the gamedesigninitiative
at cornell university

Lecture 8

2D Animation

Animation Basics: Sprite Sheets

- Animation is a sequence of **hand-drawn frames**
  - Smoothly displays action when change quickly
  - Also called flipbook animation

- Arrange animation in a **sprite sheet** (one texture)
  - Software chooses which frame to use at any time
  - Programmer is actually the one doing animation

2D Animation

CUGL Has Two Options

SpriteNode

- Scene graph node
  - Fits into JSON specification
  - Draw automatically

- Does require a scene graph
  - Limits drawing flexibility
  - Bad for some perspectives (e.g. isometric)

SpriteSheet

- A stand-alone class
  - But familiar drawing API
  - Pass SpriteBatch & transform

- Not part of a scene graph
  - Has a lot more flexibility
  - But less input from designer

2D Animation

the gamedesigninitiative
at cornell university

CUGL Has Two Options

SpriteNode

- Scene graph node
  - Fits into JSON specification
  - Draw automatically

- Does require a scene graph
  - Limits drawing flexibility
  - Bad for some perspectives (e.g. isometric)

SpriteSheet

- A stand-alone class
  - But familiar drawing API
  - Pass SpriteBatch & transform

- Not part of a scene graph
  - Has a lot more flexibility
  - But less input from designer

> Both are stateful!
> These are not assets.

2D Animation

Adjusting your Speed

- Do not want to go too fast
    - 1 animation frame = 16 ms
    - Walk cycle = 8/12 frames
    - Completed in 133-200 ms

- General solut
    - Add an int tim
    - Go to next fram
    - Reset it to > 0 at new frame

- Simple but tedious
    - Have to do for each object
    - Assumes animation is in a loop

To programmer focused.
How is designer involved?

2D Animation

the gamedesigninitiative
at cornell university

Animation Concept: Keyframes
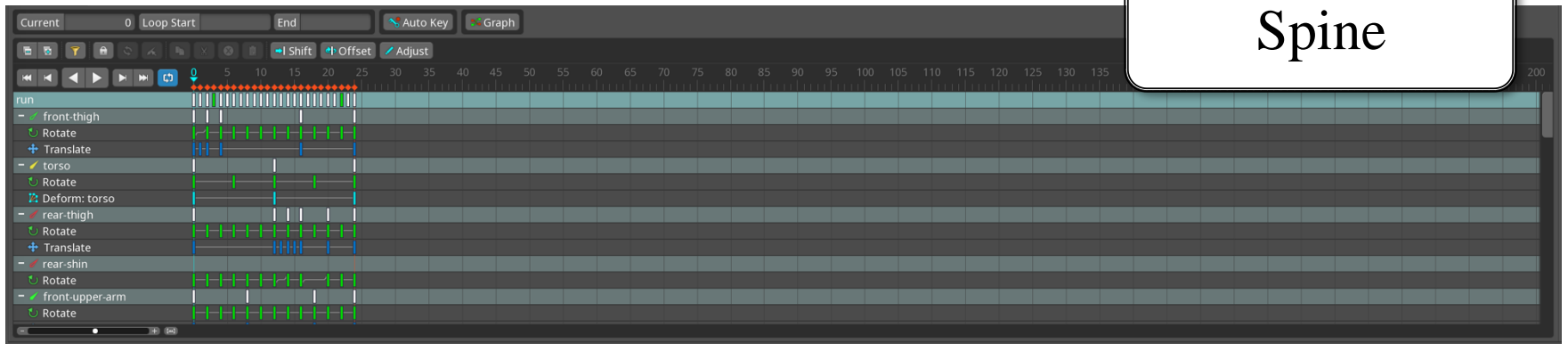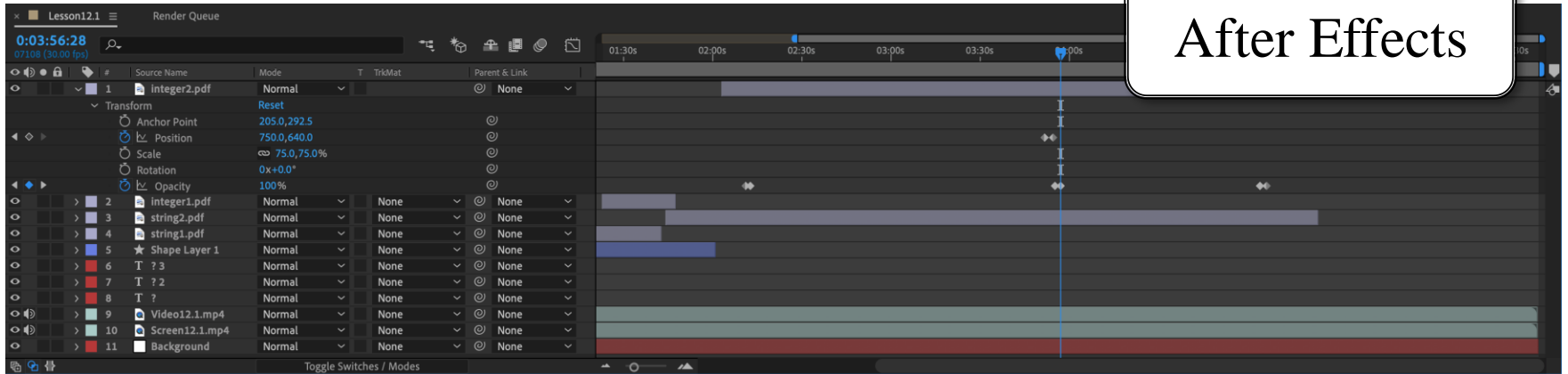

After Effects

Spine

KeyFrames

- Animator creates a timeline
  - Specifies how long animation will last
  - Specifies when a change happens (key frames)

- Ideally, want this in game engine too.

1 second

0.2 secs   0.3 secs   0.4 secs   0.1 secs

2D Animation

KeyFrames

- Animator creates a timeline
  - Specifies how long animation will last
  - Specifies when a change happens (key frames)

- Ideally, want this in game engine too.

1 second

Notice this is non-uniform

0.2 secs   0.3 secs   0.4 secs   0.1 secs
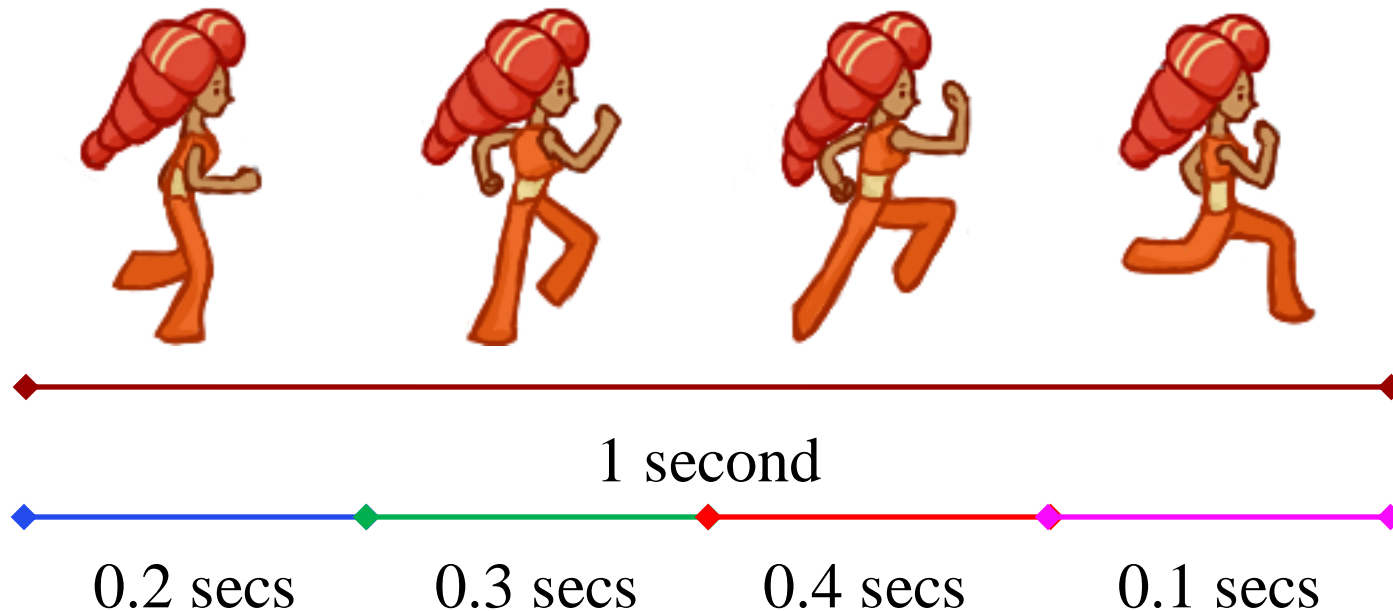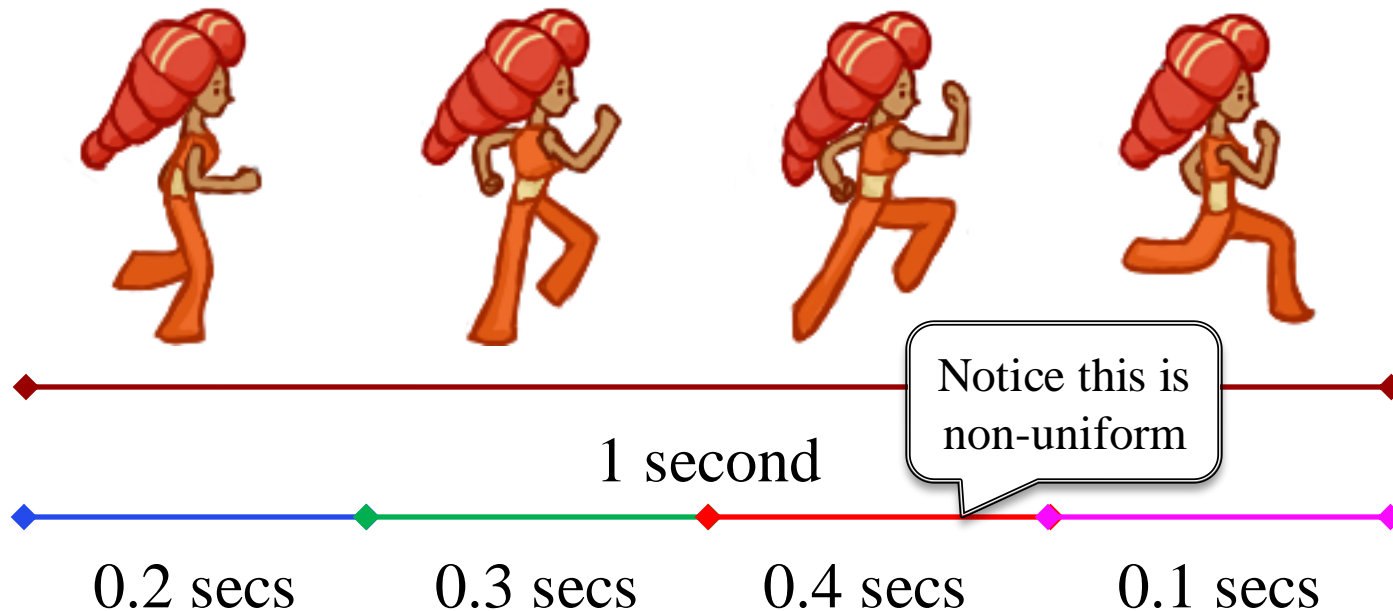
2D Animation

KeyFrames in CUGL

ActionFunction

- Represents an animation
  - User-defined function
  - No return; just animates

- Function takes
  - A value [0,1]
  - A state (begin/update/end)

- CUGL has several **factories**
  - Create common animations
  - Ideal with scene graphs

ActionTimeline

- Manages active animations
  - Assigns them a duration
  - Tracks their current state

- Has a separate update loop
  - Initialization step at start
  - Update step to increment

- Similar to **asset manager**
  - Animations have key id
  - Run `update()` to fit budget

2D Animation

the gamedesigninitiative
at cornell university

KeyFrames in CUGL

ActionTimeline

Initialize

Update

Access

- Manages active animations
  - Assigns them a duration
  - Tracks their current state

- Has a separate update loop
  - Initialization step at start
  - Update step to increment

