

Postmortem:

Using Mecanim in *Undertakers*

Ashley Egan
Animation Artist

Adam Ormsby
Technical Designer

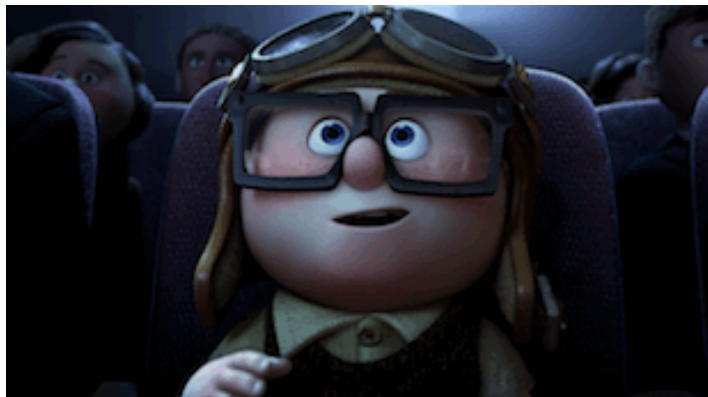
“With great power comes great responsibility...”

Or in our case, great power came with migraines and coffee runs.



In this technical post-mortem, we'll discuss:

- Using Mecanim in our senior project, Undertakers
- Key features
- Our end game result
- Problems (a lot of them)



