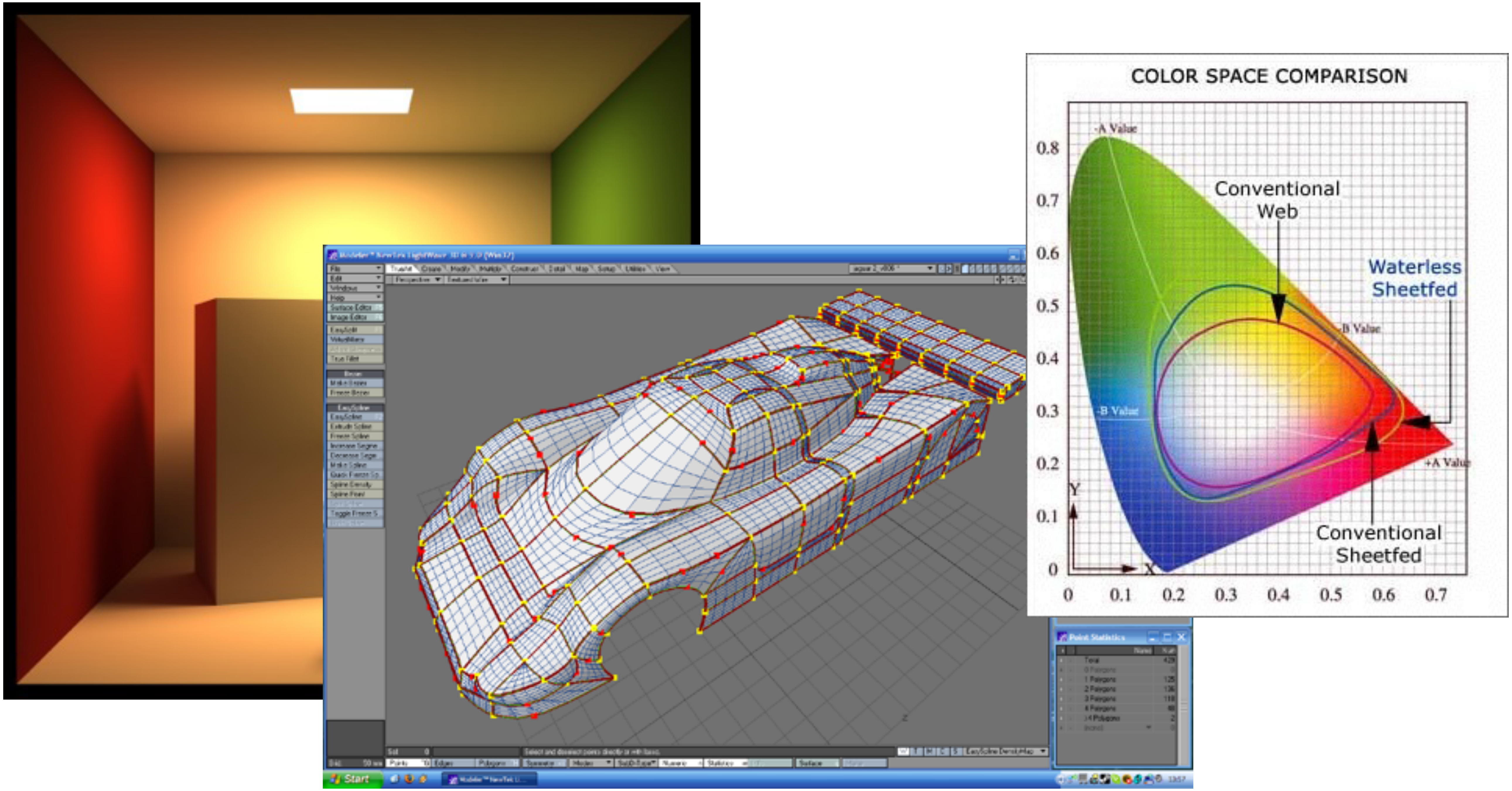


Lecture 9 : Parametric 2D curves continued (tangents, piecewise curves, continuity, intro to polynomial curves)

Thursday October 7th 2021

Logistics

- Assignment #2 due today (#3 to be released soon)
- Fate of midterm: We ***will*** offer the midterm as an online quiz on Canvas. We will continue the discussion (be on the lookout for another poll on Piazza) as to the date/time of the exam.
- Lecture time on Nov 2nd seems to be a more likely/popular option. There may be some that prefer Friday 10/29, but keep in mind that:
  - We will discuss a practice exam (maybe one of 2 …) in-class on 10/28; maybe you could use the extra time?
  - The midterm is "optional" (the final can count for both)

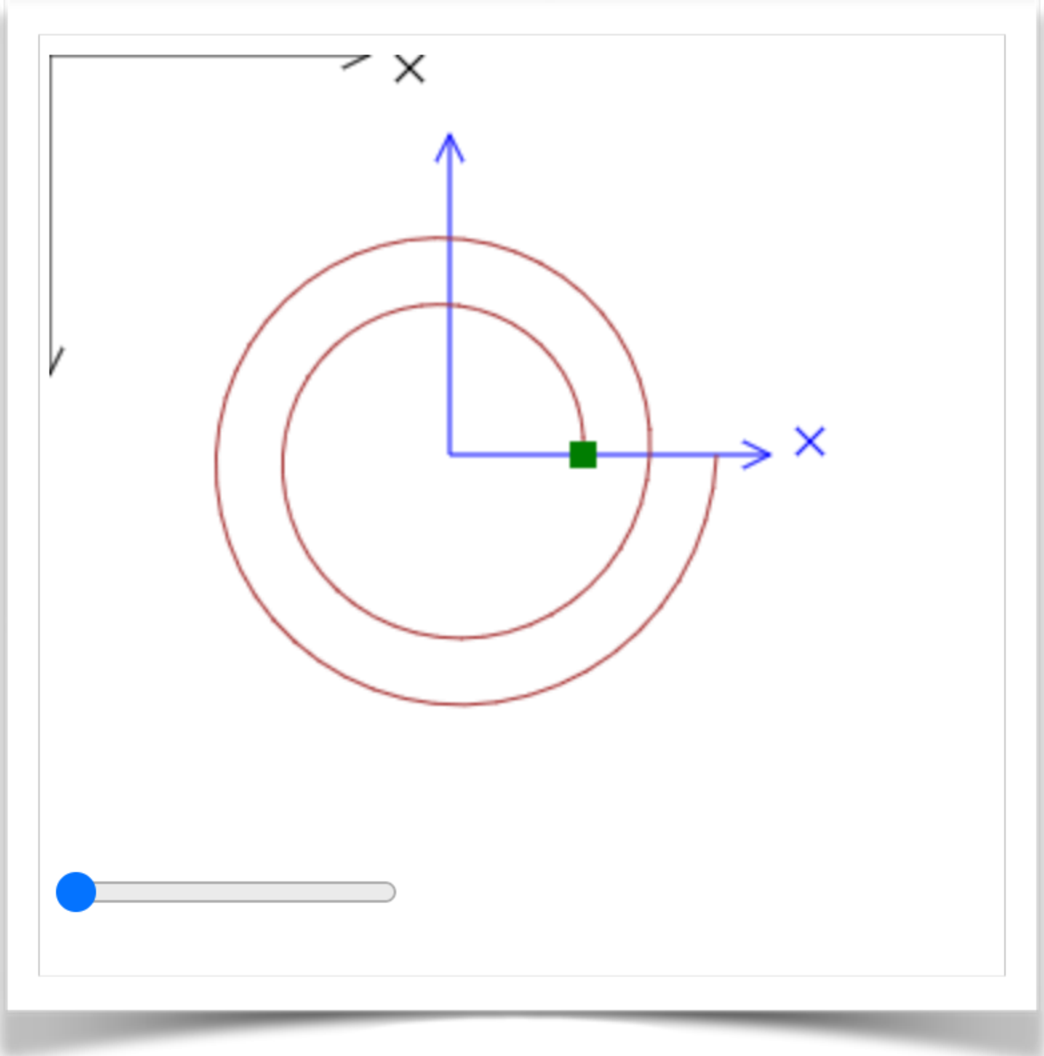
Today's lecture

- Additional practical implementation examples
  - Drawing curves, moving along curve
  - Controlling orientation
- More theory
  - Piecewise-defined curves and continuity
  - Polynomial parametric curves

Implementation example : A spiral curve

jsbin.com/ziletiq

Week5/Demo0



Curve

$$\mathcal{C}(t) = \begin{pmatrix} R(t)\cos(2\pi t) \\ R(t)\sin(2\pi t) \end{pmatrix}, \quad t \in [0, 2]$$

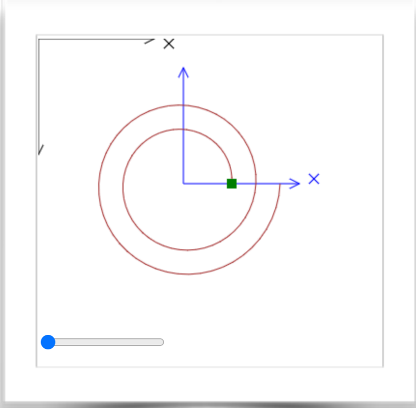
$$R(t) = \alpha t + \beta$$

(all expressions relative to the blue system!)

Implementation example : A spiral curve

jsbin.com/ziletiq

Week5/Demo0



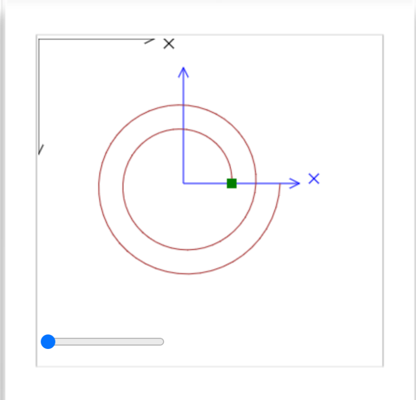
JavaScript

```
function setup() {
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');
    var slider1 = document.getElementById('slider1');
    slider1.value = -25;

    function draw() {
    canvas.width = canvas.width;
    // use the sliders to get the angles
    var tParam = slider1.value*0.01;

    function moveToTx(loc,Tx)
    {var res=vec2.create(); vec2.transformMat3(res,loc,Tx); context.moveTo(res[0],res[1]);}

    function lineToTx(loc,Tx)
    {var res=vec2.create(); vec2.transformMat3(res,loc,Tx); context.lineTo(res[0],res[1]);}

    function drawObject(color,Tx) {
        context.beginPath();
        context.fillStyle = color;
        moveToTx([-5,-5],Tx);
        lineToTx([-5,5],Tx);
        lineToTx([5,5],Tx);
        lineToTx([5,-5],Tx);
        context.closePath();
        context.fill();
    }

[...]
```

Implementation example : A spiral curve

jsbin.com/ziletiq

Week5/Demo0



JavaScript

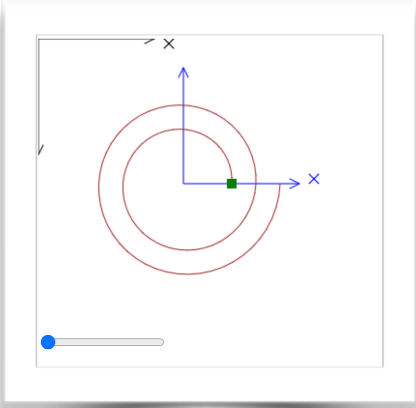
```
function setup() {
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');
    var slider1 = document.getElementById('slider1');
    slider1.value = -25;

    function draw() {
    canvas.width = canvas.width;
    // use the sliders to get the angles
    var tParam = slider1.value*0.01;

    function moveToTx(loc,Tx)
    {var res=vec2.create(); vec2.transformMat3(res,loc,Tx); context.moveTo(res[0],res[1]);}

    function lineToTx(loc,Tx)
    {var res=vec2.create(); vec2.transformMat3(res,loc,Tx); context.lineTo(res[0],res[1]);}

    function drawObject(color,Tx) {
        context.beginPath();
        context.fillStyle = color;
        moveToTx([-5,-5],Tx);
        lineToTx([-5,5],Tx);
        lineToTx([5,5],Tx);
        lineToTx([5,-5],Tx);
        context.closePath();
        context.fill();
    }

[...]
```

