Intermediate Graphics & Animation Programming

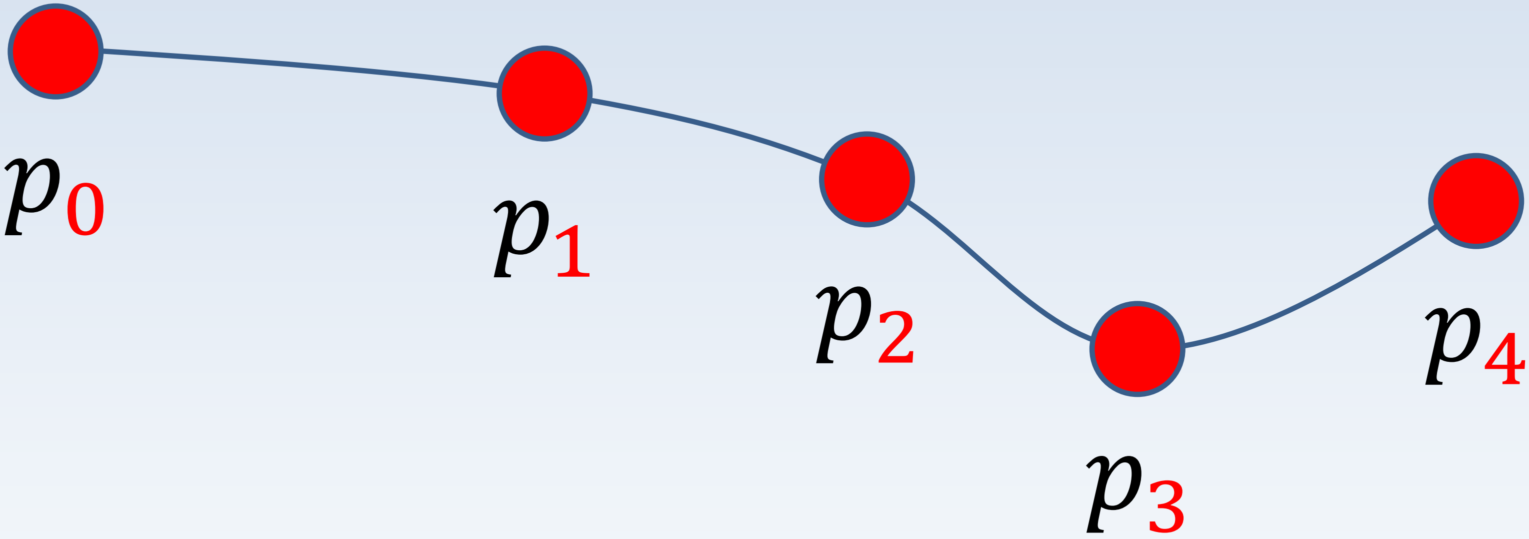
GPR-300
Daniel S. Buckstein

Interpolation Applications
Weeks 10 – 11

License

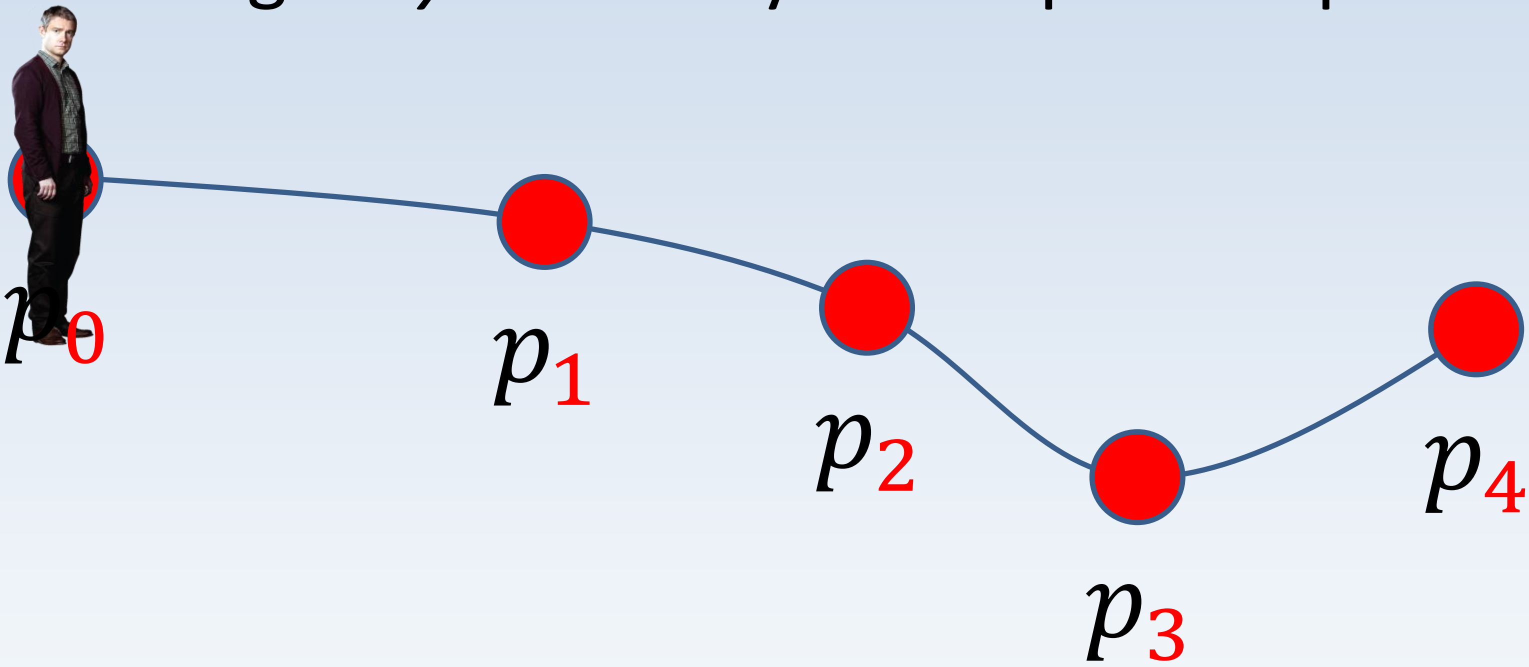
Applications of Paths & Curves

- Locomotion again; i.e. spatial application with positions, points in space!