Undertakers

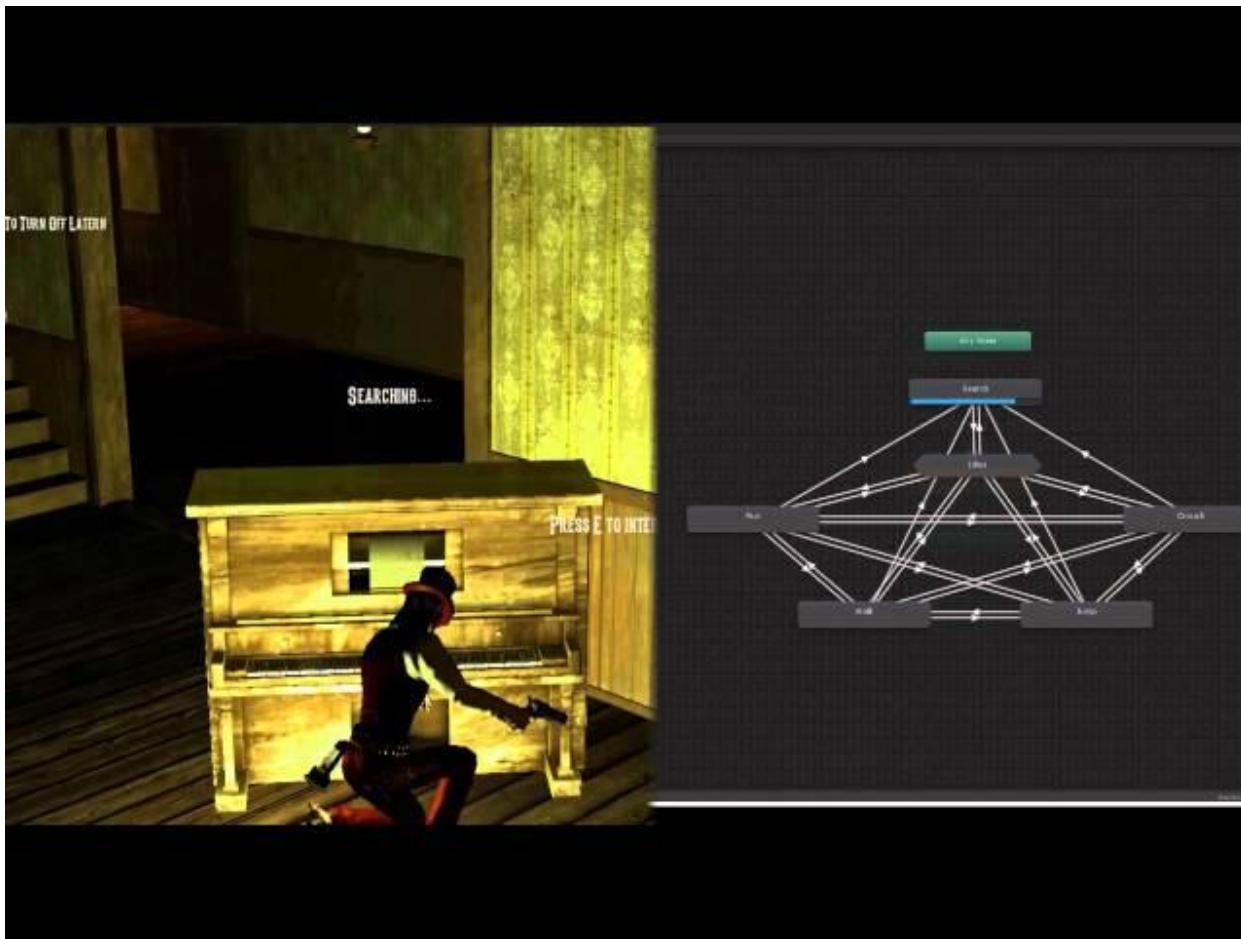
Senior Capstone Project

Two Semesters:
Pre-Production and
Production

Roles:

Adam: Designer/Scripter
Documentation Nut
Animation Scripter

Ashley: Animation Lead



Unity Upgrade



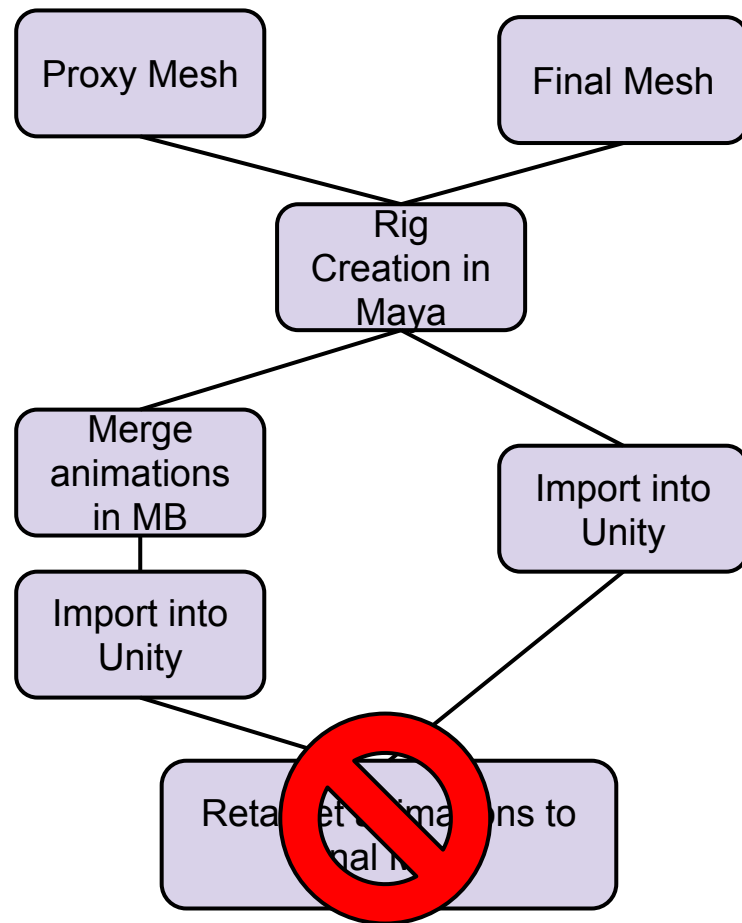
- Legacy system: Cause of many errors
- Rumor of new tool, “Mecanim”?
- No major hiccups
- Ignored professors advice

