Intermediate Graphics & Animation Programming

GPR-300
Daniel S. Buckstein

Keyframe Systems
Weeks 10 – 11

License

Keyframe Animation

- Familiarize yourself with the ***12 principles of animation***
- Main takeaway for animation programming: All else means nothing without *timing*
- Today we look at the *straight-ahead and pose-to-pose* principle
- These describe two main types of animation

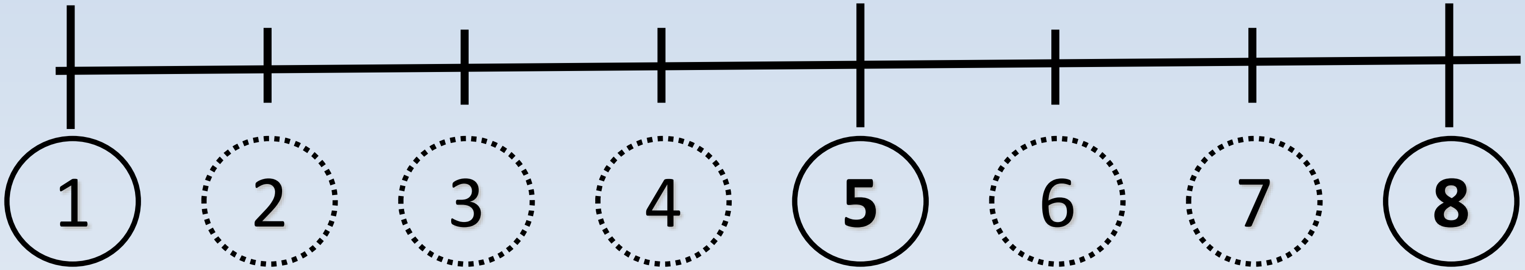
Keyframe Animation

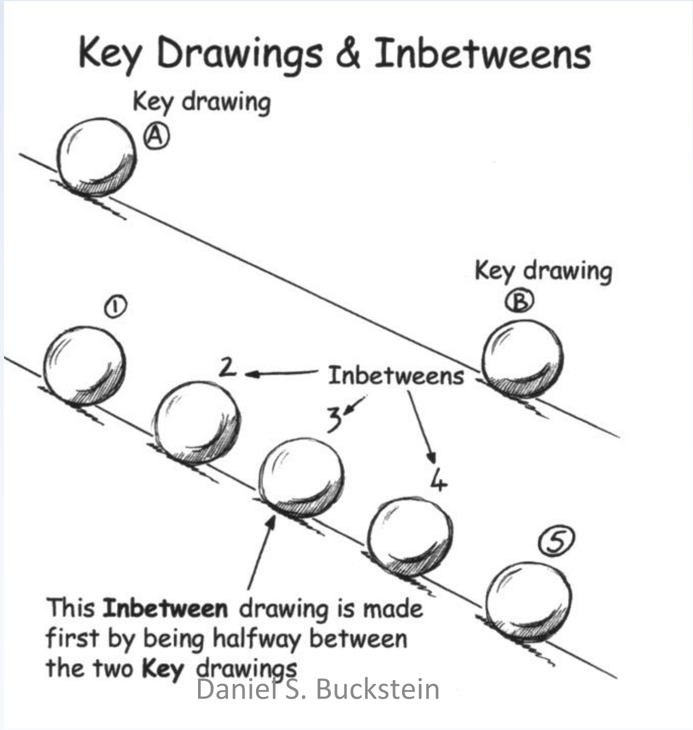
- ***Straight-ahead animation***:
- Frame-by-frame
- Example: stop-motion →
- Pros: full control over every single detail
- Cons: a lot of work, very time consuming to complete

Keyframe Animation

- ***Pose-to-pose animation***:
- Create *keyframes*, resolve in-betweens later
- Technique: onion-skinning
- Pros: keyframes are fixed, in-betweens are variable and can be adjusted as-needed

Keyframe Animation



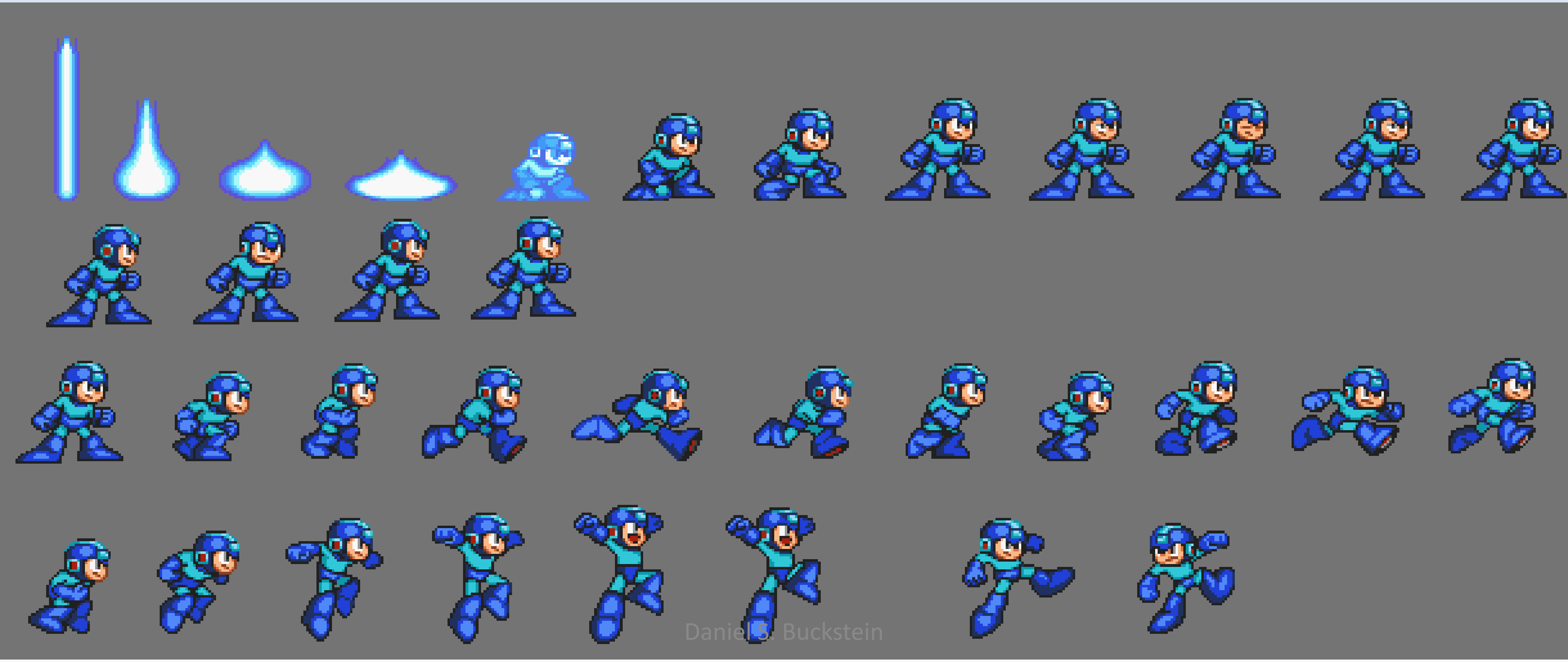

Key Drawings & Inbetweens
Key drawing
A
Key drawing
B
1
2
Inbetweens
3
4
5
This **Inbetween** drawing is made first by being halfway between the two **Key** drawings

Keyframe Animation

- Computer animation *works* because of ***pose-to-pose***
- We define keyframes and let *algorithms* take care of the in-betweens!
- Straight-ahead is useful for only some applications, like sprite animation
- For complex things, there is simply *too much data*… we need to make our lives easier!

Keyframe Animation

- ***Straight-ahead animation***:
- *Every* frame is a keyframe!



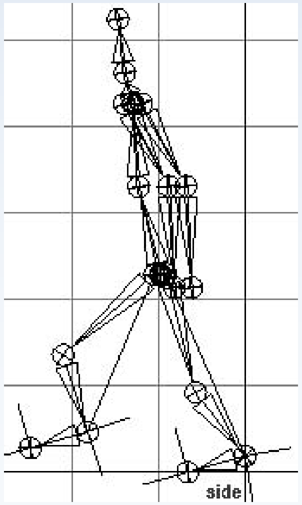
Keyframe Animation

- So what is a keyframe, really?
- Can be defined as “*Frames that are more important or denote some **key** moment or action in a sequence*”
- This is the theoretical definition
- In practice, keyframes are just *poses*
- Keyframes put the ‘*pose*’ in “pose-to-pose”

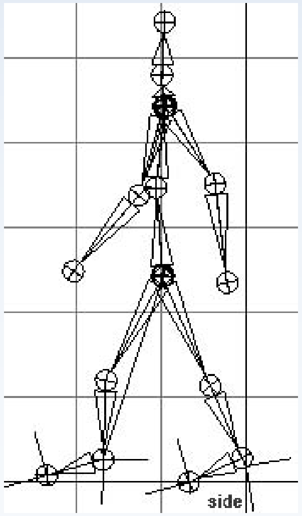
(both of them!)

