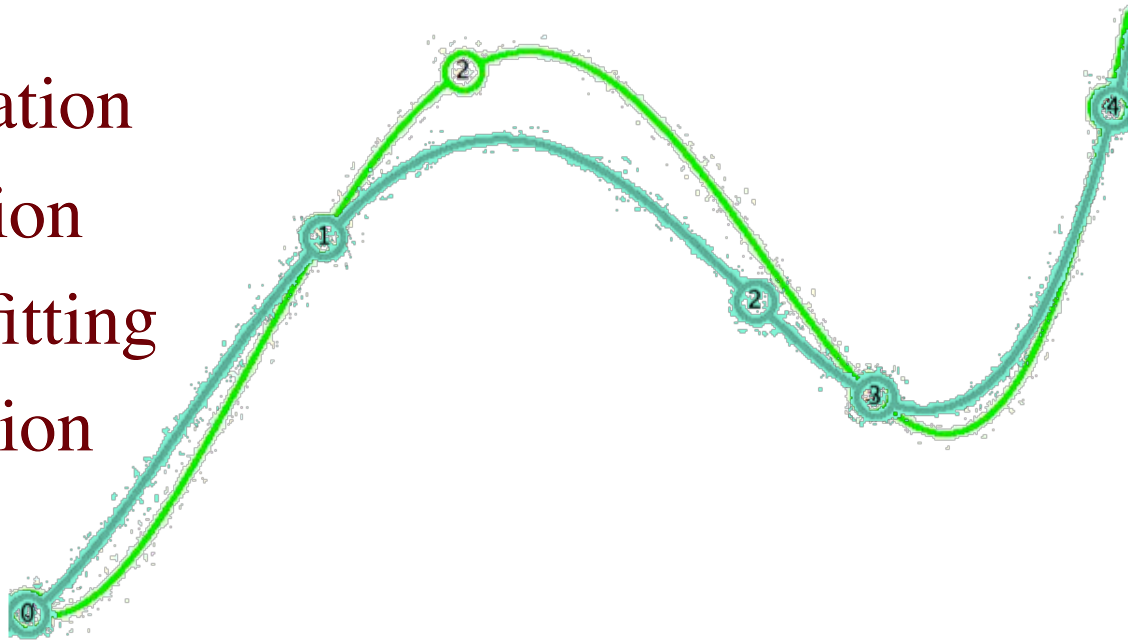
Lagrange interpolation

Jarek Rossignac

What is covered

- Linear interpolation
- Polynomial interpolation
- Neville' s algorithm
- Proof
- Recursive implementation
- Iterative implementation
- Application to curve fitting
- Application to animation



Motivation

Where/when is this useful

- **Interpolating** data samples by a smooth function
- Defining a **curve** through several points
- Designing a **motion** through timed keyframe points
- Editing time-evolution of **animation** parameters

Why is it important

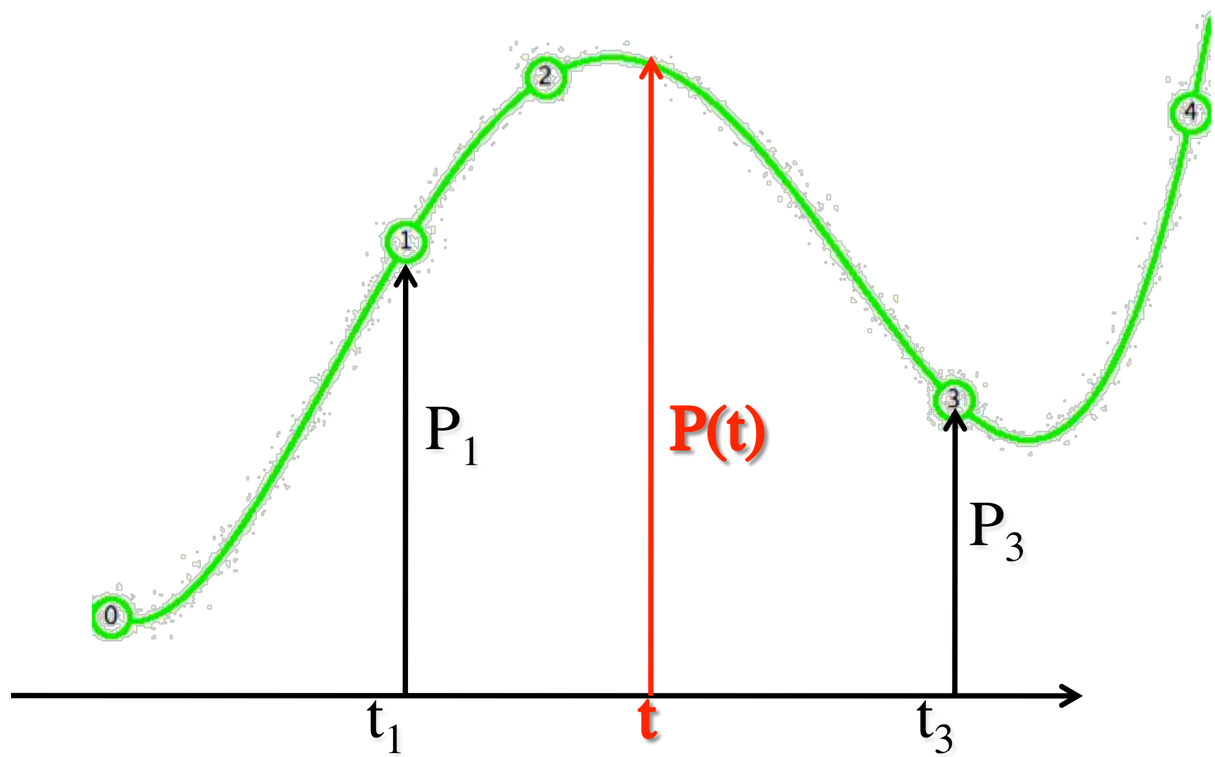
- Precisely measured and carefully designed key points should be interpolated
- Most applications need a smooth interpolation

Problem statement

Let P(t) be a polynomial

Given a set of n constraints: $P(t_i)=P_i$, for i=0, 1…n–1

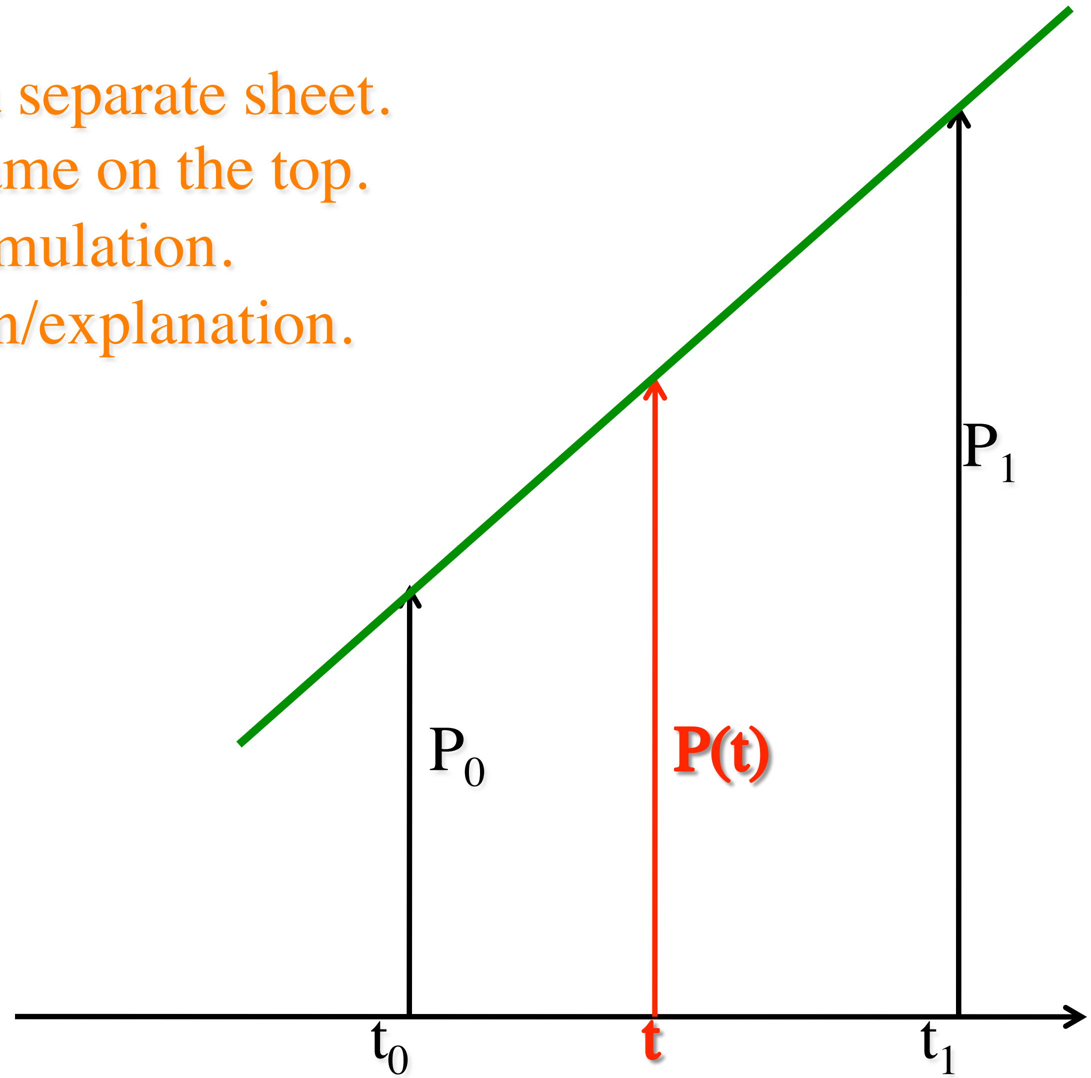
compute the value P(t) for any given t



Linear case (n=2)