- Similar to **asset manager**
  - Animations have key id
  - Run `update()` to fit budget

the
game**design**initiative
at cornell university

Executing Actions: **Sprite Sheets**

```
auto mgr =
ActionTimeline::alloc();

std::vector<int> frames;
frames.push_back(f1);
 …
frames.push_back(f8);

auto seq =
AnimateSprite::alloc(frames)
;

auto action = seq-
>attach(sprite);

mgr-
>activate(key,action,duratio
n);
while (mgr->isActive(key)) {
    mgr->update(TIMESTEP);
```

sprite

2D Animation

Executing Actions: **Sprite Sheets**

sprite

```
auto mgr =
ActionTimeline::alloc();

std::vector<int> frames;
frames.push_back(f1);
 …
frames.push_back(f8);

auto seq =
AnimateSprite::alloc(frames)
;

auto action = seq-
>attach(sprite);

mgr-
>activate(key,action,duratio
n);
while (mgr->isActive(key)) {
    mgr->update(TIMESTEP);
```

Allocate timeline

Progress the loop

the gamedesigninitiative
at cornell university

Executing Actions: **Sprite Sheets**

```
auto mgr =
ActionTimeline::alloc();

std::vector<int> f
frames.push_back(f
  …
frames.push_back(f8);

auto seq =
AnimateSprite::alloc(fr
;

auto action = seq-
>attach(sprite);

mgr-
>activate(key,action,duratio
n);
while (mgr->isActive(key)) {
    mgr->update(TIMESTEP);
```

Sequence the frames

Create a **factory**

sprite

Executing Actions: **Sprite Sheets**

```
auto mgr =
ActionTimeline::alloc();

std::vector<int> frames;
frames.push_back(f1);
 …
frames.push_back(f8);

auto seq =
AnimateSprite::alloc(fra
;

auto action = seq-
>attach(sprite);

mgr-
>activate(key,action,duratio
n);
while (mgr->isActive(key))
    mgr->update(TIMESTEP);
```

sprite

Create an **action**

Run for a given duration

2D Animation

Executing Actions: **Sprite Sheets**

```
auto mgr =
ActionTimeline::alloc();

std::vector<int> frames;
frames.push_back(f1);
 …
frames.push_back(f8);

auto seq =
AnimateSprite::alloc(fra
;

auto action = seq-
>attach(sprite);

mgr-
>activate(key,action,duratio
n);
while (mgr->isActive(key)) {
    mgr->update(TIMESTEP);
```

sprite

Can give
non-uniform
weights

Change
is abrupt

What About the Abrupt Change?

- Not an issue if animation is fast
  - Frames change too fast to notice
  - Movies are fine with 24 fps

- But what if animation is slower?
  - Could make more frames
  - But much more work on artist

- Want a way to make new frames
  - Ideally done by the computer
  - Called **tweening** (in-betweening)

- **Aside**: generative AI?

2D Animation

the
gamedesigninitiative
at cornell university

What About the Abrupt Change?

- Not an issue if animation is fast
  - Frames change too fast to notice
  - Movies are fine with 24 fps

- But what if animation is slower?
  - Could make more frames
  - But much more work on artist

- Want a way to make new frames
  - Ideally done by the computer
  - Called **tweening** (in-betweening)

- **Aside**: generative AI?

the gamedesigninitiative
at cornell university

Simple Approach: Crossfade Blending

A **+** **B** **=**

t = 0.0

- Linear interpolation on colors

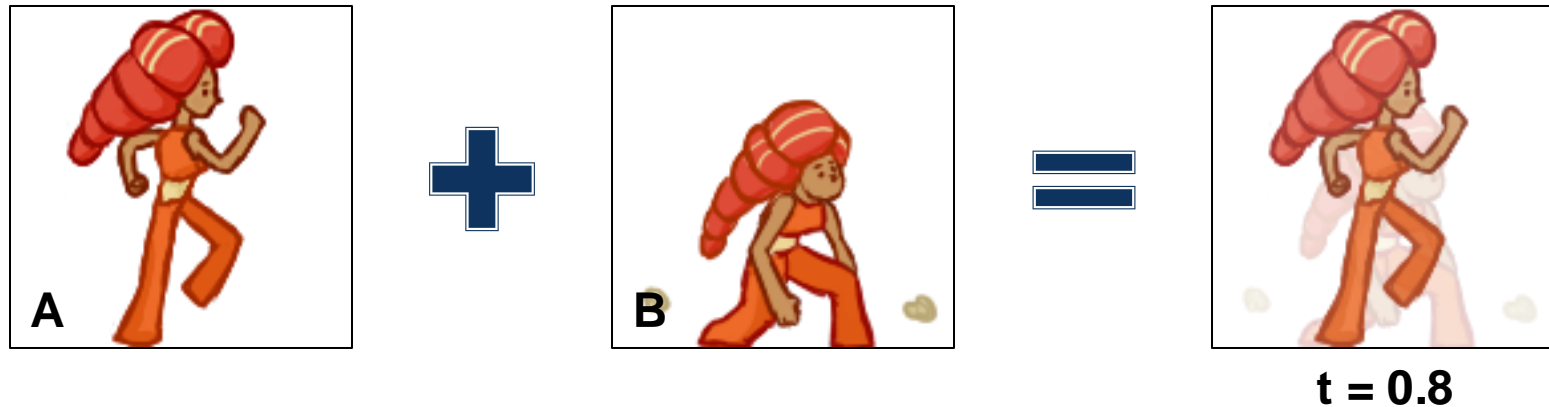$$r_c = tr_a + (1 - t)r_b$$
$$g_c = tg_a + (1 - t)g_b$$
$$b_c = tb_a + (1 - t)b_b$$

Note weights sum to 1.0

the gamedesigninitiative
at cornell university

Simple Approach: Crossfade Blending

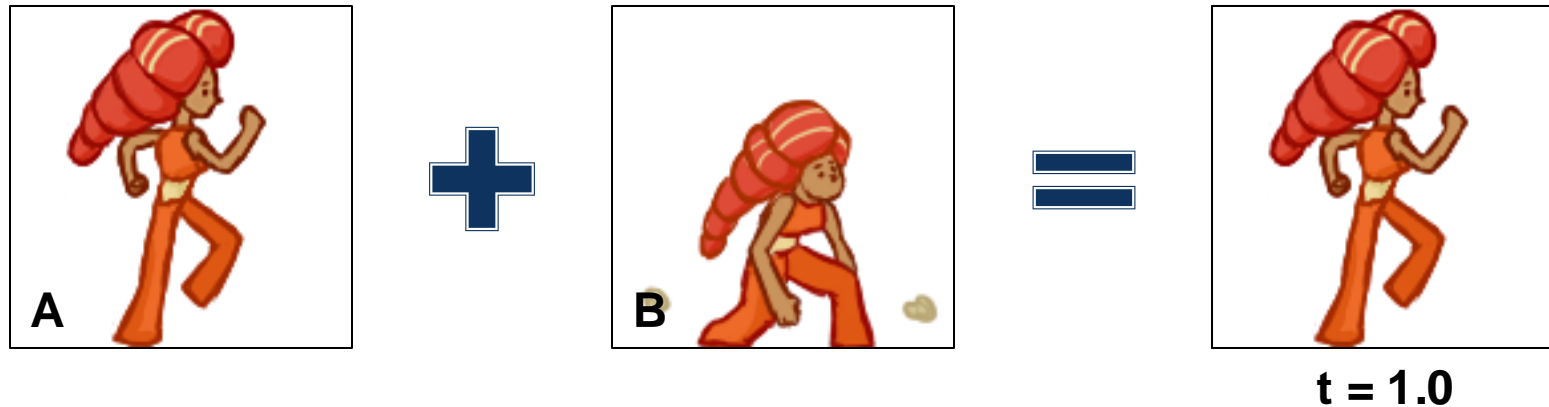**A** + **B** = , **t = 0.3**

- Linear interpolation on colors

$$r_c = tr_a + (1-t)r_b$$
$$g_c = tg_a + (1-t)g_b$$
$$b_c = tb_a + (1-t)b_b$$

Note weights sum to 1.0

2D Animation

Simple Approach: Crossfade Blending

t = 0.6

- Linear interpolation on colors

$$r_c = tr_a + (1-t)r_b$$
$$g_c = tg_a + (1-t)g_b$$
$$b_c = tb_a + (1-t)b_b$$

Note weights sum to 1.0

2D Animation

Simple Approach: Crossfade Blending

A + **B** = **t = 0.8**

- Linear interpolation on colors

$$r_c = tr_a + (1 - t)r_b$$
$$g_c = tg_a + (1 - t)g_b$$
$$b_c = tb_a + (1 - t)b_b$$

Note weights sum to 1.0

2D Animation

Simple Approach: Crossfade Blending

A **+** **B** **=**

t = 1.0

- Linear interpolation on colors

$$r_c = tr_a + (1-t)r_b$$
$$g_c = tg_a + (1-t)g_b$$
$$b_c = tb_a + (1-t)b_b$$

Note weights sum to 1.0

Tweening: Interpolating In-Betweens
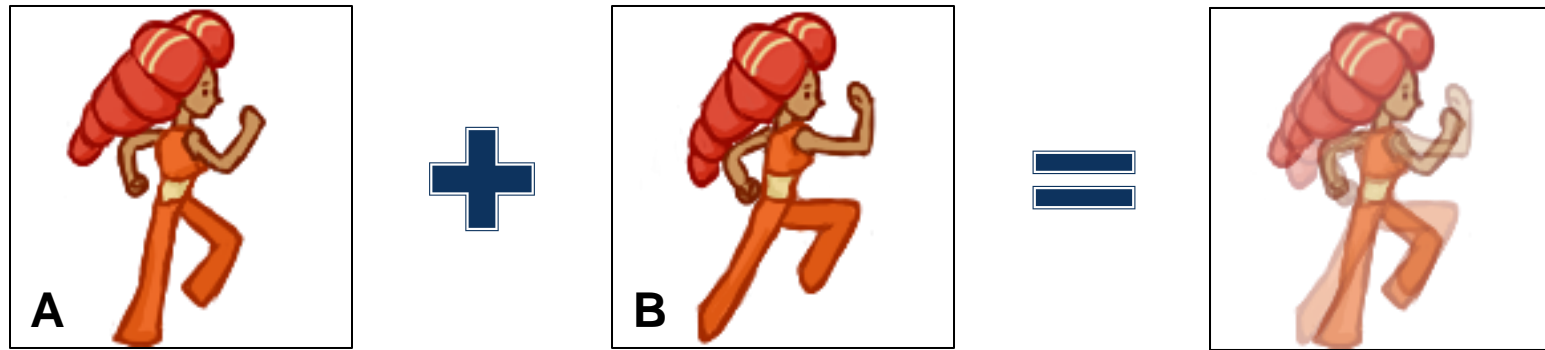
- Act of linear interpolating between animation frames
  - Because we cycle filmstrip slower than framerate
  - Implements a form of motion blur

- If animation **designed right**, makes it smoother

the gamedesigninitiative
at cornell university

Tweening and Transforms
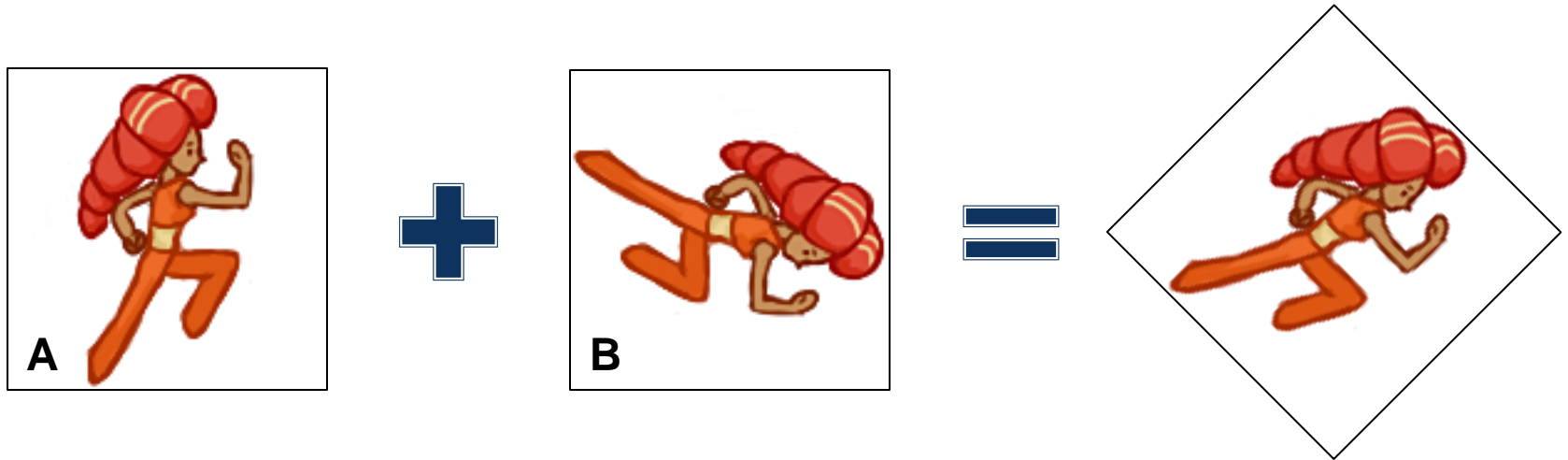
- Any transform is represented by a matrix
  - Can linearly interpolate matrix components
  - Gives a reasonable transform "in-between"