Passing "vectors" to the drawing calls

Implementation example : A spiral curve

jsbin.com/ziletiq

Week5/Demo0



$$\mathcal{C}(t) = \begin{pmatrix} R(t)\cos(2\pi t) \\ R(t)\sin(2\pi t) \end{pmatrix}, \quad t \in [0,2]$$

$$R(t) = \alpha t + \beta$$

JavaScript

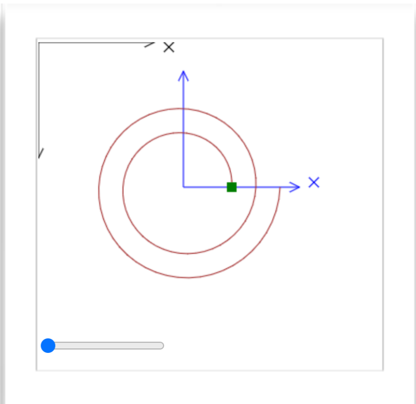
```
[...]
    function drawAxes(color,Tx) {
        context.strokeStyle=color;
        context.beginPath();
        // Axes
        moveToTx([120,0],Tx);lineToTx([0,0],Tx);lineToTx([0,120],Tx);
        // Arrowheads
        moveToTx([110,5],Tx);lineToTx([120,0],Tx);lineToTx([110,-5],Tx);
        moveToTx([5,110],Tx);lineToTx([0,120],Tx);lineToTx([-5,110],Tx);
        // X-label
        moveToTx([130,0],Tx);lineToTx([140,10],Tx);
        moveToTx([130,10],Tx);lineToTx([140,0],Tx);
        context.stroke();
    }

    var Rstart = 50.0;
    var Rslope = 25.0;
    var Cspiral = function(t) {
        var R = Rslope * t + Rstart;
        var x = R * Math.cos(2.0 * Math.PI * t);
        var y = R * Math.sin(2.0 * Math.PI * t);
        return [x,y];
    }

    function drawTrajectory(t_begin,t_end,intervals,C,Tx,color) {
        context.strokeStyle=color;
        context.beginPath();
            moveToTx(C(t_begin),Tx);
            for(var i=1;i<=intervals;i++){
        var t=((intervals-i)/intervals)*t_begin+(i/intervals)*t_end;
        lineToTx(C(t),Tx);
            }
            context.stroke();
    }

[...]
```

Implementation of the parametric curve

Implementation example : A spiral curve

jsbin.com/ziletiq

Week5/Demo0



JavaScript

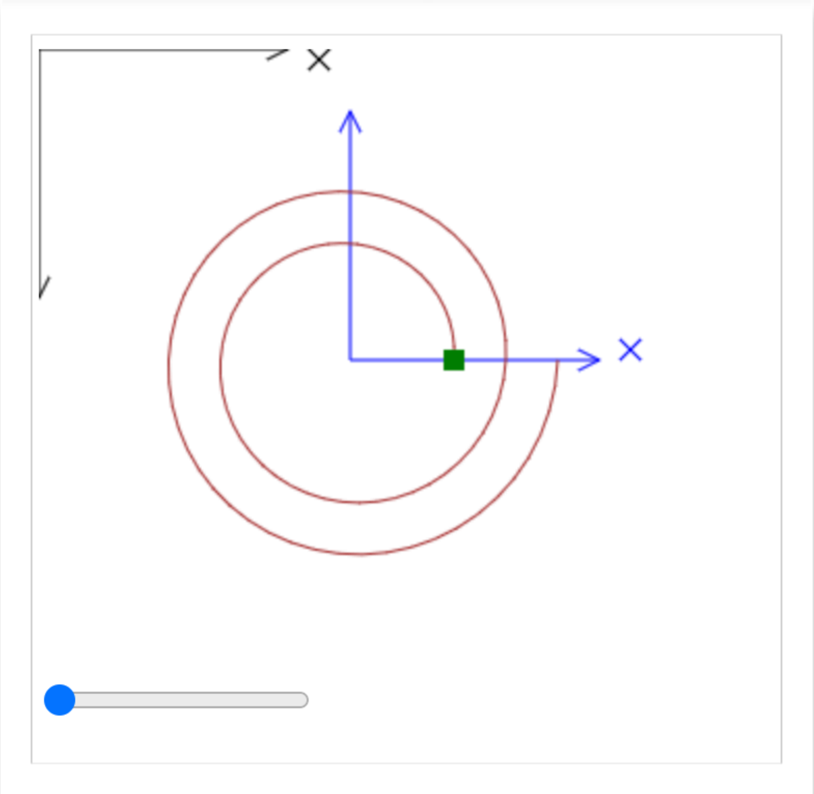
```
[...]
    function drawAxes(color,Tx) {
        context.strokeStyle=color;
        context.beginPath();
        // Axes
        moveToTx([120,0],Tx);lineToTx([0,0],Tx);lineToTx([0,120],Tx);
        // Arrowheads
        moveToTx([110,5],Tx);lineToTx([120,0],Tx);lineToTx([110,-5],Tx);
        moveToTx([5,110],Tx);lineToTx([0,120],Tx);lineToTx([-5,110],Tx);
        // X-label
        moveToTx([130,0],Tx);lineToTx([140,10],Tx);
        moveToTx([130,10],Tx);lineToTx([140,0],Tx);
        context.stroke();
    }

    var Rstart = 50.0;
    var Rslope = 25.0;
    var Cspiral = function(t) {
        var R = Rslope * t + Rstart;
        var x = R * Math.cos(2.0 * Math.PI * t);
        var y = R * Math.sin(2.0 * Math.PI * t);
        return [x,y];
    }

    function drawTrajectory(t_begin,t_end,intervals,C,Tx,color) {
        context.strokeStyle=color;
        context.beginPath();
            moveToTx(C(t_begin),Tx);
            for(var i=1;i<=intervals;i++){
        var t=((intervals-i)/intervals)*t_begin+(i/intervals)*t_end;
        lineToTx(C(t),Tx);
            }
            context.stroke();
    }

[...]
```

Drawing curve as a sequence of (small) line segments!

Implementation example : A spiral curve

jsbin.com/ziletiq

Week5/Demo0



JavaScript

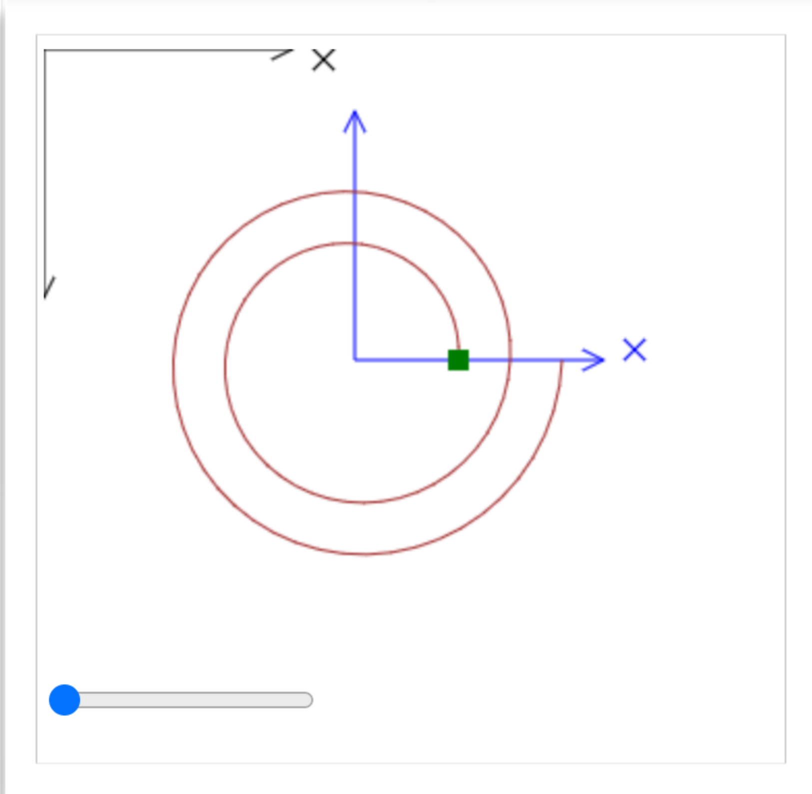
```
[...]
    function drawTrajectory(t_begin,t_end,intervals,C,Tx,color) {
        context.strokeStyle=color;
        context.beginPath();
            moveToTx(C(t_begin),Tx);
            for(var i=1;i<=intervals;i++){
        var t=((intervals-i)/intervals)*t_begin+(i/intervals)*t_end;
        lineToTx(C(t),Tx);
            }
            context.stroke();
    }

    // make sure you understand these

    drawAxes("black", mat3.create());

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[150,150]);
    mat3.scale(Tblue_to_canvas,Tblue_to_canvas,[1,-1]); // Flip the Y-axis
    drawAxes("blue",Tblue_to_canvas);

    drawTrajectory(0.0,2.0,100,Cspiral,Tblue_to_canvas,"brown");
    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,Cspiral(tParam));
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    drawObject("green",Tgreen_to_canvas);
    }

    slider1.addEventListener("input",draw);
    draw();
```