- Given P_0, t_0, P_1, t_1, and t, compute P(t)

Write your solution on a separate sheet.
Put your first and last name on the top.
Strive for an elegant formulation.
Write a brief justification/explanation.



SOLUTIONS?

Solution of the linear case (n=2)

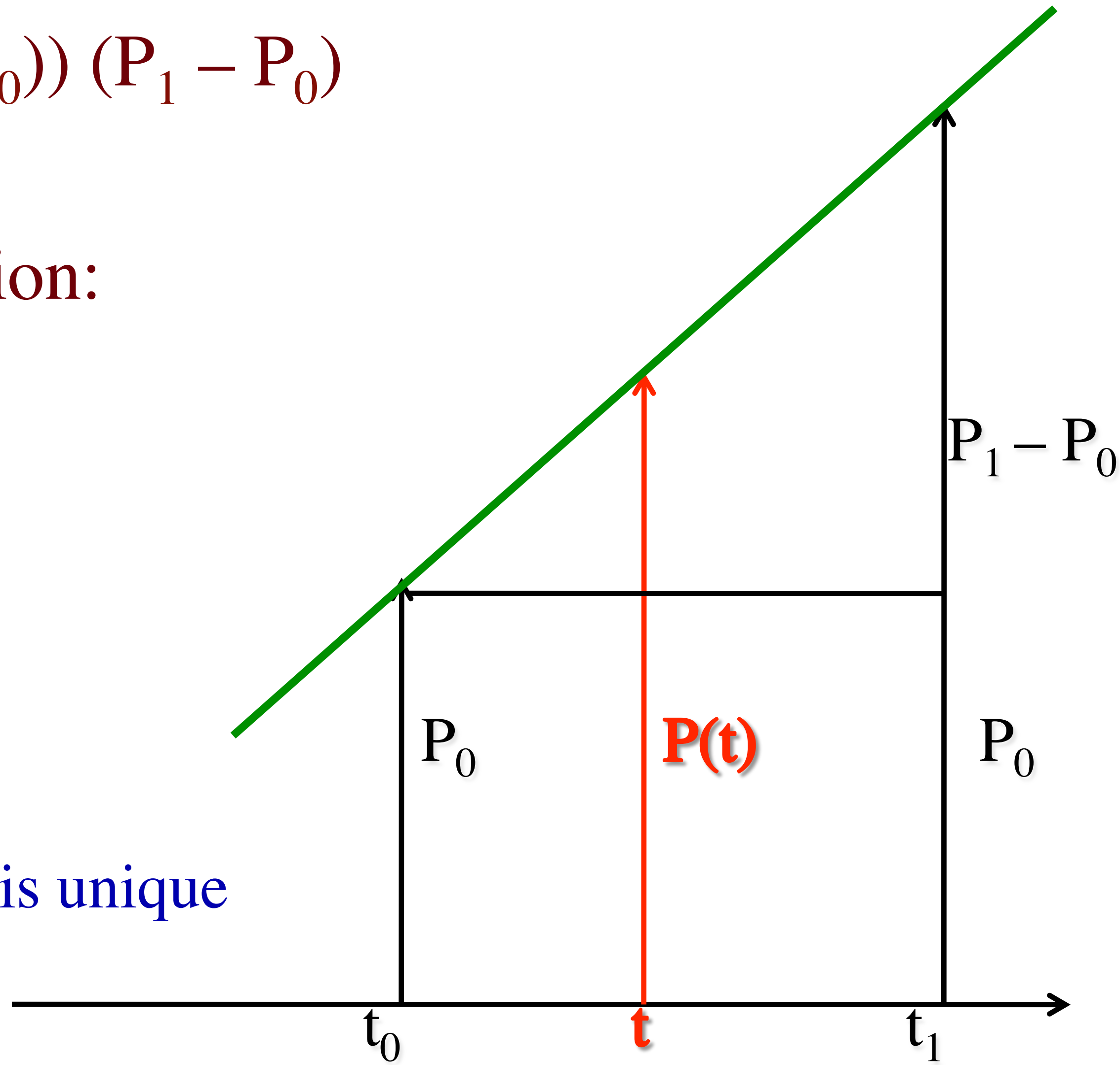
- Intuitive and semantically correct solution:

$$P(t) = P_0 + ((t-t_0) / (t_1-t_0)) (P_1 - P_0)$$

- Convenient formulation:

$$P(t) = (t_1-t) / (t_1-t_0)\ P_0 + (t-t_0) / (t_1-t_0)\ P_1$$

- Justification:
  - $P(t_0)=P_0$ and $P(t_1)=P_1$
  - line through 2 points is unique



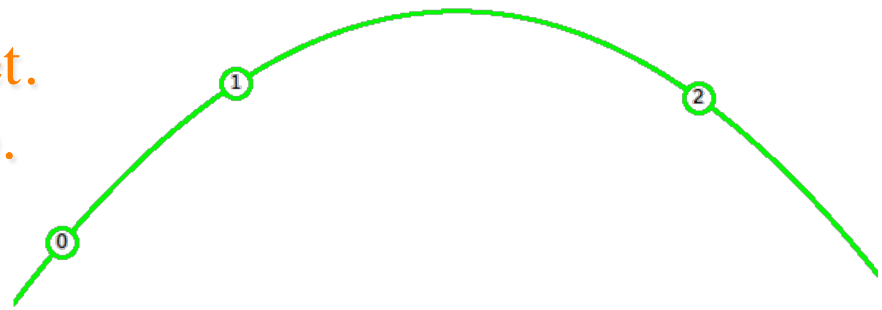
Let's turn this into a function for later

```
float P( t , t0 , P0 , t1 , P1 ) {
   return  (t1–t) / (t1–t0) * P0 + (t–t0) / (t1–t0) * P1 ; }
```

Case where n=3

- Given P_0, t_0, P_1, t_1, P_2, t_2, and t, **compute** P(t)
  - What **kind** of a curve is P(t)?

Write your solution on a separate sheet.
Put your first and last name on the top.
Strive for an elegant formulation.
Write a brief justification.



Exchange solutions with a neighbor.
Read your neighbor's solution and ask for clarifications.
Discuss and agree on a common solution.

Solutions?

Do you know your neighbor's name?

Key idea for n=3

- Progressively blend between linear interpolations:

  $G(t)=P(t,t_0,P_0,t_1, P_1)$ interpolates P_0 and P_1

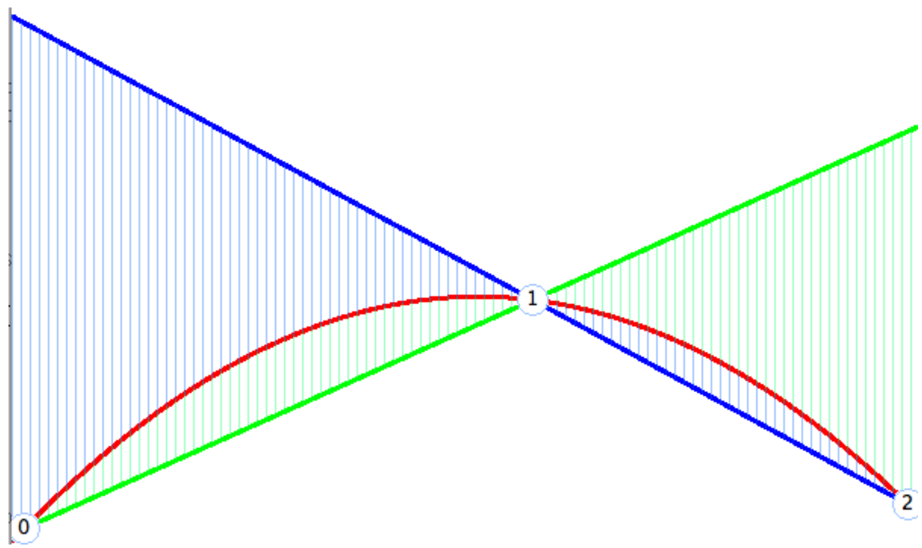
  $B(t)=P(t,t_1,P_1,t_2, P_2)$ interpolates P_1 and P_2