- **Aside**: This is a motivation for quaternions
  - Gives smoother interpolation for rotation

2D Animation

the gamedesigninitiative
at cornell university

Application to Sprite Animation

- Movement is *two* things
  - **Animation** of the filmstrip
  - **Translation** of the image
  - These two must align

- **Example:** Walking
  - Foot is point of contact
  - "Stays in place" as move
  - This constrains translation

- Make movement regular
  - Measure distance per frame
  - Keep same across frames

Application to Sprite Animation

- Movement is *two* things
  - **Animation** of the filmstrip
  - **Translation** of the image
  - These two must align

- **Example:** Walking
  - Foot is point of contact
  - "Stays in place" as move
  - This constrains translation

- Make movement regular
  - Measure distance per frame
  - Keep same across frames

Application to Sprite Animation

- Movement is *two* things
  - **Animation** of the filmstrip
  - **Translation** of the image
  - These two must align

- **Example:** Walking
  - Foot is point of contact
  - "Stays in place" as move
  - This constrains translation

- Make movement regular
  - Measure distance per frame
  - Keep same across frames

Point of contact

Distance forward

Executing Actions: **Movement**

```
// Create the factories
auto seq1 =
AnimateSprite::alloc(frames
);
auto seq2 =
MoveBy::alloc(Vec2(PACE,0))
;

// Create the actions
auto flip = seq1-
>attach(sprite);
auto shift = seq2-
>attach(sprite);

// Activate and animate
mgr->activate(key1, flip,
duration);
```

Executing Actions: **Movement**

```
// Create the factories
auto seq1 =
AnimateSprite::alloc(frames
);
auto seq2 =
MoveBy::all        CE,0))
;

// Create the actions
auto flip = seq1-
>attach(sprite);
auto shift = seq2-
>attach(sprite);

// Activate and animate
mgr->activate(key1, flip,
duration);
```

Currently only supported on scene graphs

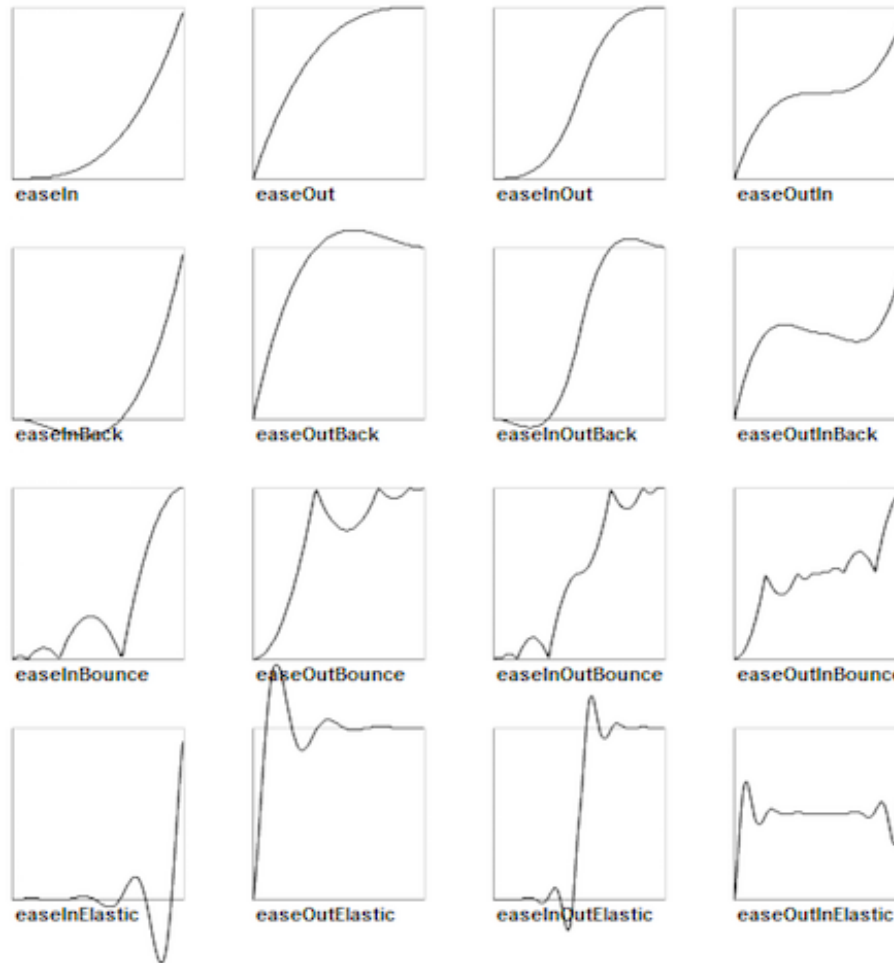Ensures animations in sync

Tweeing and Easing Functions

- Basic approach to tweening
    - Specify duration to animate
    - Set $t = 0$ at beginning
    - Normalize $t = 1$ at end
    - Interpolate value with t

- How does t change?
    - Usually done *linearly*
    - Could be some other way

- **Easing**: how to change t
    - Used for bouncing effects
    - Best used for *transforms*

2D Animation

Tweeing and Easing Functions

- Basic approach to tweening
  - Specify duration to animate
  - Set $t = 0$ at beginning
  - Normalize $t = 1$ at end
  - Interpolate value with t

- How does t change?
  - Usually done *linearly*
  - Could be some other way

- **Easing**: how to change t
  - Used for bouncing effects
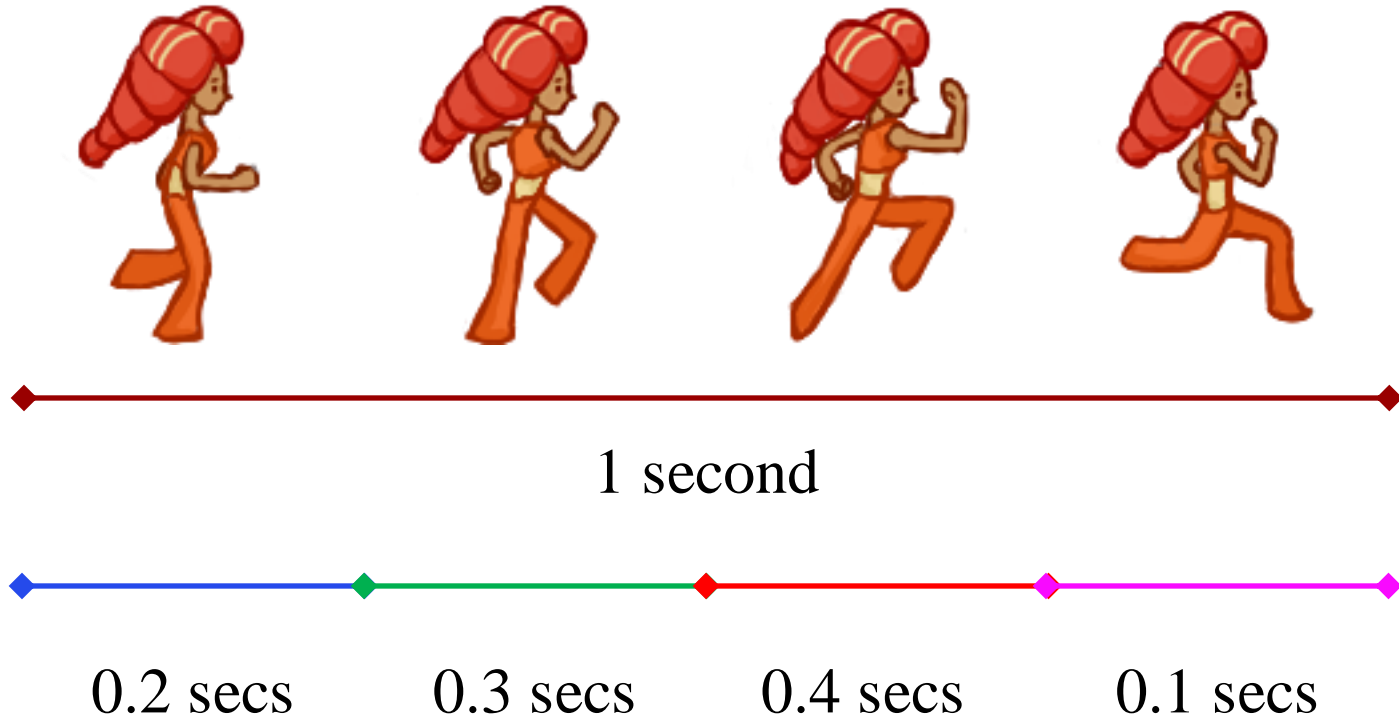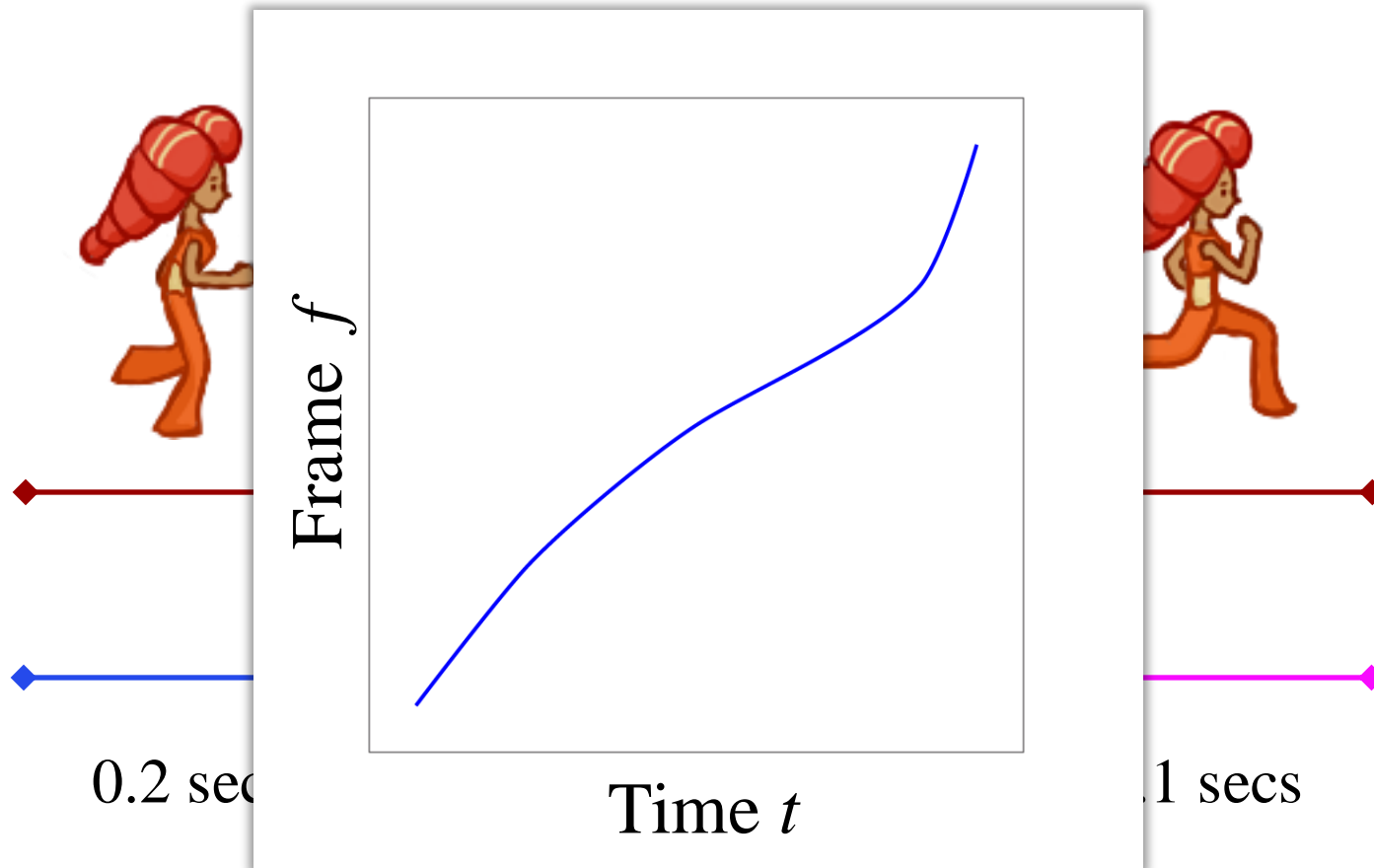 - Best used for *transforms*

