need to include enhancements like the ability to create multiple curves, assign colours to each for differentiation as well as introduce a control panel for individual curve manipulation. Besides, users may enter editing mode to alter control points intuitively right on the canvas, with real-time updates displaying the growing 3D surface. Furthermore, a history panel that logs changes and allows users to go back in time or generate keyframes for animation should be implemented. Finally, we suggest implementing animation, export capability, and a full set of 3D tools to create a user-friendly interface that stimulates creative exploration and the smooth construction of complicated 3D creations.