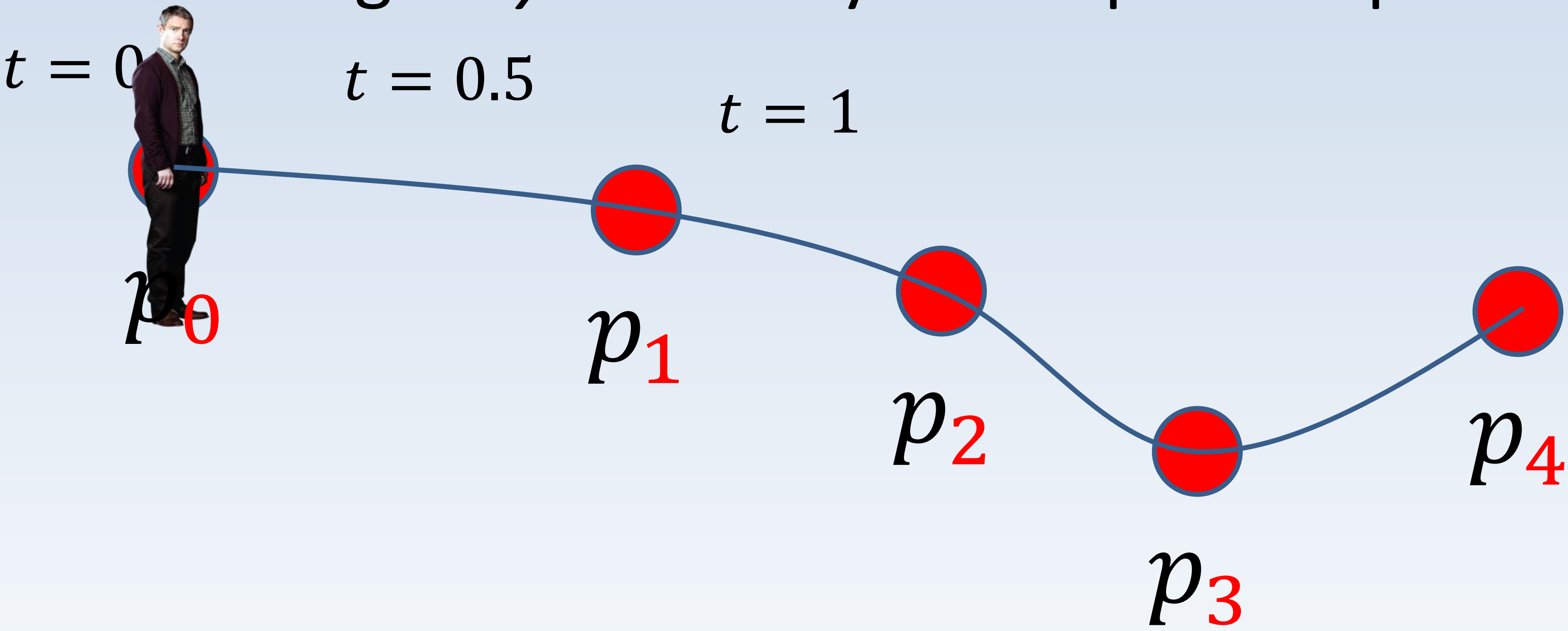
Applications of Paths & Curves

- Moving *very* smoothly from “pose to pose”:



Applications of Paths & Curves

- Moving *very* smoothly from “pose to pose”:



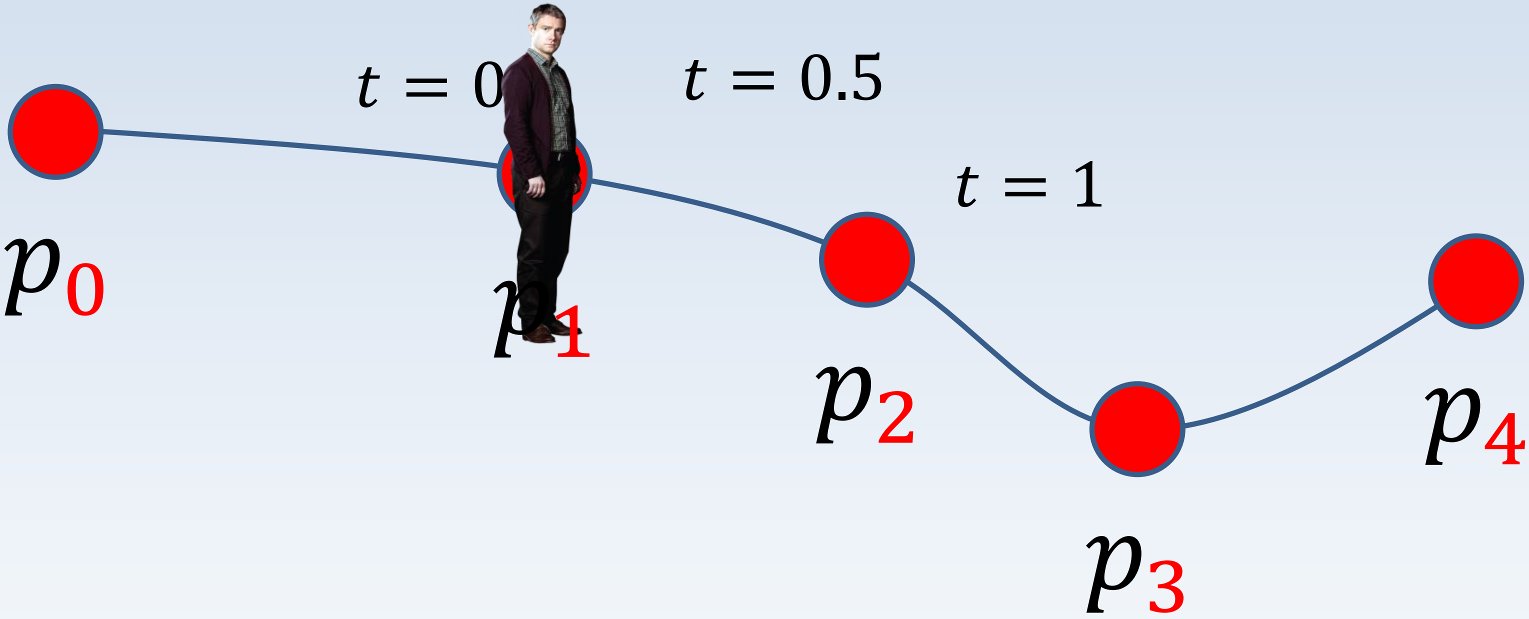
$p_{\text{Watson}} = \text{CatmullRom}(p_c, p_c, p_{c+1}, p_{c+2}, t)$

$c = 0$

Looping disabled: $p_{c-1} = p_c$

Applications of Paths & Curves

- Moving *very* smoothly from “pose to pose”:



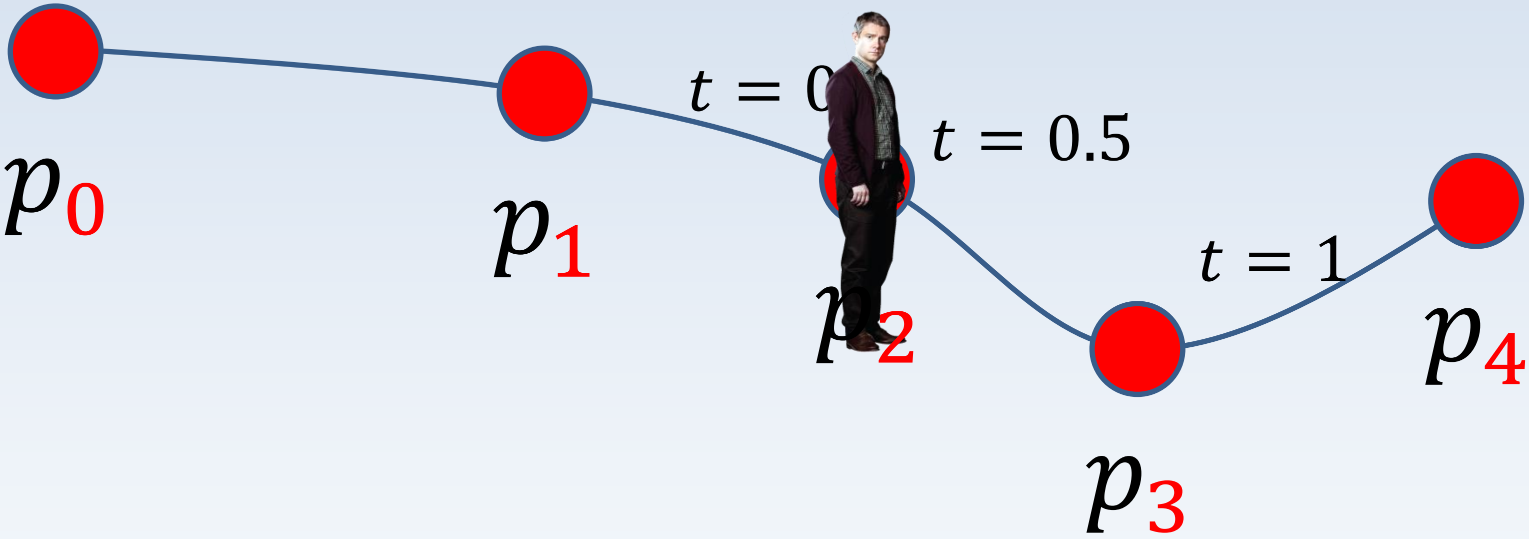
$p_{\text{Watson}} = \text{CatmullRom}(p_{c-1}, p_c, p_{c+1}, p_{c+2}, t)$

