CMPE2300 – ICA09 – Slow Draw

In this ultra-simple ICA, you will draw a line… slowly…

Queues are often used to buffer information until the system is ready to process the information. We can simulate asynchronous arrival of data by trapping mouse movement events. We can simulate synchronous data processing by drawing the lines at a fixed rate. This would mean that lines could potentially arrive much faster than they would be drawn, and would therefore need to be queued.

Begin by creating a standard forms application with `GDIDrawer` support. Continuous update on.

Add a `Queue` of `Point` to hold mouse coordinates.

Subscribe to the mouse movement event in the drawer. In this handler queue the point (with great thread safety). There is no need to redirect this handler to the thread context of the main form.

Create a manual thread for rendering the lines in the queue. This thread will run forever, using a stopwatch, scheduling, and 1ms sleep calls to pause 50ms between iterations. In the in the body of this thread, determine if there are any queued points. If there are, dequeue the next point and draw a line from the last point (dequeued in the previous iteration) to this point, then update the last point to equal this point (serving as the last point for the next line). You will need a method-level `Point` to hold the 'start' of the line segment.

NOTE: You will need to build in some sort of mechanism into this process to make the first point encountered the initial last point (there is no drawing to do on the first dequeue). Do this how you will. If you are confused, it will become obvious when you run the program.

Add a status indicator that will show the number of points in the queue, and the time it will take the application to empty the queue. To make this thread safe, `Invoke` using an `Action` and lambda to update the text of the label.

Subscribe to the mouse right-click event. In this event handler, you will add 250 arms that will radiate out from the click position in random directions, at random radii (100 pixels max). This will require that you add 500 points to the queue.

You will be judged on your ability to minimize lock times and create efficient and manageable code.

The timing operations in the rendering thread are not optional – you must implement this as specified, and your instructor will discuss this in class.