2D Animation

the gamedesigninitiative
at cornell university

Classic Easing Functions

2D Animation

Classic Easing Functions

http://easings.net

2D Animation

Application to Sprite Animation

1 second

0.2 secs   0.3 secs   0.4 secs   0.1 secs
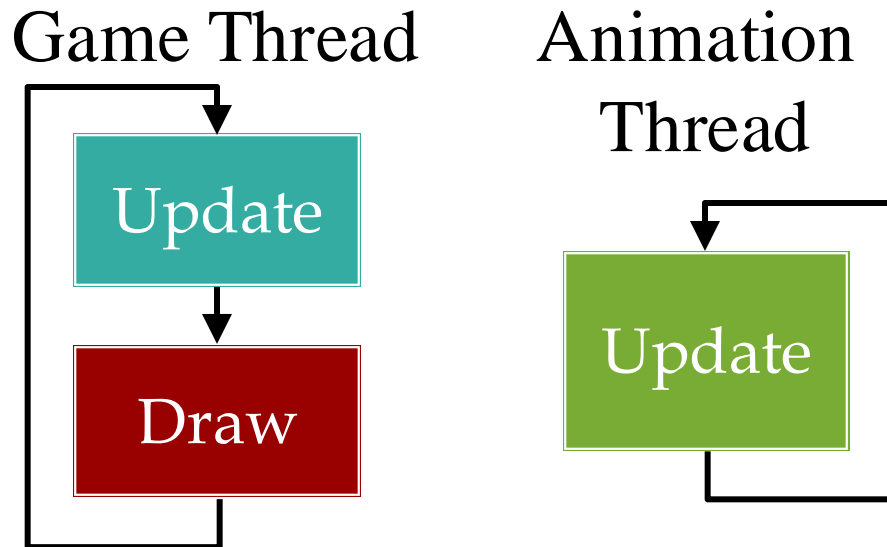
Application to Sprite Animation

Frame f

Time t

0.2 sec[s]

[0].1 secs

Aside: Animating vs Rendering

- You do not have to animate in the main thread
  - Main thread is for rendering (drawing on screen)
  - But animation is simply "posing" your models

- Allows for smoother animation (VSync problem)

Game Thread

Animation Thread

Update

Draw

Update

the gamedesigninitiative
at cornell university

Example: *Hi Fi Rush*

2D Animation

Modular Animation

- Break asset into parts
  - Natural for joints/bodies
  - Animate each separately

- Cuts down on filmstrips
  - Most steps are transforms
  - Very natural for tweening
  - Also better for physics

- Several tools to help you
  - **E.g.** *After Effects, Spine*
  - Great for visualizing design

2D Animation

the gamedesigninitiative
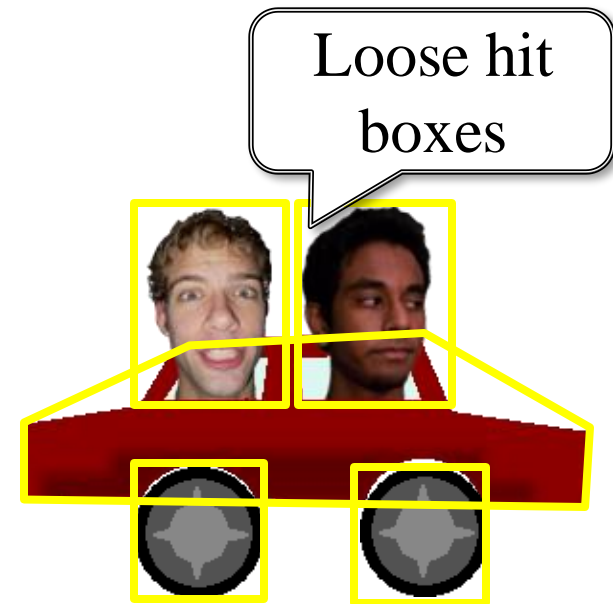at cornell university

Modular Animation

- Break asset into parts
  - Natural for joints/bodies
  - Animate each separately

- Cuts down on filmstrips
  - Most steps are transforms
  - Very natural for tweening
  - Also better for physics

- Several tools to help you
  - **E.g.** *After Effects, Spine*
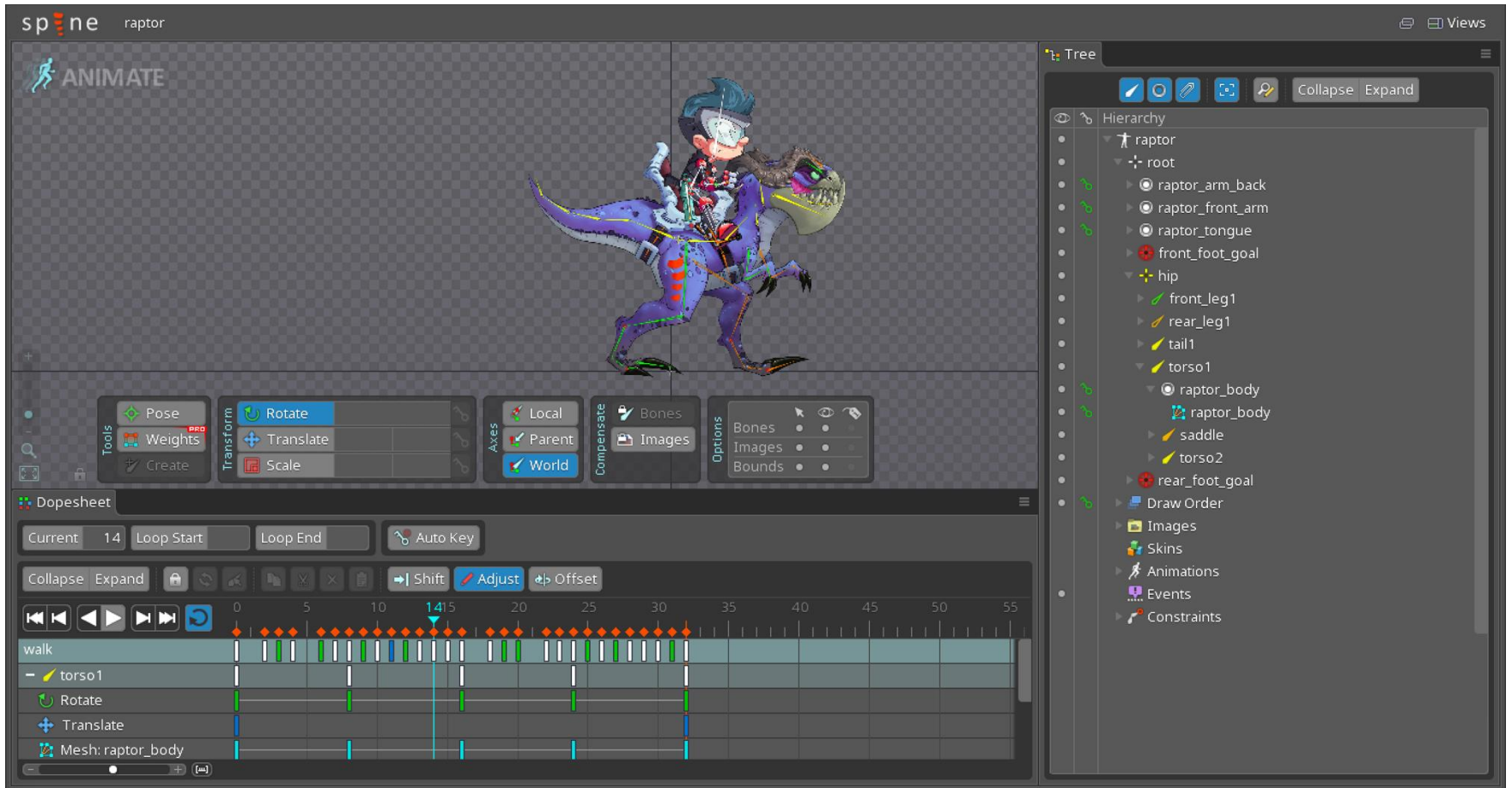 - Great for visualizing design

2D Animation

Modular Animation

- Break asset into parts
  - Natural for joints/bodies
  - Animate each separately

- Cuts down on filmstrips
  - Most steps are transforms
  - Very natural for tweening
  - Also better for physics

- Several tools to help you
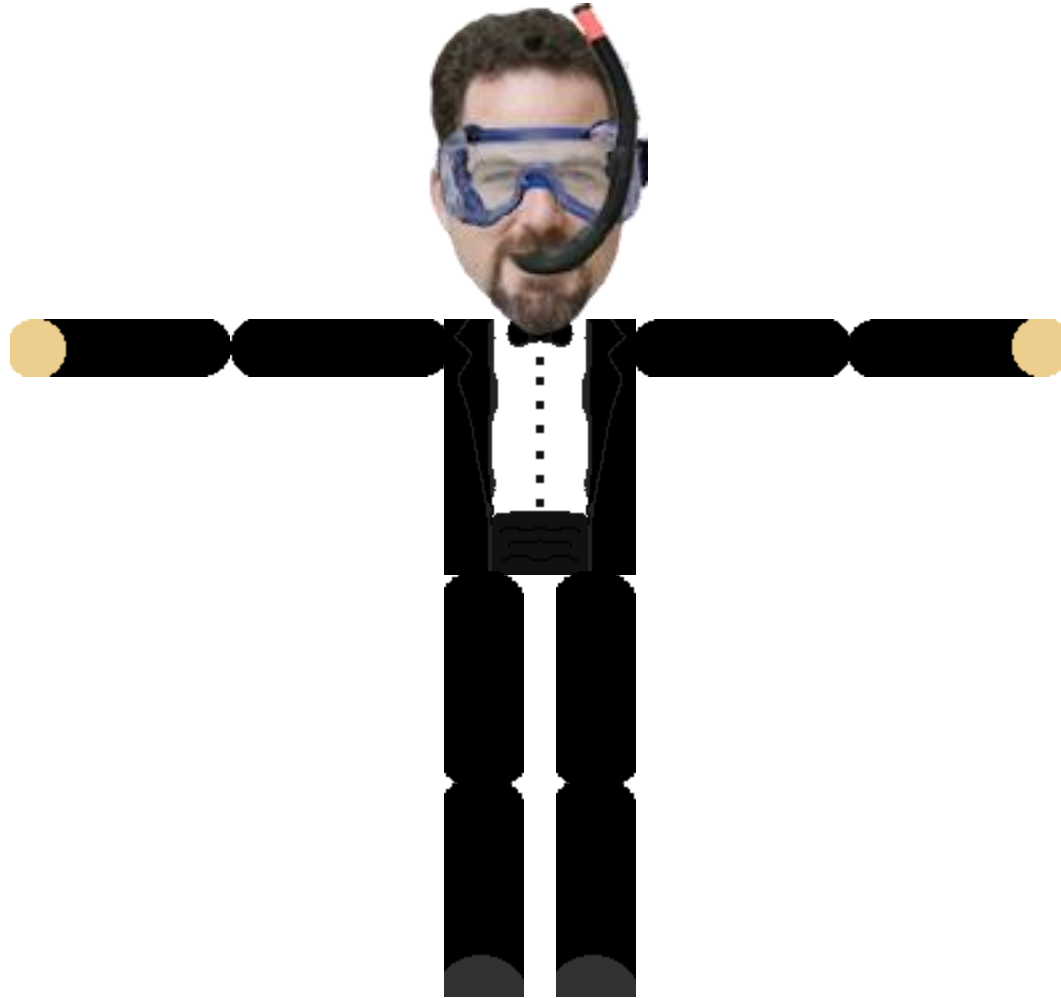 - **E.g.** *After Effects, Spine*
  - Great for visualizing design

2D Animation

Modular Animation

- Break asset into parts
  - Natural for joints/bodies
  - Animate each separately

- Cuts down on filmstrips
  - Most steps are transforms
  - Very natural for tweening
  - Also better for physics

- Several tools to help you
  - **E.g.** *After Effects, Spine*
  - Great for visualizing design