  $R(t)=G(t)+f(t)(B(t)–G(t))$ linear combination $G+f(t)(B–G)$

  - Guaranteed to interpolate P_1 , since both G and B do

- Select f(t) such that
  - $R(t_0)=P_0$: $f(t_0)=0$
  - $R(t_2)=P_2$: $f(t_2)=1$
  - Hence: f(t) = ???

$$f(t)=(t–t_0)/(t_2–t_0)$$



Solution for the case where n=3

- Given P_0, t_0, P_1, t_1, P_2, t_2, and t, compute P(t):

float P(t , t_0 , P_0 , t_1 , P_1 , t_2 , P_2) {
return P(t , t_0 , P(t , t_0, P_0 , t_1 , P_1),
t_2 , P(t , t_1, P_1 , t_2 , P_2)) ; }

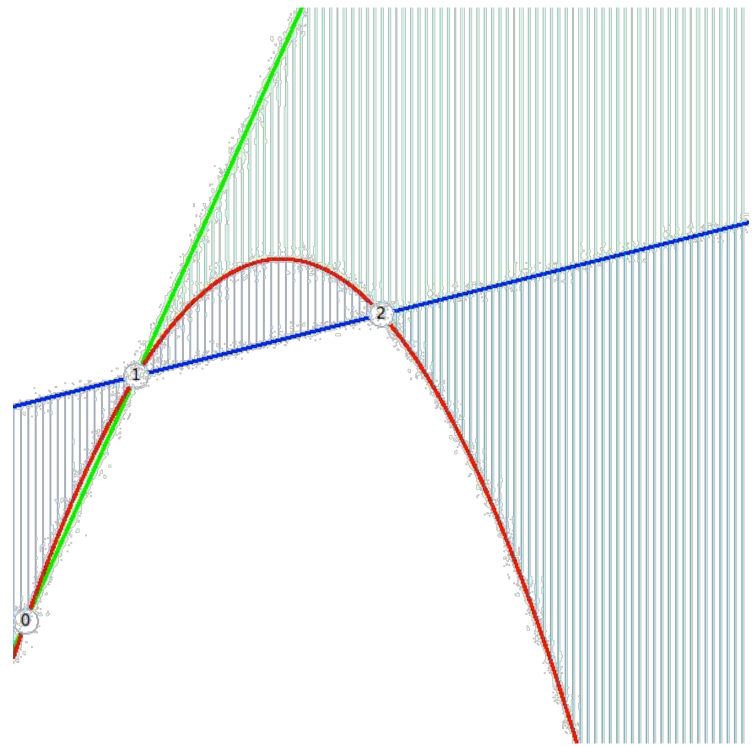
- Justification:

$P(t_0)=P_0$ because $P(t_0, t_0, A, t_2, B)=A$

$P(t_2)=P_2$ because $P(t_2, t_0, B, t_2, C)=C$

$P(t_1)=P_1$ because $P(t_1, t_0, B, t_2, B)=B$

- What kind of a curve is P(t)?



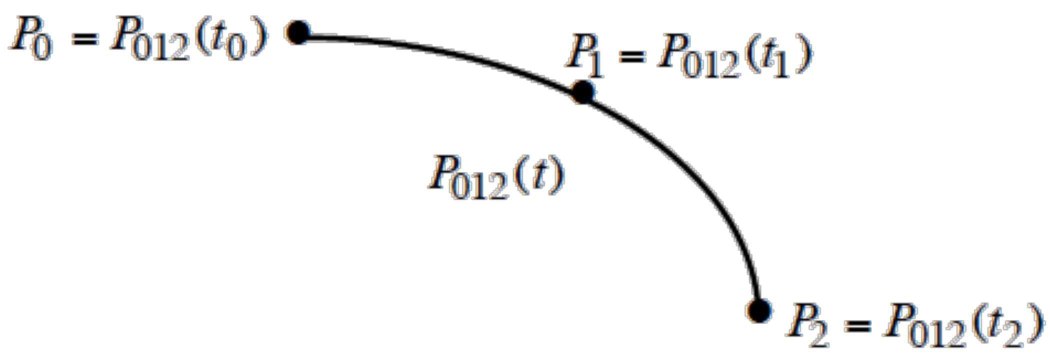
Other notation

- From Ron Goldman

  http://classes.cec.wustl.edu/~cse452/lectures/lect17_Interpolation.pdf

Linear Interpolation

- $P_{01}(t) = \dfrac{t_1 - t}{t_1 - t_0} P_0 + \dfrac{t - t_0}{t_1 - t_0} P_1$
- $P_{12}(t) = \dfrac{t_2 - t}{t_2 - t_1} P_1 + \dfrac{t - t_1}{t_2 - t_1} P_2$

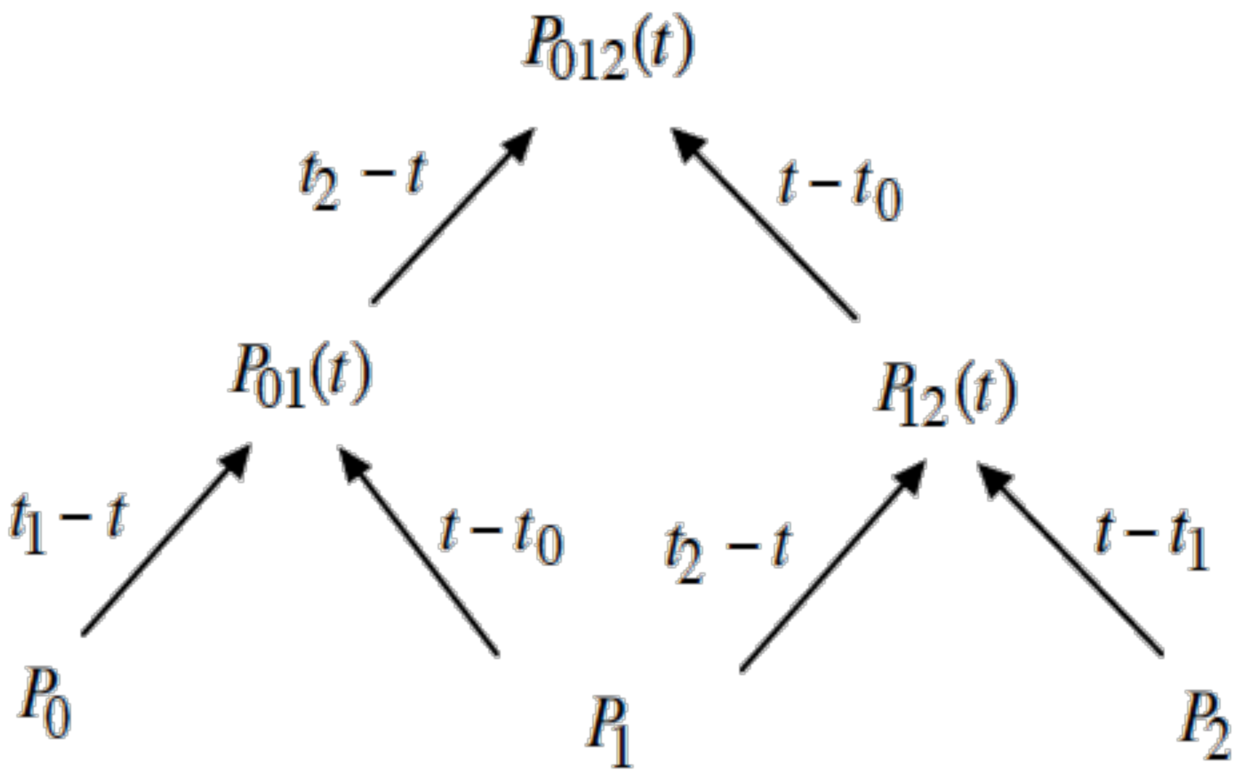


Quadratic Interpolation

- $P_{012}(t) = \dfrac{t_2 - t}{t_2 - t_0} P_{01}(t) + \dfrac{t - t_0}{t_2 - t_0} P_{12}(t)$