Keyframe Animation

- Keyframes:
- A keyframe is just a *known pose* for whatever object or character we are dealing with
- Example: walk cycle



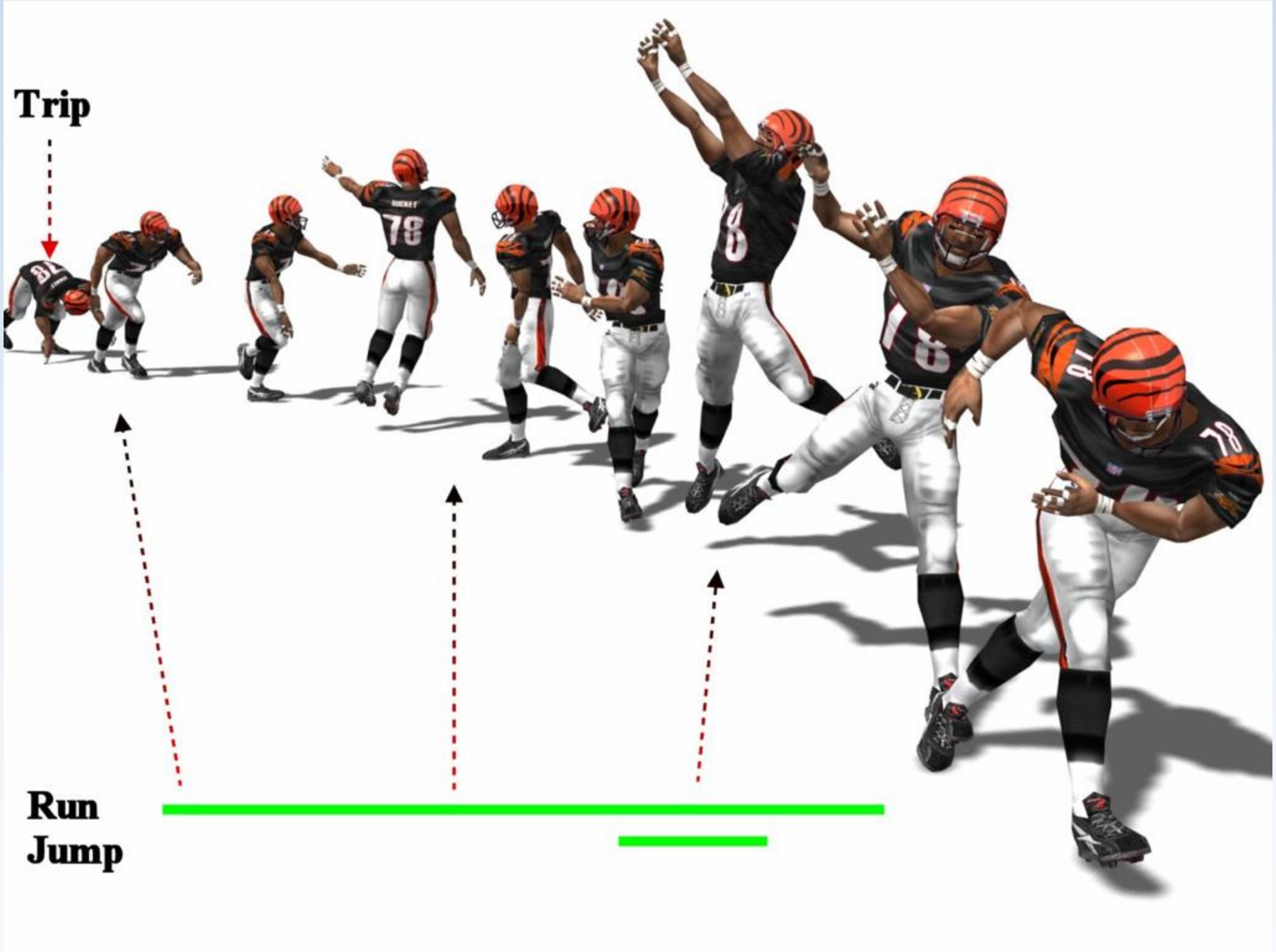
Keyframe 30



Keyframe 50

http://users.design.ucla.edu/~cariesta/MayaCourseNotes

Keyframe Animation



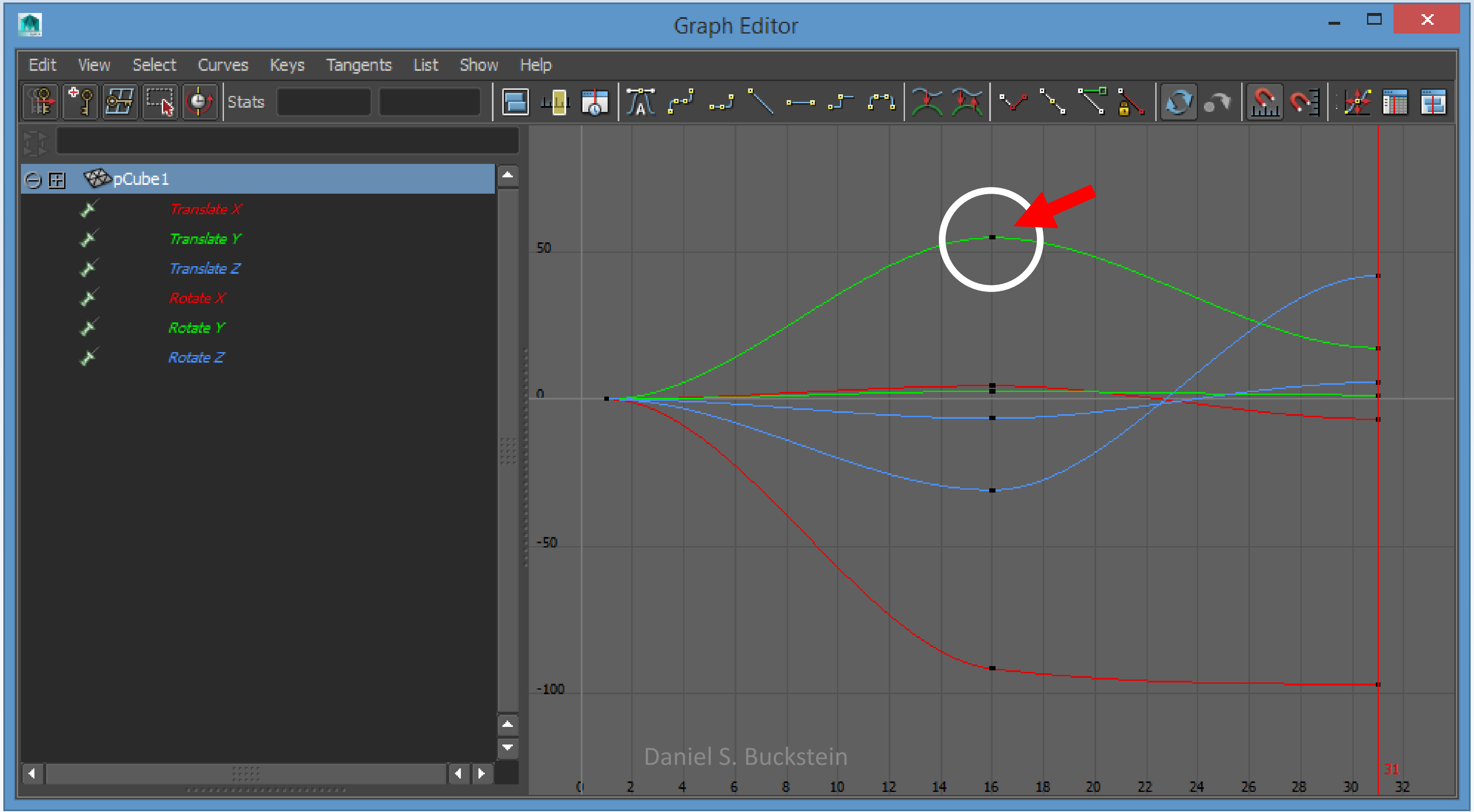
http://www.cs.utexas.edu/~okan/papers/s2003/pictures/fig1.jpg

Keyframe Animation

- Keyframes:
- Even calling a keyframe a “known pose” is too specific…
- A pose is really just the state of a set of data
- A keyframe is fundamentally just the *known values* in a set of data
- Each individual variable can have keyframes, so really keyframes are just *known numbers*!

Keyframe Animation

- Keyframes in Maya's graph editor:



Keyframe Animation

- Pose:
- A *pose* is just a set of data that is keyframed
- Has keyframes and in-betweens for all values
- Pose itself can be keyframed…
- This just means that all of its values have a keyframe at the same time

Keyframe Animation

- What kinds of data describe a pose?

- Position (vector)
- Orientation (rotation matrix or quaternion)
- Sometimes scale (uniform, non-uniform)

Keyframe Animation

- Games use keyframes extensively
- Many different applications:
- Morph targets
  - Mesh is deformed into *poses*, we just morph between full meshes
- Skeletal animation
  - Bones define the poses, mesh conforms to bones
- What else???

Quake II (.md2)
Keyframes for full mesh stored

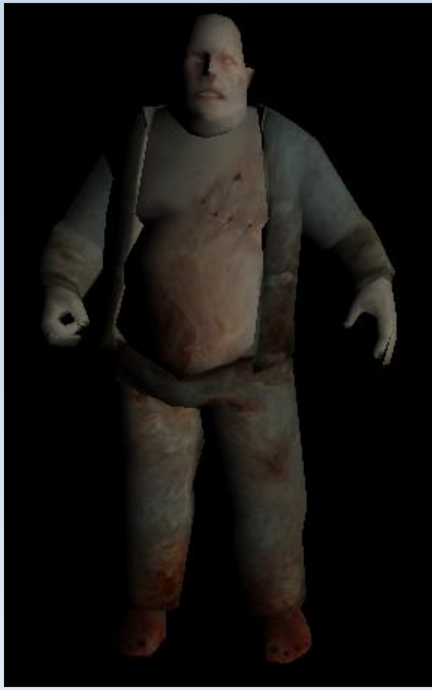
http://www.youtube.com/watch?v=NUHudbgxWfY

http://www.youtube.com/watch?v=rO37U8KLRws

.md3
separate meshes
keyframes stored

http://www.youtube.com/watch?v=vaVhcnBiob0

http://www.youtube.com/watch?v=tzyuIsdT8GY&feature=results_video&playnext=1&list=PL342E78FF3096F00E



.md5
only keyframes of
skeleton stored

http://www.youtube.com/watch?v=y
I7tMfiU&feature=related

http://www.youtube.com/watch?v=5ulV-vAqtaQ&feature=BFa&list=PL342E78FF3096F00E&lf=results_video

Keyframe Animation

- Problems:
- Inflexible
- Hard to incorporate keyframe animation into physics
  - Both have their own way of controlling things
- Hard to interact with an object while it undergoes keyframe animations
  - Frames are determinate, interactions are not!

