Assignment 5 was somewhat easy to do but took a long time for me. The provided code in C helped a lot in quickly getting the bases down for each of the shapes. The thing that took the most time was getting the outline design made and making it so that the title, input coordinates, and drawing controls of each shape were displayed only when their dropdown option was selected.

The DDA line and midpoint line were the easiest to implement and translate from C to JavaScript, with ellipse being the hardest. The DDA line compared with the simple lineTo primitive looked exactly alike, while with the midpoint line, you could clearly see the pixels of the line as the y value is incremented. The midpoint line edges aliased, unlike the anti-aliased lines of the primitive line or DDA line.

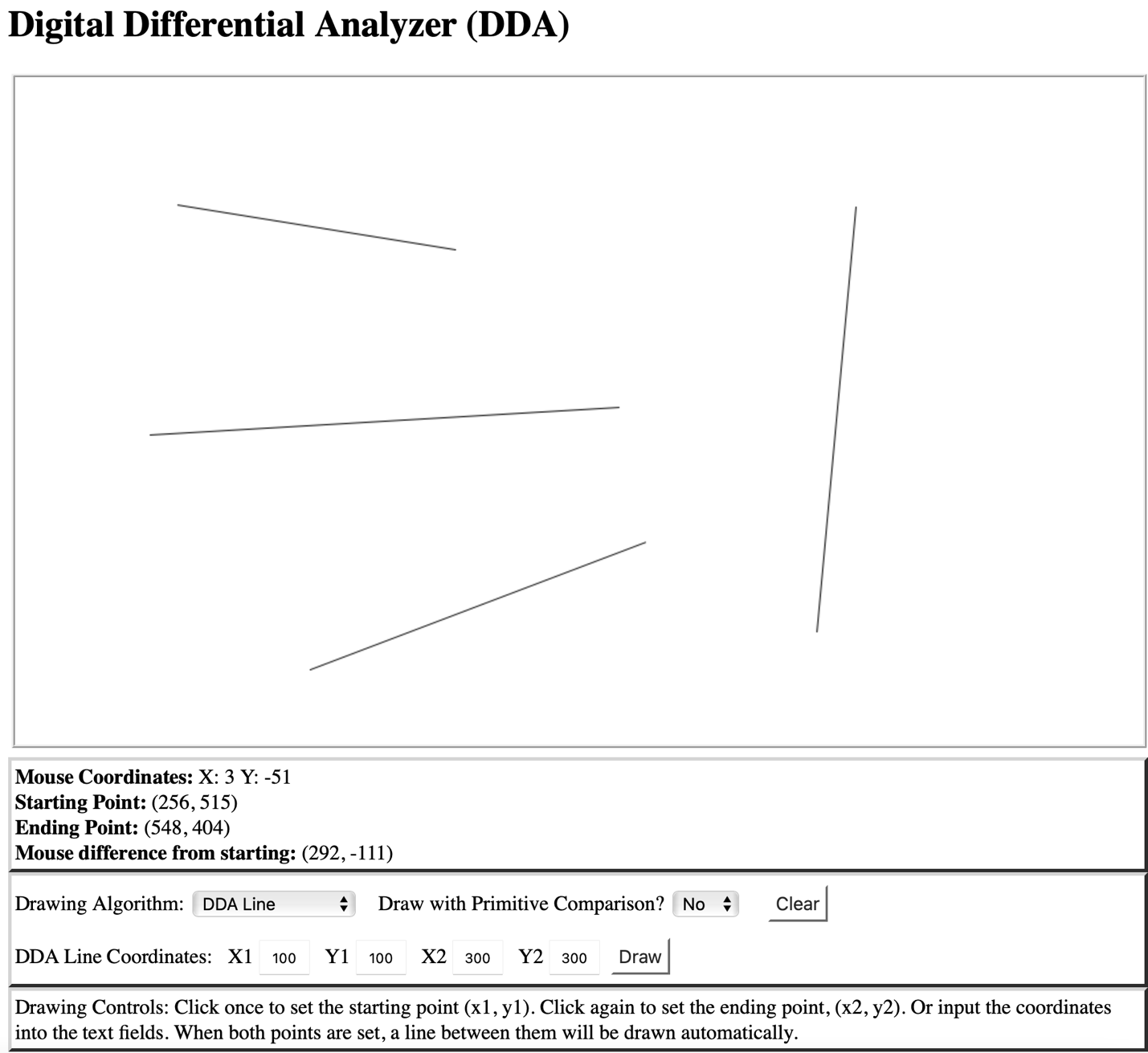
The midpoint circle was another easy one to do, only requiring x, y coordinates for the center and a radius. Like the midpoint line, the edges of the circle are hard and defined as compared to the primitive circle where they are anti-aliased. The midpoint ellipse was what took me the longest time to do beside creating the html outline. I knew it took the same x, y coordinates for the center, but it took me awhile to understand how to create the ellipse based on where the user clicked for the second time. I eventually found that the ellipse took two radiuses, one for the x-direction and one for the y-direction. Compared to its primitive, the midpoint ellipse is aliased as well, while the primitive is anti-aliased.