Using transforms to draw relatively to a better "centered" system

Implementation example : A spiral curve

jsbin.com/ziletiq

Week5/Demo0



JavaScript

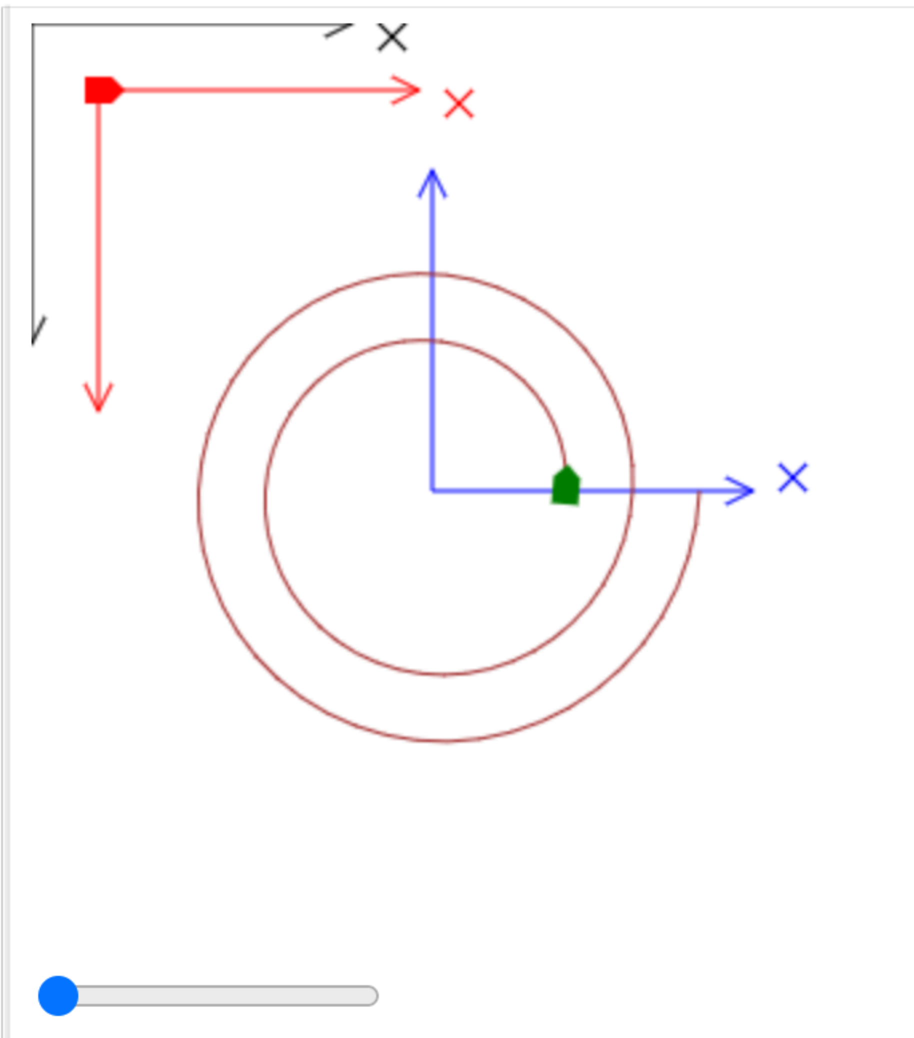
```
[...]
    function drawTrajectory(t_begin,t_end,intervals,C,Tx,color) {
        context.strokeStyle=color;
        context.beginPath();
            moveToTx(C(t_begin),Tx);
            for(var i=1;i<=intervals;i++){
        var t=((intervals-i)/intervals)*t_begin+(i/intervals)*t_end;
        lineToTx(C(t),Tx);
            }
            context.stroke();
    }

    // make sure you understand these

    drawAxes("black", mat3.create());

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[150,150]);
    mat3.scale(Tblue_to_canvas,Tblue_to_canvas,[1,-1]); // Flip the Y-axis
    drawAxes("blue",Tblue_to_canvas);

    drawTrajectory(0.0,2.0,100,Cspiral,Tblue_to_canvas,"brown");
    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,Cspiral(tParam));
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    drawObject("green",Tgreen_to_canvas);
    }

    slider1.addEventListener("input",draw);
    draw();
```

Draw extent of the curve, and animate an object translating along it

Implementation example : A spiral curve

jsbin.com/meliyax

Week5/Demo1



JavaScript

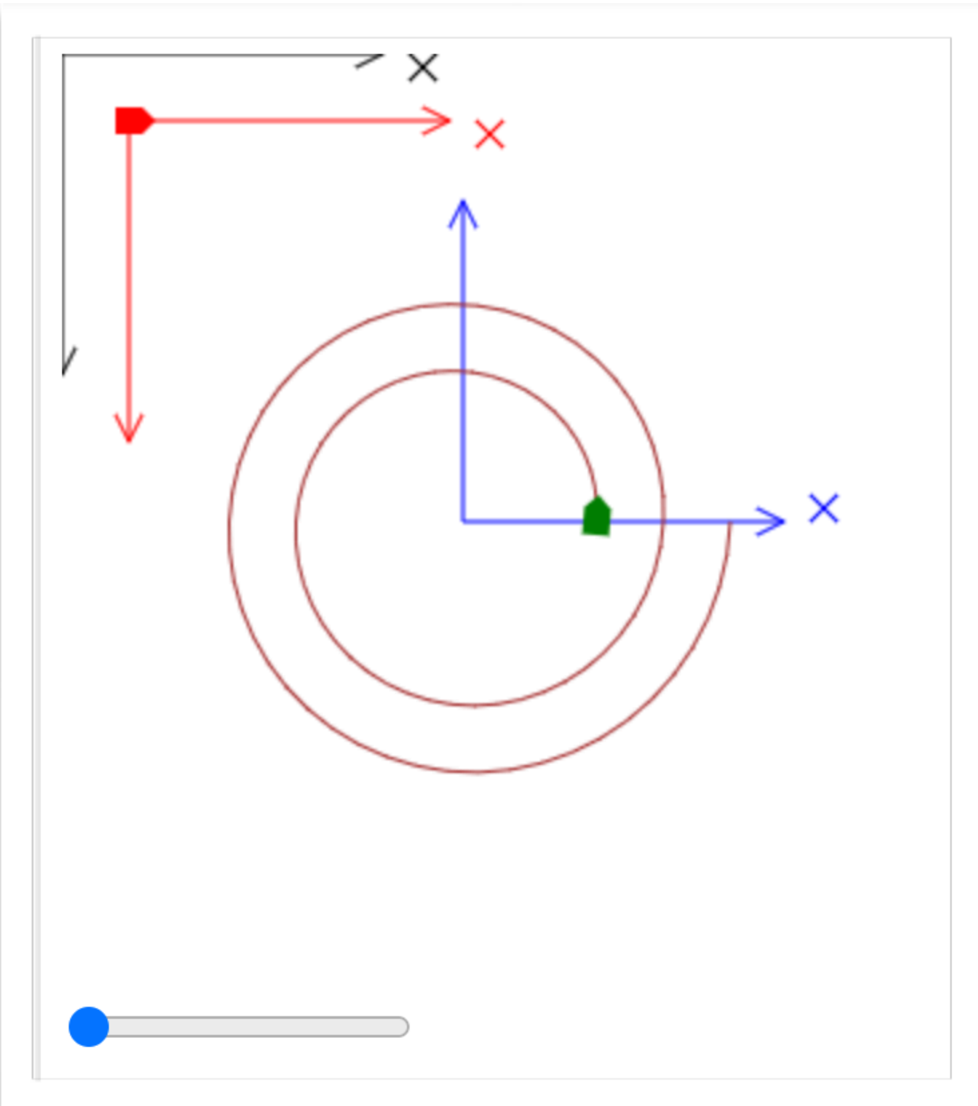
```
[...]
    function moveToTx(loc,Tx)
    {var res=vec2.create(); vec2.transformMat3(res,loc,Tx); context.moveTo(res[0],res[1]);}

    function lineToTx(loc,Tx)
    {var res=vec2.create(); vec2.transformMat3(res,loc,Tx); context.lineTo(res[0],res[1]);}

    function drawObject(color,Tx) {
        context.beginPath();
        context.fillStyle = color;
        moveToTx([-5,-5],Tx);
        lineToTx([-5,5],Tx);
        lineToTx([5,5],Tx);
        lineToTx([10,0],Tx);
        lineToTx([5,-5],Tx);
        context.closePath();
        context.fill();
    }
[...]
```

Draw a slightly different object, with a shape that suggests orientation (look at "red" system for reference)

Implementation example : A spiral curve

jsbin.com/meliyax

Week5/Demo1



JavaScript

```
[...]
    var Rstart = 50.0;
    var Rslope = 25.0;
    var Cspiral = function(t) {
        var R = Rslope * t + Rstart;
        var x = R * Math.cos(2.0 * Math.PI * t);
        var y = R * Math.sin(2.0 * Math.PI * t);
        return [x,y];
    }
    var Cspiral_tangent = function(t) {
        var R = Rslope * t + Rstart;
        var Rprime = Rslope;
        var x = Rprime * Math.cos(2.0 * Math.PI * t)
                - R * 2.0 * Math.PI * Math.sin(2.0 * Math.PI * t);
        var y = Rprime * Math.sin(2.0 * Math.PI * t)
                + R * 2.0 * Math.PI * Math.cos(2.0 * Math.PI * t);
        return [x,y];
    }
[...]
```

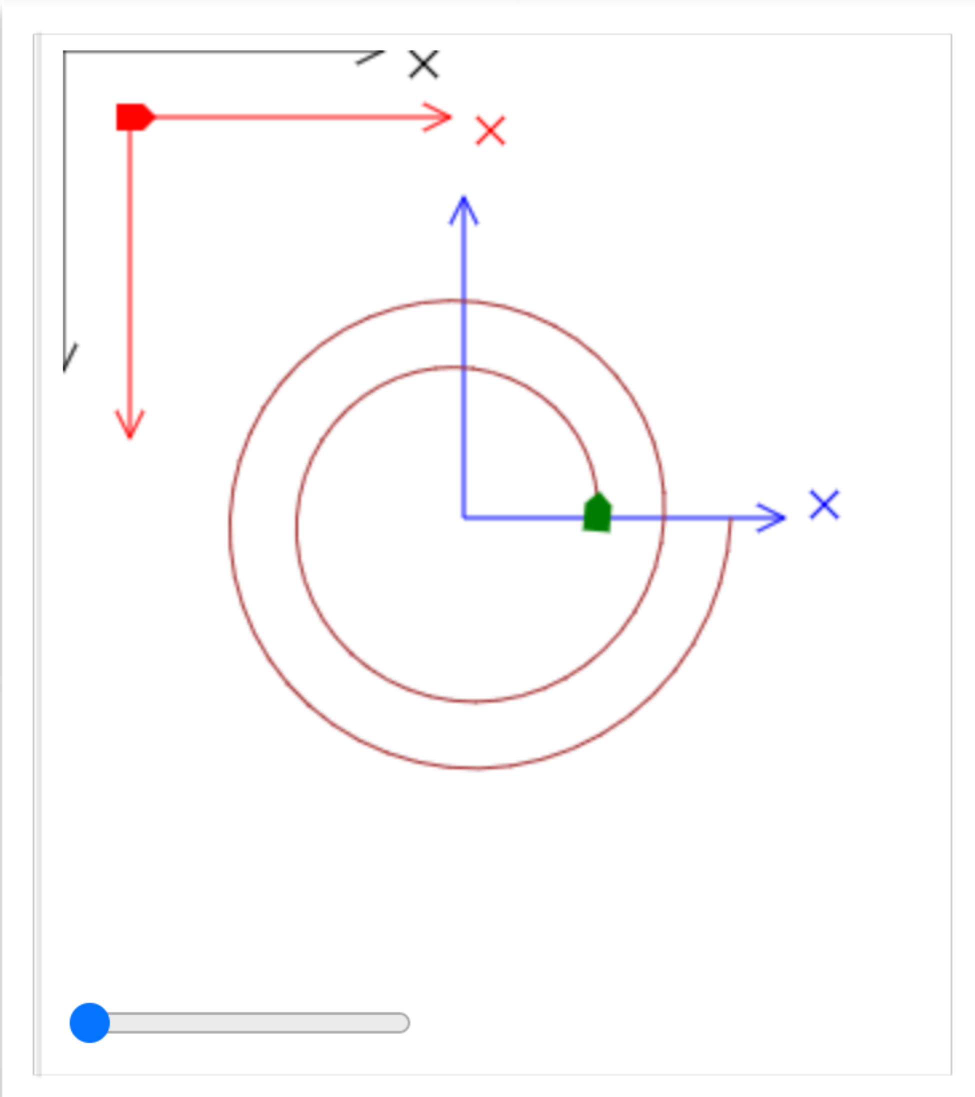
$$\mathcal{C}'(t) = \begin{pmatrix} R'(t)\cos(2\pi t) - R(t)2\pi\sin(2\pi t) \\ R'(t)\sin(2\pi t) + R(t)2\pi\cos(2\pi t) \end{pmatrix}$$

Implementation of the tangent vector (also parametric!)