Keyframe Animation

- But on the bright side, good advantages:
- Keyframe animation (pose-to-pose) is very easy to implement on a computer!
- Can tweak animations effectively to achieve desired effect, outcome, sequences

Keyframing with LERP

- Now we have all the words in “*pose-to-pose*” covered… what next?
- Let’s apply LERP to keyframe animation!
- How would you build a *simple* keyframed locomotion system?
- I.e. actual walking aside, how would we get Watson from location to location?

…instead of having him creepily teleport

Keyframing with LERP

- Determine what your *keyframes* actually represent, data-wise
- Examples (again):
  - Position
  - Orientation
  - Scale
  - Whatever other property you may be concerned with

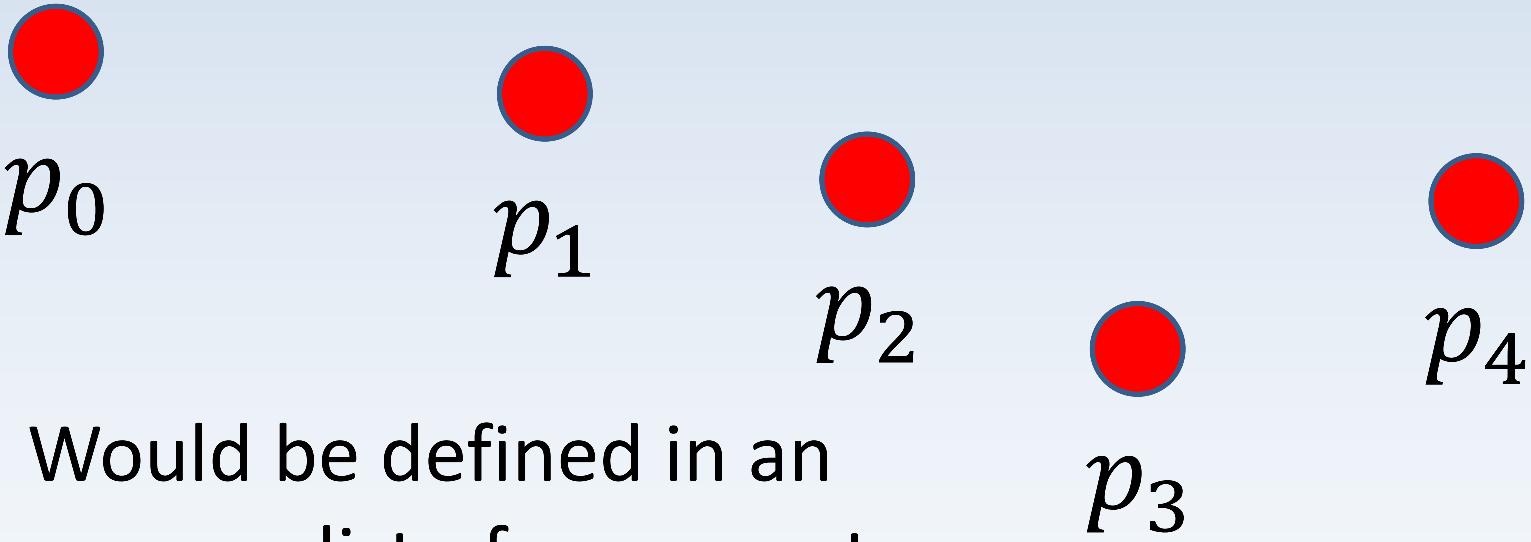
Keyframing with LERP

- For Watson, let’s keep it simple and say that one keyframe is just a *known position in space*

- Create a bunch of these to determine our “*path*” that Watson is to follow, store in some kind of list

Keyframing with LERP

- Watson’s locomotion keyframes:

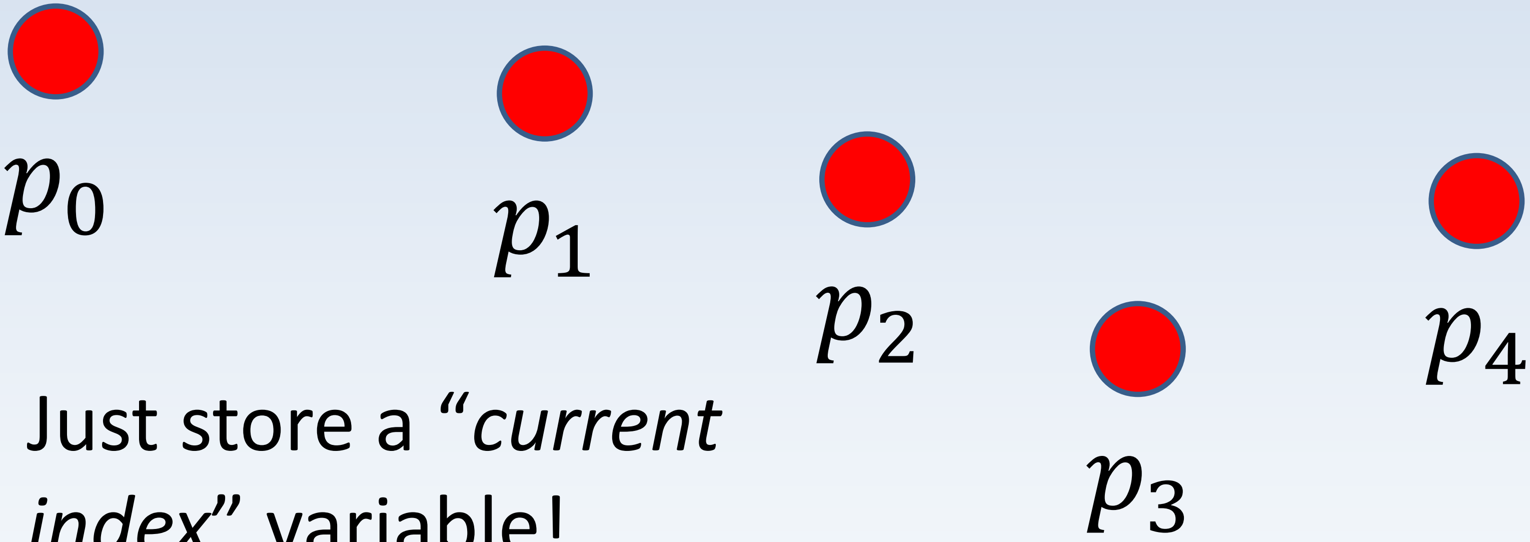


- Would be defined in an array or list of some sort…

The *subscripts* are just ***indices***!!!

Keyframing with LERP

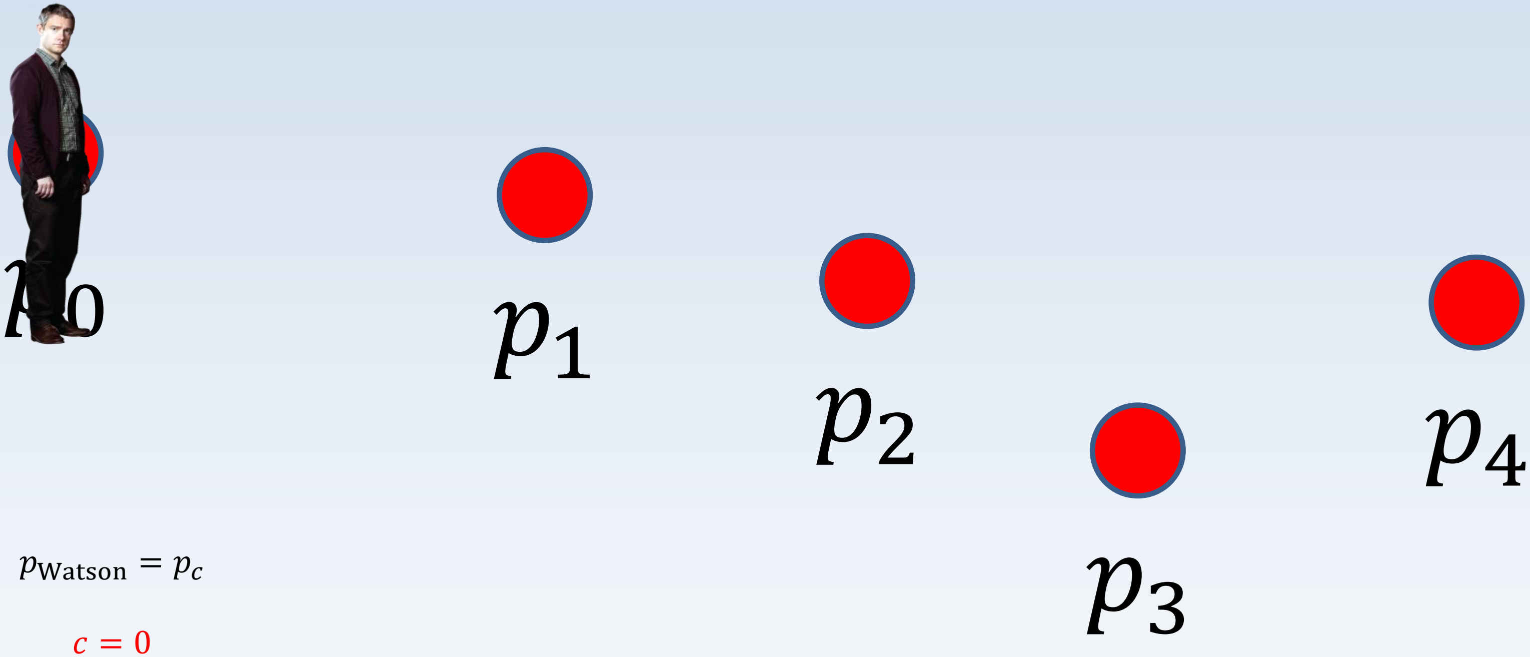
- How do we toggle the *current keyframe*?
  - I.e. jump from keyframe to keyframe?