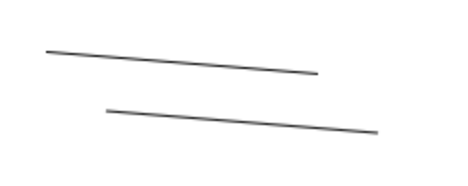
For the Curves, these were just a tad bit easier to implement than the midpoint ellipse. Translating them from C to JavaScript was easy, but while I could find a primitive for the Bezier curve, I was not able to for the Hermite or B Spline curve (though I was still able to implement the non-primitive counterparts).

For the Bezier curve, I made a curvePoints object that has x and y coordinates. The Bezier function took a variable **n** for the number of points displayed on the line, and four coordinates, p1, p2, p3 and p4. To draw the curve was simple; click once to set the beginning, click twice to set the first control point for the curve, click a third time for the second control point, and then click a final time for the end of the curve. The variable **n** defined how many points were drawn on the curve, with the higher the number leading to a more distinguished curve. With less points, you could see the majority of the points are centered around the middle of the curve, with fewer points towards the beginning and end. When compared to its primitive, the only difference in the Bezier curve was the number of points displayed. The primitive had a constant number of points, while the function could be faint with very few points scattered along the line, or many points to make it dark and noticeable.

For the Hermite curve, it is similar to the Bezier curve, with the variable **n** to control the number of points drawn on the line and the coordinates, p1, p2, p3, and p4, but instead of drawing the points in order, the first two points, p1, p2, are used for the start and end of the curve, while the points p3 and p4 are used for the control points. The first two clicks set how long and in what general direction the Hermite curve is facing, while the next two clicks control the curves. I was unable to make a function for the primitive for Hermite Curve.

