CS606: Computer Graphics / Term 1 (2025-26) / Programming Assignment 1 (A1)

Subha Chakraborty
Roll Number: MT2024156

1. Explain your strategy for user interactions?

In my code, the user interactions follow a clear update cycle to keep the geometry and visualization consistent:

- **Obstacle Movement and Rotation:** When keys `w`, `a`, `s`, `d` are pressed, the code adds or subtracts a fixed movement speed from the obstacle's x and y coordinates. Rotation keys `r` and `R` increase or decrease the rotation angle `rotDeg`. After these updates, the obstacle's corners are recalculated using trigonometric rotation formulas that rotate the rectangle around its center.
- **Obstacle Scaling:** Scaling smaller (`b`) or larger (`B`) multiplies or divides the obstacle's width and height by a constant scale factor `sclFac`. The scaling preserves the obstacle center, ensuring uniform growth or shrinkage.
- **Points Interaction:** Points ("people") are dragged with mouse events. The drag index tracks which point follows the mouse position, updating its coordinates. Dragging triggers recalculation of densities inside triangles.
- **Edge Creation and Deletion:** Left clicking two base points one after another adds an edge connecting them. Left clicking near existing edges deletes them. Right-click cancels any selections. Changing edges causes triangles and densities to be rebuilt.
- **Triangle and Density Updates:** Using the updated edges, triangles are found by looking for triplets of mutually connected points. The code counts how many "people" points lie inside each triangle with a point-in-triangle test using area-based coordinate calculations. These counts become density values, which the rendering code uses to assign blue, green, or red colors to triangles dynamically.
- **Rendering:** Each user interaction triggers geometry and density updates, which are sent to the GPU for real-time drawing. This maintains an

interactive and accurate visual representation of the spatial data based on user input.

2. How is scaling about a corner of the quadrilateral different from that of the centre?

When we scale a quadrilateral about its centre, the centre point stays fixed, and all four corners move outward or inward evenly. This keeps the shape balanced, and it grows or shrinks uniformly in all directions. It feels intuitive because the quadrilateral essentially expands or contracts around its middle.

On the other hand, if we scale about a corner, that corner stays in place while the other corners move. This can make the shape appear skewed or shifted because the movement is uneven. The quadrilateral no longer grows symmetrically, and its overall position changes, which can feel less predictable. Scaling around the center usually gives a smoother and more controlled effect, especially when interacting with it in real time.

References

- Delaunator Library (Delaunay Triangulation): Used for triangulating points. `https://github.com/mapbox/delaunator`
- WebGL and GLSL Shader Basics: For rendering with shaders and WebGL API usage. `https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API/Tutorial`