Exercise – Curves and Splines

The exercises for this make use of the AIE Bootstrap codebase.

1. Implement a Bezier class or namespace that contains methods for calculating a Quadratic Bezier Curve and a Cubic Bezier Curve.

Note: Bezier Curves work on many different types of variables, not just Points and Vectors. Consider making your methods or class templated so that it can handle various data types.

1. Test out your Bezier Curve methods graphically using Bootstrap.

The following code uses the Renderer2D class to draw a 2D Quadratic Bezier Curve. You will need to replace the curve() method with your own. Also test your Cubic Bezier Curve method.

// within Application::draw()...

// draw curve

float t0 = 0;

float t1 = 0.1f;

while (t0 < 1) {

// replace curve() with your methods

// cp0, 1 and 2 are Vector2 in this example

Vector2 p0 = curve(cp0, cp1, cp2, t0);

Vector2 p1 = curve(cp0, cp1, cp2, t1);

m\_2dRenderer->drawLine(p0.x, p0.y, p1.x, p1.y, 2);

t0 = t1;

t1 += 0.1f;

}

1. Implement methods for calculating Splines.

The below code is an example of a template method for calculating linear interpolation between a collection of control points. Look over the code; each control point takes the same amount of time to interpolate between as the others, regardless of distance between the points. It does not use any tangents as it is simple Linear Interpolation.

template <typename T>

T linearSpline(T\* controlPoints, int count, float t) {

// scale between each control point

float step = 1.0f / count;

// find the two control points to interpolate between

int low = int(t / step);

int high = low + 1;

// if out of range return last

if (high >= count)

return controlPoints[count - 1];

// clamp the interpolation scale between the two control points

t = (t - step \* low) / step;

// return the interpolated value

return controlPoints[low] \* (1 - t) + controlPoints[high] \* t;

}

Using the provided code as an example, attempt to implement functions for **Hermite Splines**, **Cardinal Splines** and **Catmull-Rom Splines**. You will need to include parameters for tangents for Hermite Splines.

Using the sample code from step 2, add the ability to graphically display a spline.