Winter Break Purgatory

- 1 TA/30 TM = Not enough TAs
- *Unity 4.0 Mecanim Tutorial*

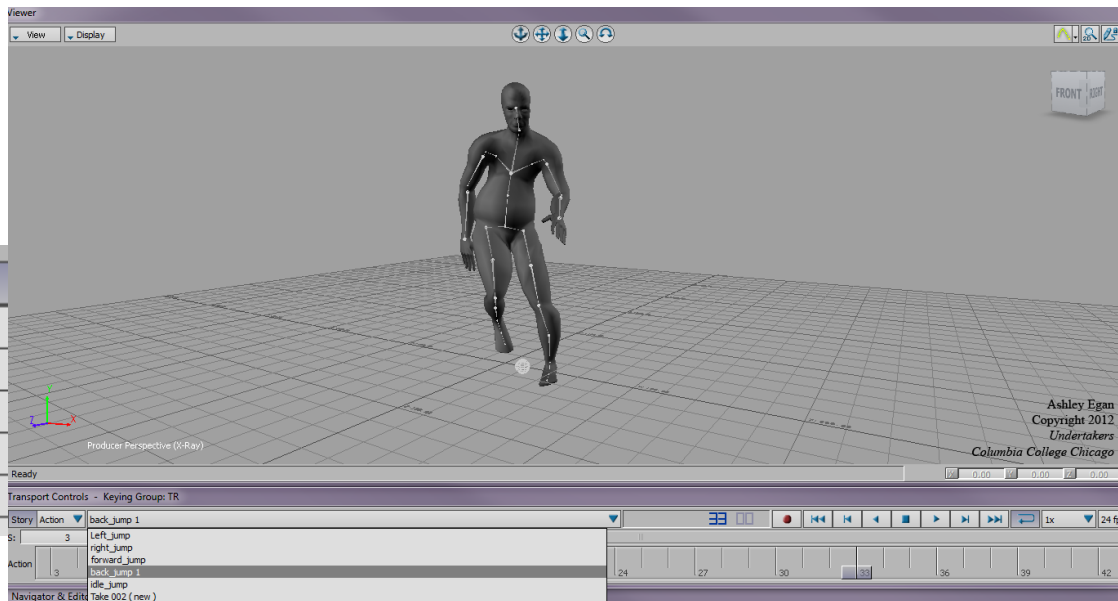


Initial pipeline setup

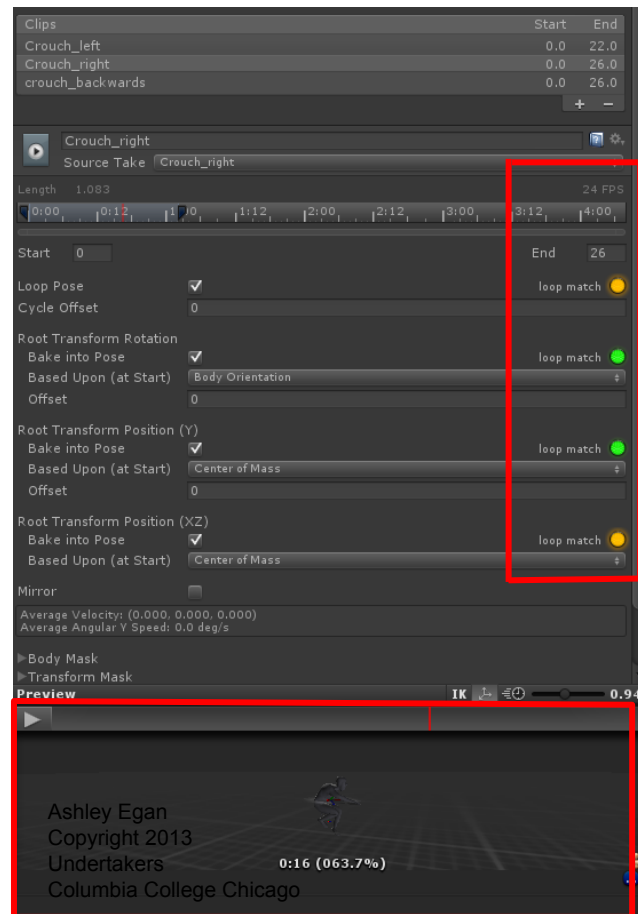
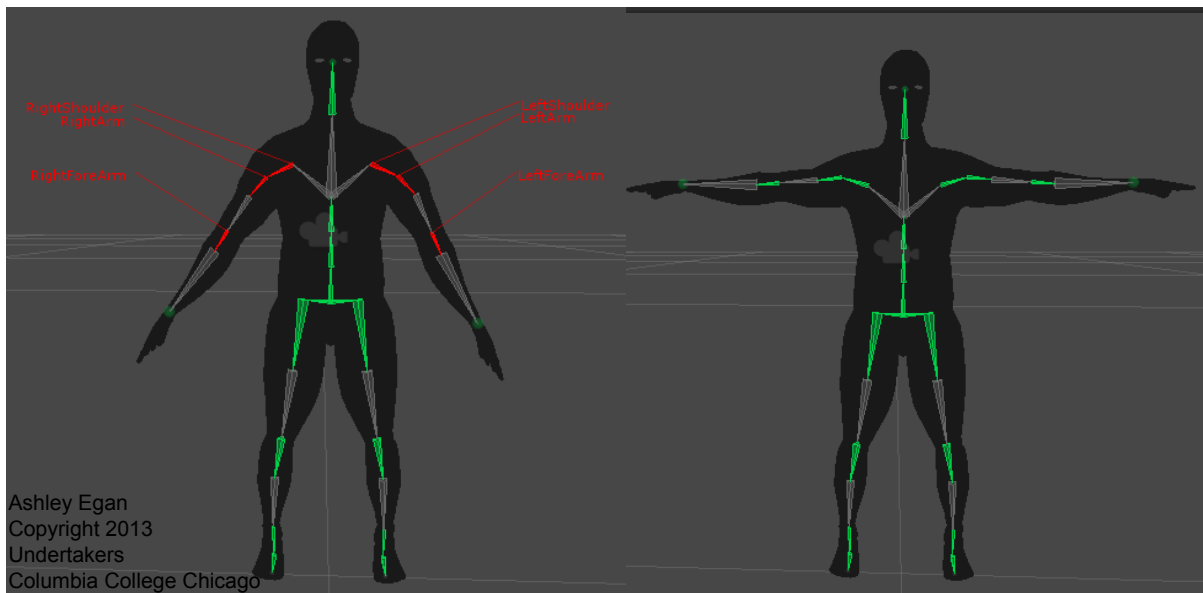