Loose hit boxes

- Inside hit box can safely
  - Transform with duration
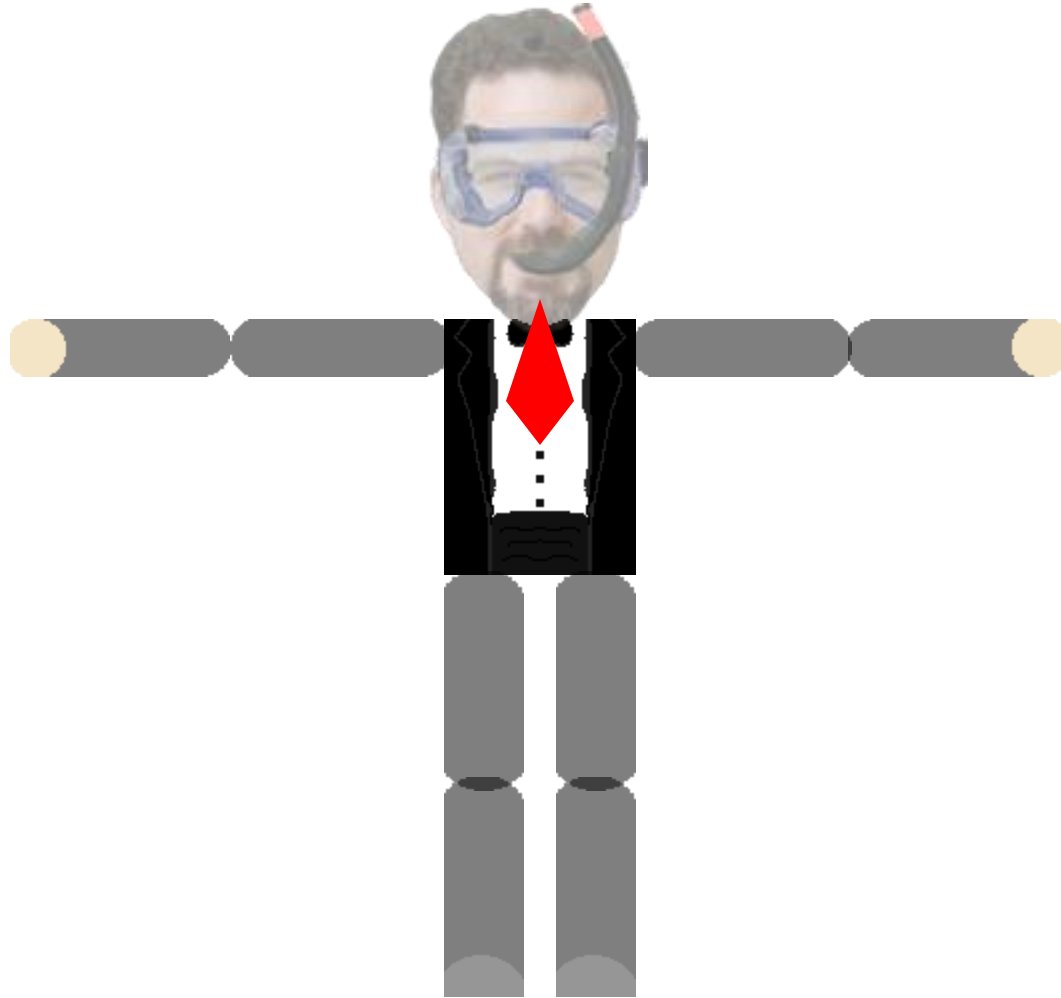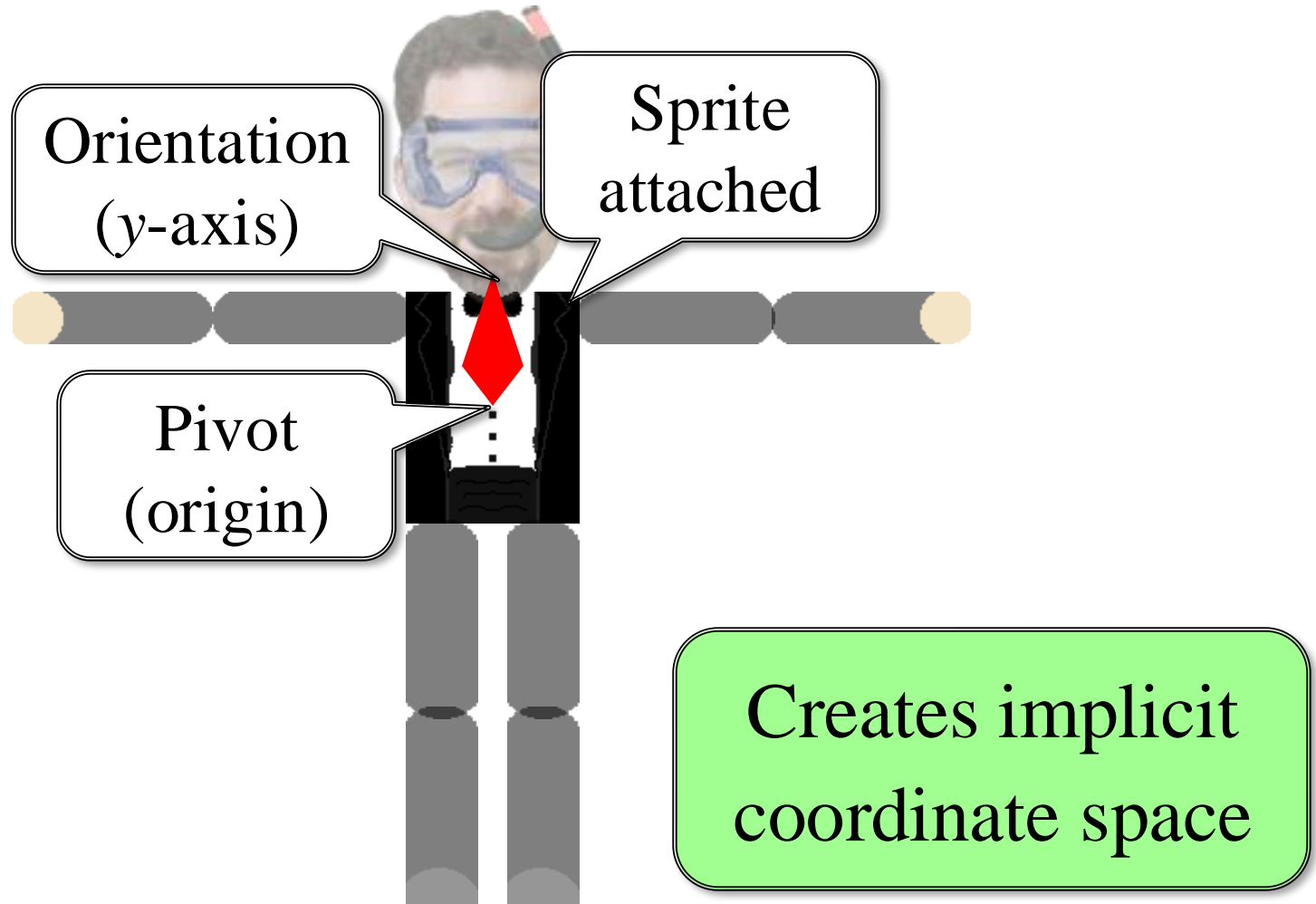 - Tween animations
  - Manage multiple actions

the gamedesigninitiative
at cornell university

Spine Demo

2D Animation

Basic Idea: **Bones**

2D Animation

Basic Idea: **Bones**

2D Animation

Basic Idea: **Bones**

Orientation (*y*-axis)

Sprite attached

Pivot (origin)

Creates implicit coordinate space

Bones are Heirarchical

Child

Parent

2D Animation

Bones are Heirarchical

Transforms apply to children

Bones are Heirarchical

Transforms
do not affect
the parent

2D Animation
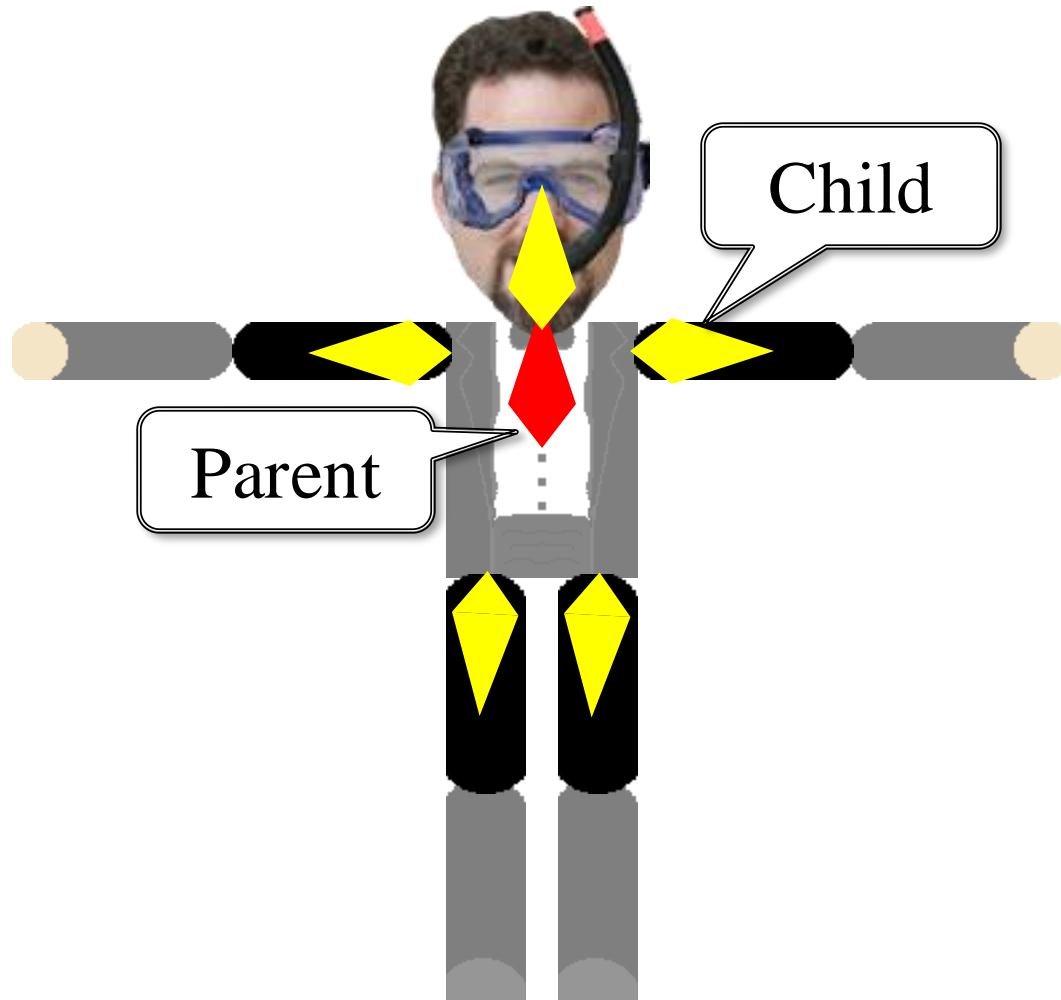
Recall: Scene Graph Hierarchy