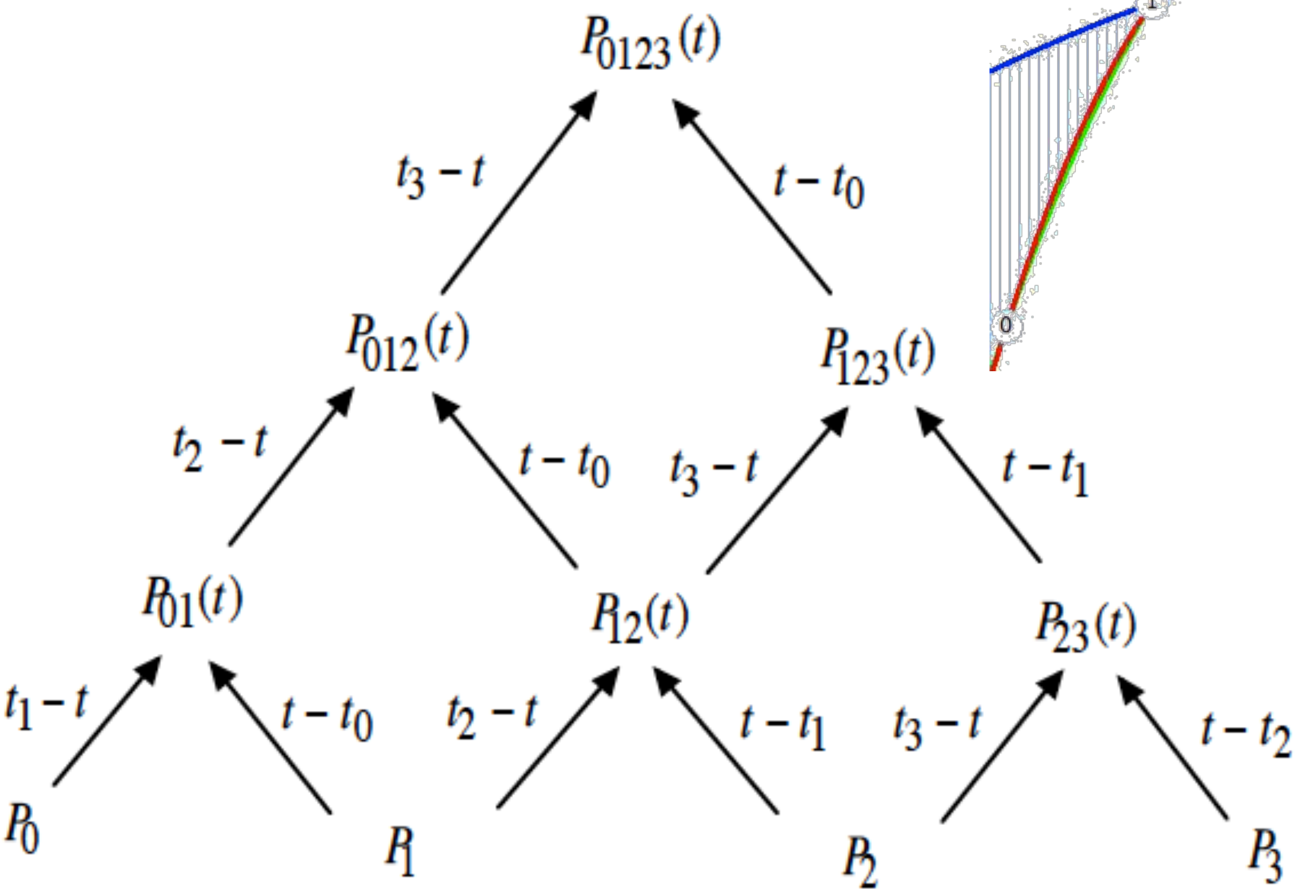
Neville's algorithm for a quadratic curve

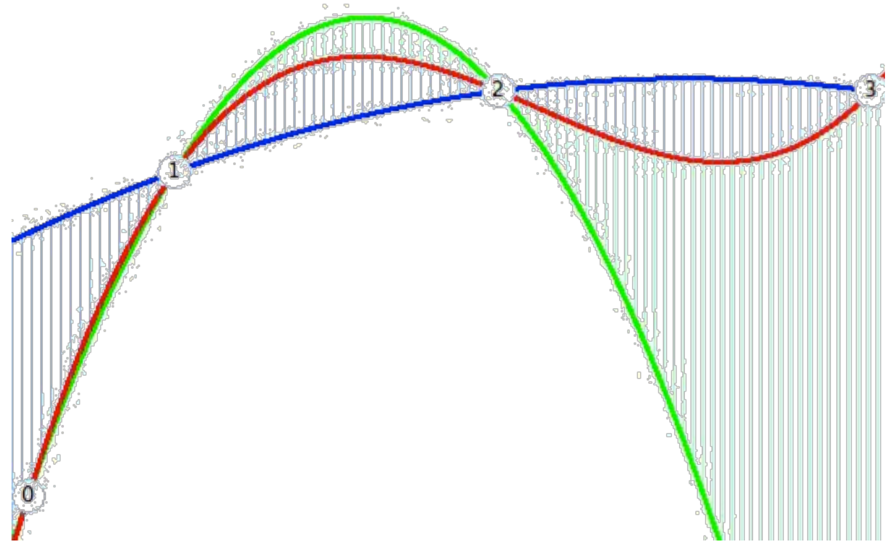
- Denominators omitted (divide the 2 coeffs by their sum)
- Add the 2 results



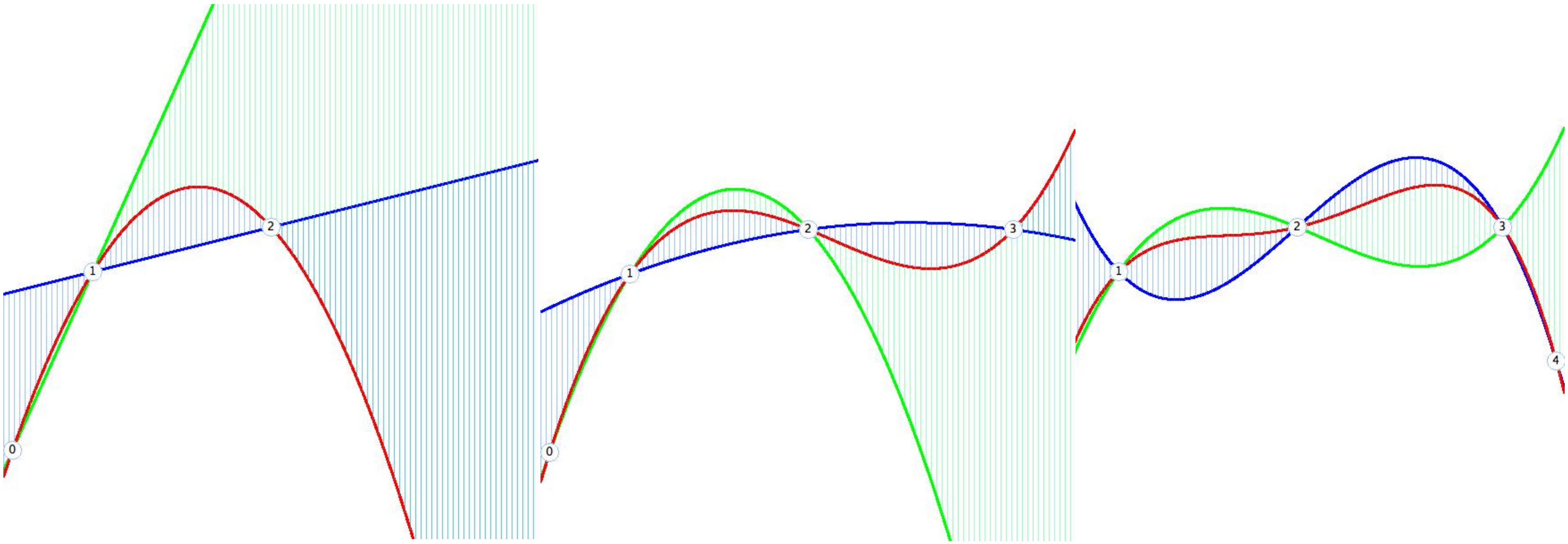
$$P_{012}(t_k) = P_k$$

Neville’s algorithm for n=4





Example for n=5



Uniqueness of Lagrange interpolation

- There exists one and only one polynomial of degree n that interpolates n+1 data points $(P_0,t_0)\ldots(P_n,t_n)$
  - A non-zero polynomial of degree ≤ n can have at most n roots
    - $P(t) = k\,(t-r_1)\,(t-r_2)\,(t-r_n) = k\,(t^n+\ldots)$
  - A polynomial of degree ≤ n with more than n roots is 0
  - If $P_n(t)$ and $Q_n(t)$ agree on the n+1 data points, they are equal
    - $P_n(t) - Q_n(t)$ cancels at n+1 data points and hence is zero