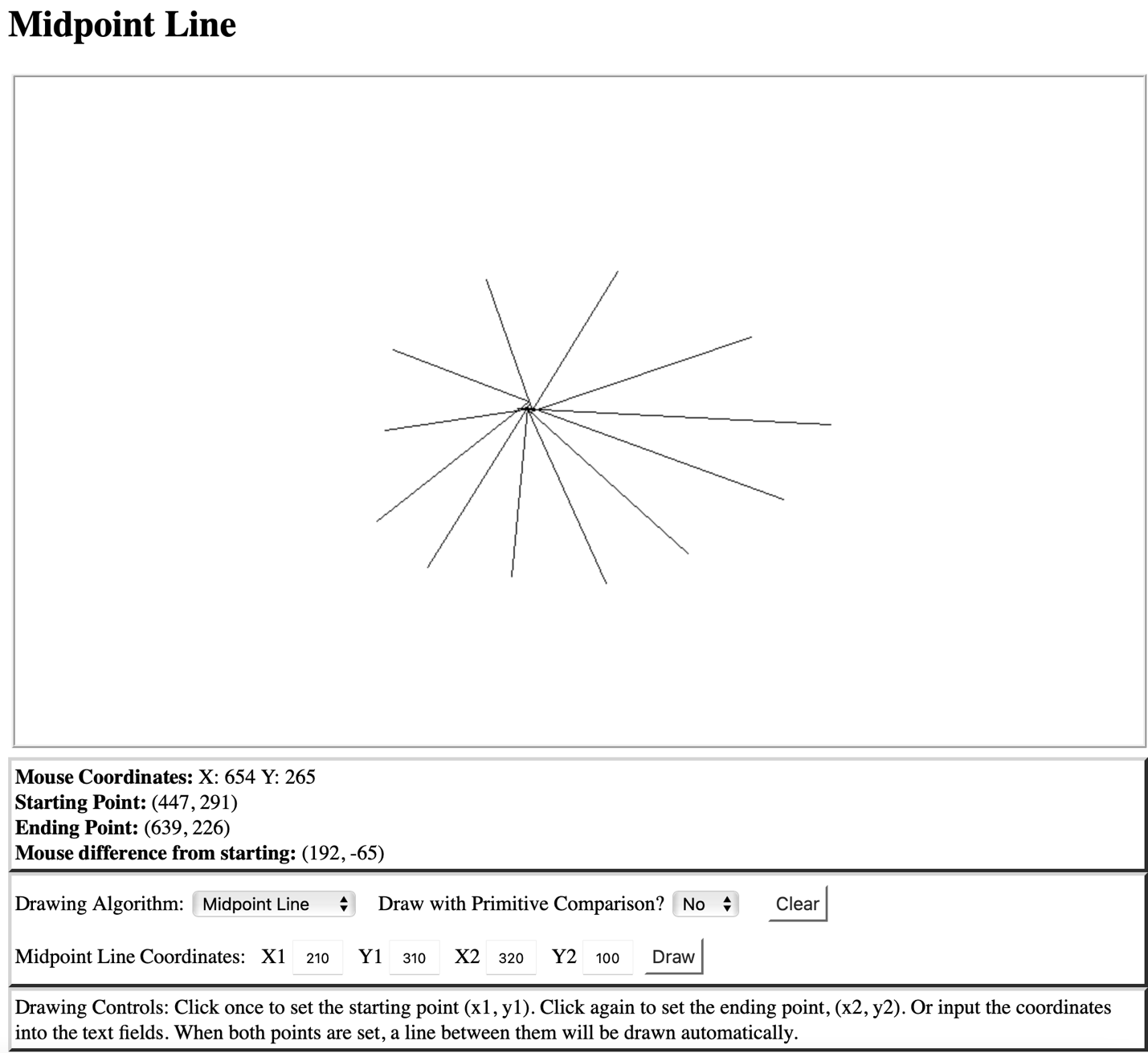
For the B Spline curve, it is similar to the Bezier curve in how it is drawn with variable **n** to control how many points are drawn and coordinates, p1, p2, p3 and p4, but instead of the curve touching the points, it is sandwiched between them. The best I can describe it is the curve being drawn halfway between the coordinates; p1, p2; then p2, p3; then p3 and p4. I was unable to make a function for the primitive of B Spline