$c = 1$

Looping disabled

Applications of Paths & Curves

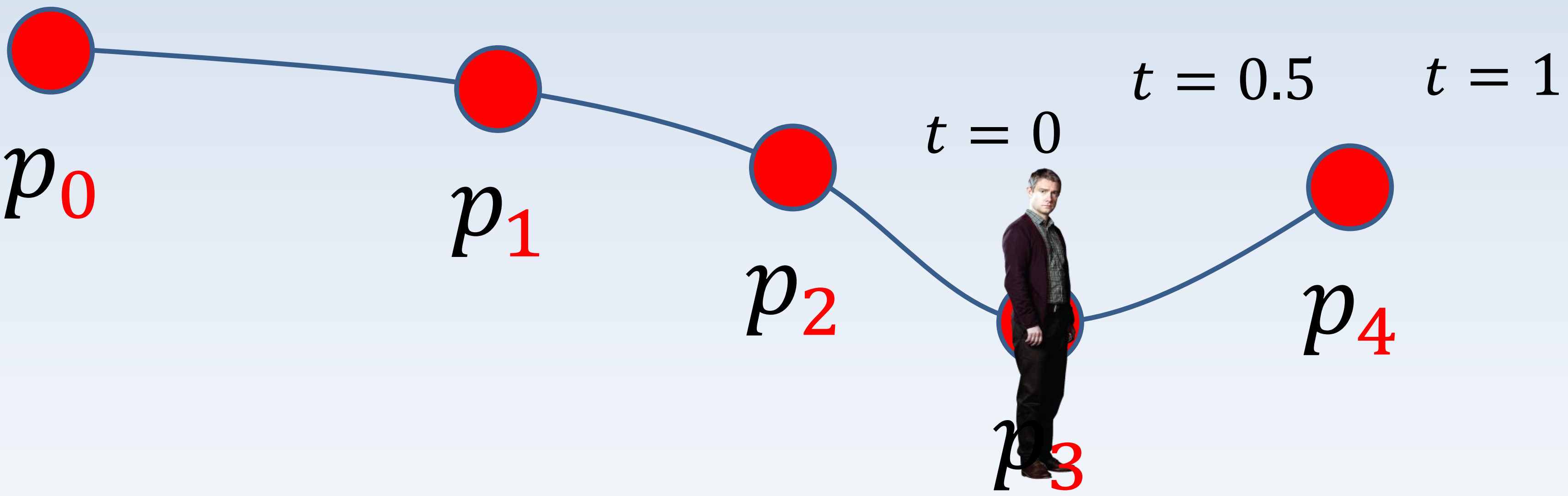
- Moving *very* smoothly from "pose to pose":



$p_{\text{Watson}} = \text{CatmullRom}(p_{c-1}, p_c, p_{c+1}, p_{c+2}, t)$
$c = 2$
Looping disabled

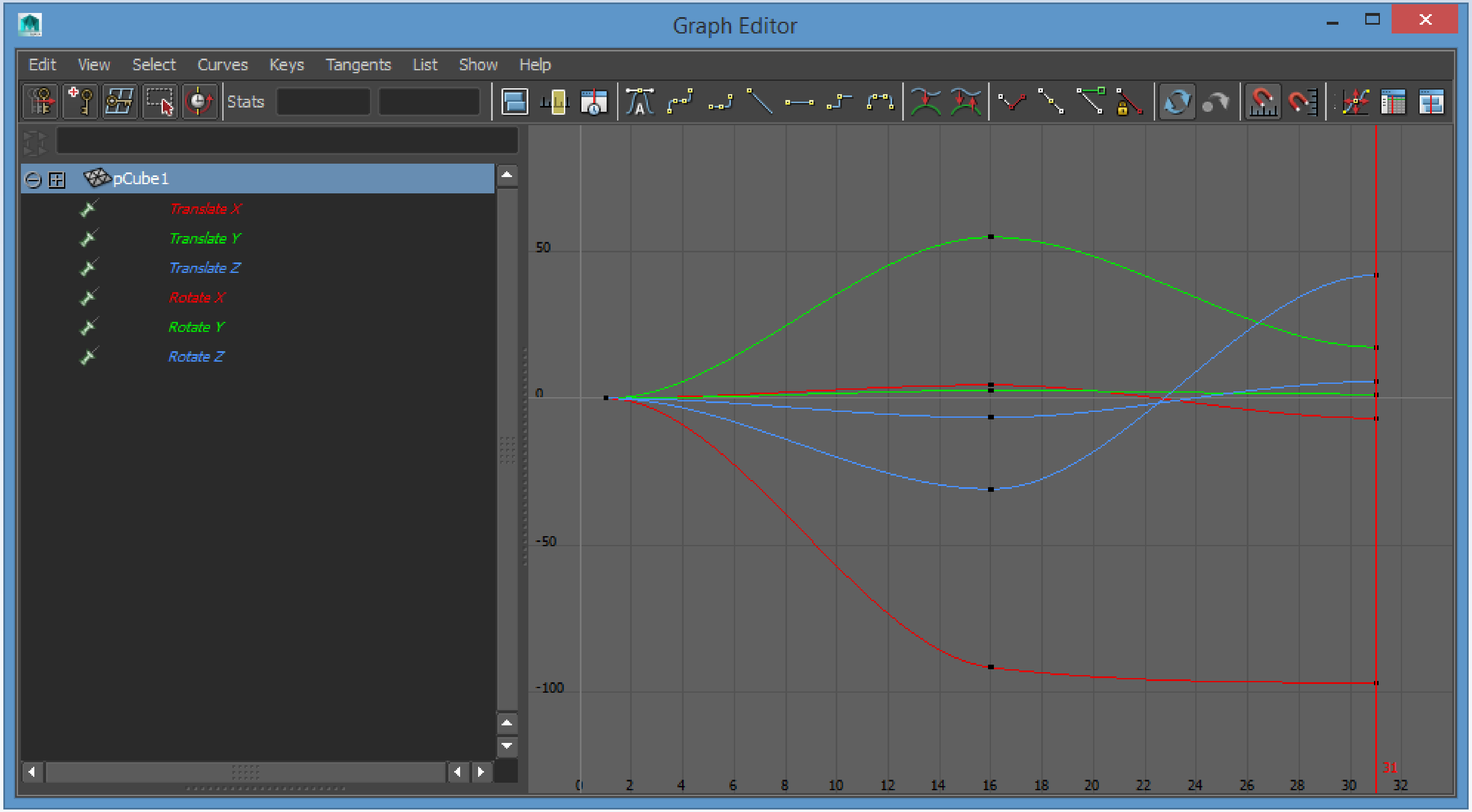
Applications of Paths & Curves

- Moving *very* smoothly from “pose to pose”:

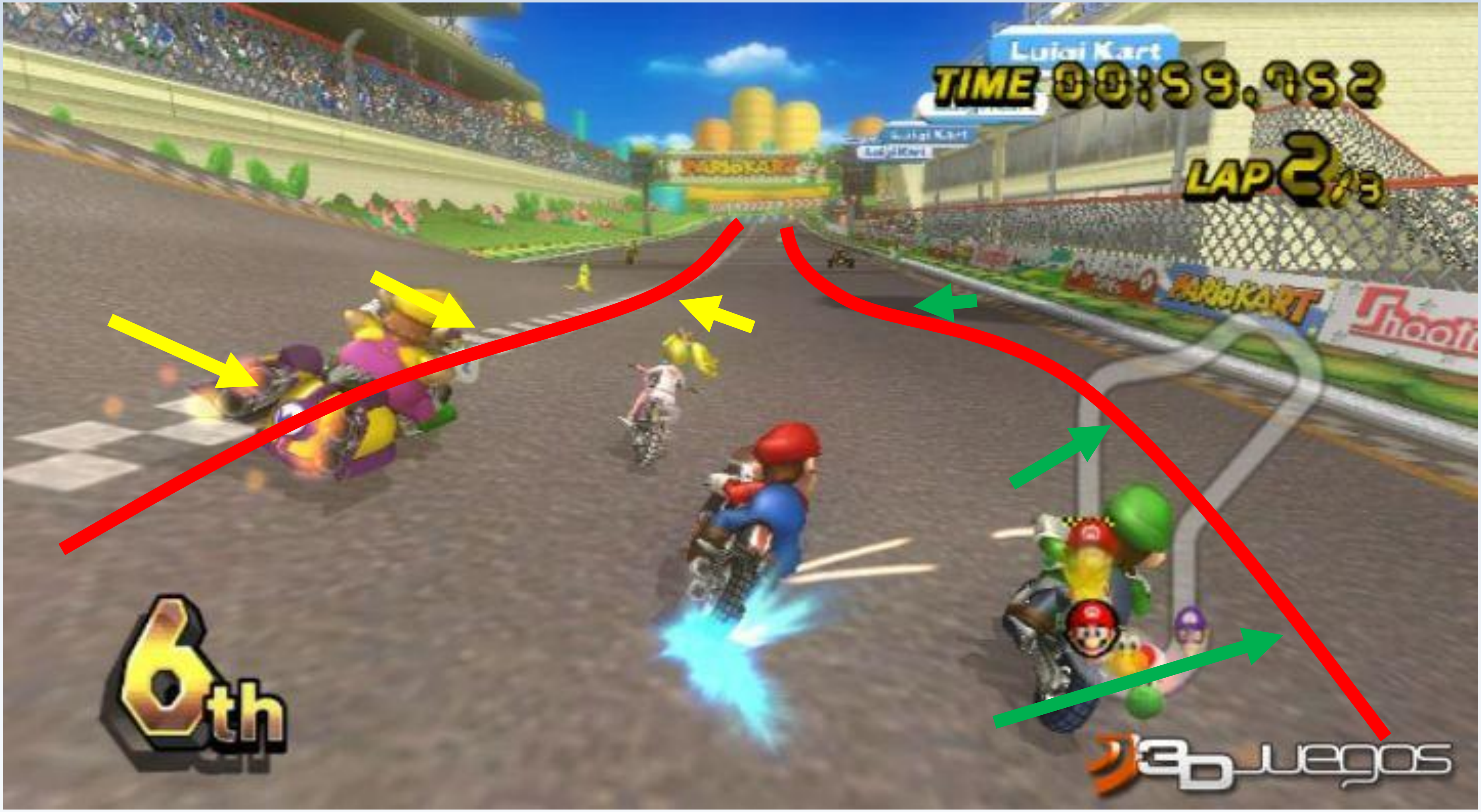


$p_{\text{Watson}} = \text{CatmullRom}(p_{c-1}, p_c, p_{c+1}, p_{c+1}, t)$
$c = 3$
Looping disabled: $p_{c+2} = p_{c+1}$

Applications of Paths & Curves



Applications of Paths & Curves



Applications of Paths & Curves

- Smooth locomotion
  - (this is only *one* application!)