Recursive Implementation

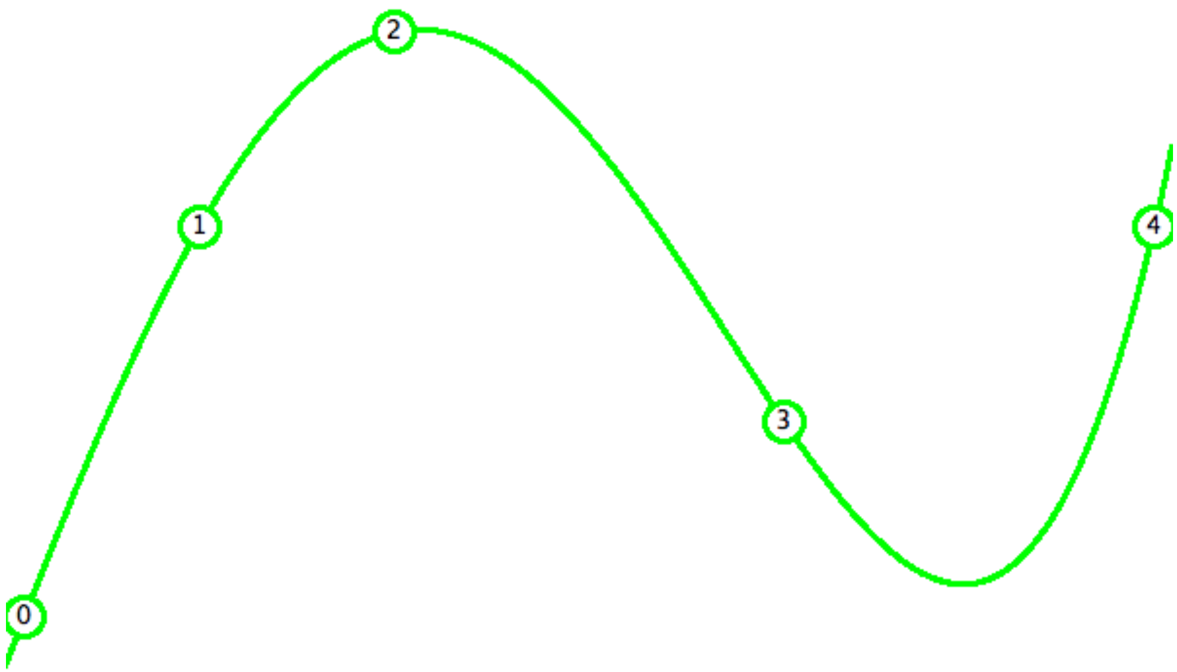
inspired by http://en.wikipedia.org/wiki/Neville's_algorithm

```
// input constraints: P(T[i])=P[i]
float [] P = {10,100,200,400,590};    // parameters
float [] T = {500,300,200,400,300};   // values

float f(int i, float t, int j) {
    if ( i == j ) return P[i];
    float s = ( t - T[i] ) / ( T[j] - T[i] ); // blending weight
    return lerp( f(i,t,j-1), f(i+1,t,j), s); // linear combination
    }
```

Demo

CS3451 Fall 2011, Project 1

Jarek Rossignac



press ('0','1'...) + click&drag to move selected point, press 'p' to snap a picture

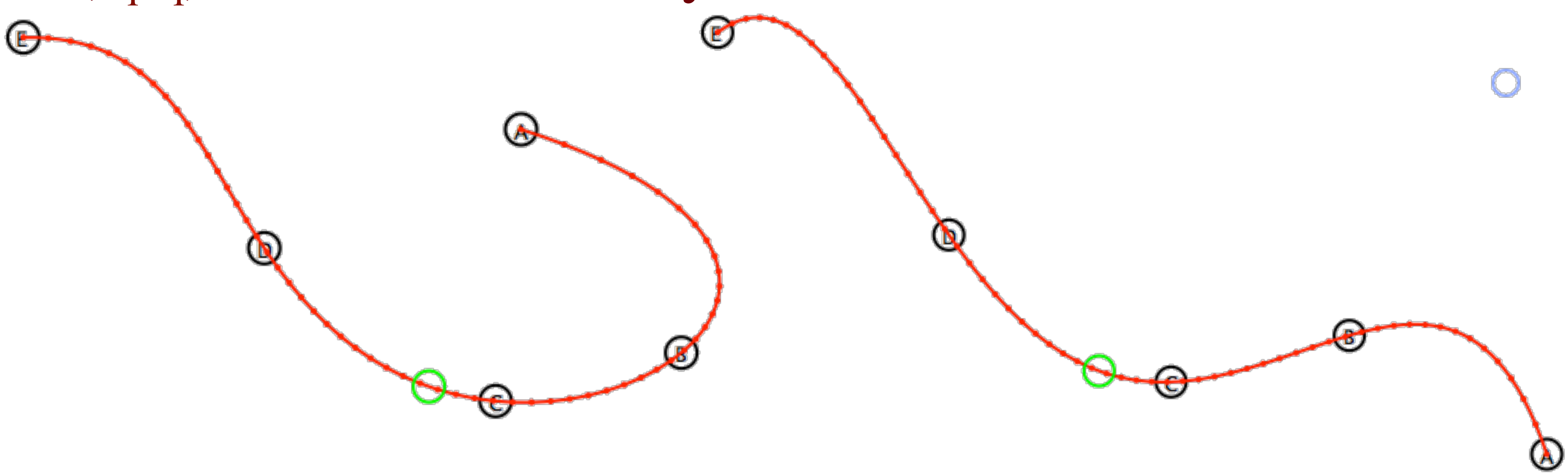
Iterative implementation

// from Rene at
http://www.torkian.info/Site/Research/Entries/2008/2/29_Nevilles_algorithm_Java_Code.html

```
float f (float Y[], float [] T, float t) {
  float F[]=Y.clone();
  for (int j=1; j<n; j++)
    for (int i=n-1; i>=j; i--)
      F[i]=((t-T[i-j])*F[i]-(t-T[i])*F[i-1])/(T[i]-T[i-j]);
  return F[n-1] ;
  }
```

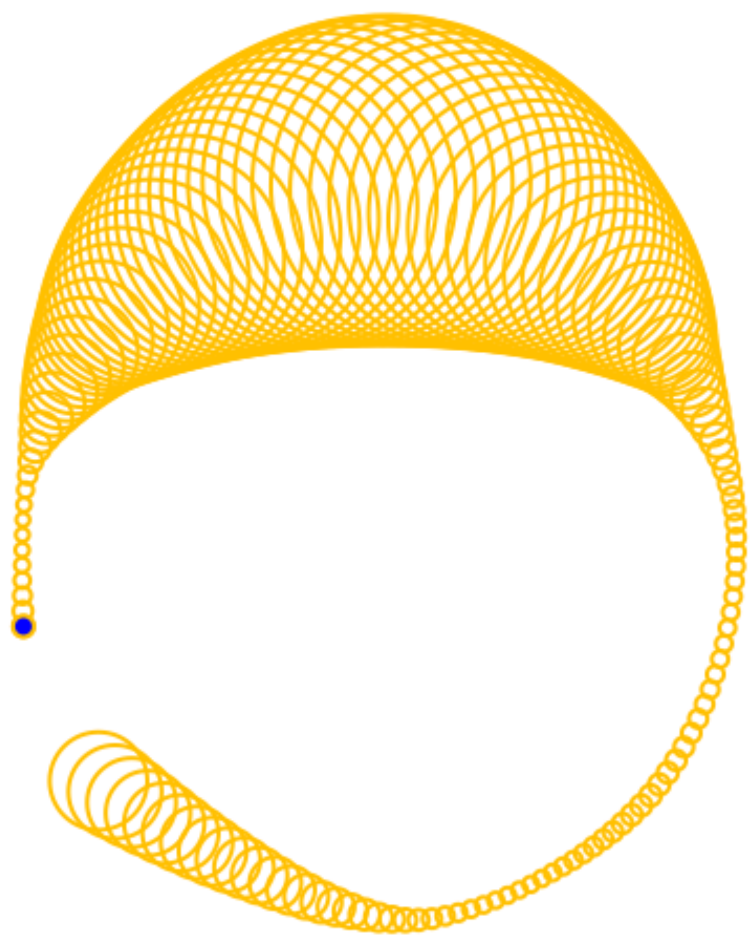
From functions to curve and animations

- Consider n samples $P_i=(x_i,y_i)$ stored in point array P[] and associated time values t_i stored in array T[]
- We can compute a point P(t) on the interpolating polynomial curve as P(t)=(x(t), y(t)) where x(t) is computed by the Neville's algorithm using constraints (x_i,t_i), and the same for y



