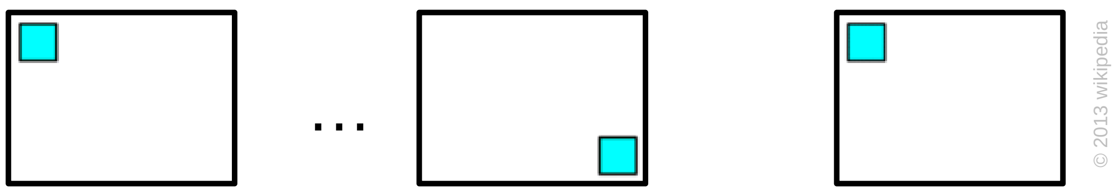
Smooth Interpolation

Chapter 9

Keyframe animation

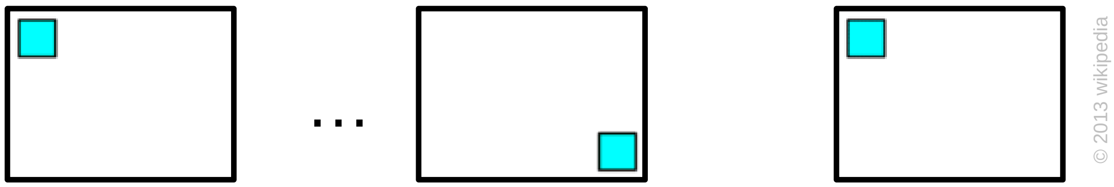
- An animator describes snapshots of a 3D computer graphics animation at a set of discrete times.
  - So-called *key frames*



Starting keyframe Ending keyframe Completed animation

Keyframe animation

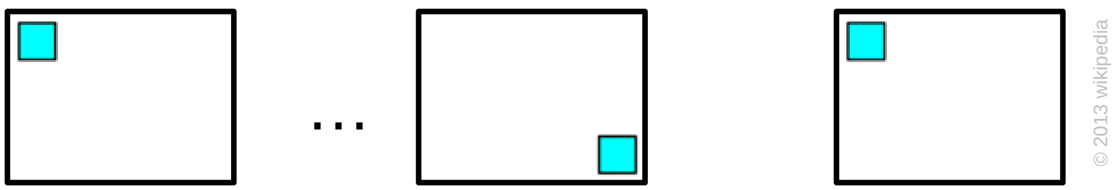
- Each keyframe is defined by some set of modeling parameters.
  - In our case, this is a bunch of RBTs.
  - The translations are 3 real scalars.
  - The rotations are quaternions.



Starting keyframe | Ending keyframe | Completed animation

Keyframe animation

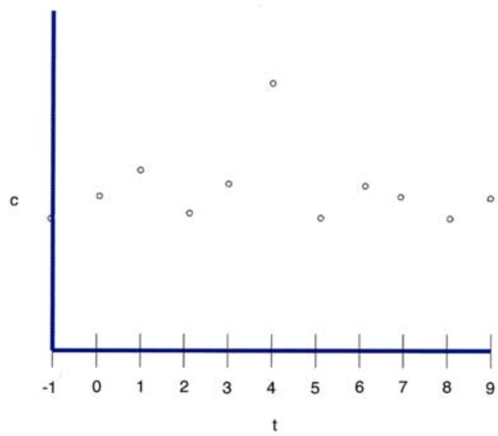
- To create a smooth animation, the computer's job is to smoothly 'fill in' the parameter values over a continuous range of times.



Starting keyframe | Ending keyframe | Completed animation

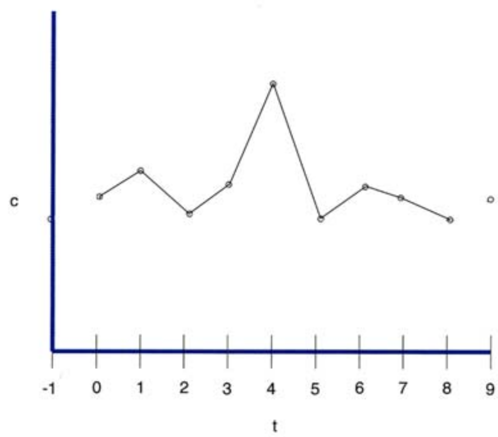
Interpolation

- Let one of these animated parameters be called c, and each of our discrete snapshots is called c_i where $c(t)$ is some range of integers
- Our job is to go from the c_i to a continuous function of time,
  - We will typically want the function $c(t)$ to be sufficiently smooth, so that the animation does not appear too jerky.



