CS411 COMPUTER GRAPHICS FALL 2021 BRACK TURNER

Implemented a Pointprocs() function to store user clicks into ctrlPts array. Separating point procs function from the click function do interpolation computations separately instead of inside of click() from the skeleton code.

```
    // store the point
    console.log('storing point (%f,%f)',x,y);
    ctrlPts.push(x);
    ctrlPts.push(y);

    Pointprocs();
    // generate interpolation points

  }


  function Pointprocs() {
    intrPts = [];
    console.log("Remaking!");

    var nCtrlPts = ctrlPts.length/2;
    if (nCtrlPts > 3){
      for (var i = 0; i < nCtrlPts - 3; i++) {
      // there are 4 points - interpolate between pk and pk+1
      console.log('adding interpolation points');
      var pk_n1 = {x: ctrlPts[0 + i * 2], y: ctrlPts[1 + i * 2]}; // p_{k-1}
      var pk    = {x: ctrlPts[2 + i * 2], y: ctrlPts[3 + i * 2]}; // p_{k}
      var pk_p1 = {x: ctrlPts[4 + i * 2], y: ctrlPts[5 + i * 2]}; // p_{k+1}
      var pk_p2 = {x: ctrlPts[6 + i * 2], y: ctrlPts[7 + i * 2]}; // p_{k+2}
      interpolate(pk_n1,pk,pk_p1,pk_p2);
      }
    }
    refreshFlag = 1;
    //show a plot update
  }
```

Created a McUpdate() function to update the interpolated points and is dependent on the tension variable. Added it to the main function to constantly update and create a Cardinal Spline between two midpoints.

```
//update Mc MAtrix
function McUpdate() {
  Mc = [
    -tension, 2 - tension, tension - 2, tension,
    2 * tension, tension - 3, 3 - 2 * tension, - tension,
    -tension, 0, tension, 0,
    0, 1, 0, 0
  ];
}
```

Once the mc matrix is set and we use the point procs() function we then need the interpolate function to then interpolate the points chosen.
Function interpolate() then takes the 4 control points and creates x and y coordinates for each control point. We then Multiply those x an y coordinates with the Mc Matrix in McMultiplypoint()

```
function McMultiplyPoint(point) {
  return [
    Mc[0] * point[0] + Mc[1] * point[1] + Mc[2] * point[2] + Mc[3] * point[3],
    Mc[4] * point[0] + Mc[5] * point[1] + Mc[6] * point[2] + Mc[7] * point[3],
    Mc[8] * point[0] + Mc[9] * point[1] + Mc [10] * point[2] + Mc[11] * point[3],
    Mc[12] * point[0] + Mc[13] * point[1] + Mc[14] * point[2] + Mc[15] * point[3]
  ];
}
function findPoint( x, x1) {
  return x[0] * x1[0] + x[1] * x1[1] + x[2] * x1[2] + x[3] * x1[3];
 }
function interpolate(pk_n1,pk,pk_p1,pk_p2)
{
  var xpoint = [pk_n1.x, pk.x,pk_p1.x, pk_p2.x];
  var ypoint = [pk_n1.y,pk.y, pk_p1.y,pk_p2.y];

  var xpointMc = McMultiplyPoint(xpoint);
  var ypointMc = McMultiplyPoint(ypoint);
```

Finally we then take the matrix multiplied points and multiply the u coefficients giving us x and y values for each given value of u. findPoint() then multiplies the Mc Matrix and u coefficients and pushes the results to the intrPts array.

```
  for(var u=0; u<=1; u+=uStep){
    var Ux = [Math.pow(u, 3), u * u, u, 1];
    var Uy = [Math.pow(u, 3), u * u, u, 1];

    var xpointNew = findPoint(xpointMc, Ux);
    var ypointNew = findPoint(ypointMc, Uy);

    intrPts.push(xpointNew);
    intrPts.push(ypointNew);
  }
```

Then UpdateMc and Pointprocs was added to the tension up and down functions to have the buttons be affected by the matrix and spline calculations respectively.

```
function tensionUp() {
  tension += 0.1;
  console.log('tension = %f', tension);
  McUpdate();
  Pointprocs();

}

function tensionDown() {
  tension -= 0.1;
  console.log('tension = %f', tension);
  McUpdate();
  Pointprocs();

}
```

Also added intrPts = []; emptied list to reset the array in case new points are added and more calculations need to be done. That way the tension changes can be properly incremented and decremented without error.

```
function Pointprocs() {
 intrPts = [];
 console.log("Remaking!");
```