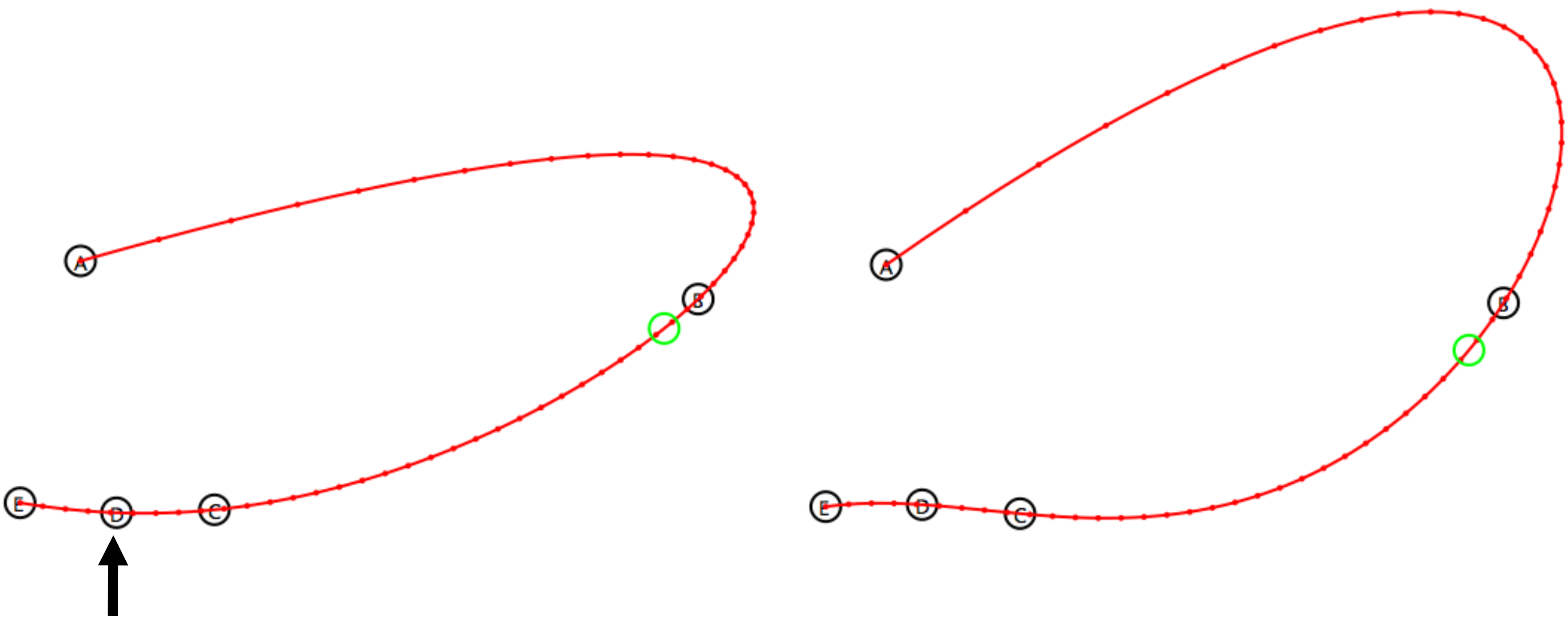
Use it to blend other attributes

- Size (radius of disk)
- Color
- Orientation



Drawback of polynomial interpolation

- Small changes to a sample (D below) may have large effects elsewhere (lack of local control)



Other schemes

- Want local control
- Trade-off between
  - smoothness (degree of continuity)
  - interpolation (how close the curve passes through the data)
- Options:
  - Four-points
  - Catmull-Rom
  - B-splines
  - J-splines
  - …

What must be retained

- Formula for linear interpolation
- Neville's algorithm for Lagrange interpolation
- Its implementation (recursive or iterative code)
- How to plot the result
- How to apply this to draw interpolating curves

Further reading

- Neville's algorithm.
  http://en.wikipedia.org/wiki/Neville's_algorithm
- Prof. Ron Goldman's lecture notes:
  http://classes.cec.wustl.edu/~cse452/lectures/lect17_Interpolation.pdf

Practice problem 1: linear interpolation

- Position P(t) of particle that moves with constant speed and is at A when t=a and at B when t=B.

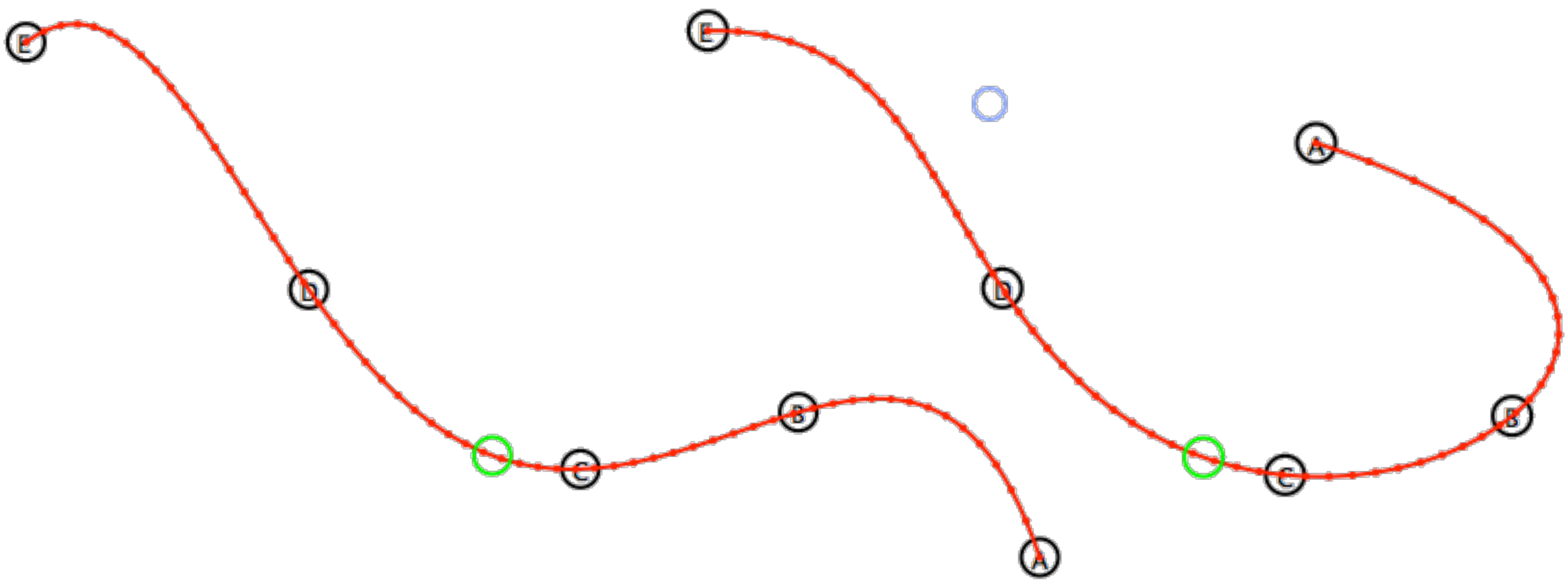
Solution to practice problem

- P(t) = A + (t-a)/(b-a) AB

  where AB is the vector from A to B, such that B = A + AB

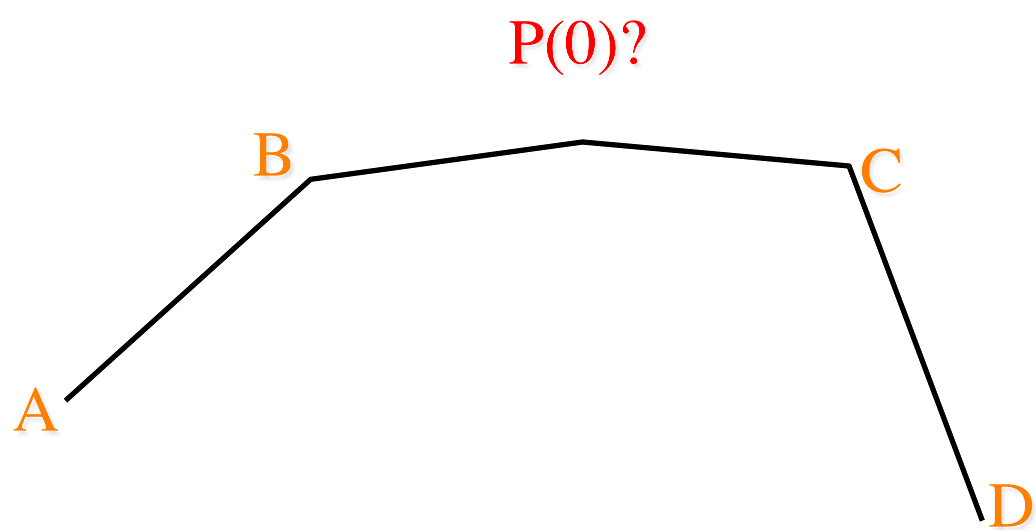
Practice problem 2: Parabolic

- Position P(t) of particle that moves with constant speed and is at A when t=a, at B when t=B, and at C when t=c.
- Code for Neville's algorithm given n positions P[i] and associated times T[i]