Layer — Device/Screen Coordinates

Bounded box inside

Node

Node

Node

Coords relative to parent box

Node Node Node Node Node Node

2D Animation

the gamedesigninitiative
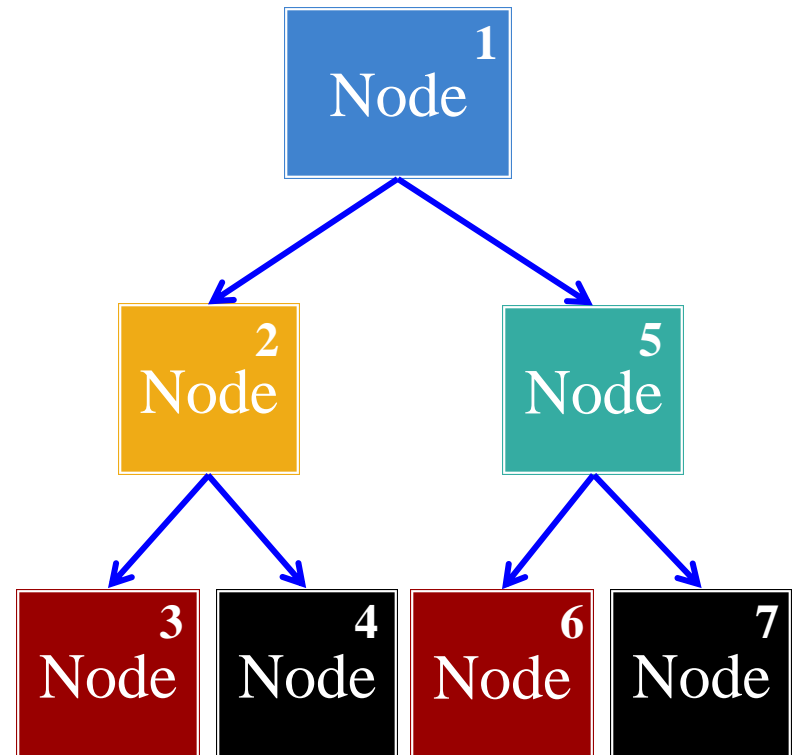at cornell university
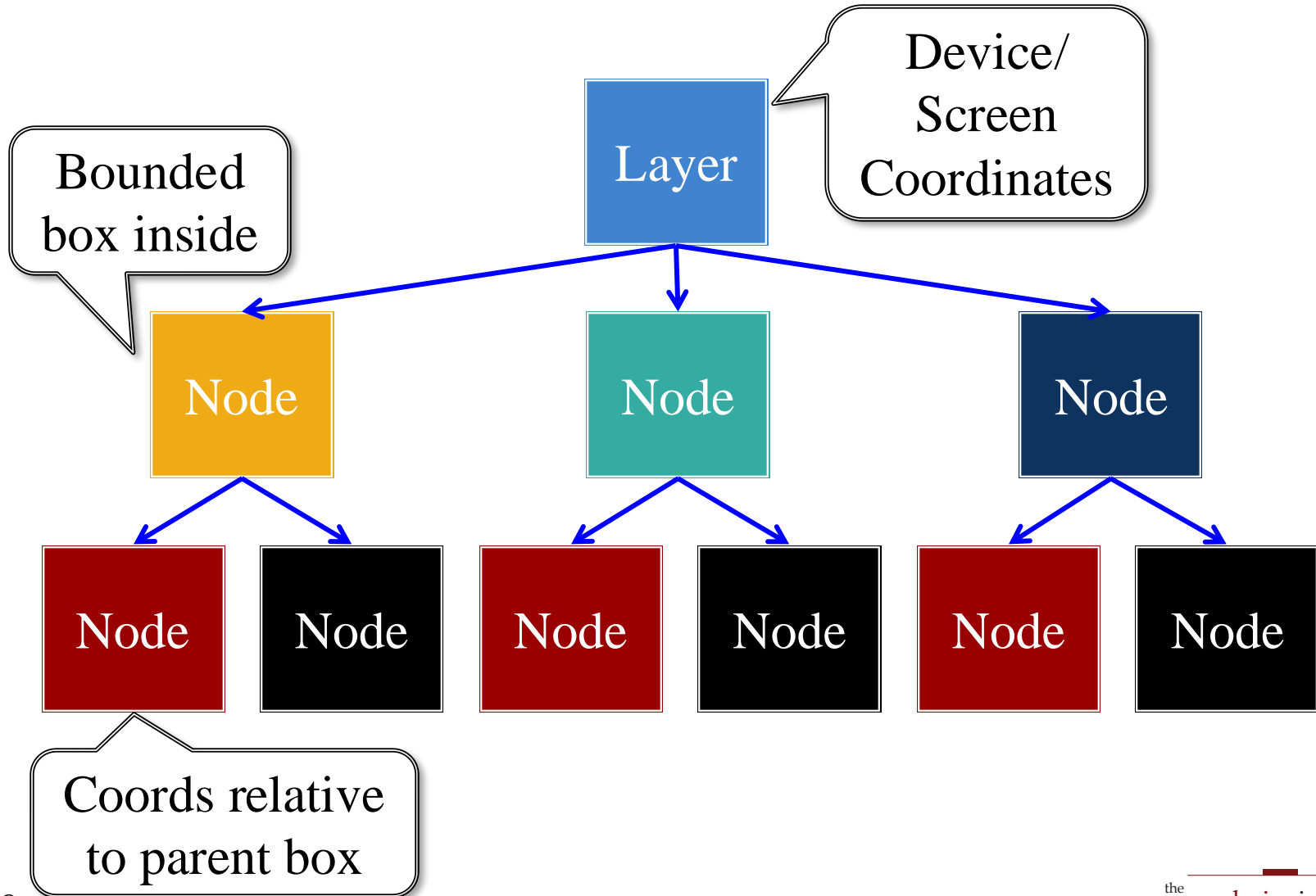
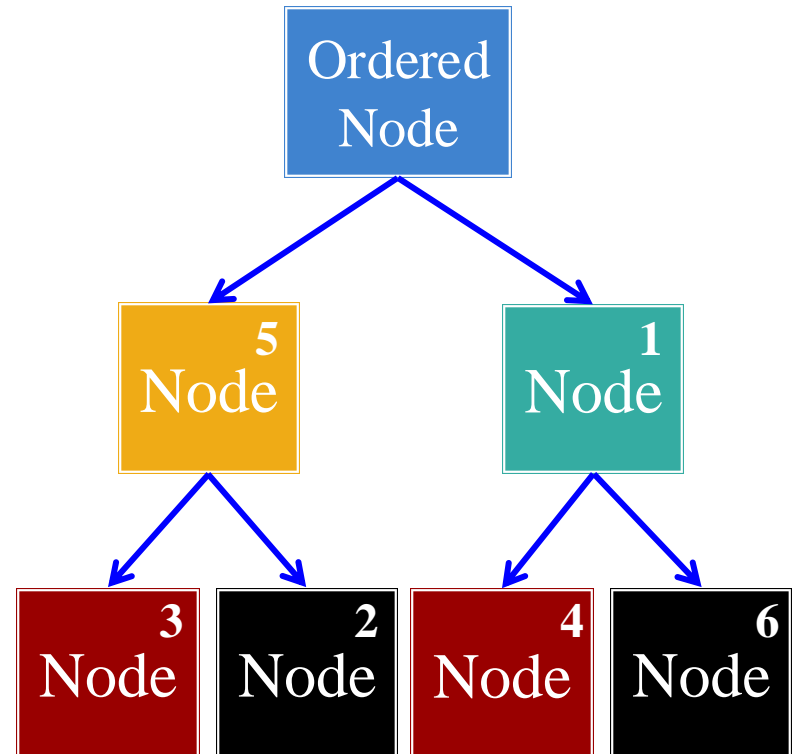# Bones are a Scene Graph Visualization

2D Animation

Problem: Scene Graphs are Preorder

- **Parents are drawn first**
  - Children are drawn in front
  - Ideal for UI elements
  - Bad for modular animation

- Solution: `OrderedNode`
  - Puts descendents into a list
  - Sorts based on priority value
  - Draws nodes in that order

- Acts as a render barrier
  - What if nested `OrderedNode`?
  - Each `OrderedNode` is a unit
  - Priorities do not mix

2D Animation

Problem: Scene Graphs are Preorder

- Parents are drawn first
  - Children are drawn in front
  - Ideal for UI elements
  - Bad for modular animation

- **Solution:** `OrderedNode`
  - Puts descendents into a list
  - Sorts based on priority value
  - Draws nodes in that order

- Acts as a render barrier
  - What if nested `OrderedNode`?
  - Each `OrderedNode` is a unit
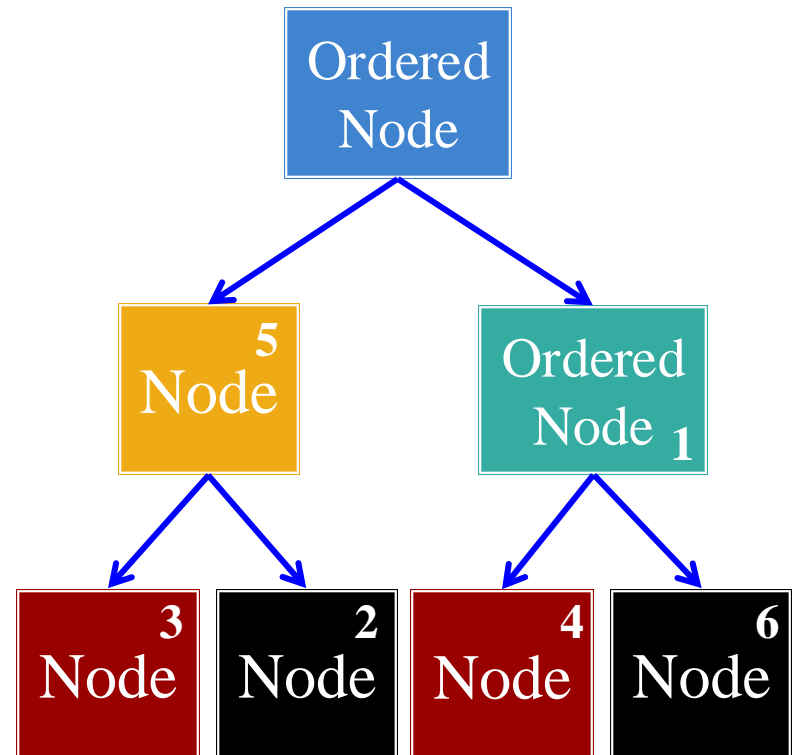 - Priorities do not mix

2D Animation

Problem: Scene Graphs are Preorder

- Parents are drawn first
  - Children are drawn in front
  - Ideal for UI elements
  - Bad for modular animation

- Solution: `OrderedNode`
  - Puts descendents into a list
  - Sorts based on priority value
  - Draws nodes in that order

- Acts as a **render barrier**
  - What if nested `OrderedNode`?
  - Each `OrderedNode` is a unit
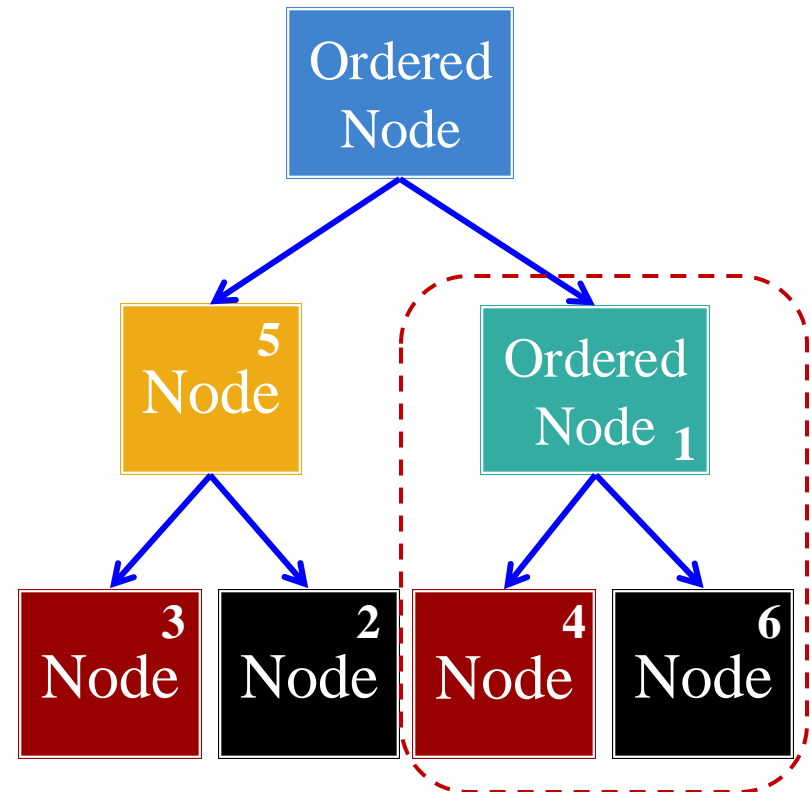 - Priorities do not mix

53

Problem: Scene Graphs are Preorder

- Parents are drawn first
  - Children are drawn in front
  - Ideal for UI elements
  - Bad for modular animation

- Solution: `OrderedNode`
  - Puts descendents into a list
  - Sorts based on priority value
  - Draws nodes in that order

- Acts as a **render barrier**
  - What if nested `OrderedNode`?
  - Each `OrderedNode` is a unit
  - Priorities do not mix

Transform Tweening

+

Physical Animation

=

Complete Disaster

Problems With Tweening

```
auto seq =
RotateBy::alloc(90.0f);

auto action – seq-
>attach(sprite);

mgr-
>activate(key,action,2.0f)
;
```

What if we change our mind before 2 seconds?