Workflow Progression

Takes	Export	As Take ...	
Left_jump	<input checked="" type="checkbox"/>	Left_jump	
right_jump	<input checked="" type="checkbox"/>	right_jump	
forward_jump	<input checked="" type="checkbox"/>	forward_jump	
back_jump 1	<input checked="" type="checkbox"/>	back_jump	
idle_jump	<input checked="" type="checkbox"/>	idle_jump	

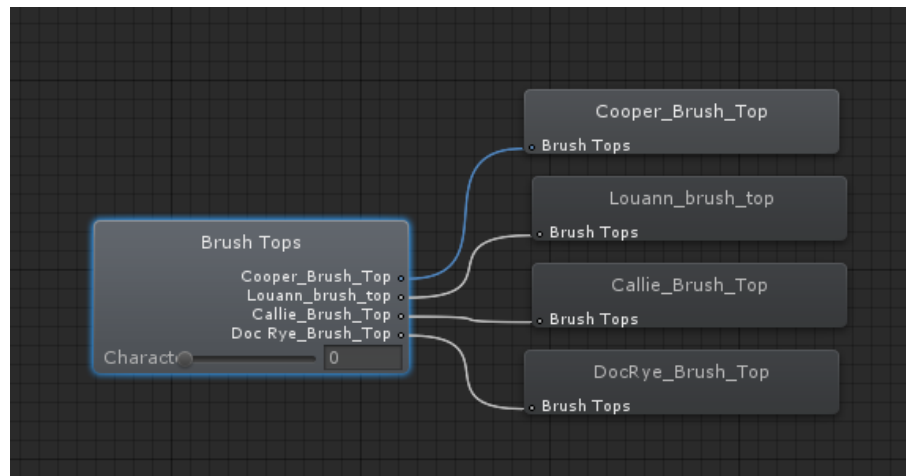


Humanoid Rig Setting



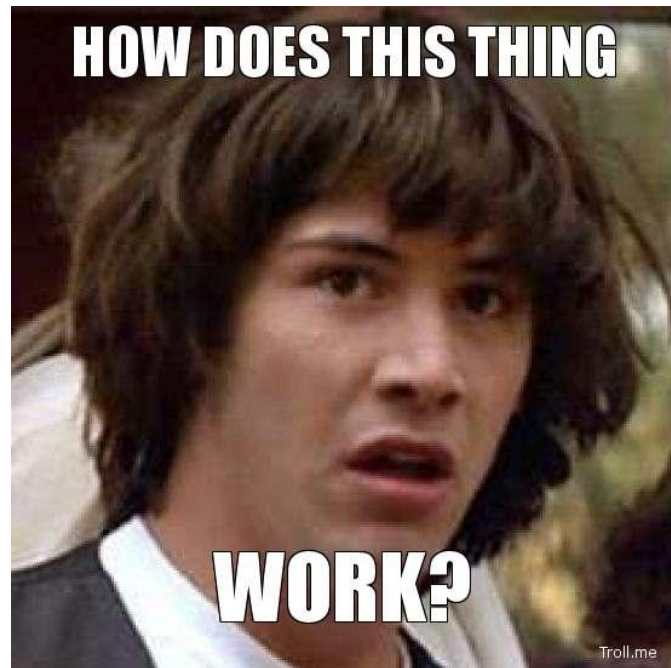
Retargeting

- 95% of animations Used globally
- 1:4
- Fallback: Generic animations on differently sized characters
- Resolved: Create separate animations for each character.