Practice problem 3: Cubic smoothing

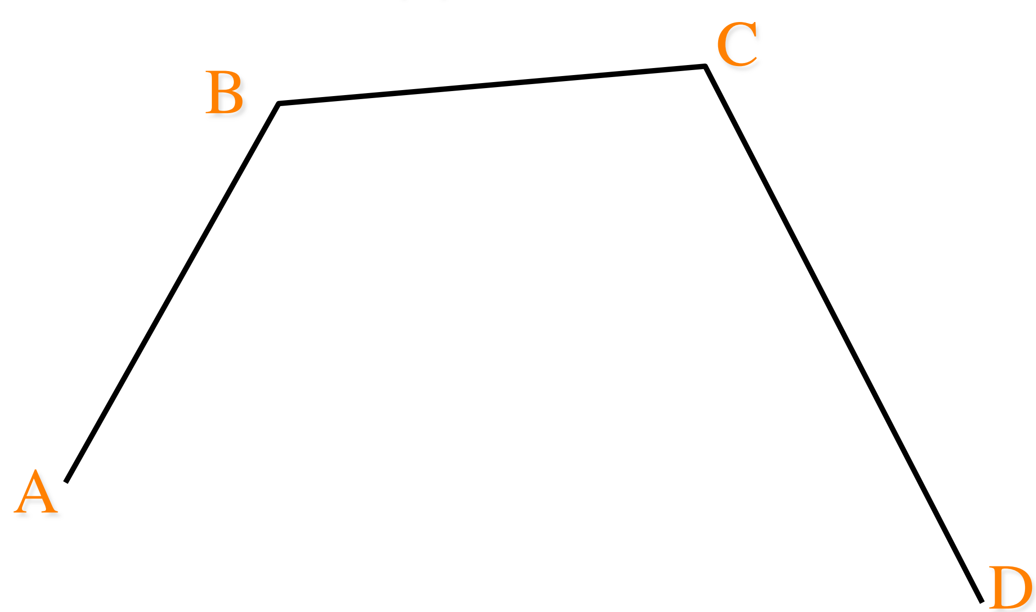
- Find P(0) given 4 constraints
  - P(-2) = A
  - P(-1) = B
  - P(1) = C
  - P(2) = D
- Compute a, b, c, d so that
  - P(0) = aA+bB+cC+dD



Practice problem 4: Cubic subdivision

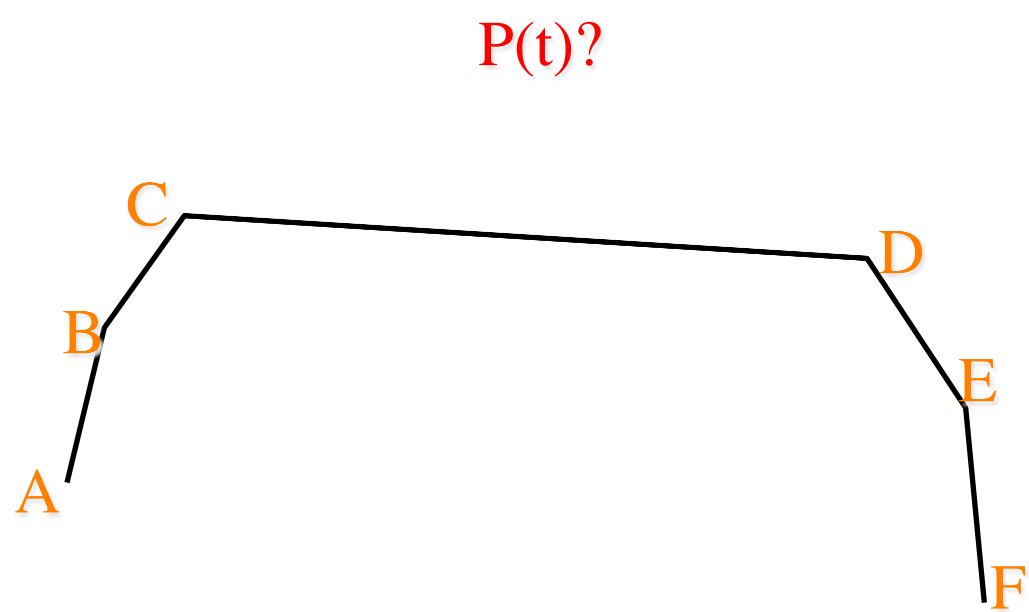
- Find P(0) given 4 constraints
  - P(-3) = A
  - P(-1) = B
  - P(1) = C
  - P(3) = D
- Compute a, b, c, d so that
  - P(0) = aA+bB+cC+dD

P(0)?



Practice problem 5: C^2 fit

- Program P(t) given 6 constraints
  - P(a-d) = A, P(a) = B, P(a+d) = C,
  - P(e-d) = D, P(e) = E, P(e+d) = F
- Write procedure P(t,a,A,B,C,e,D,E,F,d)



Personal Project Page

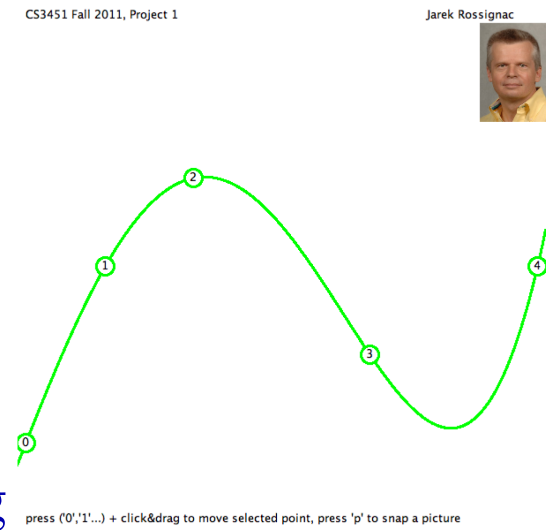
- Create a Personal Project Page containing:
  - TITLE: Projects for CS3451 Fall 2011
  - Your first and LAST name
  - A picture clearly showing your face
  - Your email
- Post it on the web
- Email the link to the TA

For each project, you will update this page by adding a link to the page with your project applet and other deliverables (but only AFTER class, on the due date)

Project 1 deliverables

- Part A (due next class): Neville

  Bring typed sheet to class containing:
  - Image produced by your code
    - with YOUR name and YOUR picture
  - Source code YOU wrote (yourself)
  - YOUR typed explanation of Neville's alg
    - High level intuition for teaching and remembering



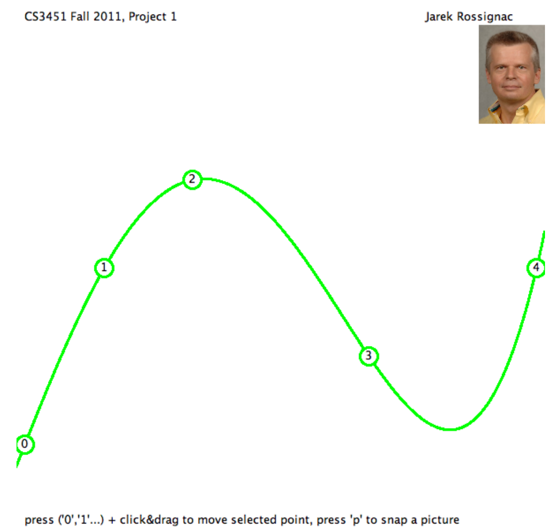
- Part B (due Sept 6 in class): Curve
  - Bring printout to class containing:
    - Project title, your name & your picture
    - Image of your program & print of you source code
    - URL of your applet
  - Post link to interactive applet on your PPP (only after class)