Implementation example : A spiral curve

jsbin.com/meliyax

Week5/Demo1



JavaScript

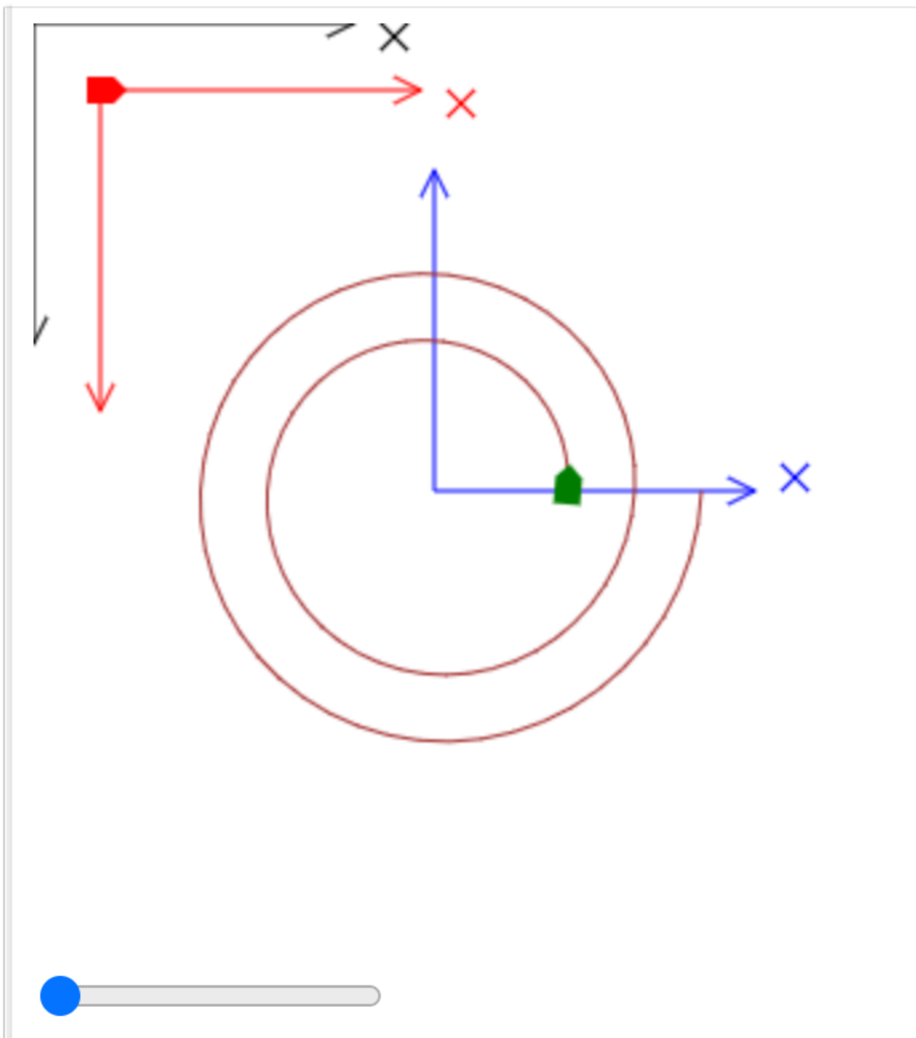
```
[...]
    // make sure you understand these

    drawAxes("black", mat3.create());

    var Tred_to_canvas = mat3.create();
    mat3.fromTranslation(Tred_to_canvas,[25,25]);
    drawAxes("red",Tred_to_canvas);
    drawObject("red",Tred_to_canvas);

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[150,175]);
    mat3.scale(Tblue_to_canvas,Tblue_to_canvas,[1,-1]); // Flip the Y-axis
    drawAxes("blue",Tblue_to_canvas);

    drawTrajectory(0.0,2.0,100,Cspiral,Tblue_to_canvas,"brown");
    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,Cspiral(tParam));
    var tangent = Cspiral_tangent(tParam);
    var angle = Math.atan2(tangent[1],tangent[0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,angle);
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    // drawAxes("green",Tgreen_to_canvas); // Un-comment this to view axes
    drawObject("green",Tgreen_to_canvas);

[...]
```

Draw a slightly different object, with a shape that suggests orientation (look at "red" system for reference)

Implementation example : A spiral curve

jsbin.com/meliyax

Week5/Demo1



JavaScript

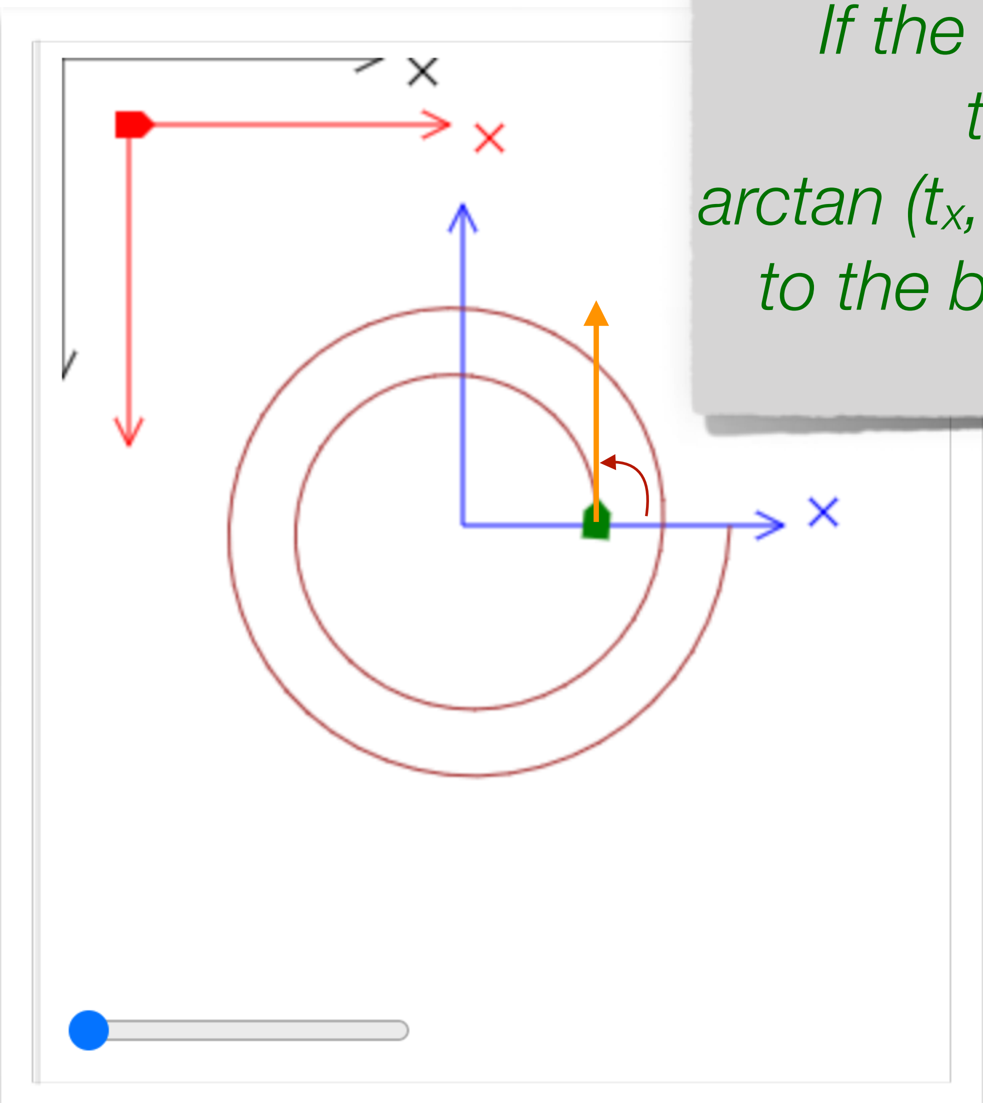
```
[...]
    // make sure you understand these

    drawAxes("black", mat3.create());

    var Tred_to_canvas = mat3.create();
    mat3.fromTranslation(Tred_to_canvas,[25,25]);
    drawAxes("red",Tred_to_canvas);
    drawObject("red",Tred_to_canvas);

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[150,175]);
    mat3.scale(Tblue_to_canvas,Tblue_to_canvas,[1,-1]); // Flip the Y-axis
    drawAxes("blue",Tblue_to_canvas);

    drawTrajectory(0.0,2.0,100,Cspiral,Tblue_to_canvas,"brown");
    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,Cspiral(tParam));
    var tangent = Cspiral_tangent(tParam);
    var angle = Math.atan2(tangent[1],tangent[0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,angle);
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    // drawAxes("green",Tgreen_to_canvas); // Un-comment this to view axes
    drawObject("green",Tgreen_to_canvas);

[...]
```

Compute angle of rotation, from the tangent vector (this will be aligned with the "x" axis vector)

Implementation example : A spiral curve

jsbin.com/meliyax

Week5/Demo1



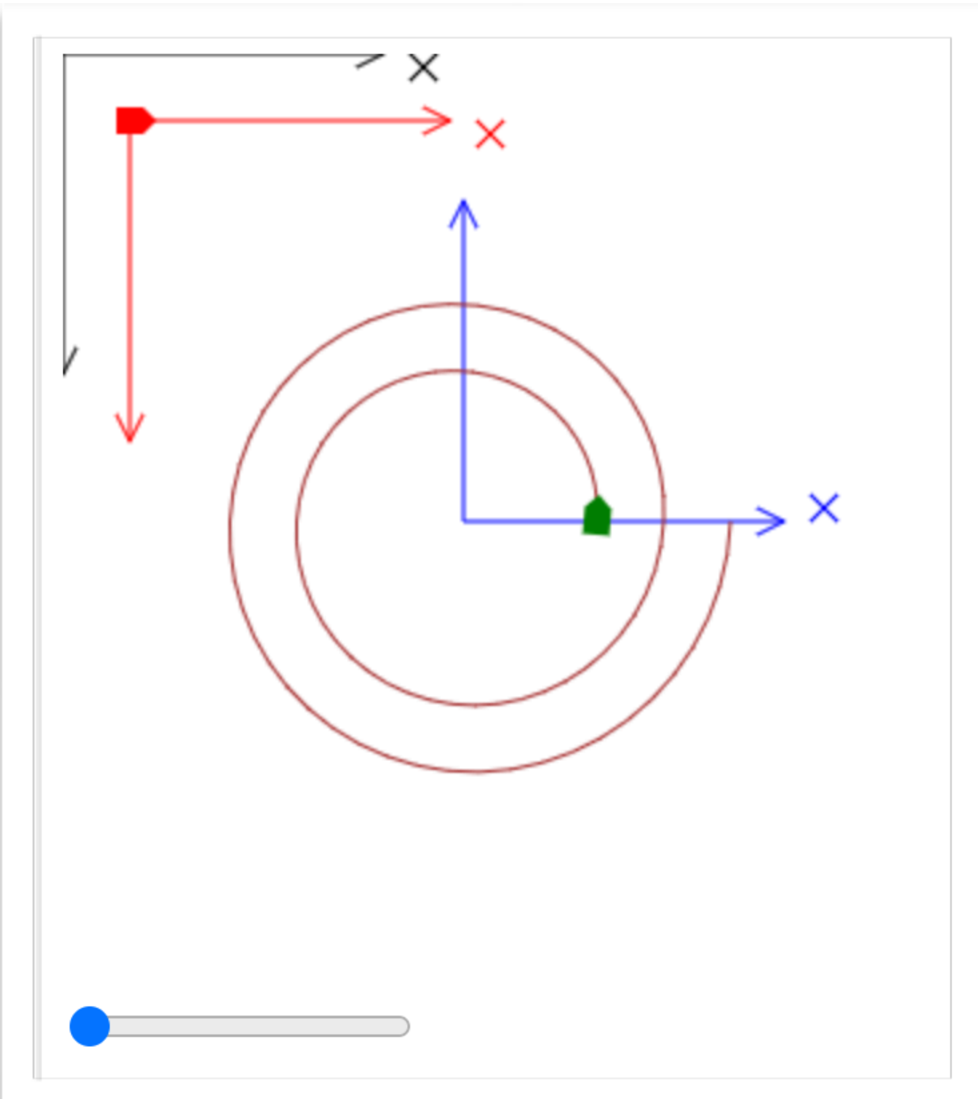
If the tangent vector is (t_x,t_y) the angle it forms with the blue coordinate system (as drawn) is arctan (t_x,t_y) … this is the rotation that has to be appended to the blue system (after translation) to have it oriented properly with the curve

```
    drawAxes("black", mat3.create());

    var Tred_to_canvas = mat3.create();
    mat3.fromTranslation(Tred_to_canvas,[25,25]);
    drawAxes("red",Tred_to_canvas);
    drawObject("red",Tred_to_canvas);

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[150,175]);
    mat3.scale(Tblue_to_canvas,Tblue_to_canvas,[1,-1]); // Flip the Y-axis
    drawAxes("blue",Tblue_to_canvas);

    drawTrajectory(0.0,2.0,100,Cspiral,Tblue_to_canvas,"brown");
    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,Cspiral(tParam));
    var tangent = Cspiral_tangent(tParam);
    var angle = Math.atan2(tangent[1],tangent[0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,angle);
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    // drawAxes("green",Tgreen_to_canvas); // Un-comment this to view axes
    drawObject("green",Tgreen_to_canvas);

[...]
```