- Just store a "*current index*" variable!
- Example: 'c'

Keyframing with LERP

- Jumping between keyframes:

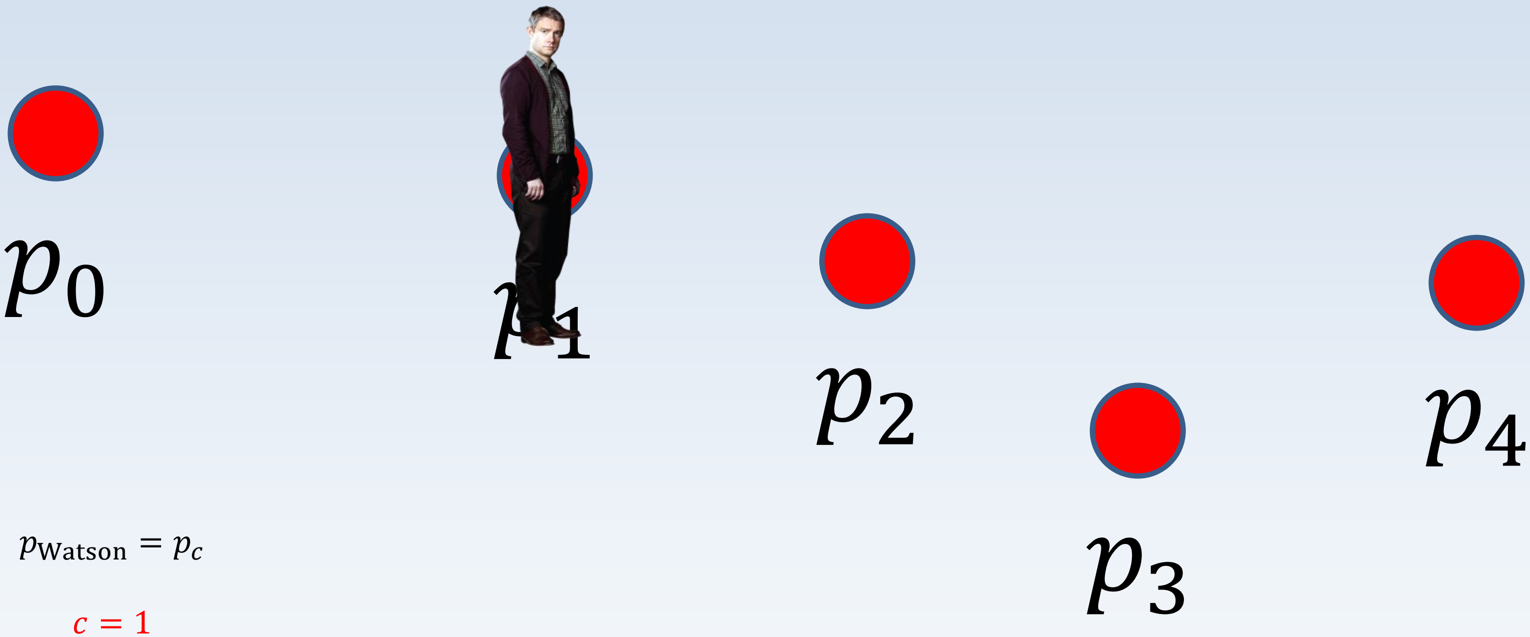


$p_{\text{Watson}} = p_c$

$c = 0$

Keyframing with LERP

- Jumping between keyframes:



p_0

p_1

p_2

p_3

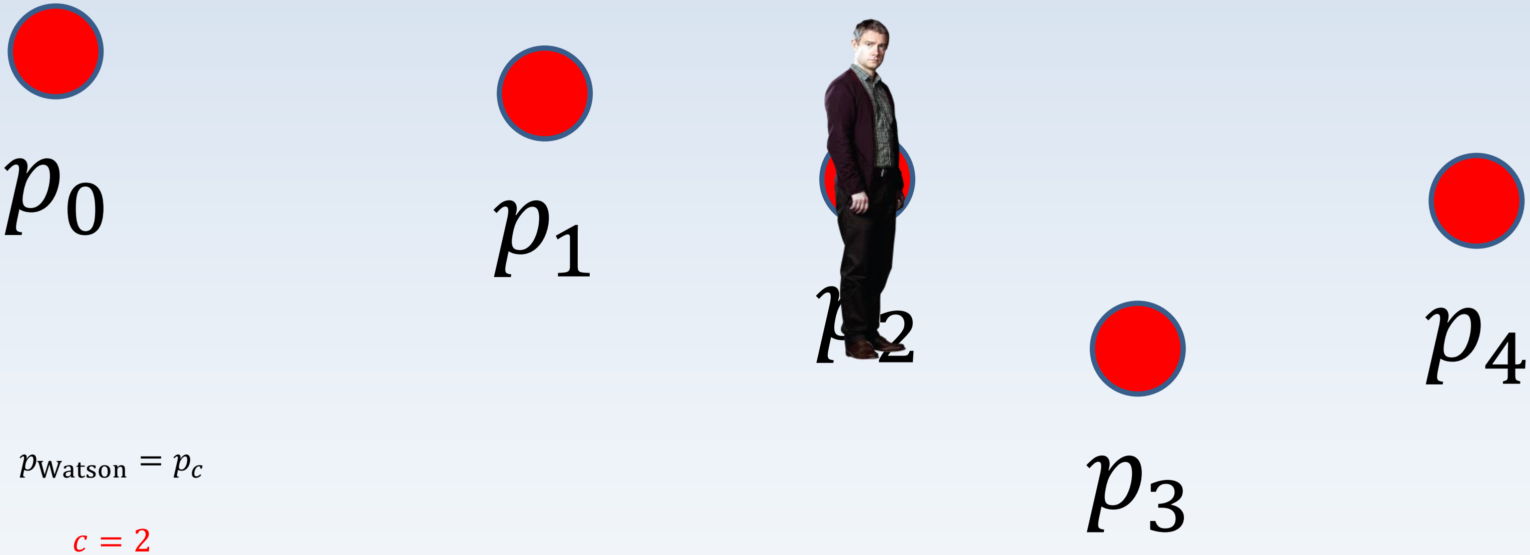
p_4

$p_{\text{Watson}} = p_c$

$c = 1$

Keyframing with LERP

- Jumping between keyframes:

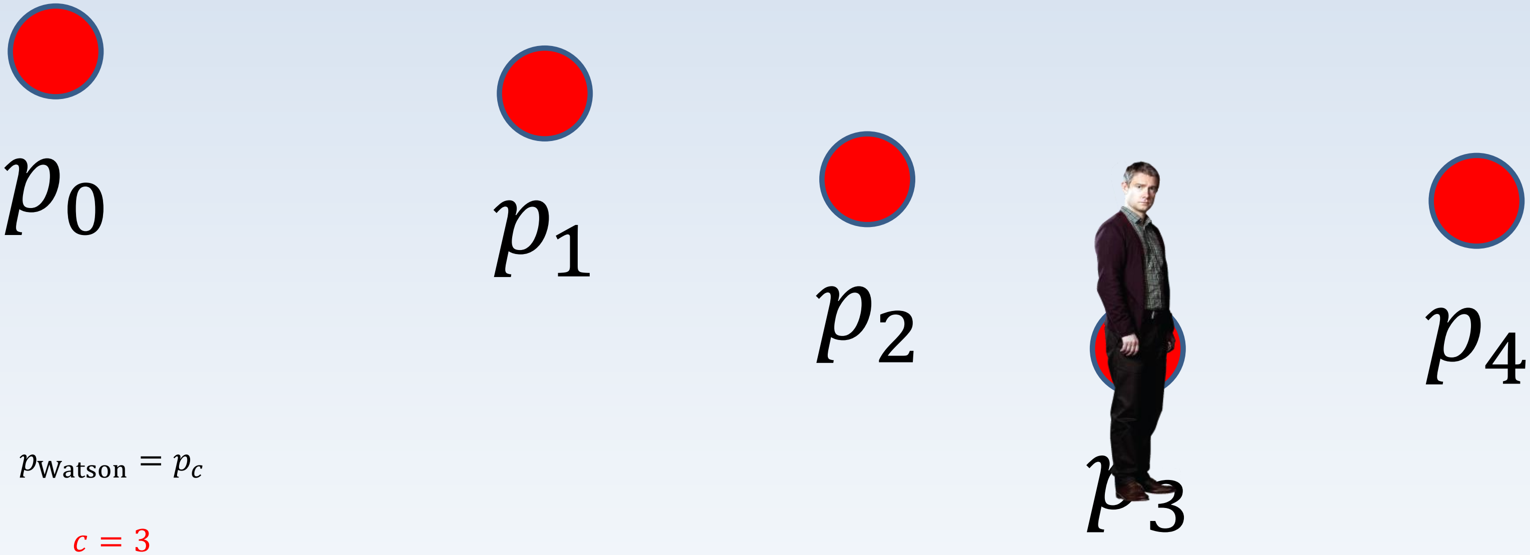


$p_{\text{Watson}} = p_c$

$c = 2$

Keyframing with LERP

- Jumping between keyframes:



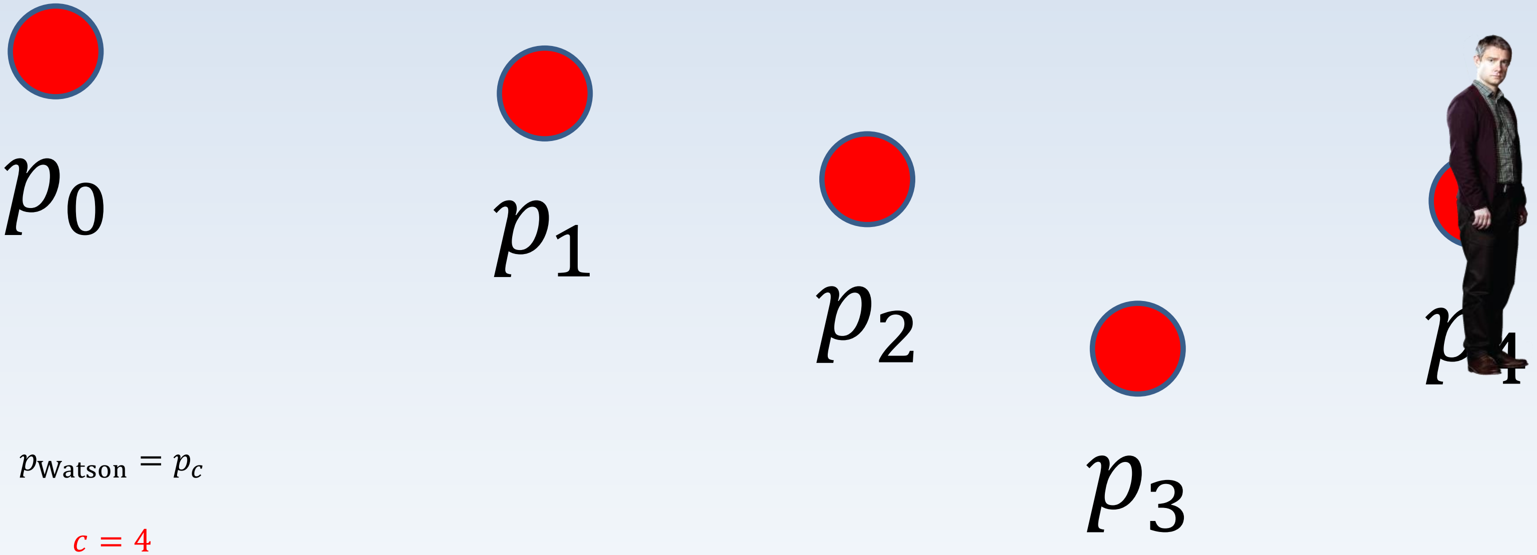
p_0 p_1 p_2 p_3 p_4

$p_{\text{Watson}} = p_c$

$c = 3$

Keyframing with LERP

- Jumping between keyframes:

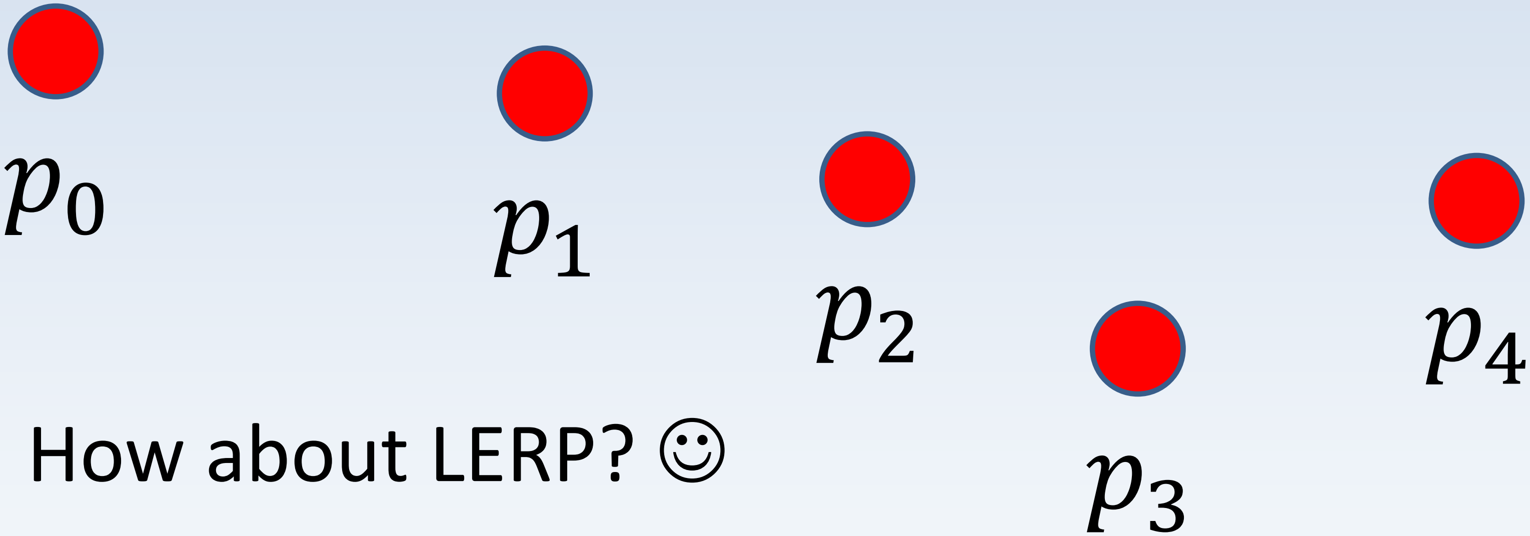


$p_{\text{Watson}} = p_c$

$c = 4$

Keyframing with LERP

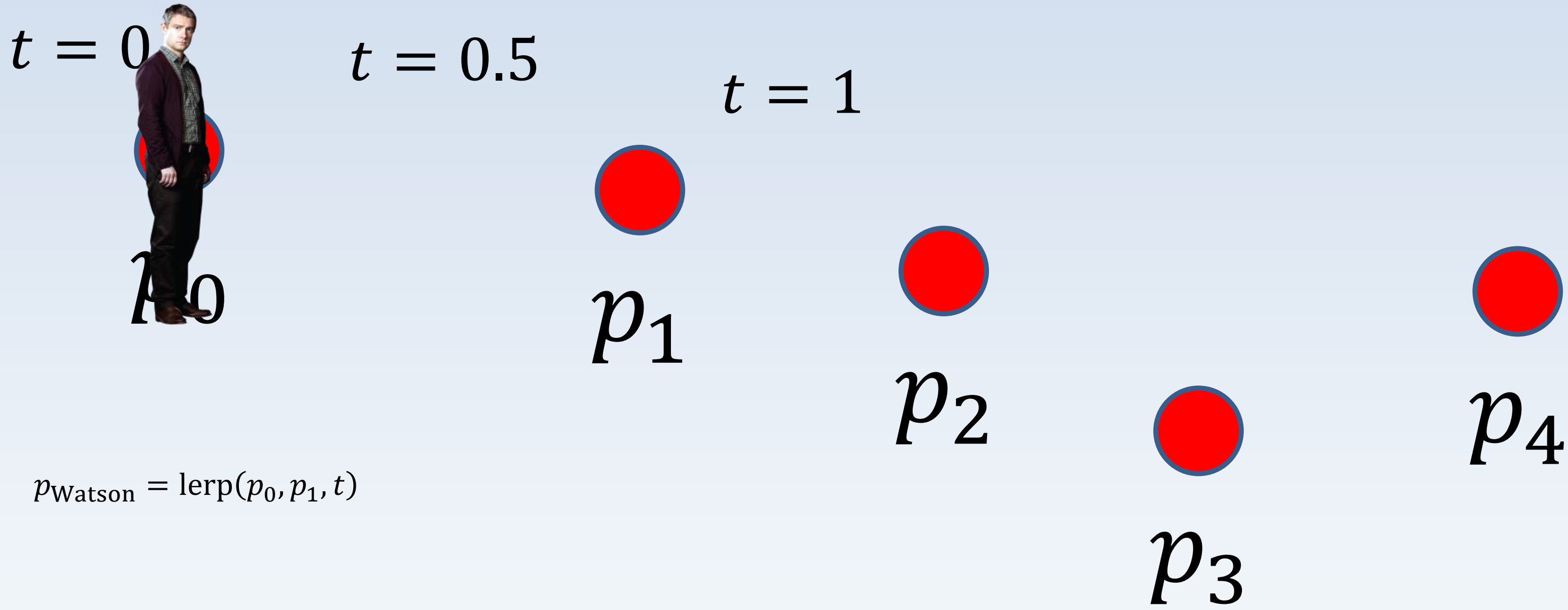
- How do we move smoothly from p0 to p1?