Humanoid Rig Setting: Root Motion

- Uses the character translations and moves the game object based off of it
- Contradicted the TPC script we were currently using (and modified)
- Already too far into production to make Root Motion work

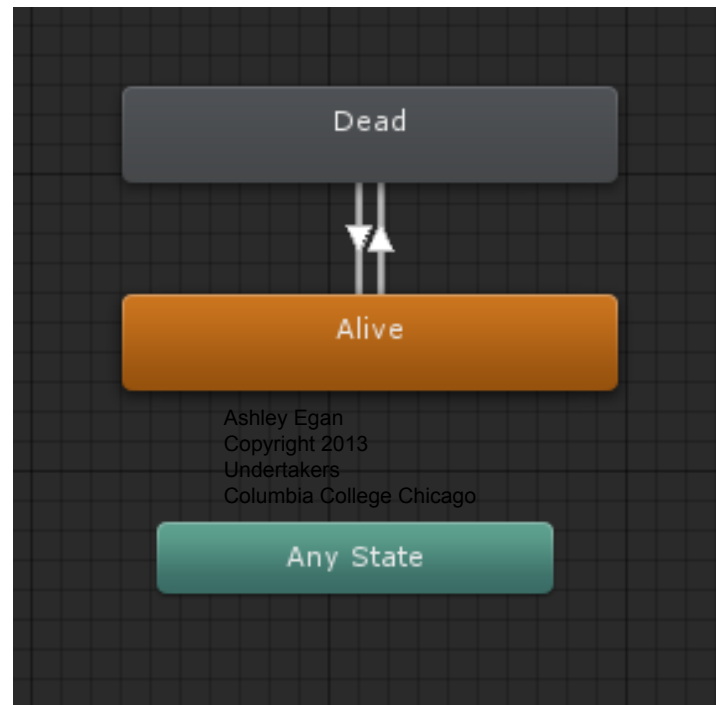
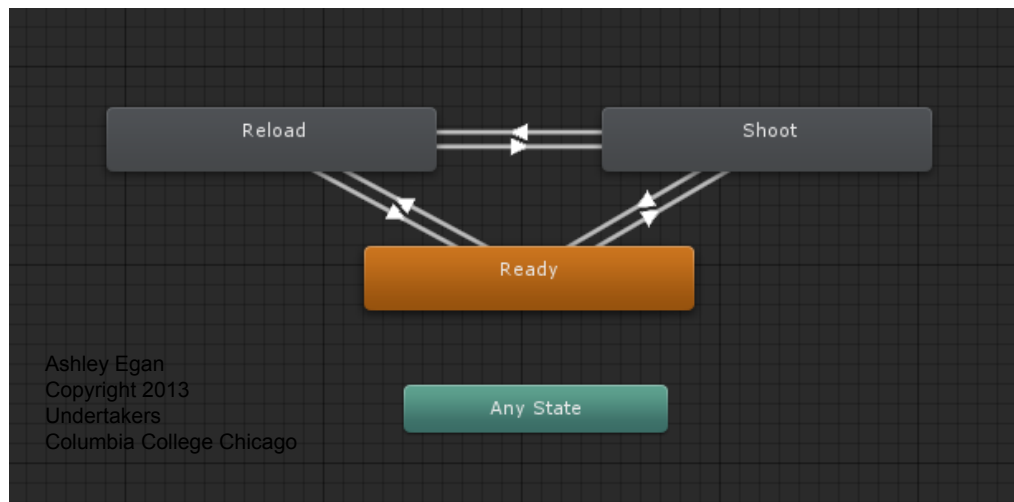


Animation Skating

- Not the same type of skating.
- Methods of resolving problem:
 - Added or reduce # of frames in MB, then tested in Mecanim
 - Adjusted speed in script
- Would have not been a problem if we used Root Motion...