Part A: Given source code to modify

```
PImage myFace;                        // picture of author's face, read from file pic.jpg in data folder
float [] T = {10,100,200,400,590};   // parameters for which values are specified (constraints)
float [] Y = {500,300,200,400,300};  // function values at these parameters (to be interpolated)
int n;                               // number of constraint points (is set in setup)
int s=0;                             // selected constraint for interactive dragging

String title ="CS3451 Fall 2011, Project 1", // text that will be displayed in the sketch window
       name ="Jarek Rossignac",
       help="press ('0','1'...) + click&drag to move selected point, press 'p' to snap a picture";

void setup() {  // executed once at the beginning of the program
  size(600, 600);  // specifies size of window
  n=T.length;      // sets the number of constraints to be the length of array T
  myFace = loadImage("data/pic.jpg");  // load image from file pic.jpg in folder data
  }

void draw() {    // exected at each frame to refresh the screen
  background(255);  // erases the screen by painting a white background
  image(myFace, width-myFace.width/2,25,myFace.width/2,myFace.height/2); // displays the author's face at the top right
  fill(0); text(title,10,20); text(name,width-name.length()*9,20); text(help,10,height-10); noFill(); // writes the title, nc
  stroke(255,0,0); strokeWeight(2); myPlot();  // plots the function in red using color as (R,G,B) between 0 and 255
  fill(255); for(int i=0; i<n; i++) ellipse(T[i],Y[i],20,20);  // displays disks at each constraint (filled in white)
  fill(0);  for(int i=0; i<n; i++) text(str(i),T[i]-4,Y[i]+4); // displays in black the constrain number in the correspondinc
  }

void mouseDragged() { // interrupt executed each time the mouse is dragged while the mouse button is down (pressed)
  if (s<0 || n<=s) return; // does nothing if the key is not a number between o and n-1
  Y[s]+=mouseY-pmouseY; T[s]+=mouseX-pmouseX;  // changes the constraint
  }

void keyPressed() {  // interrupt executed each time a key is pressed
  int w = int(key)-48; if (w<0 || w<n) s=w; // sets s if the key is not a number between o and n-1
  if(key=='p') snapPicture();  // when 'p' is pressed, an image of the window is saved into the pictures subfolder of your sk
  if(key=='r') recursive=true;  if(key=='i') recursive=false;
  if(key==',') n--;  if(key=='.') n++;
  println("n="+n);
  }
int pictureCounter=0; // counter used to give different names to the pictures you snap in the same session (save them elsewhe
void snapPicture() {saveFrame("pictures/P"+nf(pictureCounter++,3)+".jpg");} // creates file P000.jpg, P001.jpg... in /picture
```

What you must implement for P1A

- Replace my name and picture by yours
- Implement function that plots the interpolating polynomial using Neville's algorithm
- Make a picture of the result



Extra credit options:

- Provide 2 implementations
  - Recursive
  - Non-recursive (updates an array)
- Let the user click&drag data points

Part A: Solution (with extra credit)

```
// MY SOLUTION
void myPlot() {if(recursive) myPlotR(); else myPlotI();} // switch for EXTRA CREDIT OPTION 1

void myPlotI() {stroke(0,255,0); beginShape(); for(int i=0; i<width; i++) vertex(i,f2(Y,T,i)); endShape();}
float f2(float Y[], float [] T, float t) {// from Rene at http://www.torkian.info/Site/Research/Entries/2008/2/29_Nevilles_a
   float F[]=Y.clone(); // local copy to avoid overwriting the input data
   for (int j=1; j<n; j++) for (int i=n-1; i>=j; i--) F[i]=((t-T[i-j])*F[i]-(t-T[i])*F[i-1])/(T[i]-T[i-j]);
   return(F[n-1]);
   }

// EXTRA CREDIT OPTION 1: Iterative implementation
boolean recursive=false;  // selects method, changed in keyPressed()

void myPlotR() {stroke(255,0,0); beginShape(); for(int i=0; i<width; i++) vertex(i,f1(0,i,n-1)); endShape();}

float f1(int i, float t, int j) { // inspired by http://en.wikipedia.org/wiki/Neville's_algorithm
  if(i==j) return Y[i]; float s=(t-T[i])/(T[j]-T[i]);
  return lerp(f1(i,t,j-1),f1(i+1,t,j),s);
  }

// EXTRA CREDIT OPTION 2: Automatic pic
void mousePressed() {s=0; for(int i=0; i<n; i++) {println("i="+i+", s="+s); if(abs(mouseX-T[i])<abs(mouseX-T[s])) s=i;} }
```

Project 1 part B

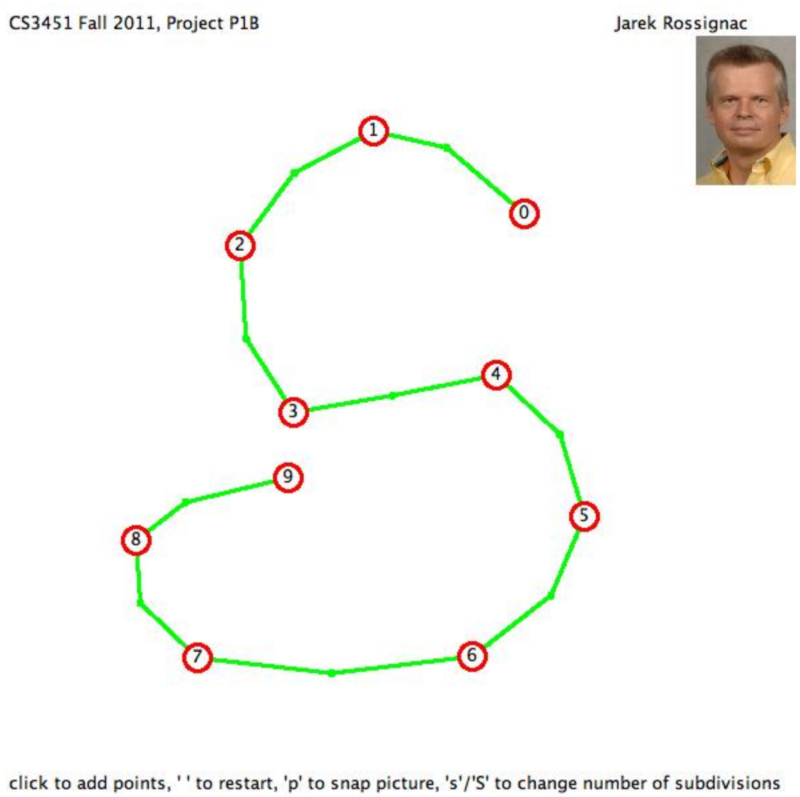
Implement the following functionality

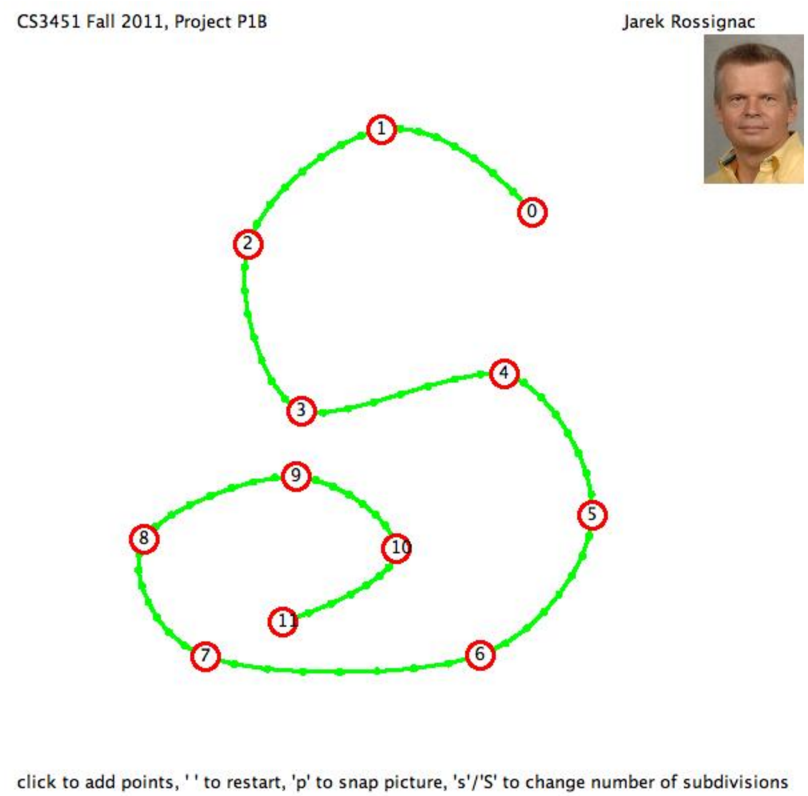
- Each mouse click appends mouse location to array P of points
- Pressing SPACE deletes all points, so you can restart
- Display a smooth curve that interpolates these points in order

Use subdivision to compute the smooth curve

- Iterate r times (user presses 'r' or 'R' to change it):
  - Insert a new points for each edge
  - Use Neville's algorithm to compute the new point
    - For x(t) and y(t)
    - Using 4 neighbors (or 3 for the first and last edge)
  - Use edge-lengths to compute the t-values

P1B demo


CS3451 Fall 2011, Project P1B
Jarek Rossignac
1
0
2
4
3
9
5
8
7
6
click to add points, ' ' to restart, 'p' to snap picture, 's'/'S' to change number of subdivisions


CS3451 Fall 2011, Project P1B
Jarek Rossignac
1
0
2
4
3
9
5
8
10
11
7
6
click to add points, ' ' to restart, 'p' to snap picture, 's'/'S' to change number of subdivisions