Implementation example : A spiral curve

jsbin.com/meliyax

Week5/Demo1



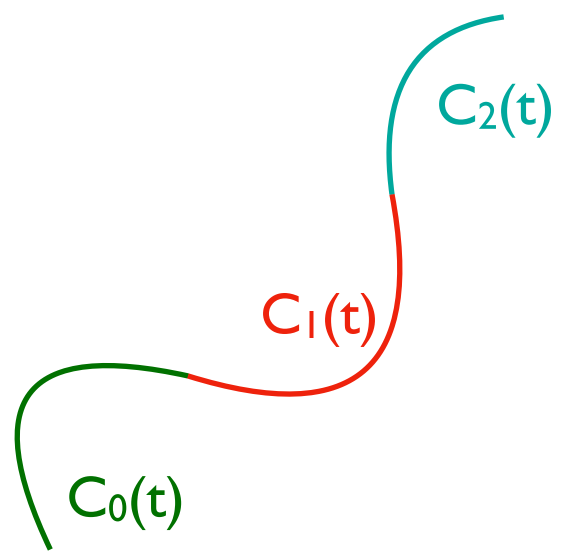
JavaScript

```
[...]
    // make sure you understand these

    drawAxes("black", mat3.create());

    var Tred_to_canvas = mat3.create();
    mat3.fromTranslation(Tred_to_canvas,[25,25]);
    drawAxes("red",Tred_to_canvas);
    drawObject("red",Tred_to_canvas);

    var Tblue_to_canvas = mat3.create();
    mat3.fromTranslation(Tblue_to_canvas,[150,175]);
    mat3.scale(Tblue_to_canvas,Tblue_to_canvas,[1,-1]); // Flip the Y-axis
    drawAxes("blue",Tblue_to_canvas);

    drawTrajectory(0.0,2.0,100,Cspiral,Tblue_to_canvas,"brown");
    var Tgreen_to_blue = mat3.create();
    mat3.fromTranslation(Tgreen_to_blue,Cspiral(tParam));
    var tangent = Cspiral_tangent(tParam);
    var angle = Math.atan2(tangent[1],tangent[0]);
    mat3.rotate(Tgreen_to_blue,Tgreen_to_blue,angle);
    var Tgreen_to_canvas = mat3.create();
    mat3.multiply(Tgreen_to_canvas, Tblue_to_canvas, Tgreen_to_blue);
    // drawAxes("green",Tgreen_to_canvas); // Un-comment this to view axes
    drawObject("green",Tgreen_to_canvas);

[...]
```

Draw the axes in the "moving object coordinate system" if you need more visual orientation!

Piecewise parametric curves

- It can be convenient to define different extents of a parametric curve using different formulas (rather than attempting to over-complicate a *single* formula)

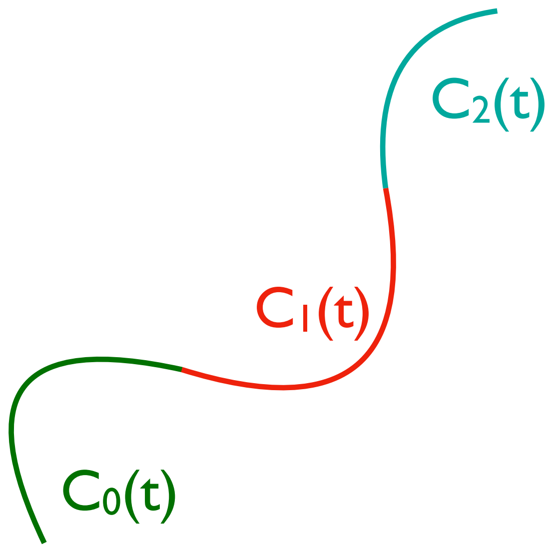


$$\mathcal{C}(t) = \begin{cases} \mathcal{C}_0(t), & t \in [t_0, t_1] \\ \mathcal{C}_1(t), & t \in [t_1, t_2] \\ \vdots & \\ \mathcal{C}_{N-1}(t), & t \in [t_{N-1}, t_N] \end{cases}$$

(let's not dwell right now on what function applies to the common point of these intervals … not so important for drawing the bulk of the curve, anyways)

Piecewise parametric curves

- Derivatives of such piecewise-defined functions are simply obtained by differentiating the component formulas …



$$\mathcal{C}'(t) = \begin{cases} \mathcal{C}_0'(t), & t \in [t_0, t_1] \\ \mathcal{C}_1'(t), & t \in [t_1, t_2] \\ \vdots & \\ \mathcal{C}_{N-1}'(t), & t \in [t_{N-1}, t_N] \end{cases}$$

Piecewise parametric curves

jsbin.com/vupevab

Week5/Demo2

- Implementing such curve definitions typically reduces to an "if-statement" in JavaScript …

$$\mathcal{C}(t) = \begin{cases} \mathcal{C}_0(t), & t \in [t_0, t_1] \\ \mathcal{C}_1(t), & t \in [t_1, t_2] \\ \vdots & \\ \mathcal{C}_{N-1}(t), & t \in [t_{N-1}, t_N] \end{cases}$$

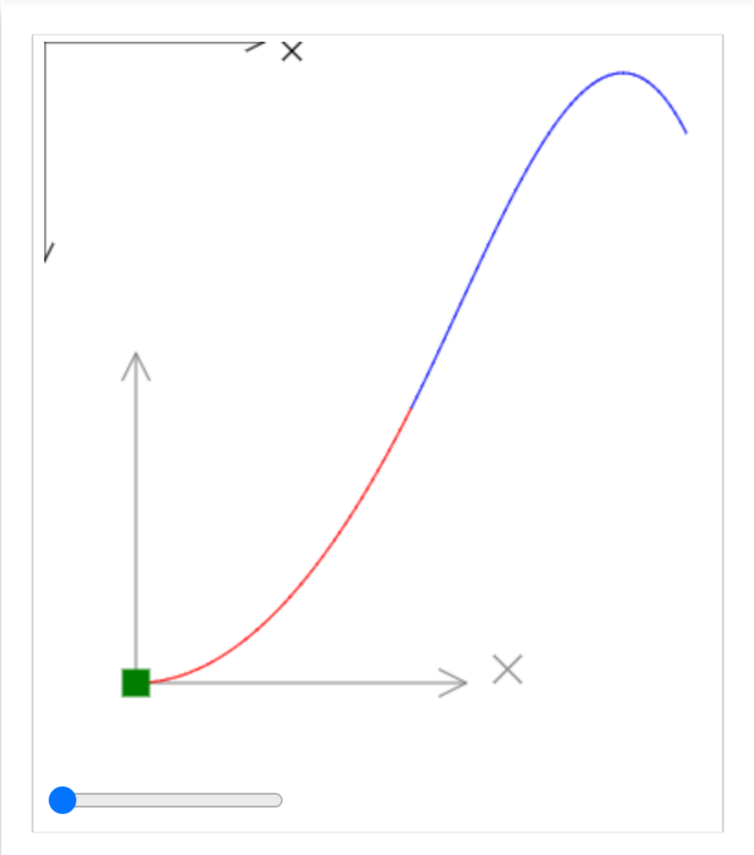
JavaScript

```
[...]
    var C0 = function(t) {
            var x = t;
            var y = t*t;
            return [x,y];
    }

[...]
    var C1e = function(t) { // C2 continuity at t=1
            var x = t;
            var y = t*t-2*(t-1)*(t-1)*(t-1);
            return [x,y];
    }

    var C1 = C1e;

    var Ccomp = function(t) {
            if(t<1) {
        return C0(t);
            }else{
        return C1(t);
            }
    }
[...]
```

Piecewise parametric curves

jsbin.com/vupevab

Week5/Demo2



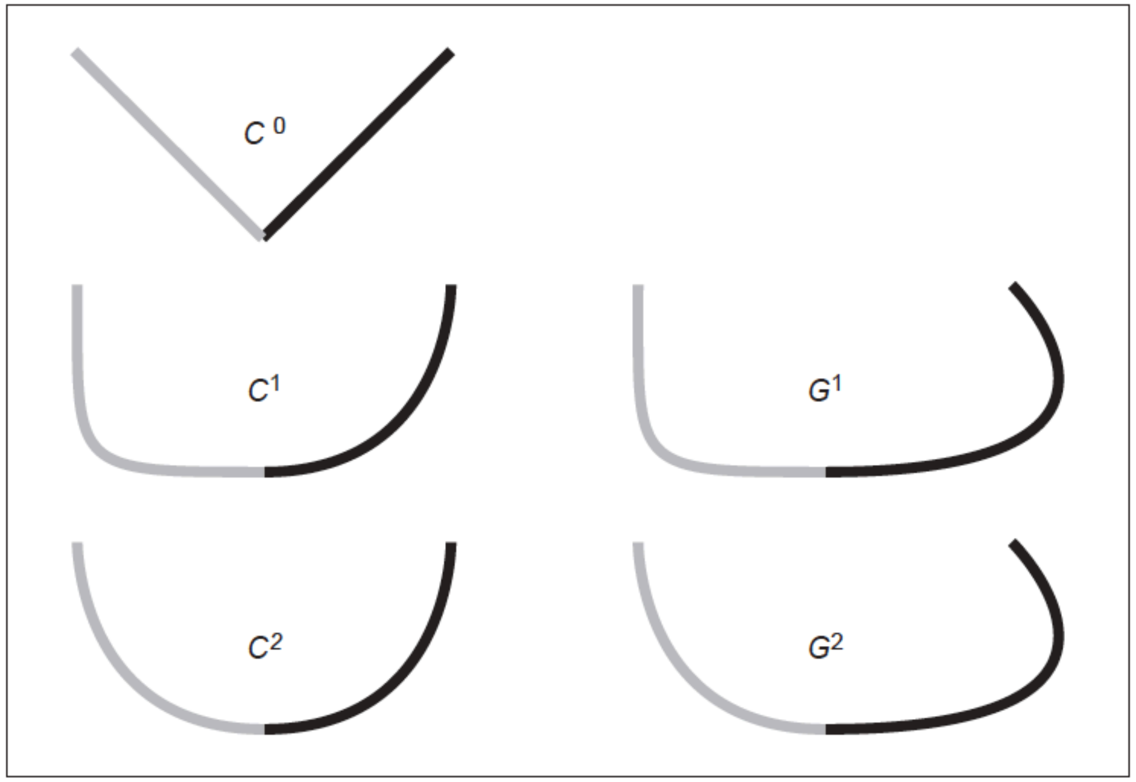
Demo incorporates several distinct versions of piecewise-defined curves! (Set variable "C1" to appropriate version to try them all!)