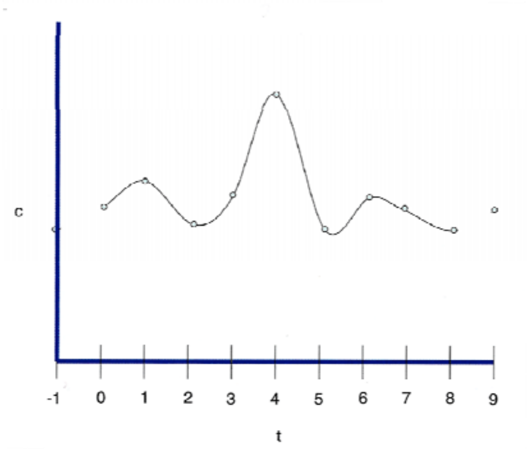
Interpolation

- We show a function $c(t)$, with $t \in [0..8]$ that interpolates the discrete values associated with the integers c_i with $t \in [-1..9]$
  - The need for the extra non-interpolated values at -1 and 9 will be made clear later
- Until now, we have used piecewise linear interpolation, which is not smooth!



Splines

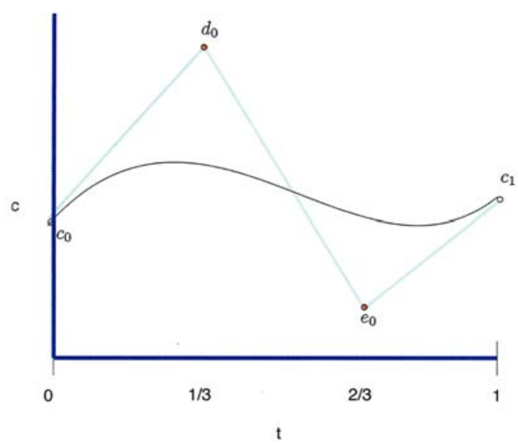
- Our spline is made up of individual pieces, where each piece is some <u>low-order polynomial function</u>
- These polynomial pieces will be selected so that they ‘stitch up’ smoothly.
- Easy to present, evaluate and control.
  - Spline behavior much easier to predict than, say, a single high-order polynomial function.
- Also useful for curves in the plane and space, and the basis for theory of smooth surfaces in space.



Cubic Bezier function

- We start by just looking at how to represent a cubic polynomial function $c(t)$ with $t \in [0..1]$
- Bezier representation
  - The parameters have a direct geometric interpolation
  - Evaluation reduces to repeated linear interpolation.
- Specify a cubic function using four *‘control values’*

$$c_0, d_0, e_0, \text{ and } c_1$$



Bezier evaluation

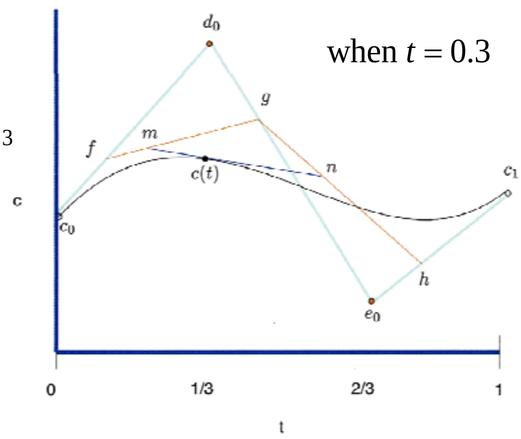
- To evaluate the function $c(t)$ at any value of t,
  we perform the following sequence of linear interpolations:

$$f = (1-t)c_0 + td_0 \qquad m = (1-t)f + tg \qquad c(t) = (1-t)m + tn$$
$$g = (1-t)d_0 + te_0 \qquad n = (1-t)g + th$$
$$h = (1-t)e_0 + tc_1$$

- We can verify $c(t)$ has the form:

$$c(t) = c_0(1-t)^3 + 3d_0t(1-t)^2 + 3e_0t^2(1-t) + c_1t^3$$

  - It is a cubic function
  - The c_i are interpolated:
    $c(0) = c_0$ and $c(1) = c_1$