- Smooth morphing

- Smooth general variables

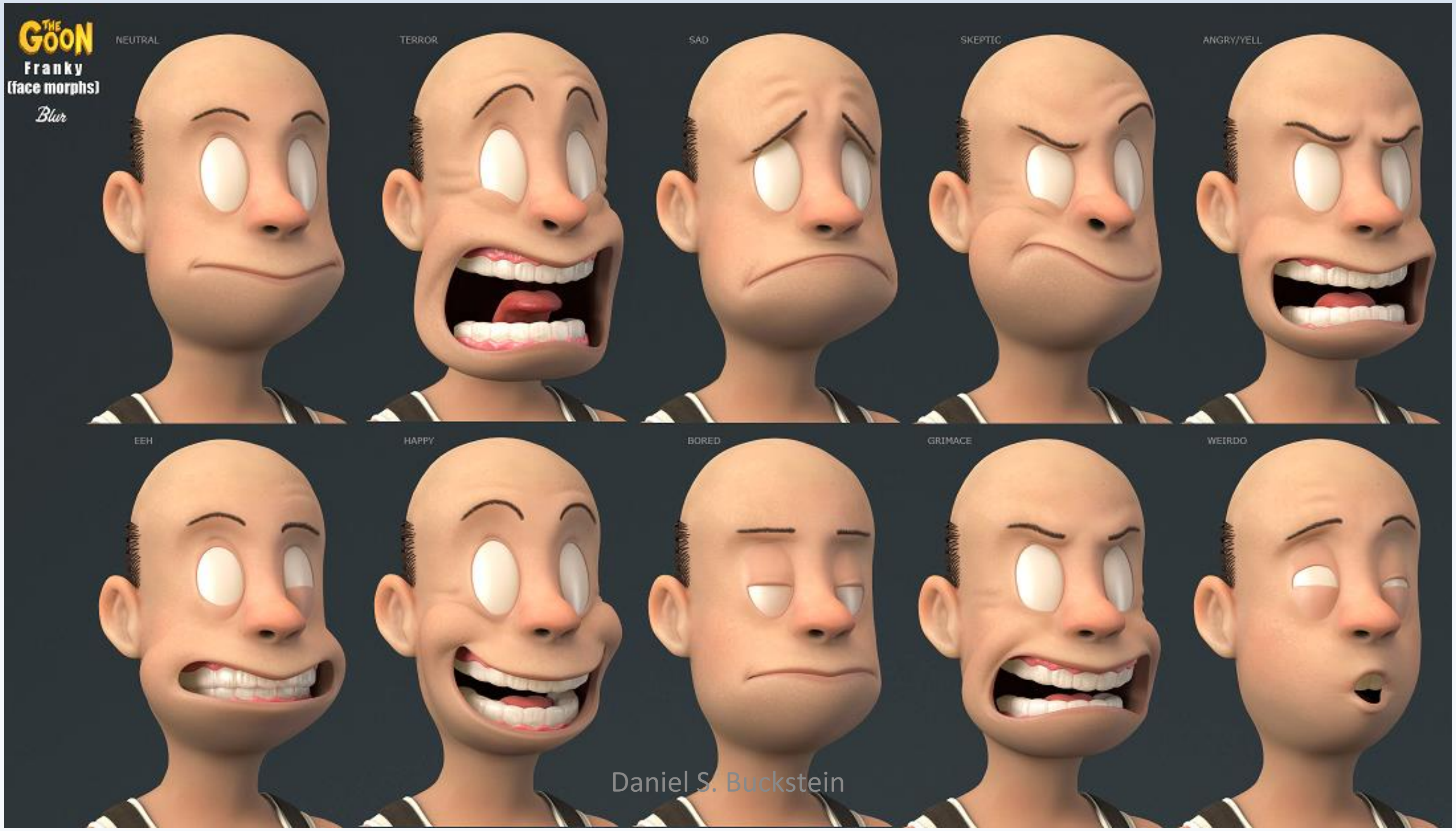
- Smooth ***everything***

Intro to Morphing

- Morphing: given a set of *morph targets* (keyframes for morphing), we can use morphing algorithms to find the in-betweens
- We’re familiar with *locomotion* as a “spatial” type of pose-to-pose animation
- Morphing is another kind of pose-to-pose that changes the way something *looks*

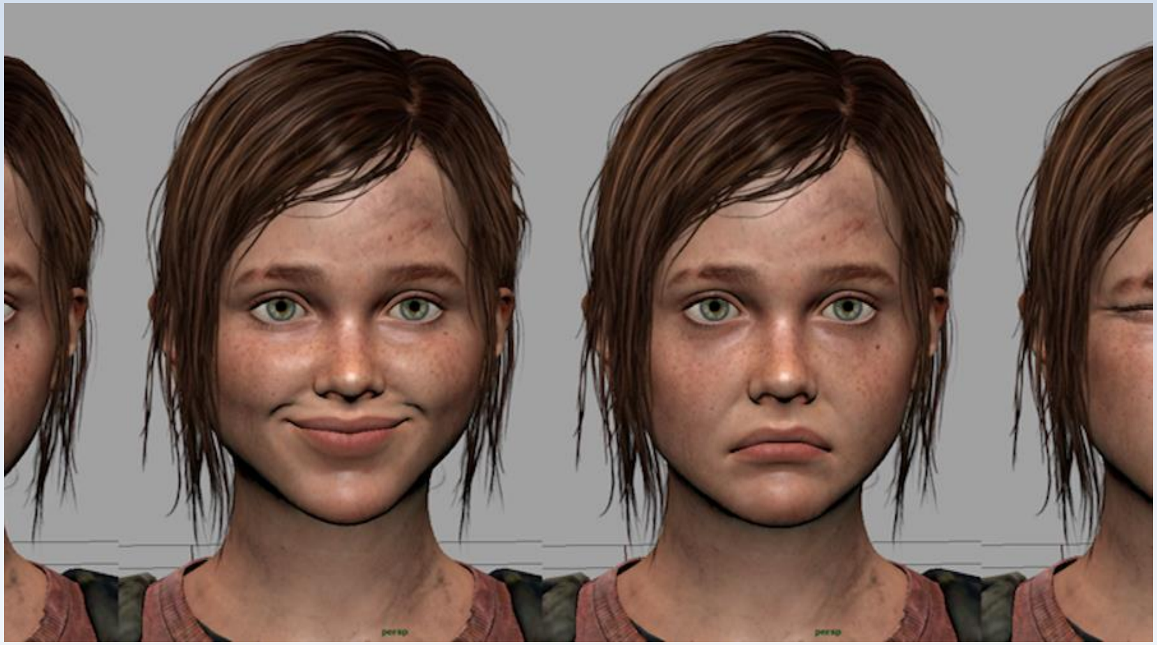
Intro to Morphing

- Facial expressions are a very common application of morph targets:



Intro to Morphing

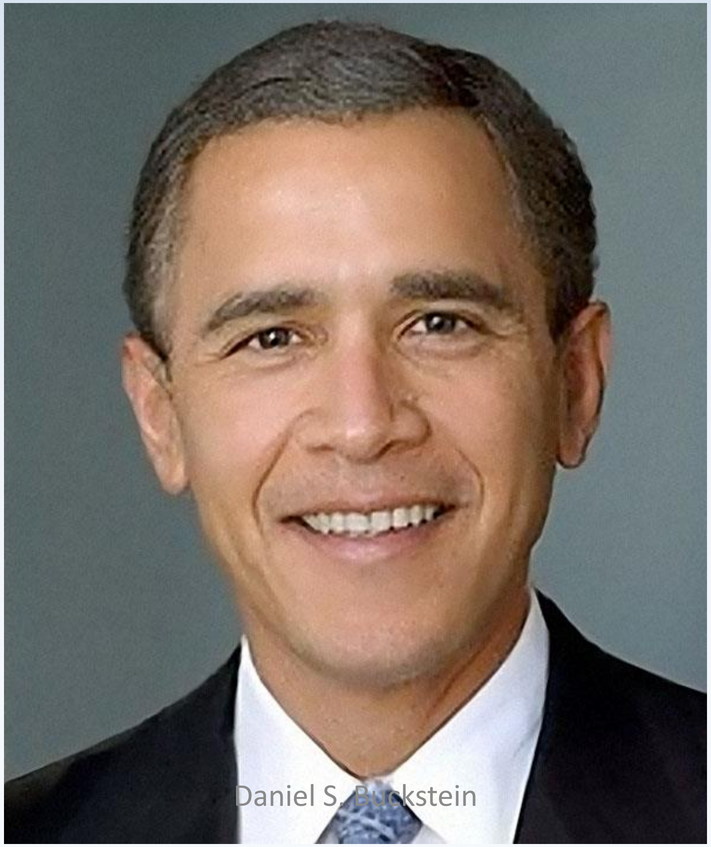
- Mesh animation technique: *morph targets*



- http://www.gameenginebook.com/

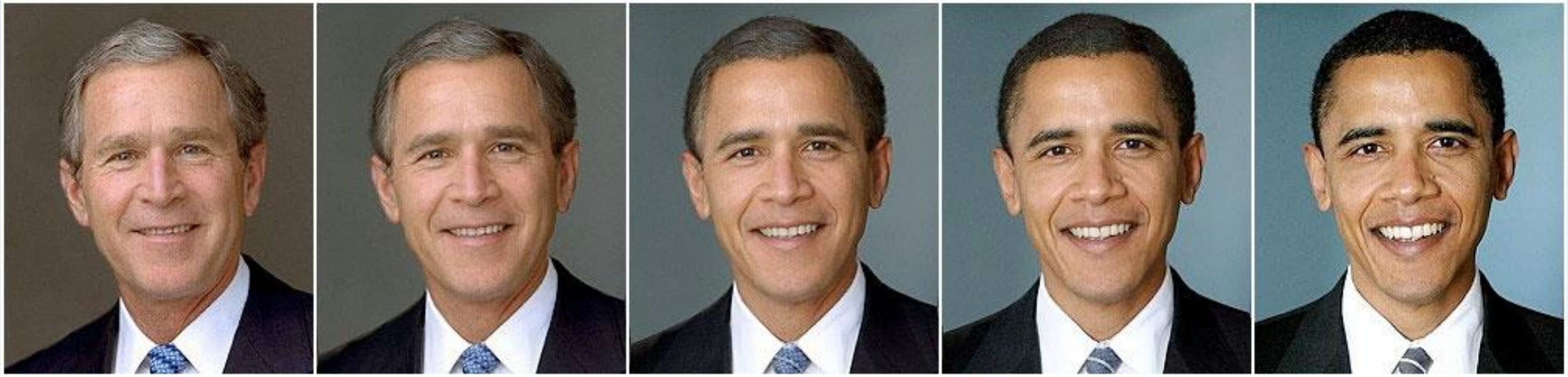
Intro to Morphing

- Works with images too…
- Can be used to do some very uncanny things…



Intro to Morphing

- Morphing between targets in-game is just an application of linear interpolation:

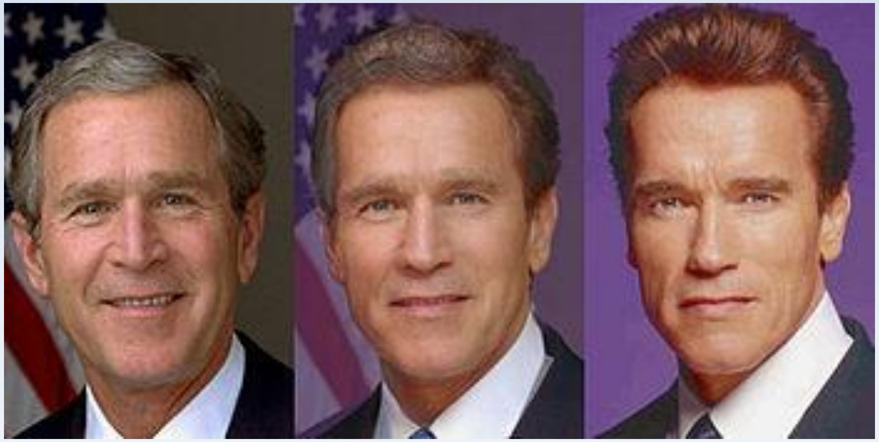


$t = 0$ $t = 0.25$ $t = 0.5$ $t = 0.75$ $t = 1$

$$\text{president} = \text{morph}(\text{Bush}, \text{Obama}, t)$$

Intro to Morphing

- Morphing between targets in-game is just an application of linear interpolation:



$t = 0.5$

$$\mathrm{theBushinator} = \mathrm{morph}(\mathrm{Bush}, \mathrm{Governator}, 0.5)$$

Intro to Morphing

- How does morph target animation actually work (for meshes)???
- Very simple!!!
- Apply interpolation to every vertex in the mesh!
- BTW it doesn’t matter if you know this in graphics yet… just an algorithm
- You can apply it whenever you are ready ☺

Intro to Morphing

- Assuming the meshes being morphed are from the same character set
- Same number of vertices, stored in the same order within each mesh
- ***The algorithm in its simplest form:***

```
$\mathrm{result} = \mathrm{morph}(\mathrm{target}_0, \mathrm{target}_1, t)$:
    for each vertex $v$ in target:
        $v_{\mathrm{result}} = \mathrm{lerp}\left(v_{\mathrm{target}_0}, v_{\mathrm{target}_1}, t\right)$
```

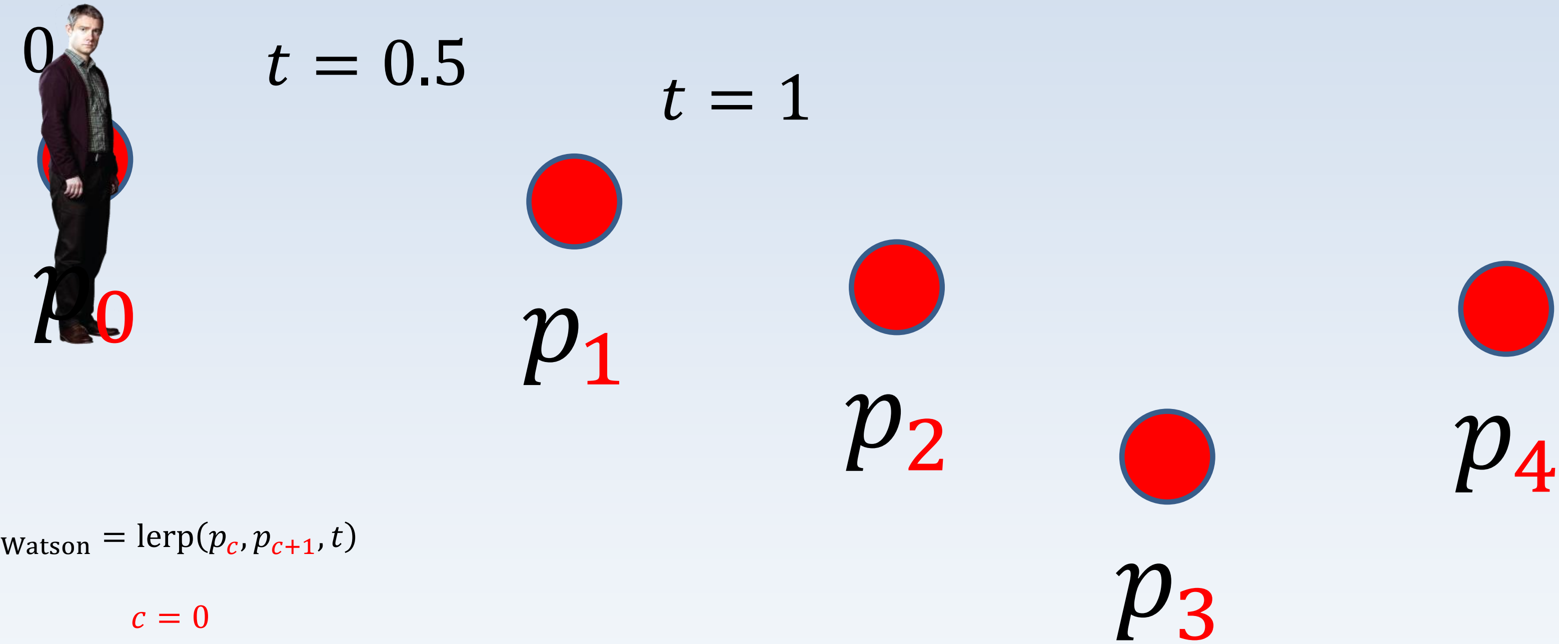
Intro to Morphing

- How do we morph through a *series* or *sequence* of targets instead of between just two???
- Remember paths?
- *Just an algorithm*
- *Can apply it to anything*
- *Why not apply it to morph targets?!*

Intro to Morphing

- Moving smoothly from “pose to pose”:

$t = 0$ $t = 0.5$ $t = 1$



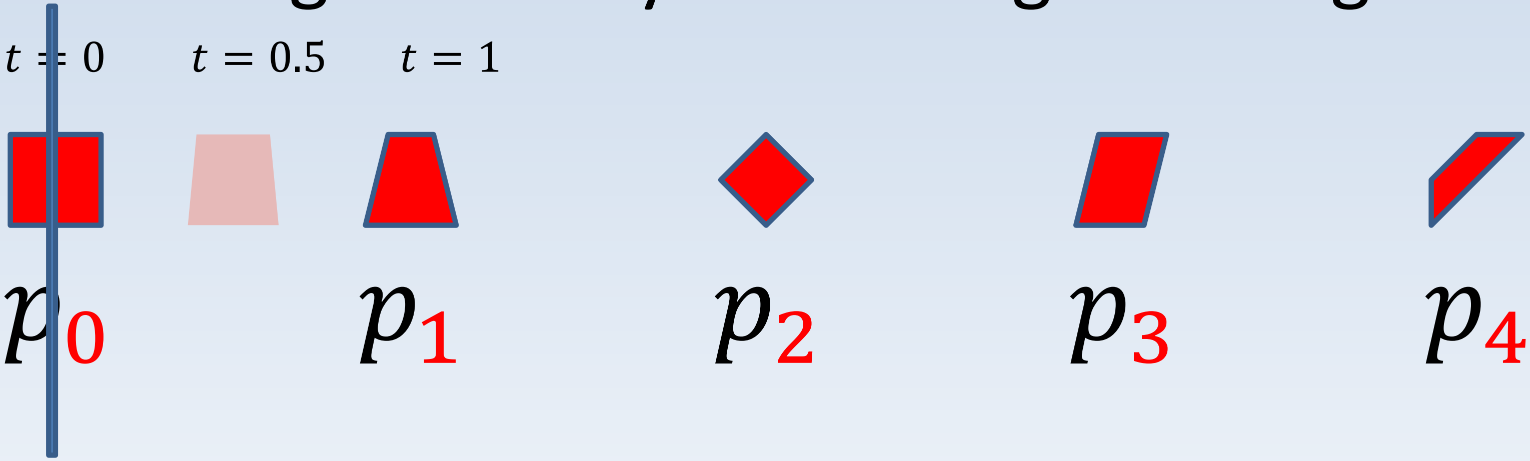
p_0 p_1 p_2 p_3 p_4

$p_{\text{Watson}} = \text{lerp}(p_c, p_{c+1}, t)$

$c = 0$

Intro to Morphing

- Blending smoothly from “target to target”:

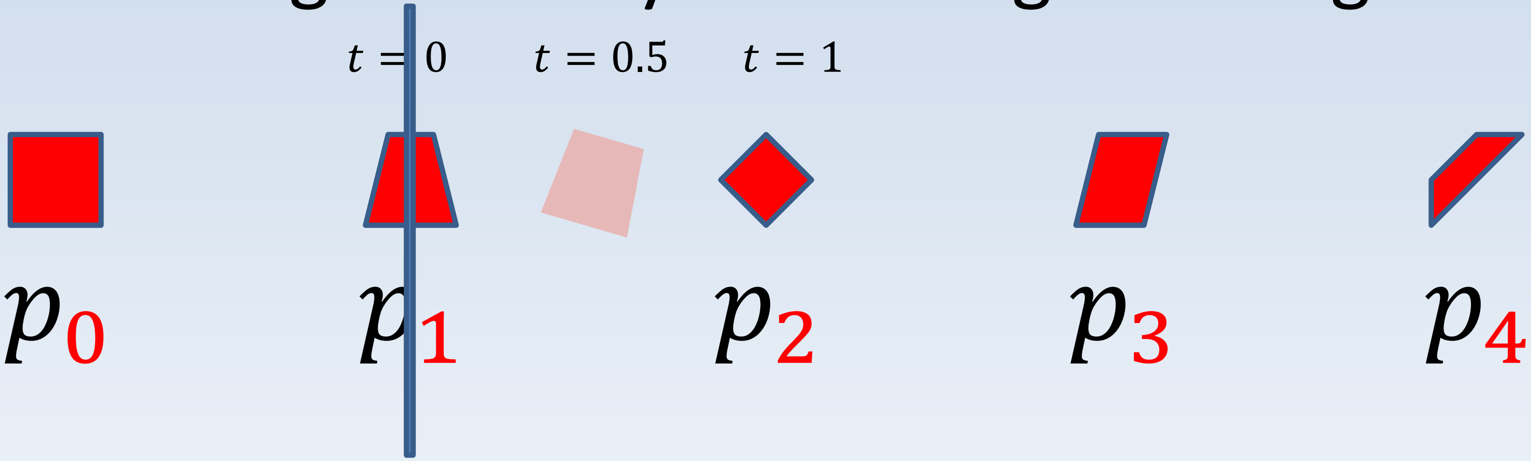


$\text{mesh} = \text{morph}(p_c, p_{c+1}, t)$

$c = 0$

Intro to Morphing

- Blending smoothly from “target to target”:



$t = 0$ $t = 0.5$ $t = 1$

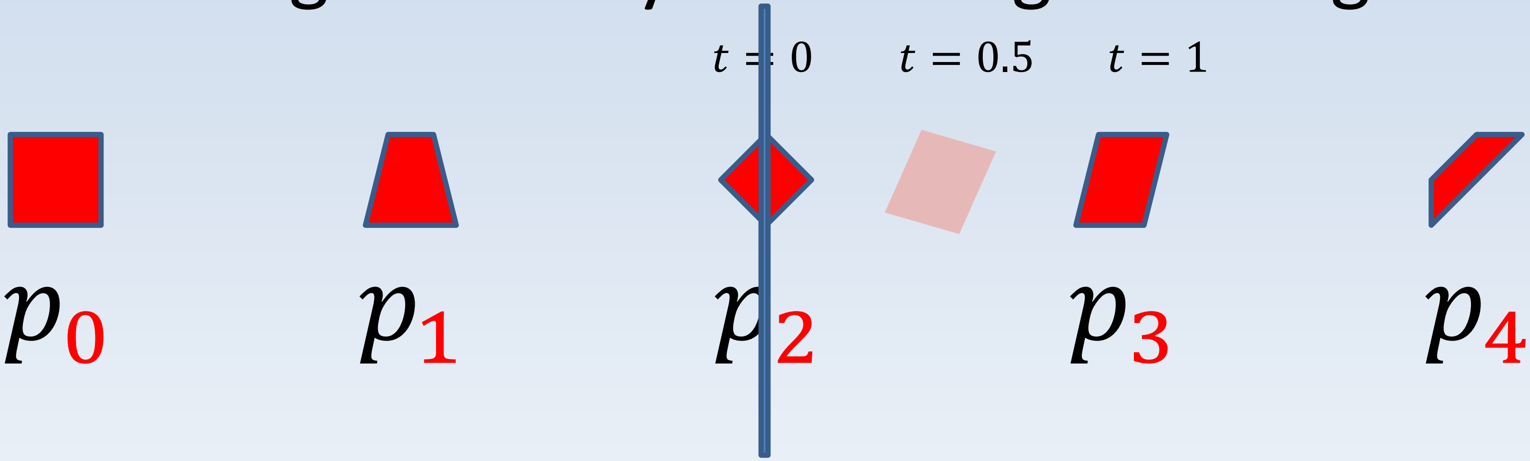
p_0 p_1 p_2 p_3 p_4

$\text{mesh} = \text{morph}(p_c, p_{c+1}, t)$

$c = 1$

Intro to Morphing

- Blending smoothly from “target to target”:



$\text{mesh} = \text{morph}(p_c, p_{c+1}, t)$

$c = 2$

Intro to Morphing

- If you’re familiar with Flash, *morph target animation* is the same as a “shape tween”
- The path interpolation algorithm is the same as a “motion tween”
- Same tool, different applications! ☺