JavaScript

```
[...]
    var C0 = function(t) {
            var x = t;
            var y = t*t;
            return [x,y];
    }

    var C1a = function(t) { // discontinuity at t=1
            var x = t*(t-1);
            var y = t;
            return [x,y];
    }

    var C1b = function(t) { // C0 continuity at t=1
            var x = t*t-3*t+3;
            var y = t;
            return [x,y];
    }

    var C1c = function(t) { // C1 continuity at t=1
            var x = t;
            var y = -t*t+4*t-2;
            return [x,y];
    }

    var C1d = function(t) { // G1 continuity at t=1
            var x = 0.25*(t+3);
            var y = -0.0625*(t+3)*(t+3)+(t+3)-2;
            return [x,y];
    }

    var C1e = function(t) { // C2 continuity at t=1
            var x = t;
            var y = t*t-2*(t-1)*(t-1)*(t-1);
            return [x,y];
    }

    var C1 = C1e;

    var Ccomp = function(t) {
            if(t<1) {
        return C0(t);
            }else{
        return C1(t);
            }
    }
[...]
```

Continuity

- When the formula of a piecewise-defined parametric curve switches (from an interval to the next), *continuity* becomes a point of concern …



$$\mathcal{C}(t) = \begin{cases} \mathcal{C}_0(t), & t \in [t_0, t_1] \\ \mathcal{C}_1(t), & t \in [t_1, t_2] \end{cases}$$

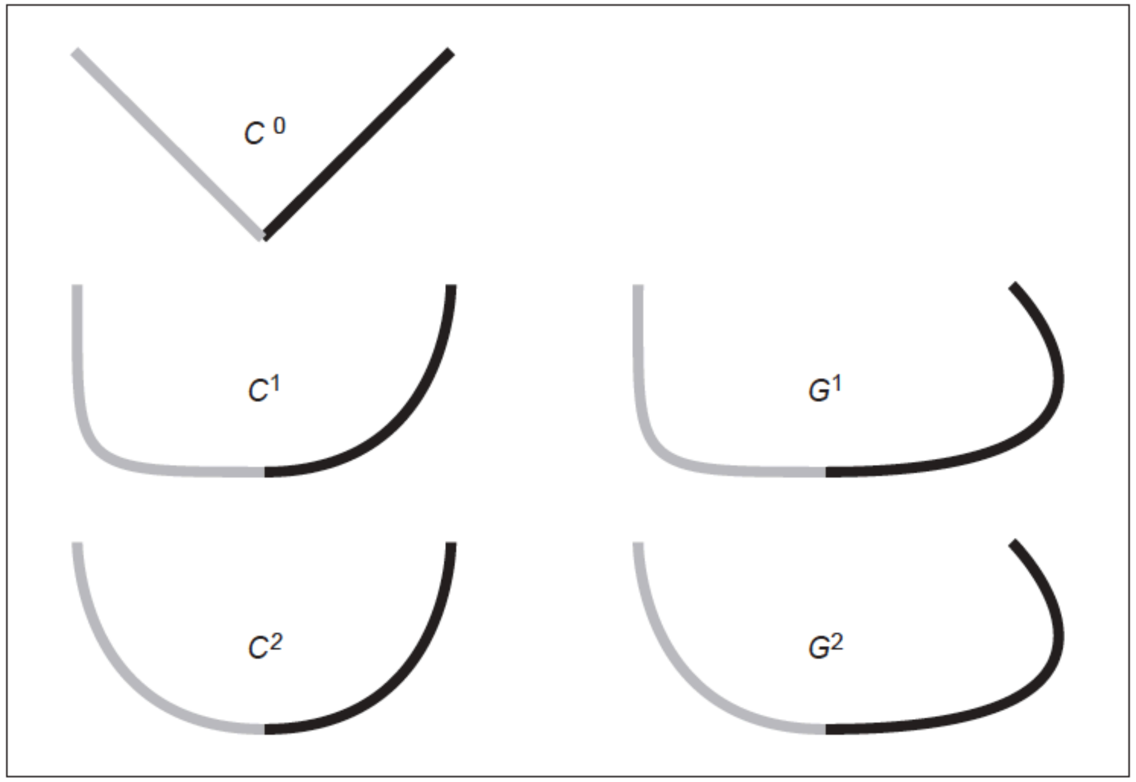
Continuity

- When the formula of a piecewise-defined parametric curve switches (from an interval to the next), *continuity* becomes a point of concern …

It is possible for the two curves
to *not meet at all* at the point
where the parameterization switches over!

Such a piecewise-defined curve
will be called discontinuous at the
parameter value of such transition.

(Set var Ccomp=C1a in our demo)



$$\mathcal{C}_0(t_1) \neq \mathcal{C}_1(t_1)$$

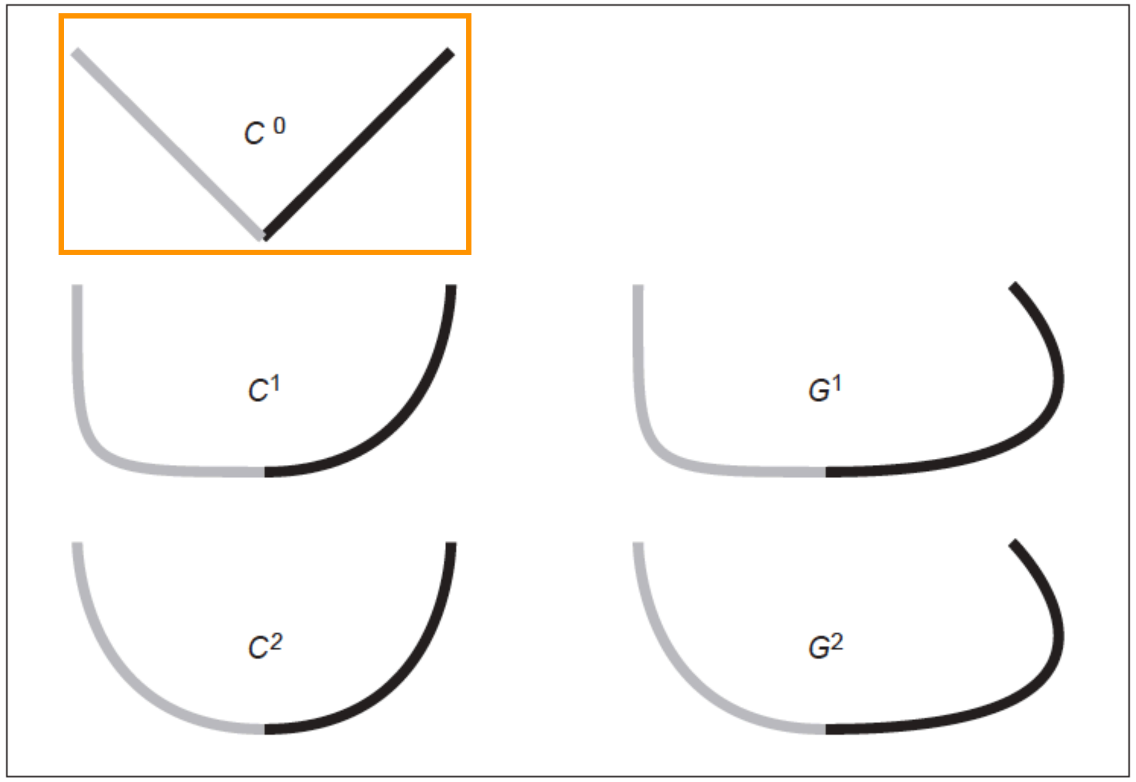
$$\mathcal{C}(t) = \begin{cases} \mathcal{C}_0(t), & t \in [t_0, t_1] \\ \mathcal{C}_1(t), & t \in [t_1, t_2] \end{cases}$$

Continuity

- When the formula of a piecewise-defined parametric curve switches (from an interval to the next), *continuity* becomes a point of concern …

A curve is said to be ***C0-continuous*** if the values of C(t) match on each side of the parametric transition

(intuitively: the curve is not broken, although it might have a "kink")



$$\mathcal{C}_0(t_1) = \mathcal{C}_1(t_1)$$

$$\mathcal{C}(t) = \begin{cases} \mathcal{C}_0(t), & t \in [t_0, t_1] \\ \mathcal{C}_1(t), & t \in [t_1, t_2] \end{cases}$$

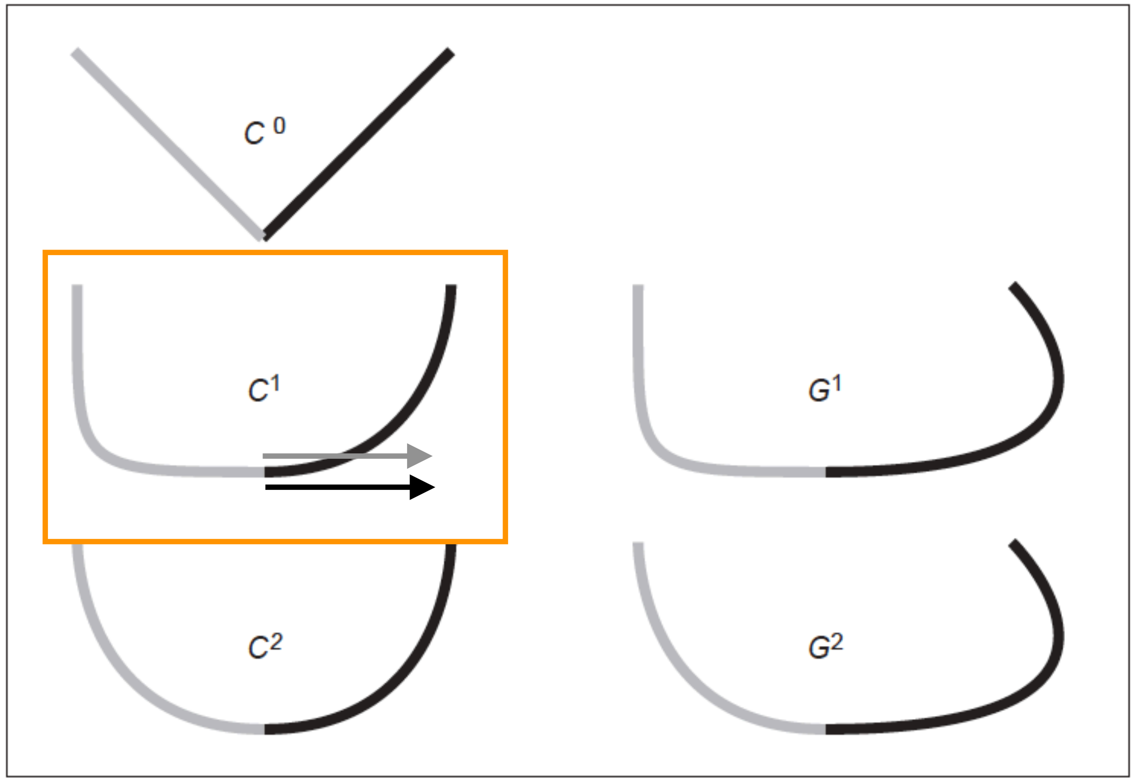
Continuity

- When the formula of a piecewise-defined parametric curve switches (from an interval to the next), *continuity* becomes a point of concern …