2D Animation

Problems With Tweening

```
auto seq =
RotateBy::alloc(90.0f);

auto action – seq-
>attach(sprite);

mgr-
>activate(key,action,2.0f)
;
```

Compatible: Combine
Incompatible: Replace

the gamedesigninitiative
at cornell university

Combining Animations
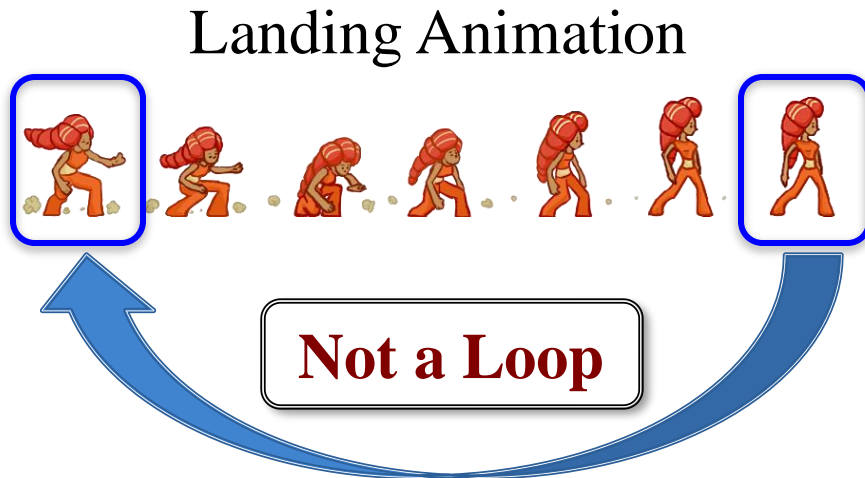
Landing Animation

Idling Animation

- Characters do a lot of things
  - Run, jump, duck, slide
  - Fire weapons, cast spells
  - Fidget while player AFK

- Want animations for all
  - Is loop appropriate for each?
  - How do we transition?

- **Idea**: shared boundaries
  - End of loop = start of another
  - Treat like advancing a frame

2D Animation

Combining Animations

Landing Animation

Not a Loop

Idling Animation

- Characters do a lot of things
  - Run, jump, duck, slide
  - Fire weapons, cast spells
  - Fidget while player AFK

- Want animations for all
  - Is loop appropriate for each?
  - How do we transition?

- **Idea**: shared boundaries
  - End of loop = start of another
  - Treat like advancing a frame

the gamedesigninitiative
at cornell university

Combining Animations

Landing Animation

Transition

Idling Animation

- Characters do a lot of things
  - Run, jump, duck, slide
  - Fire weapons, cast spells
  - Fidget while player AFK

- Want animations for all
  - Is loop appropriate for each?
  - How do we transition?

- **Idea**: shared boundaries
  - End of loop = start of another
  - Treat like advancing a frame

Combining Animations

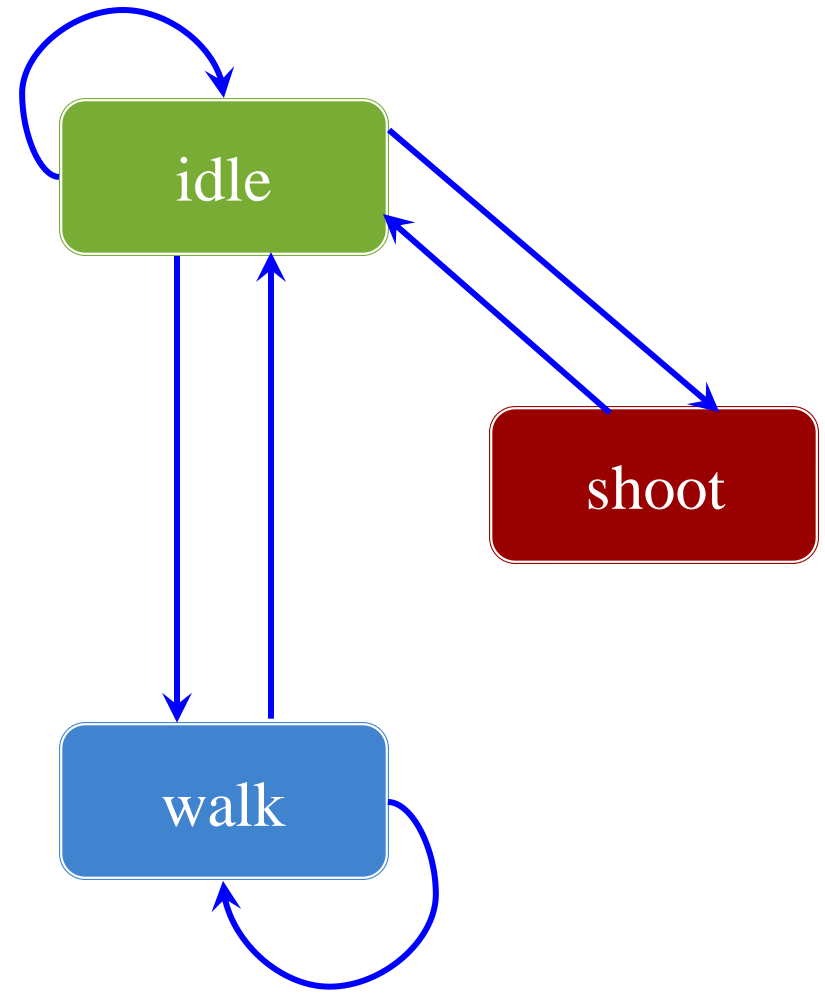Landing Animation

Transition

Idling Animation

- Characters do a lot of things
  - Run, jump, duck, slide
  - Fire weapons, cast spells
  - Fidget while player AFK

- Want animations for all
  - Is loop appropriate for each?
  - How do we transition?

- **Idea**: shared boundaries
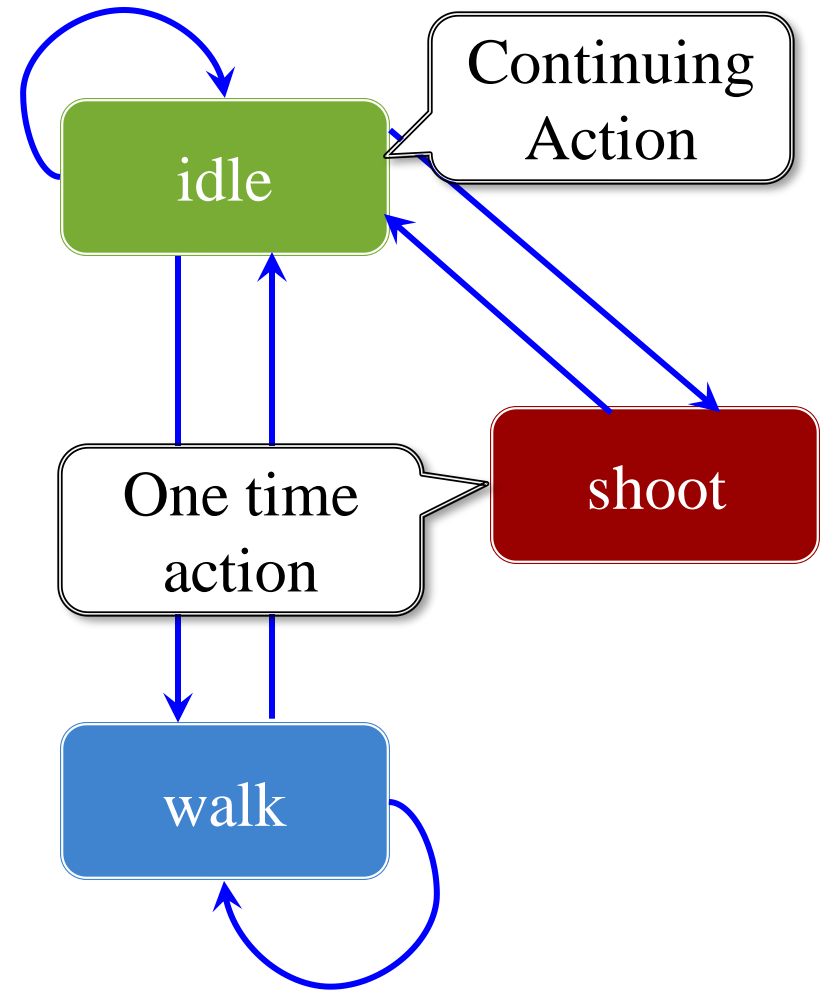
But do not draw ends twice!

Animation and State Machines

- Idea: Each sequence a state
  - Do sequence while in state
  - Transition when at end
  - Only loop if loop in graph

- A graph edge means…
  - Boundaries match up
  - Transition is allowable

- Similar to data driven AI
  - Created by the designer
  - Implemented by programmer
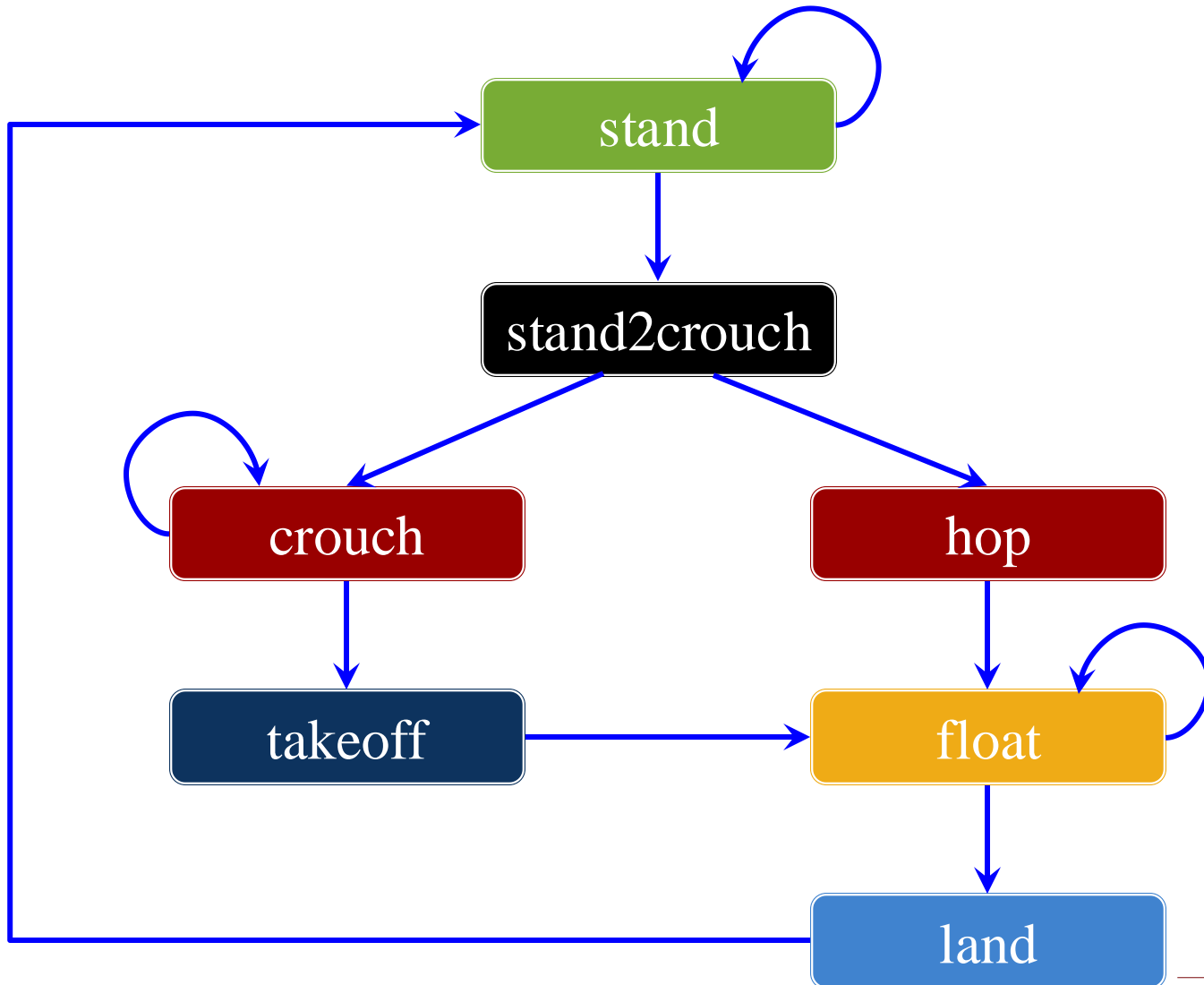 - Modern engines have tools
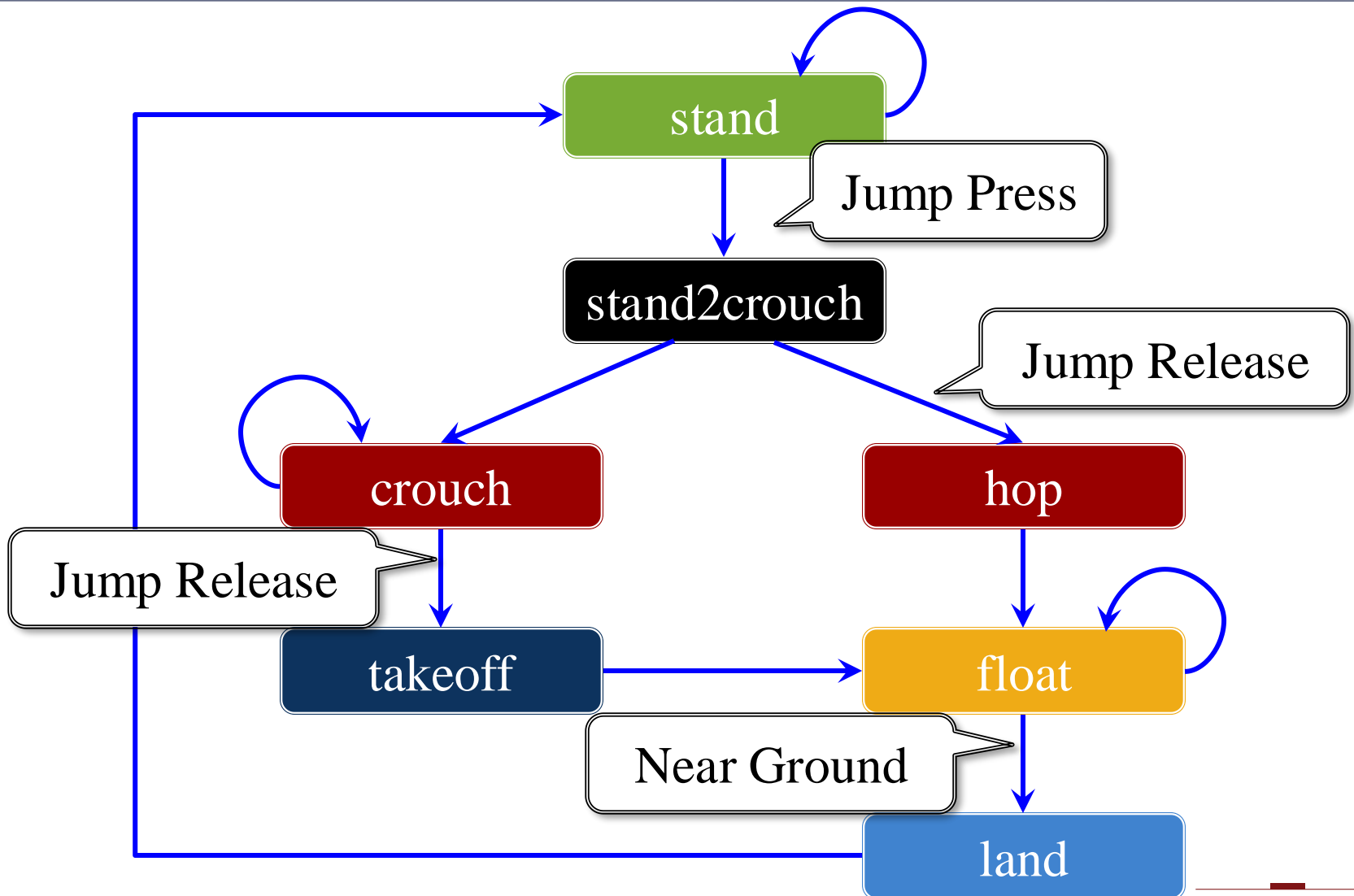
2D Animation