Intro to Morphing

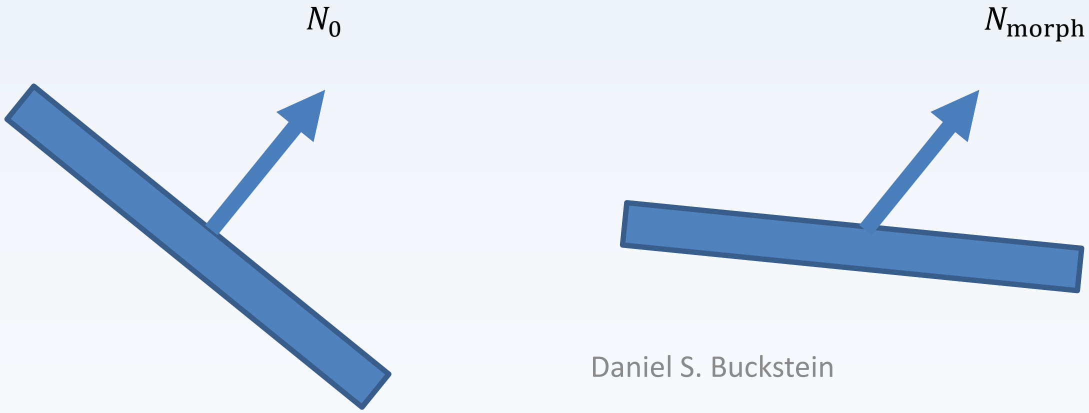
- Food for thought:
- Inheriting facial features from your parents
- Also why siblings kinda look alike…
- …you’re not identical at any given age because you have a slightly different *t* value!

$$\mathrm{RobertFace} = \mathrm{morph}(\mathrm{momFace}, \mathrm{dadFace}, 0.45)$$
$$\mathrm{HubertFace} = \mathrm{morph}(\mathrm{momFace}, \mathrm{dadFace}, 0.51)$$
$$\mathrm{DilbertFace} = \mathrm{morph}(\mathrm{momFace}, \mathrm{dadFace}, 0.62)$$

Intro to Morphing

- Food for thought:
- What happens to the lighting if we only morph vertices?

- Is there something else we need to do?



Intro to Morphing

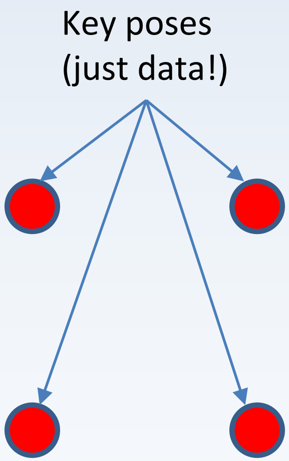
- Now… if we only use *LERP* to transition between morph target keyframes…
- …how will the animation look?

- Let’s learn some more types of interpolation!
- Remember, interpolation is just an algorithm!
- Can be applied to *anything*!

Applications of Paths & Curves

- Morph target smoothing:
- Given a morphing sequence of four keyframes, can create a *smooth cycle* using Catmull-Rom interpolation!

p_0


Key poses (just data!)

p_1

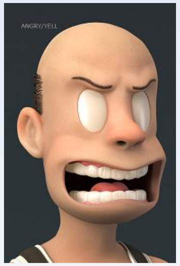
p_3

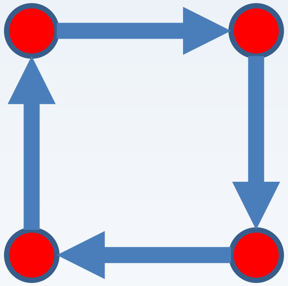
p_2

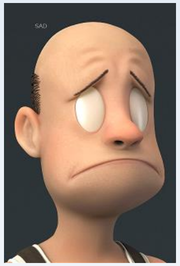
Applications of Paths & Curves

- Morph target smoothing:
- Standard morph algorithm would transition between targets as if it were a looping *linearly-segmented path*:

p_0

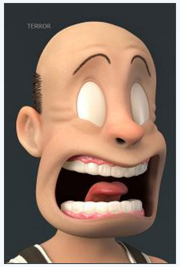


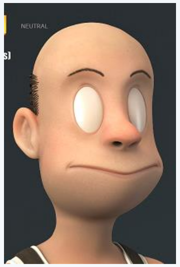




p_1

p_3

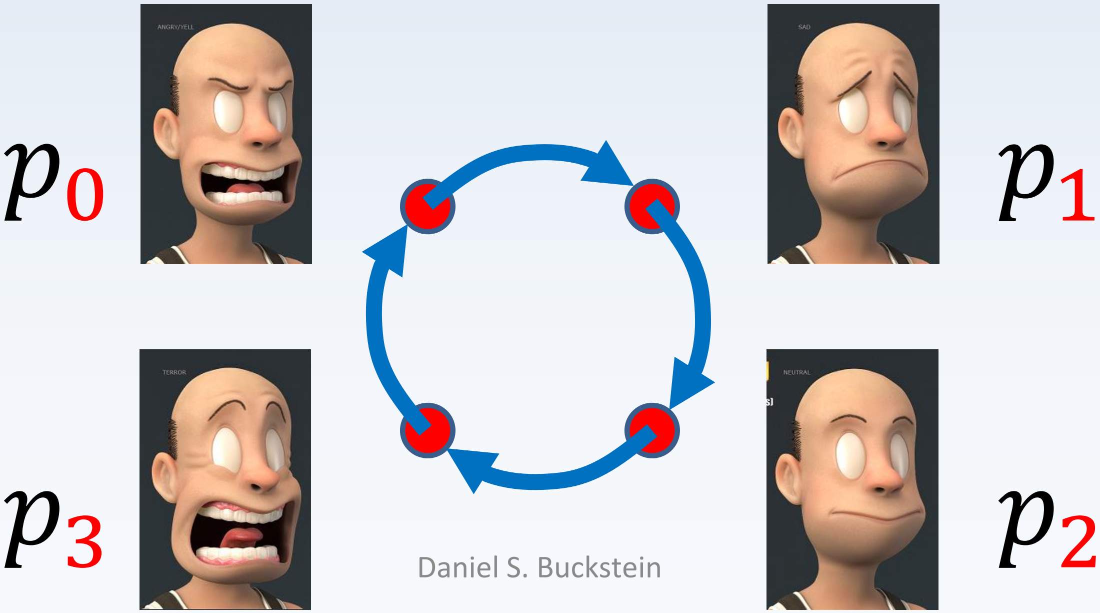




p_2

Applications of Paths & Curves

- Morph target smoothing:
- We can modify the algorithm a bit to use *all four targets at once* as inputs to Catmull-Rom



Applications of Paths & Curves

- Morph target smoothing:
- We can modify the algorithm a bit to use *all four targets at once* as inputs to Catmull-Rom

```
$\text{result} = \text{morph}(\text{target}_{c-1}, \text{target}_{c}, \text{target}_{c+1}, \text{target}_{c+2}, t)$:
    for each vertex $v$ in target:
        $v_{\text{result}} = \text{CatmullRom}\left(v_{\text{target}_{c-1}}, v_{\text{target}_{c}}, v_{\text{target}_{c+1}}, v_{\text{target}_{c+2}}, t\right)$
```

where ‘c’ is the current target keyframe!

The end.

- Questions? Comments? Concerns?