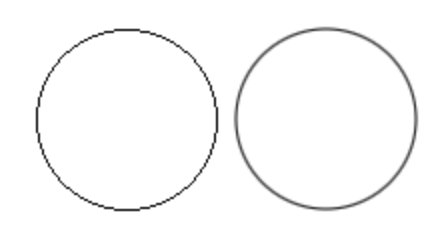
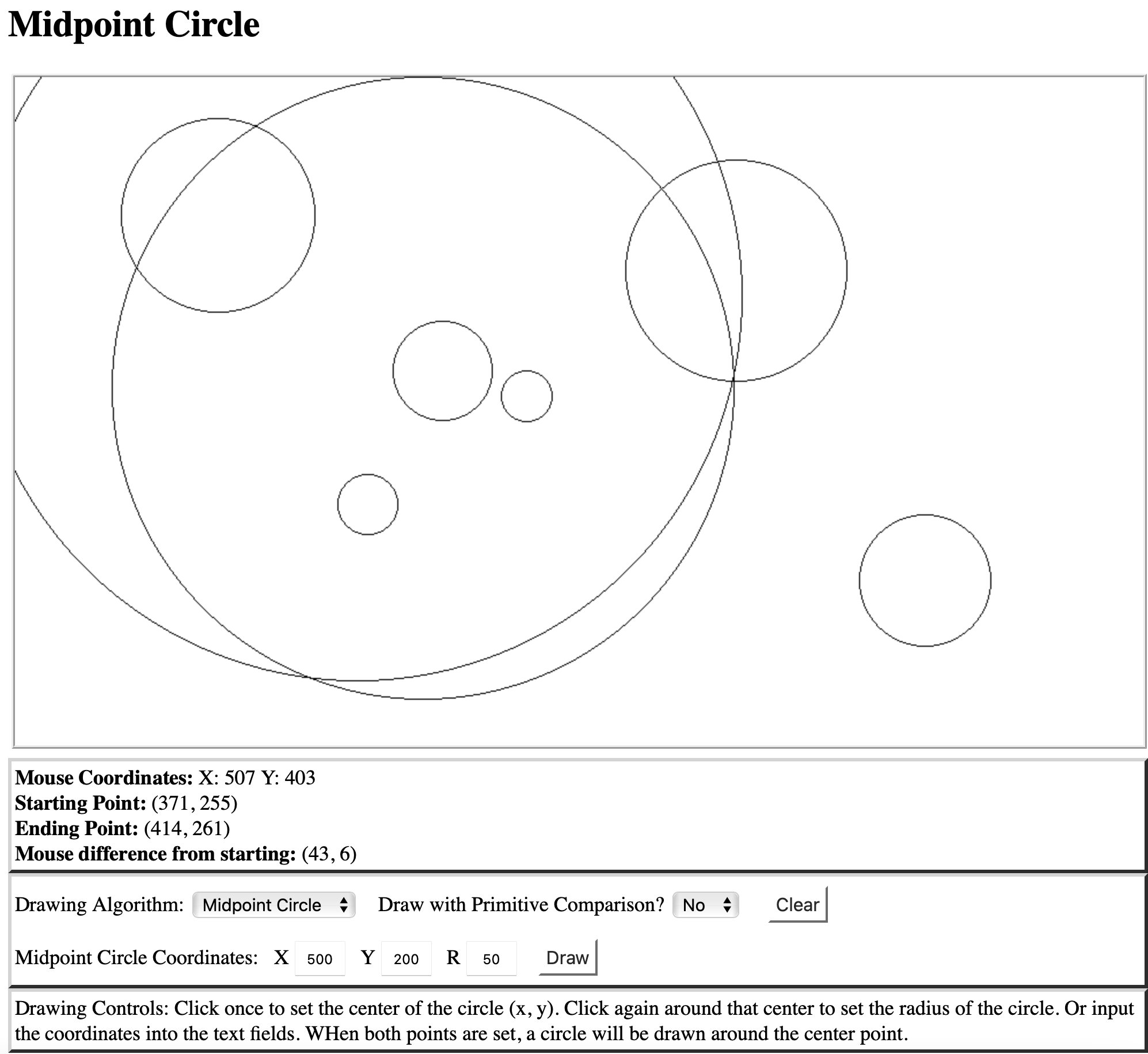
Top: function