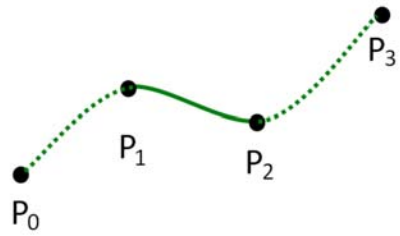


Cubic Bezier Splines

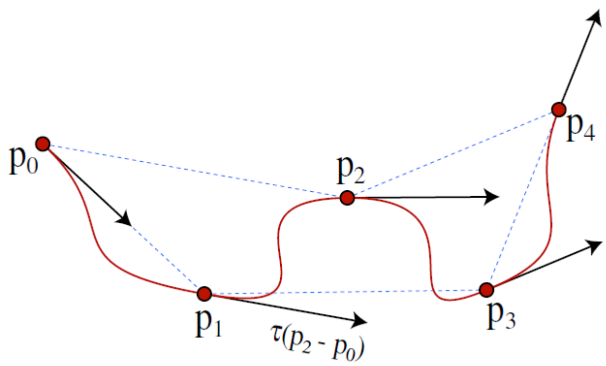
http://www.math.ucla.edu/~baker/java/hoefer/Spline.htm

Catmull-Rom Splines (CRS)

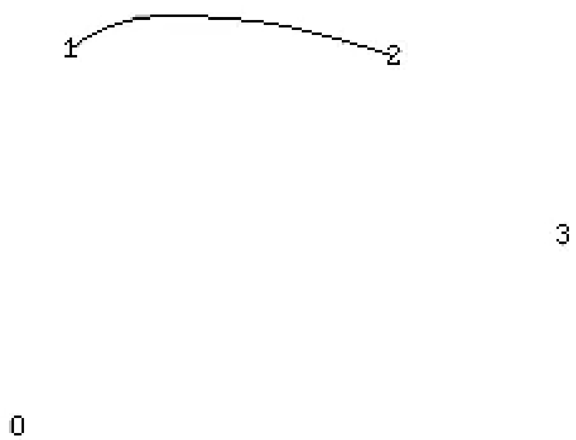
- Defined by 4 control points p_0, p_1, p_2, p_3, with the curve drawn only from p_1 to p_2.



- Cubic Hermite spline



Catmull-Rom Splines

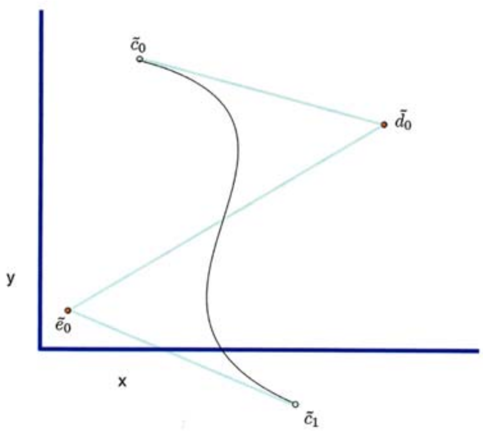


http://www.cse.unsw.edu.au/~lambert/splines/CatmullRom.html

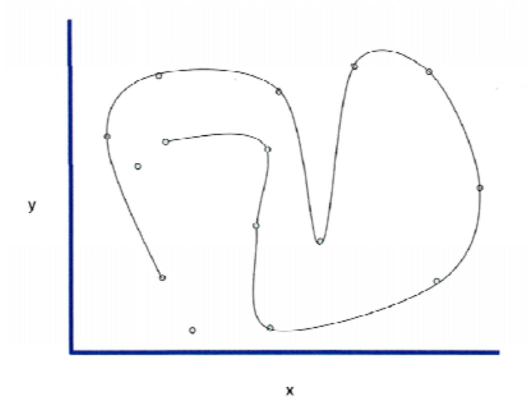
Curves

- The spline curve is controlled by a set of *control points* $\tilde{c}_i$ in 2D or 3D.

Bezier curve (2D)



Catmull-Rom curve (2D)



Curves

- Applying the spline construction independently to the x, y and z coordinates, one gets a point-valued spline function $\tilde{c}_i$;
- Think of this as a point flying through space over time, tracing out the spline curve, γ .
- This can be further developed into a theory for surfaces.