- How about LERP? ☺

Keyframing with LERP

- Moving smoothly from p0 to p1 using LERP:

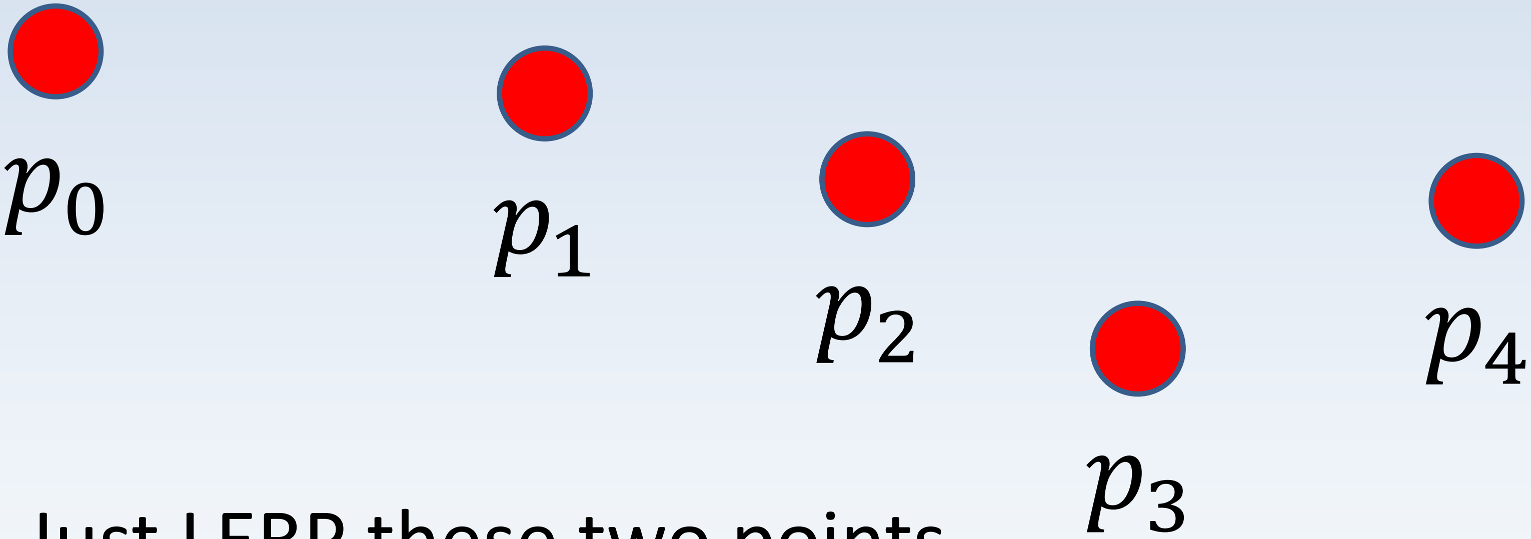


$p_{\text{Watson}} = \text{lerp}(p_0, p_1, t)$

- Remember, right now we have control over ‘*t*’

Keyframing with LERP

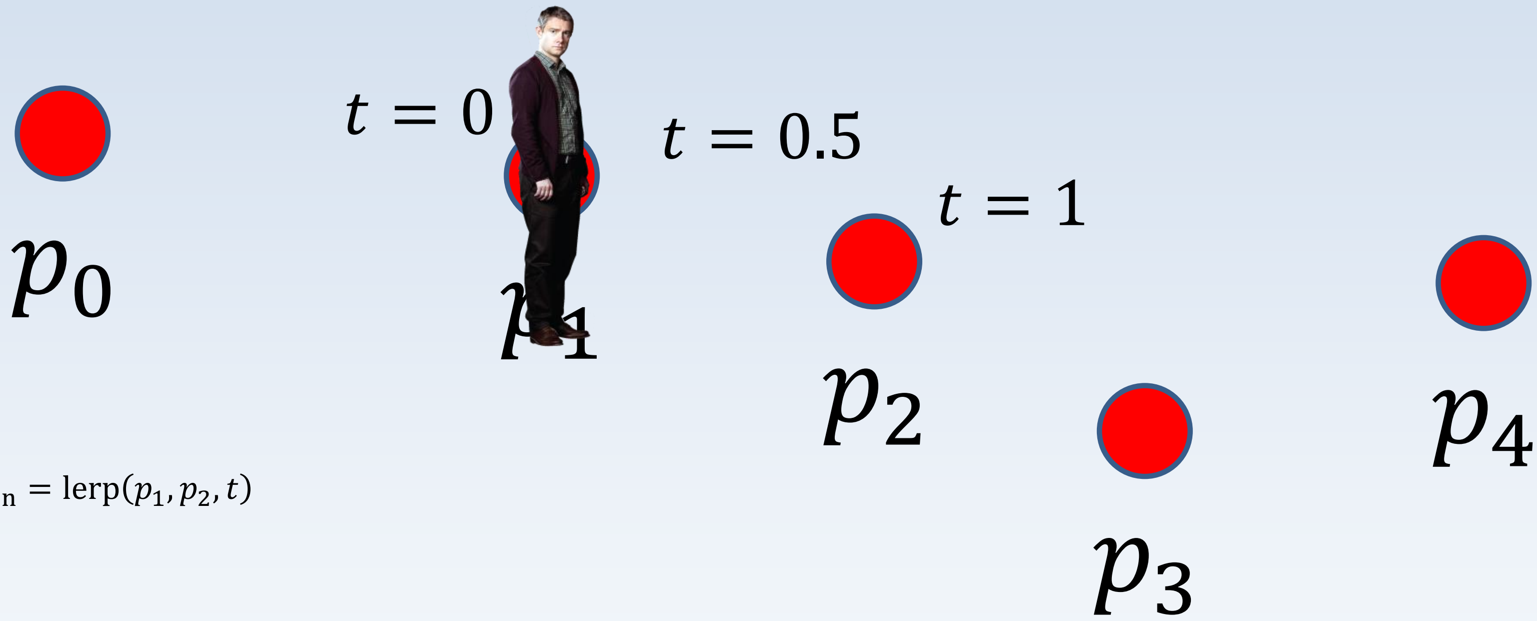
- How do we move smoothly from p1 to p2?



- Just LERP these two points instead of p0 and p1, rinse & repeat!

Keyframing with LERP

- Moving smoothly from p1 to p2 using LERP:

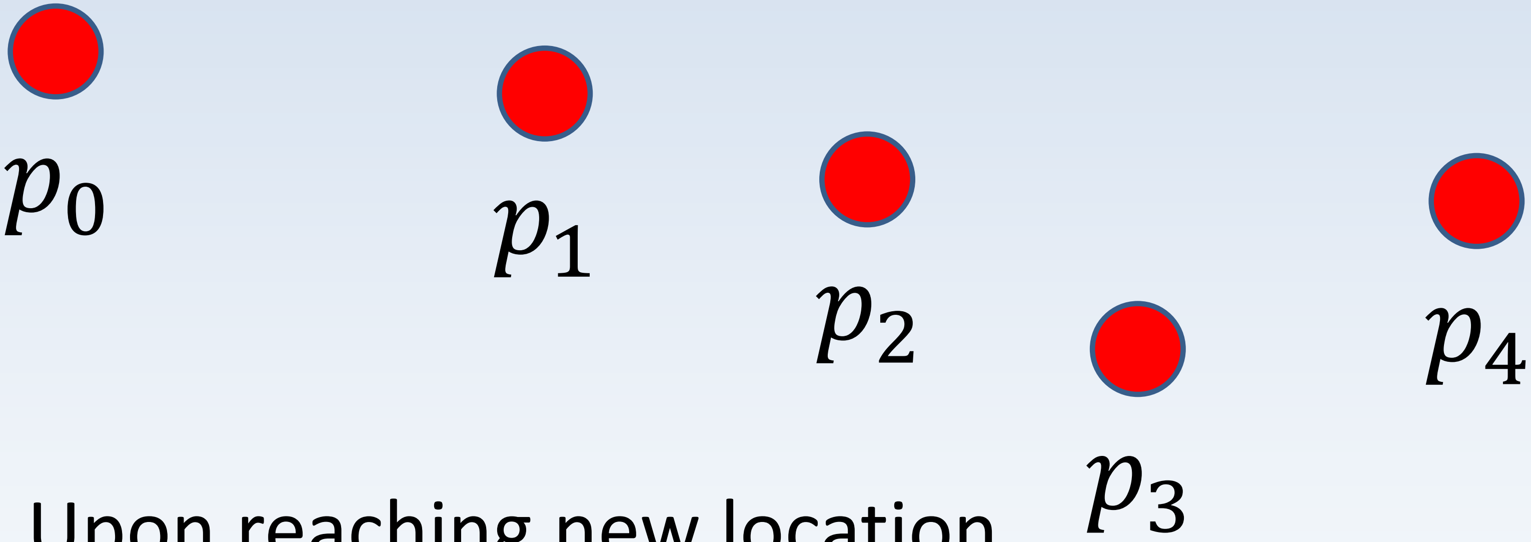


$p_{\text{Watson}} = \text{lerp}(p_1, p_2, t)$

- …still have control over '*t*'

Keyframing with LERP

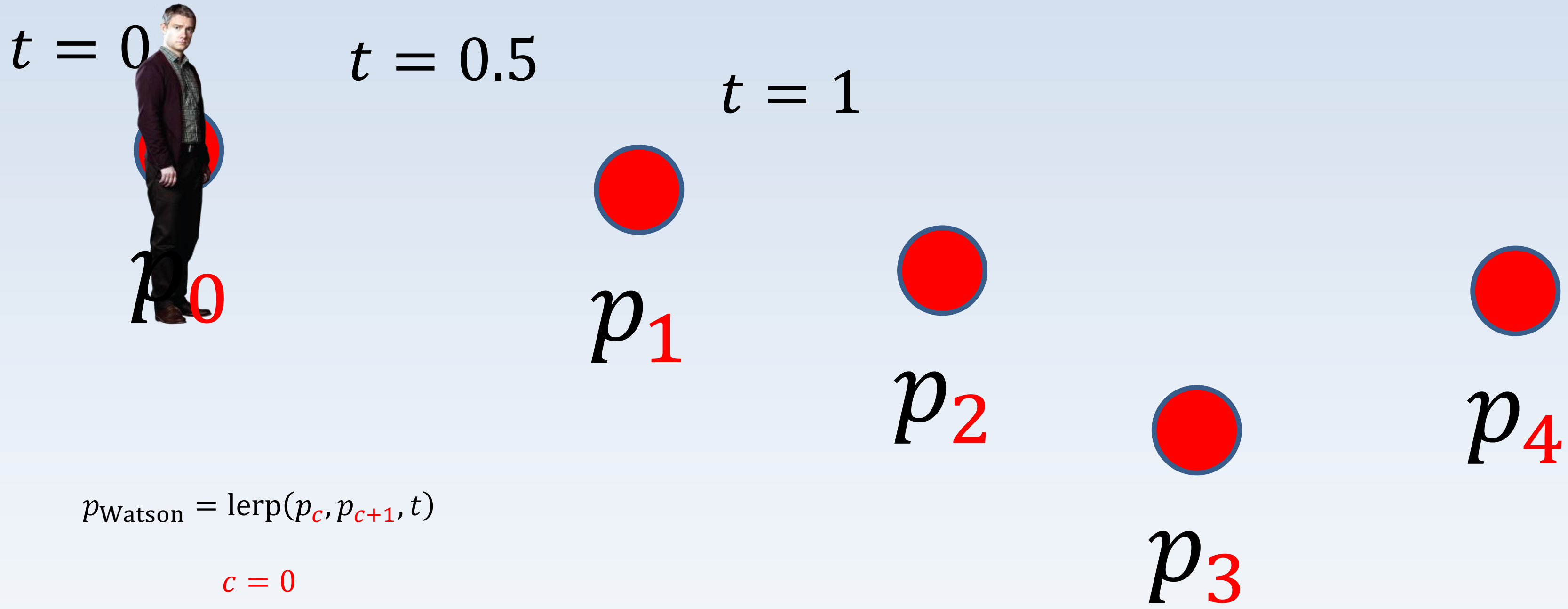
- So what is the actual *algorithm* used here?