Bottom: primitive



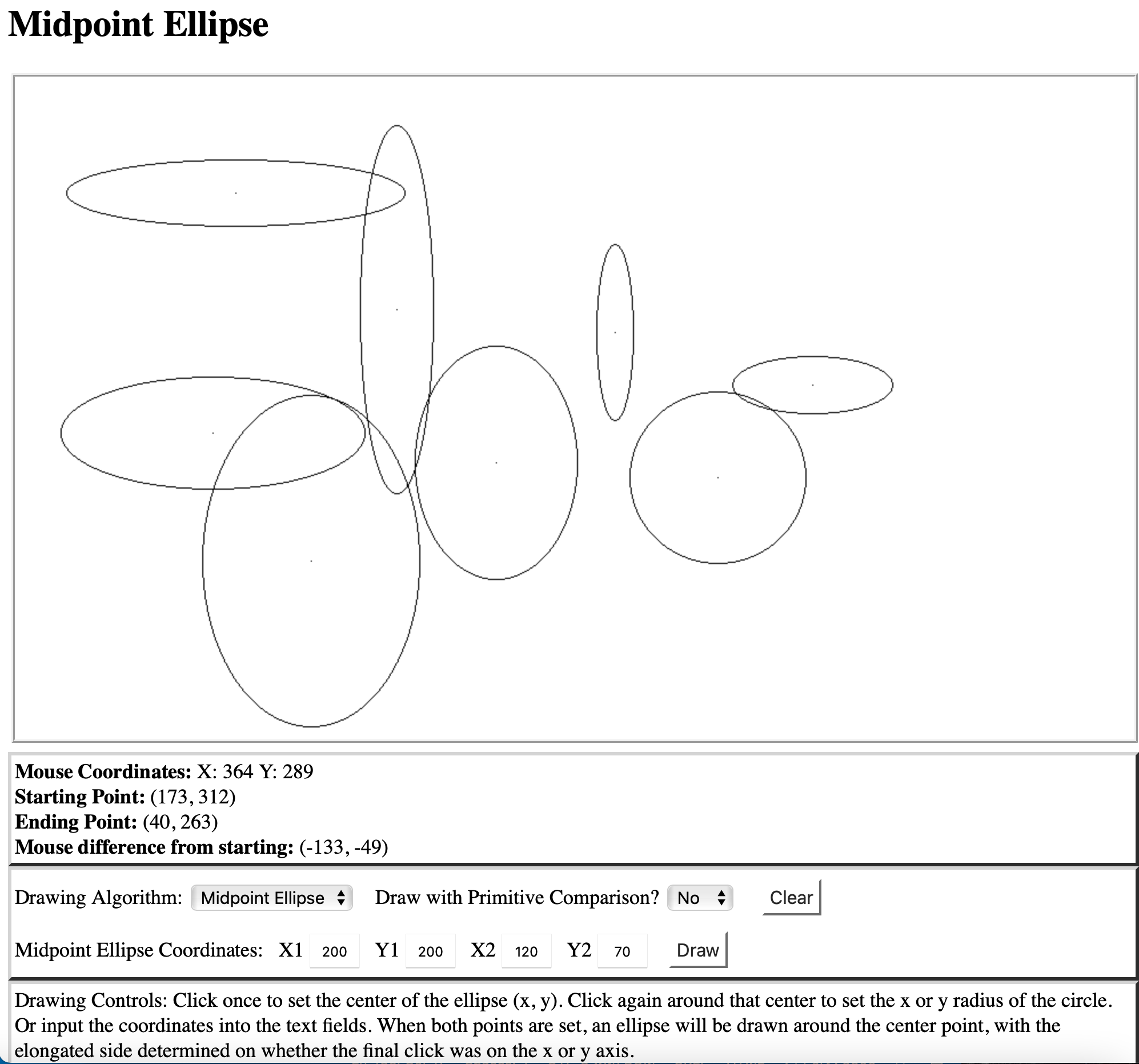
Top: function

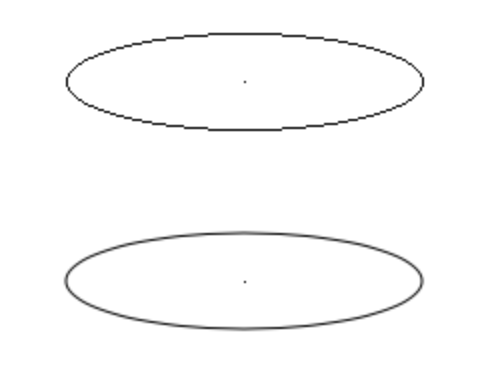
Bottom: primitive



left: function