A curve is said to be ***C1-continuous*** if both the values of C(t) and the tangent vector match on each side of the parametric transition

(intuitively: the curve is connected, and in a way that there's no visible "kink" at the joint point)

$$\mathcal{C}_0(t_1) = \mathcal{C}_1(t_1)$$
$$\mathcal{C}_0'(t_1) = \mathcal{C}_1'(t_1)$$



$$\mathcal{C}(t) = \begin{cases} \mathcal{C}_0(t), & t \in [t_0, t_1] \\ \mathcal{C}_1(t), & t \in [t_1, t_2] \end{cases}$$

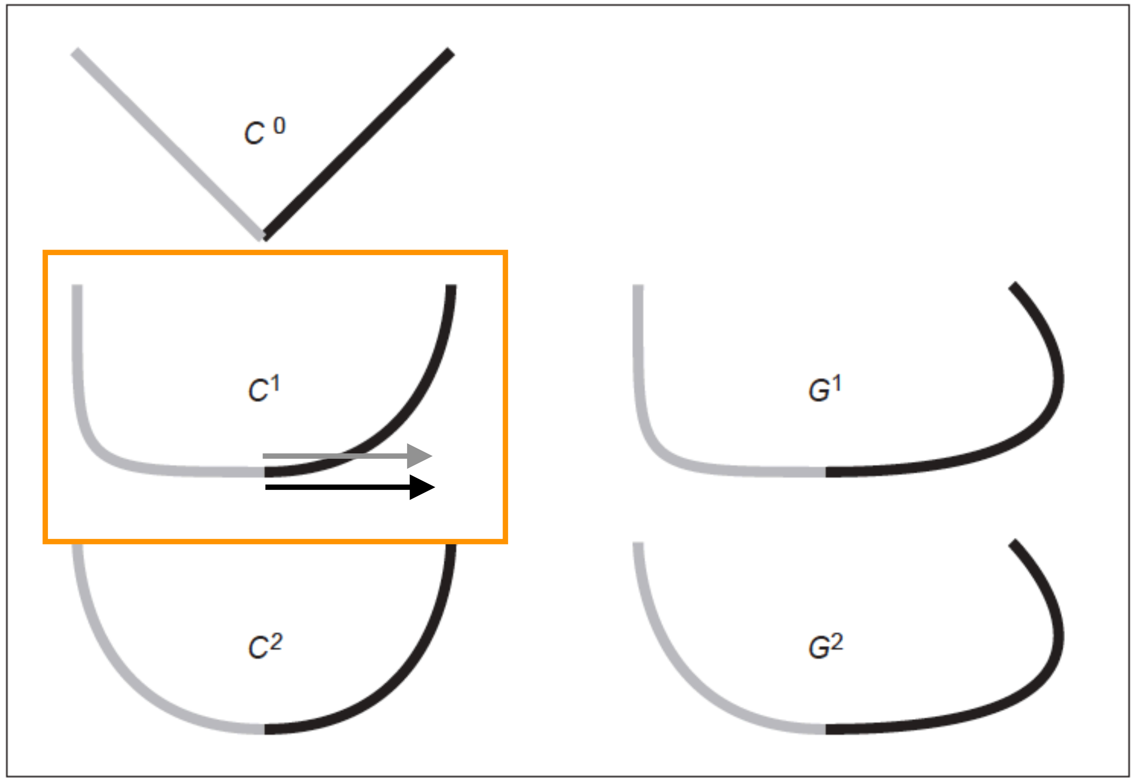
Continuity

- When the formula of a piecewise-defined parametric curve switches (from an interval to the next), *continuity* becomes a point of concern …

Additional intuition: If you animate an object along a C1-continuous curve the velocity will be continuous

Special case (no need to worry too much about this in practice …)

If $C_0(t)=C_1(t)=0$, it is still possible for the curve to show a sharp turn …



$$\mathcal{C}_0(t_1) = \mathcal{C}_1(t_1)$$
$$\mathcal{C}_0'(t_1) = \mathcal{C}_1'(t_1)$$

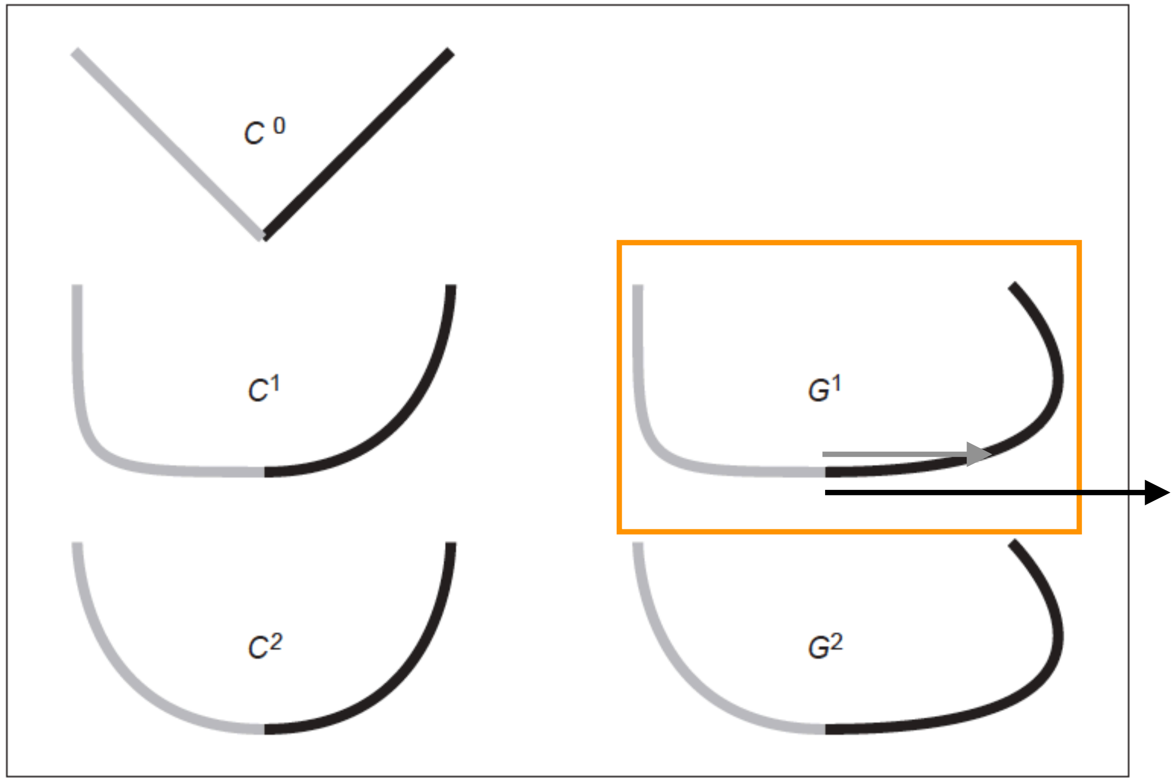
$$\mathcal{C}(t) = \begin{cases} \mathcal{C}_0(t), & t \in [t_0, t_1] \\ \mathcal{C}_1(t), & t \in [t_1, t_2] \end{cases}$$

Continuity

- When the formula of a piecewise-defined parametric curve switches (from an interval to the next), *continuity* becomes a point of concern …

A curve is said to be ***G1-continuous***
if the values of C(t) match and the
tangent vectors are multiples of each other
on each side of the parametric transition

(intuitively: there's no visible kink, but the parameterization "speeds-up" after the transition point)



$$\mathcal{C}_0(t_1) = \mathcal{C}_1(t_1)$$
$$\mathcal{C}_0'(t_1) = \lambda \mathcal{C}_1'(t_1)$$

$$\mathcal{C}(t) = \begin{cases} \mathcal{C}_0(t), & t \in [t_0, t_1] \\ \mathcal{C}_1(t), & t \in [t_1, t_2] \end{cases}$$

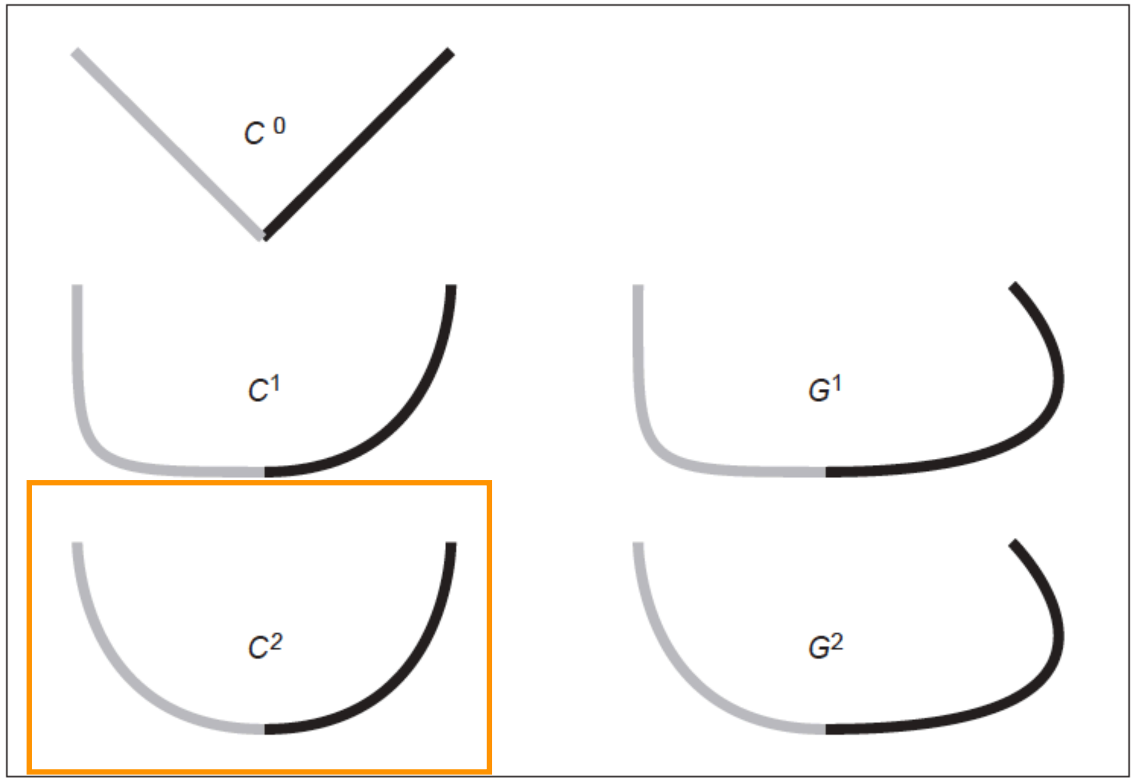
Continuity

- When the formula of a piecewise-defined parametric curve switches (from an interval to the next), *continuity* becomes a point of concern …

A curve is said to be ***C2-continuous***
if the values of C(t), C'(t), and C"(t) match
on each side of the parametric transition

(intuitively: matching locations, tangents
and *curvatures*)

$$\mathcal{C}_0(t_1) = \mathcal{C}_1(t_1)$$
$$\mathcal{C}_0'(t_1) = \mathcal{C}_1'(t_1)$$
$$\mathcal{C}_0''(t_1) = \mathcal{C}_1''(t_1)$$