- Upon reaching new location…
- …reset *t* and increment *current index!!!*

Keyframing with LERP

- Moving smoothly from “pose to pose”:

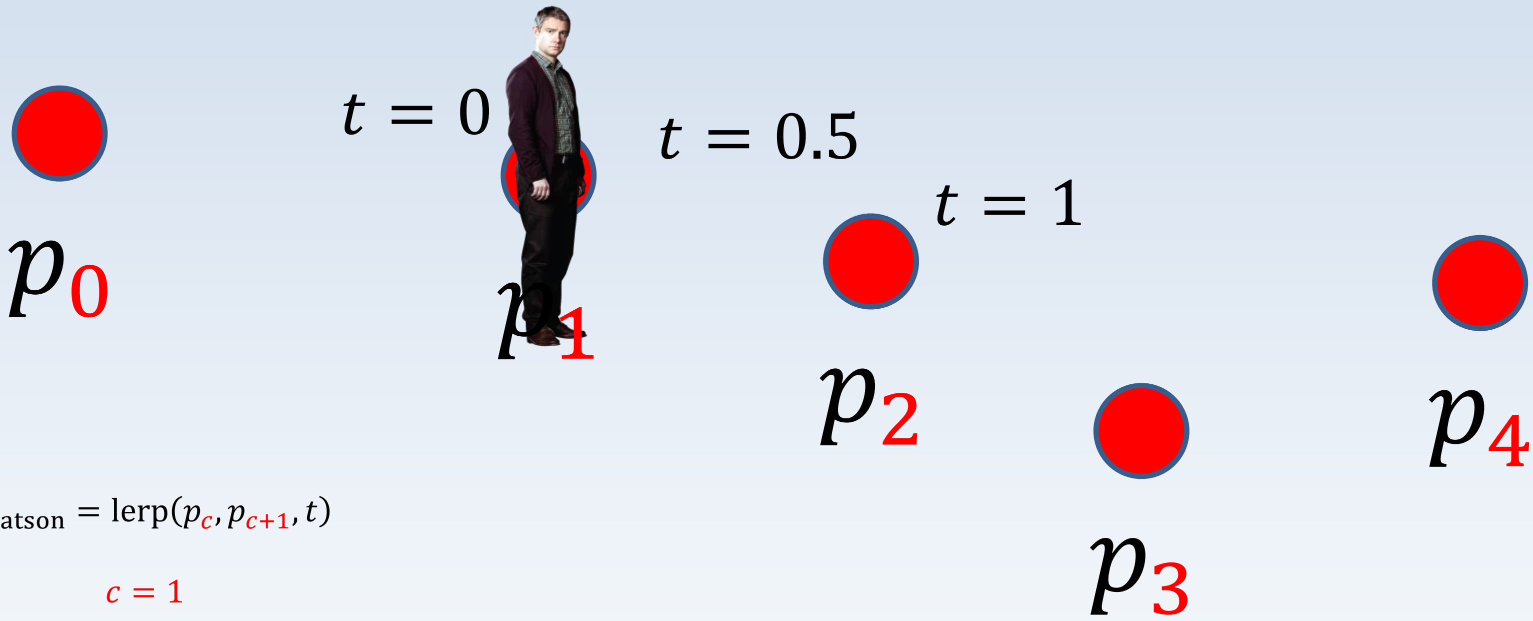


$p_{\text{Watson}} = \text{lerp}(p_c, p_{c+1}, t)$

$c = 0$

Keyframing with LERP

- Moving smoothly from “pose to pose”:



$t = 0$

$t = 0.5$

$t = 1$

p_0

p_1

p_2

p_3

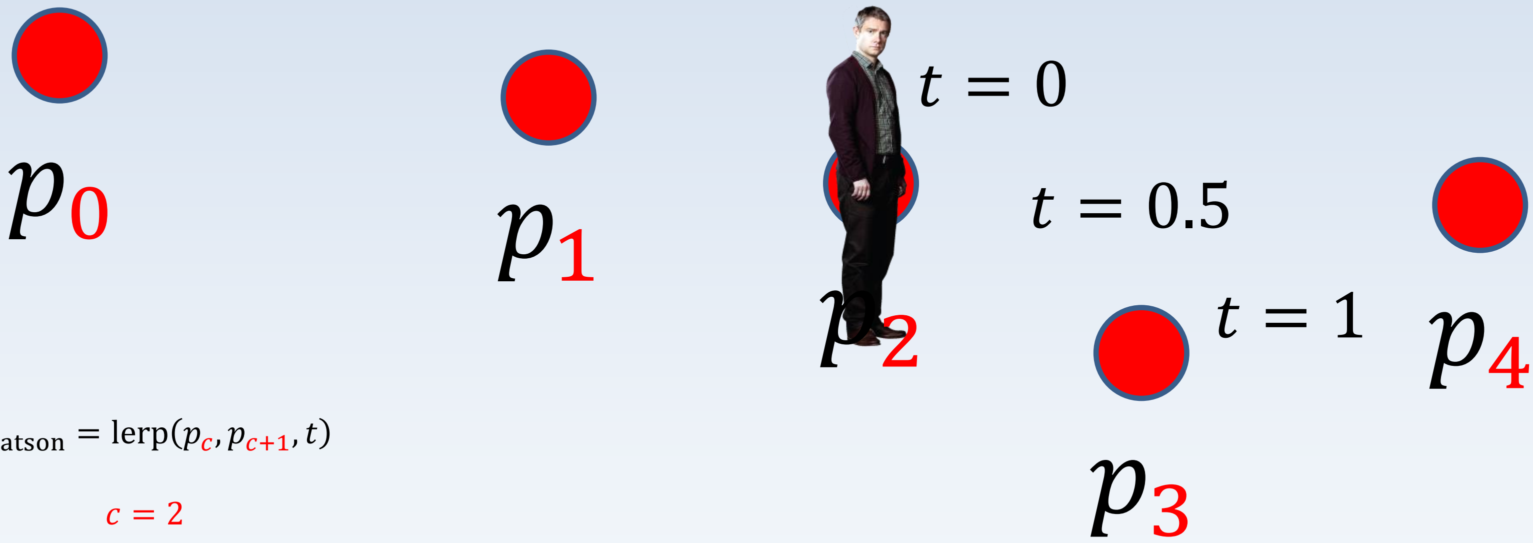
p_4

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

$c = 1$

Keyframing with LERP

- Moving smoothly from "pose to pose":

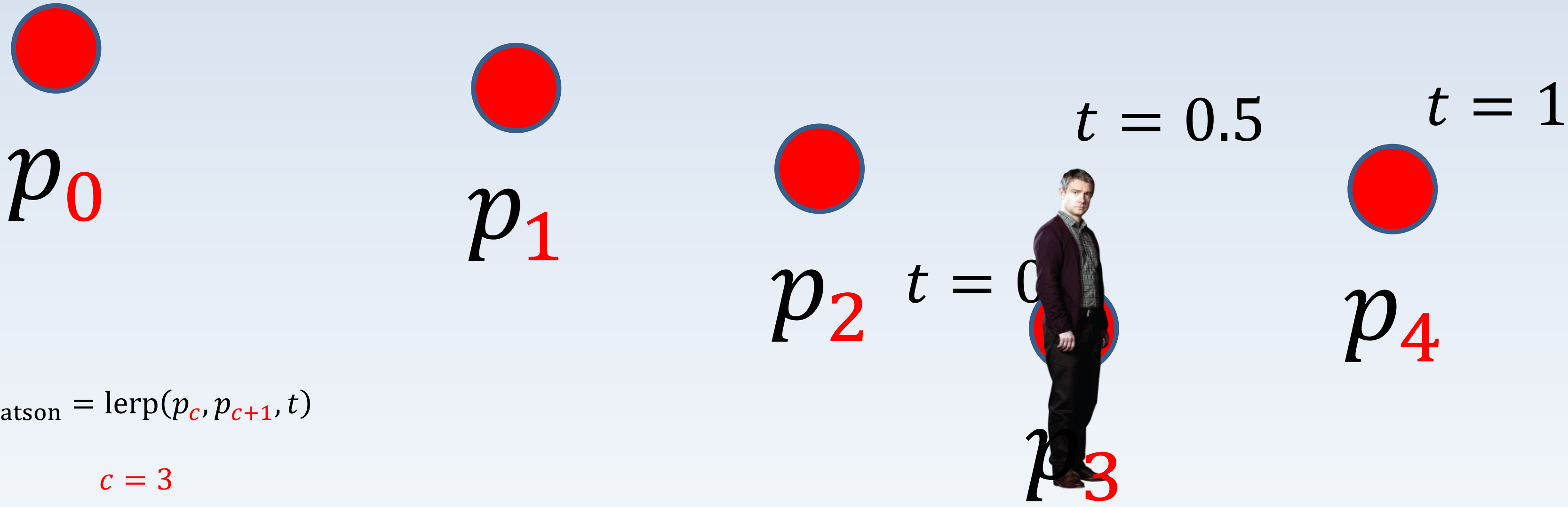


$p_{\text{Watson}} = \text{lerp}(p_c, p_{c+1}, t)$

$c = 2$

Keyframing with LERP

- Moving smoothly from “pose to pose”:

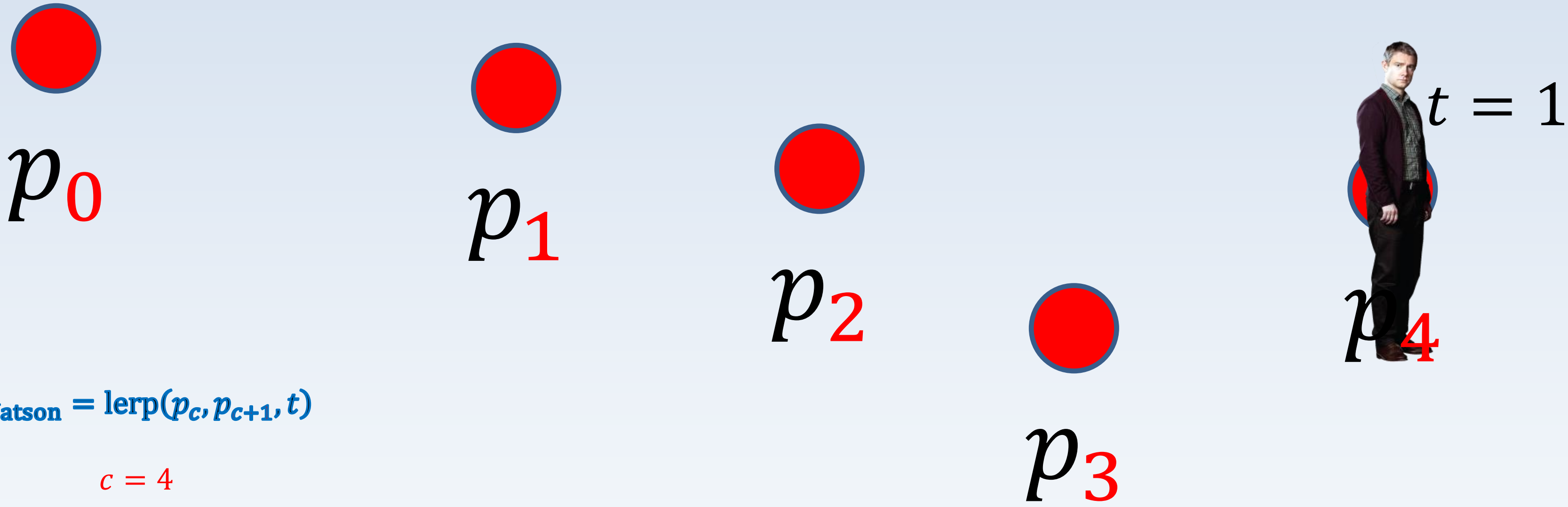


$p_{\text{Watson}} = \text{lerp}(p_c, p_{c+1}, t)$

$c = 3$

Keyframing with LERP

- Notice how the LERP call did not change once during that entire process!!!

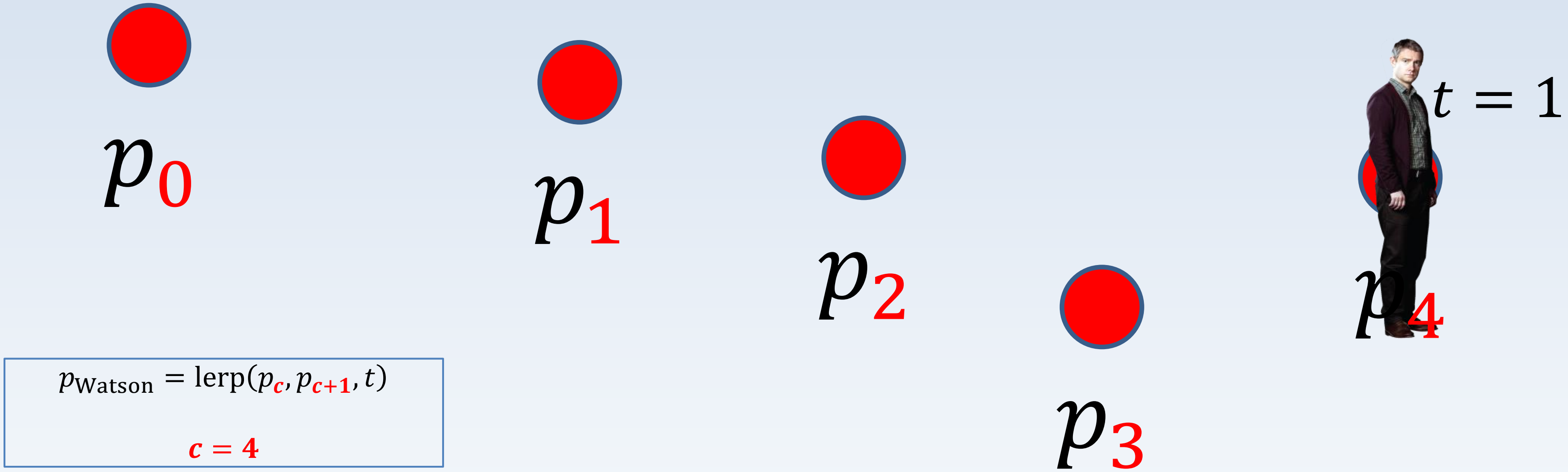


$p_{\text{Watson}} = \text{lerp}(p_c, p_{c+1}, t)$

$c = 4$

Keyframing with LERP

- All we did was keep track of the *current keyframe index*

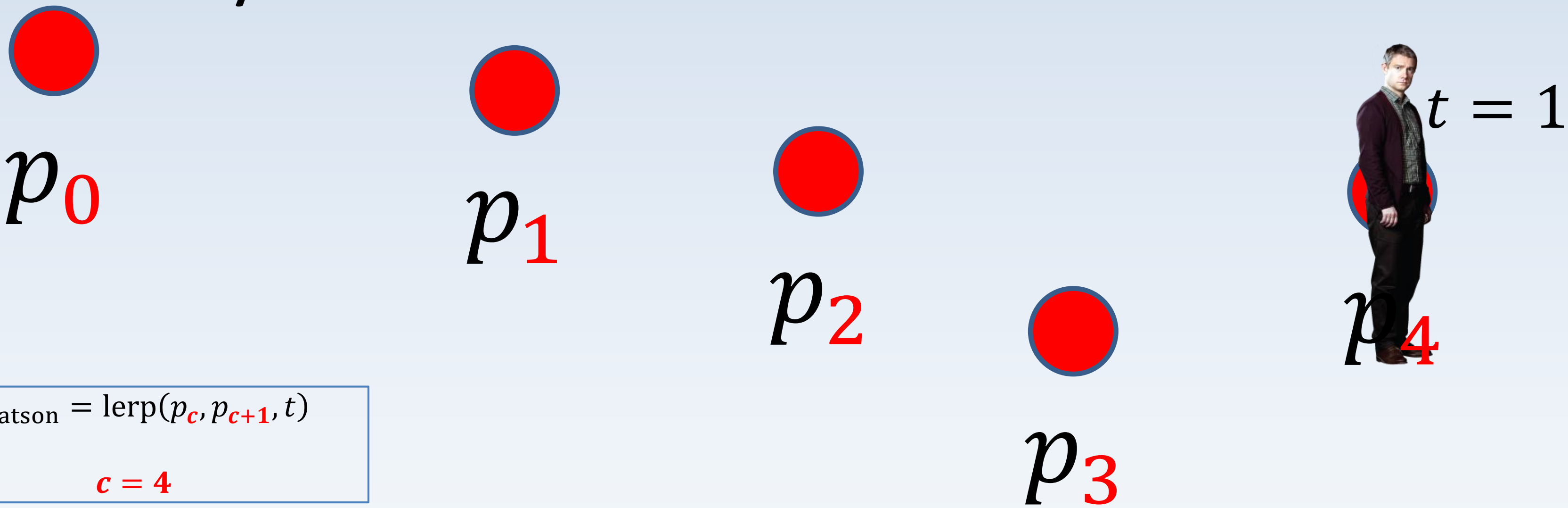


$p_{\text{Watson}} = \text{lerp}(p_{\boldsymbol{c}}, p_{\boldsymbol{c+1}}, t)$

$\boldsymbol{c = 4}$

Keyframing with LERP

- The index value is incremented… when???
- …every time $t \geq 1$



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

$$\boldsymbol{c = 4}$$

Keyframing with LERP

- How would we *play in reverse* using the exact same LERP *algorithm*???
- Actually *two* very fast and simple solutions
- Think about it… what simple changes would result in reverse playback?

$$\text{lerp}(\underbrace{p_0, p_1}_{\text{METHOD 1 HINT}}, t) = (\underbrace{1 - t}_{\text{METHOD 2 HINT}})p_0 + (t)p_1$$

Keyframing with LERP

- **Summary**:
- LERP is incredibly important for keyframed animation
- Locomotion is just one application
- Position is just one kind of data
- Literally everything else *this year* builds on this concept

Keyframing with LERP

- **Summary**:
- *All of the above examples* are written in their pure mathematical forms
- Luckily, *all of it translates* ***directly*** *and* ***easily*** *into code!!!*
- Algorithms are just functions
- Functions are math… get used to it ☺

The end.

- Questions? Comments? Concerns?