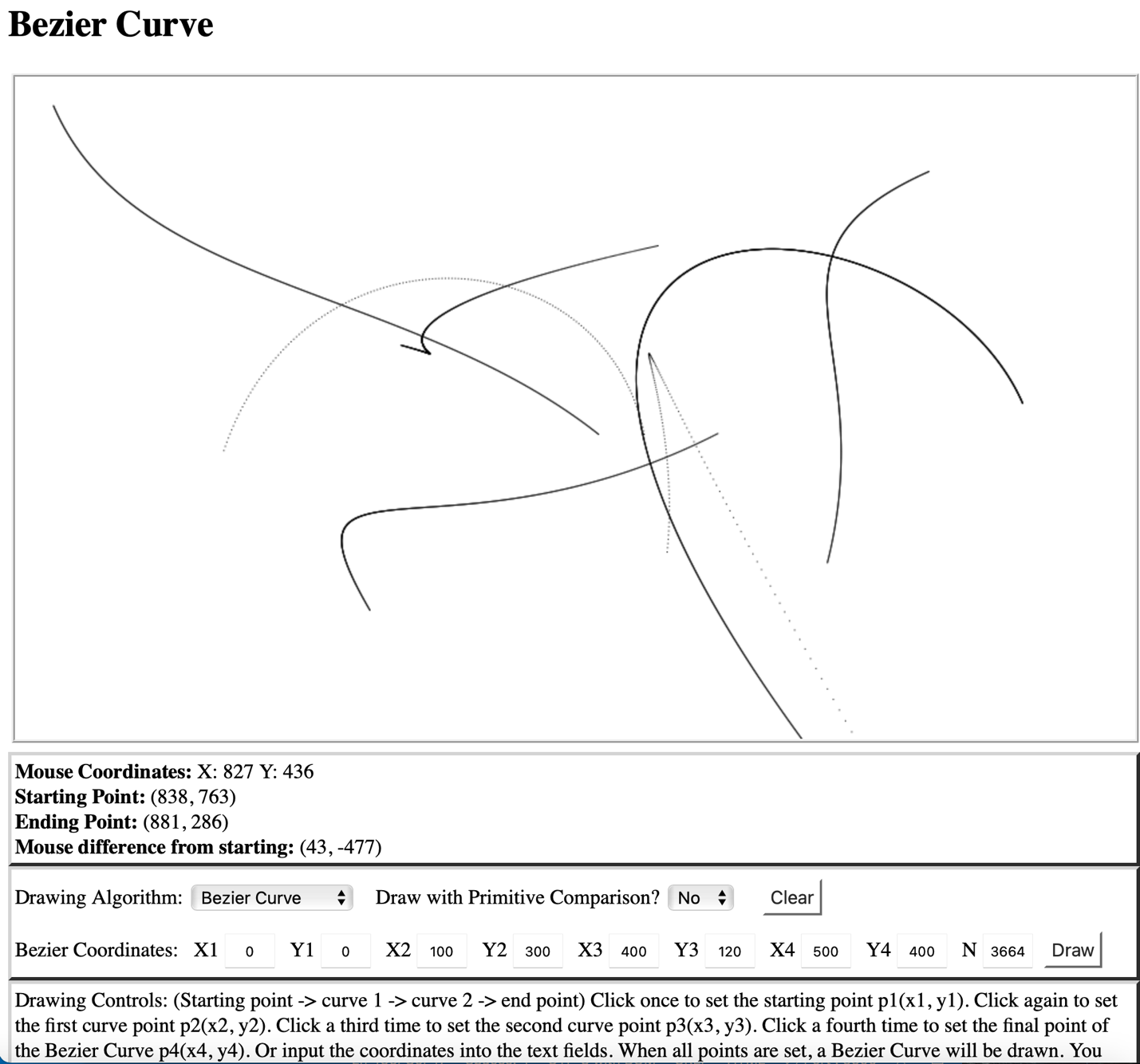
right: primitive





Top: function

Bottom: primitive





Top: primitive

Bottom: function (n = 100)



Top: primitive

Bottom: function (n = 1000)