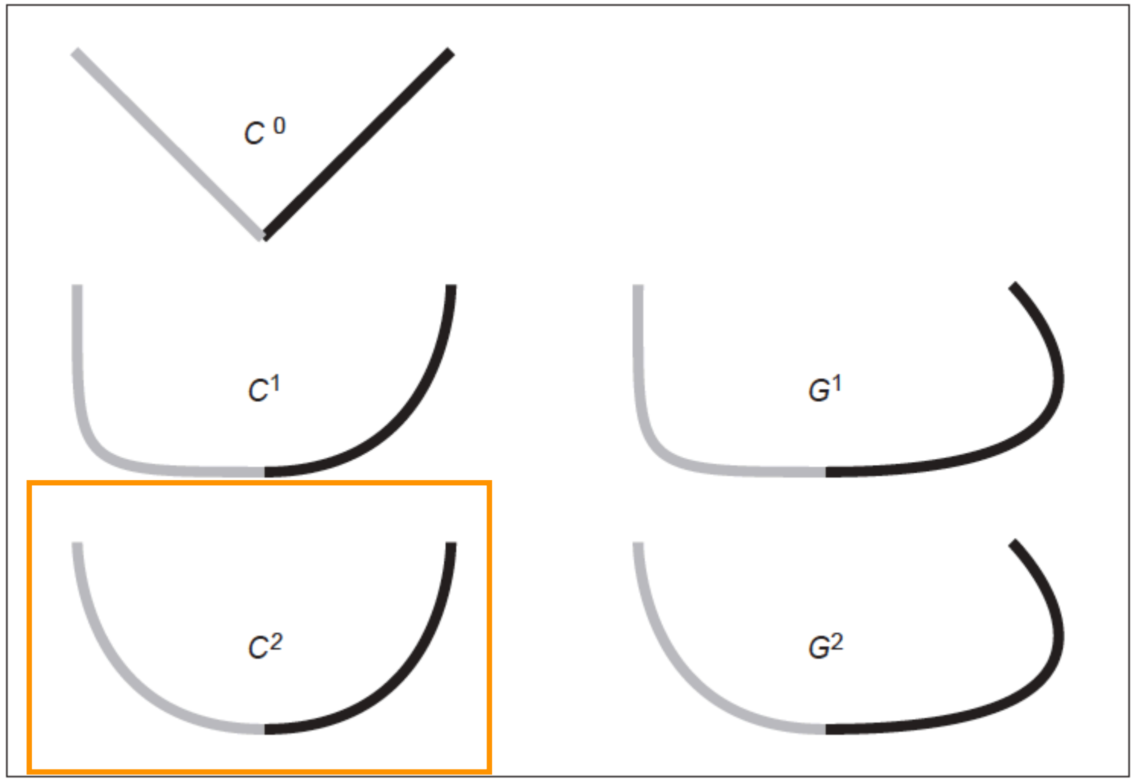
$$\mathcal{C}(t) = \begin{cases} \mathcal{C}_0(t), & t \in [t_0, t_1] \\ \mathcal{C}_1(t), & t \in [t_1, t_2] \end{cases}$$

Continuity

- When the formula of a piecewise-defined parametric curve switches (from an interval to the next), *continuity* becomes a point of concern …

In practice … C2-continuity looks so smooth and continuous that most non-experts would not be able to discern any higher degree of continuity …

$$\mathcal{C}_0(t_1) = \mathcal{C}_1(t_1)$$
$$\mathcal{C}_0'(t_1) = \mathcal{C}_1'(t_1)$$
$$\mathcal{C}_0''(t_1) = \mathcal{C}_1''(t_1)$$



$$\mathcal{C}(t) = \begin{cases} \mathcal{C}_0(t), & t \in [t_0, t_1] \\ \mathcal{C}_1(t), & t \in [t_1, t_2] \end{cases}$$

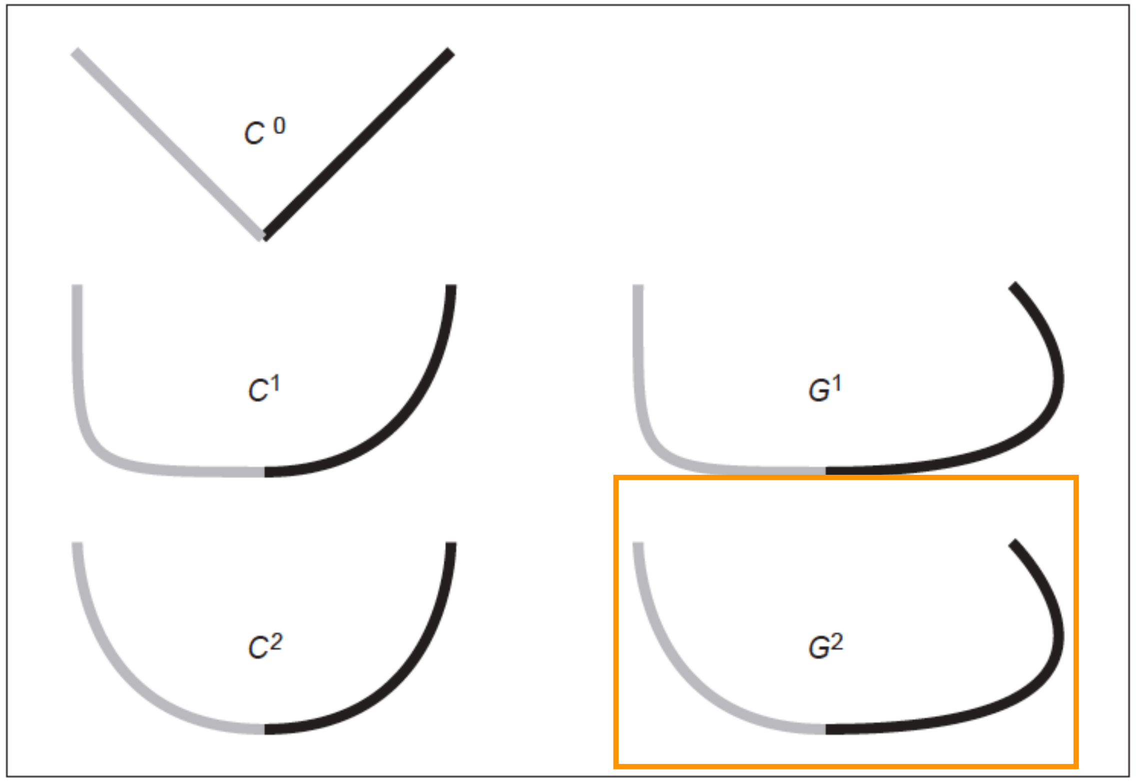
Continuity

- When the formula of a piecewise-defined parametric curve switches (from an interval to the next), *continuity* becomes a point of concern …

G2-continuity relaxes just the 2nd derivative match into just being a multiple of each other …

Almost as smooth as C2 for many practical purposes … we will not focus on G2-continuity much …

$$\mathcal{C}_0(t_1) = \mathcal{C}_1(t_1)$$
$$\mathcal{C}_0'(t_1) = \mathcal{C}_1'(t_1)$$
$$\mathcal{C}_0''(t_1) = \lambda\mathcal{C}_1''(t_1)$$



$$\mathcal{C}(t) = \begin{cases} \mathcal{C}_0(t), & t \in [t_0, t_1] \\ \mathcal{C}_1(t), & t \in [t_1, t_2] \end{cases}$$

Continuity

jsbin.com/vupevab

Week5/Demo2

- When examining continuity, we typically test what is the <u>highest</u> degree of continuity we can prove.
- Exercise to practice: Can you determine the degree of continuity for the curves ("C0 relative to "C1x", where x=a,b,c,d,e) in this demo program?
- (We'll do quite a few more of these on paper, including in preparation for the midterm)

(Piecewise) Polynomial curves

- Before we start … a significant comment on notation (for purposes of alignment with your textbook)
- We will follow the convention that point coordinates (i.e. the curve point location C(t), or tangent C'(t)) are expressed as <u>row vectors!</u>

$$\mathcal{C}(t) = \left[\begin{array}{cc} x(t) & y(t) \end{array}\right]$$

$$\mathcal{C}'(t) = \left[\begin{array}{cc} x'(t) & y'(t) \end{array}\right]$$

(Piecewise) Polynomial curves

$$\mathcal{C}(t) = \left[\begin{array}{cc} x(t) & y(t) \end{array} \right]$$

- In earlier examples, we crafted curves using "custom" expressions for x(t) and y(t) … not an easy way to maneuver the curve around!
- Idea: use polynomial expressions to define x(t), y(t)!

$$x(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_N t^N$$
$$y(t) = b_0 + b_1 t + b_2 t^2 + \cdots + b_N t^N$$

(Piecewise) Polynomial curves

$$\mathcal{C}(t) = \begin{bmatrix} x(t) & y(t) \end{bmatrix}$$

$$x(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_N t^N$$

$$y(t) = b_0 + b_1 t + b_2 t^2 + \cdots + b_N t^N$$

- Benefits :
  - The "knobs" we can turn to maneuver the curve around are clearly defined (coefficients a_k and b_k)
  - Polynomials <u>can</u> represent the simplest curve: a line segment (it only requires linear polynomials)
  - Manipulation of these curves can be facilitated by matrix algebra (we'll see this next!)
  - We can adjust polynomial cubes for desired features (where to start and stop, tangents, and continuity)

(Piecewise) Polynomial curves

$$\mathcal{C}(t) = \begin{bmatrix} x(t) & y(t) \end{bmatrix}$$

$$x(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_N t^N$$

$$y(t) = b_0 + b_1 t + b_2 t^2 + \cdots + b_N t^N$$

Matrix representation

$$\mathcal{C}(t) = \begin{bmatrix} x(t) & y(t) \end{bmatrix} = \begin{bmatrix} 1 & t & t^2 \cdots & t^N \end{bmatrix} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_N & b_N \end{bmatrix}$$

(Piecewise) Polynomial curves

$$\mathcal{C}(t) = \left[\begin{array}{cc} x(t) & y(t) \end{array} \right]$$

$$x(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_N t^N$$

$$y(t) = b_0 + b_1 t + b_2 t^2 + \cdots + b_N t^N$$

Matrix representation
(switching to textbook notation)

$$\mathbf{f}(u) = \left[\begin{array}{cc} x(u) & y(u) \end{array} \right] = \underbrace{\left[\begin{array}{cccc} 1 & u & u^2 \cdots & u^N \end{array} \right]}_{\mathbf{u}} \underbrace{\left[\begin{array}{cc} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_N & b_N \end{array} \right]}_{\mathbf{A}}$$

(Piecewise) Polynomial curves

$$\mathbf{f}(u) = \left[\begin{array}{cc} x(u) & y(u) \end{array}\right] = \underbrace{\left[\begin{array}{cccc} 1 & u & u^2 \cdots & u^N \end{array}\right]}_{\mathbf{u}} \underbrace{\left[\begin{array}{cc} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_N & b_N \end{array}\right]}_{\mathbf{A}}$$

Tangents:

$$\mathbf{f}'(u) = \underbrace{\left[\begin{array}{cccc} 0 & 1 & 2u \cdots & Nu^{N-1} \end{array}\right]}_{\mathbf{u}'} \underbrace{\left[\begin{array}{cc} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_N & b_N \end{array}\right]}_{\mathbf{A}}$$

We can solve for the matrix A if we have a specification of what we want the curve to be able to do!

(we'll see how to do that in the next lectures!)