Animation and State Machines

- Idea: Each sequence a state
  - Do sequence while in state
  - Transition when at end
  - Only loop if loop in graph

- A graph edge means…
  - Boundaries match up
  - Transition is allowable

- Similar to data driven AI
  - Created by the designer
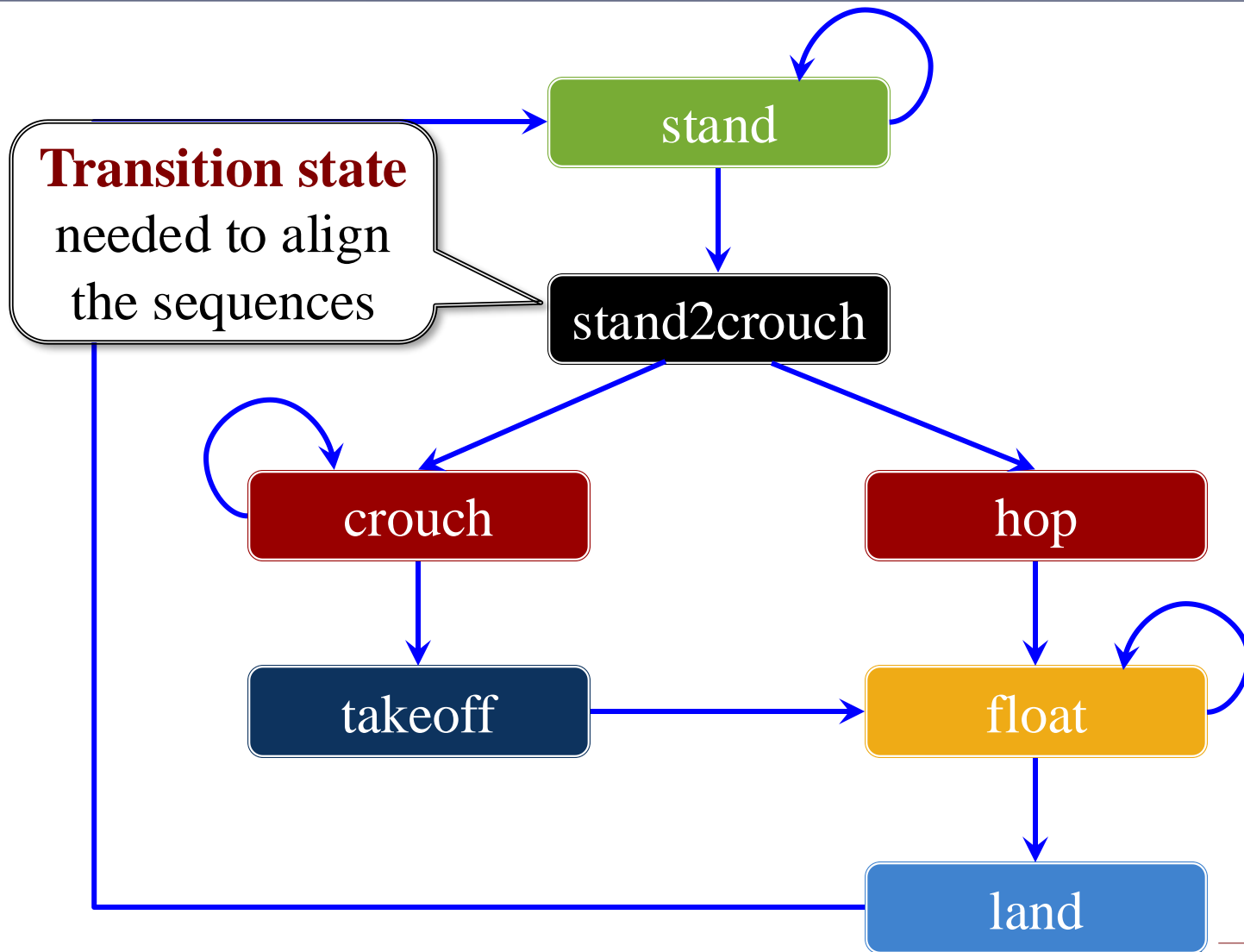 - Implemented by programmer
  - Modern engines have tools

idle

Continuing Action

shoot

One time action

walk

2D Animation

Complex Example: Jumping

2D Animation

Complex Example: Jumping

2D Animation

the gamedesigninitiative
at cornell university

Complex Example: Jumping

Transition state needed to align the sequences

stand

stand2crouch

crouch

hop

takeoff

float

land

2D Animation

The Responsiveness Issue

Tightness of the gameplay

stand

stand2crouch

Additional delay preventing jump

crouch

hop

takeoff

float

land
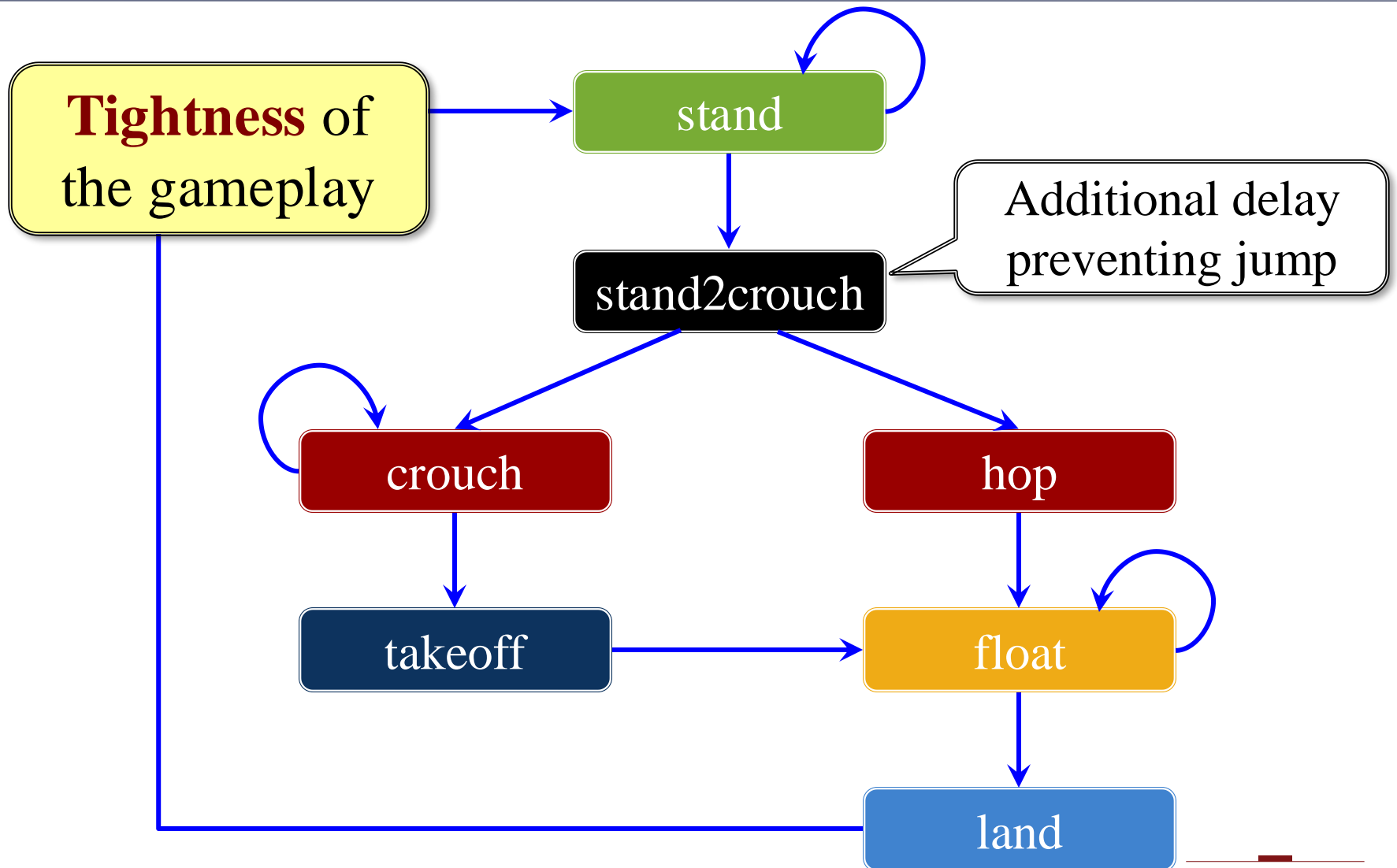
2D Animation

Summary

- Standard 2D animation is **flipbook** style
  - Create a sequence of frames in sprite sheet
  - Switch between sequences with state machines

- **Tweening** supports interpolated transitions
  - Helpful for motion blur, state transitions
  - Transforms can be combined with easing functions

- Professional 2D animation uses **modular sprites**
  - Scene graphs are a simplified form of model rigging
  - State machine coordination can be very advanced

the game**design**initiative
at cornell university