Animation Layers



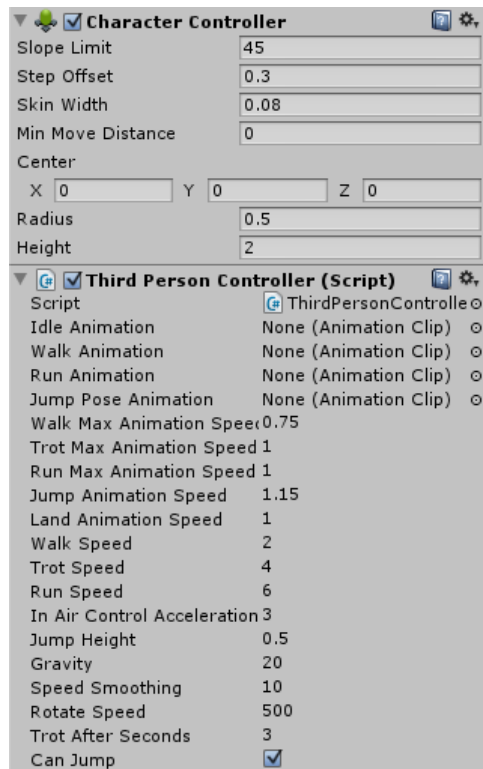
And then there was Adam...





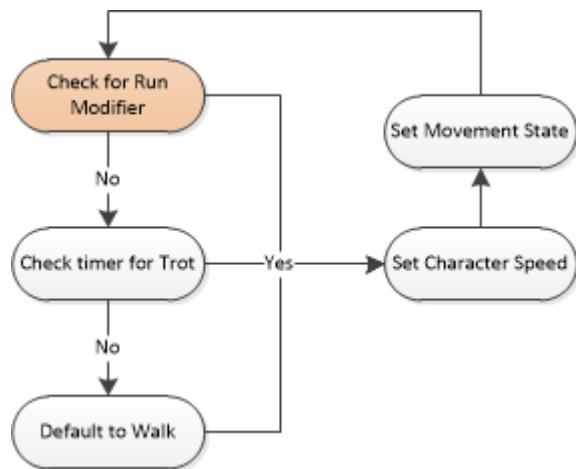
Player Movement, Unity 3.5

- Prototype - Third Person Controller script (Unity)
- Production - The TPC was modified heavily to fit our needs
- Built for the Legacy animation system

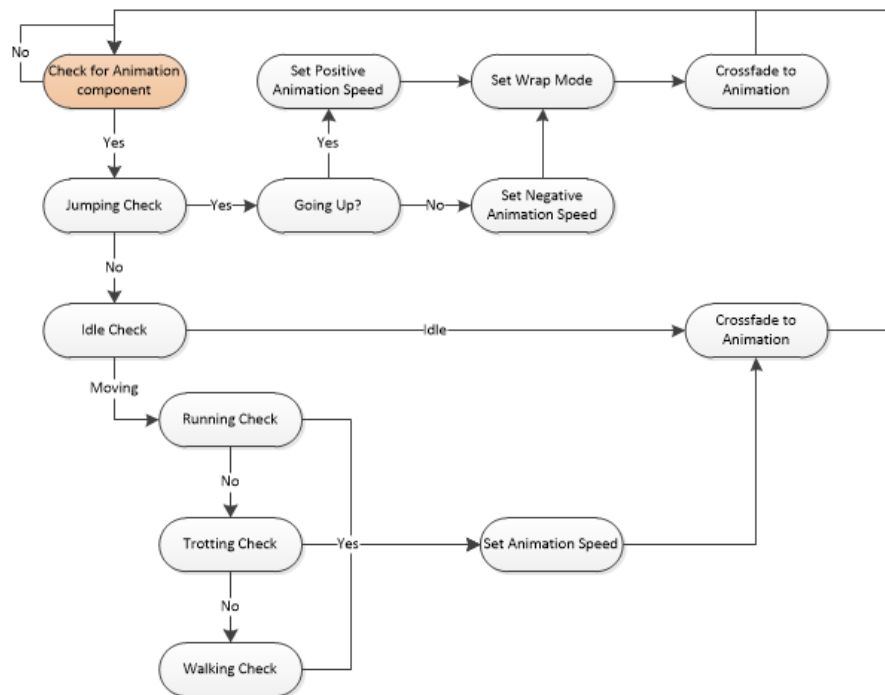


Legacy Animation Flow

Get Input



Play Animations



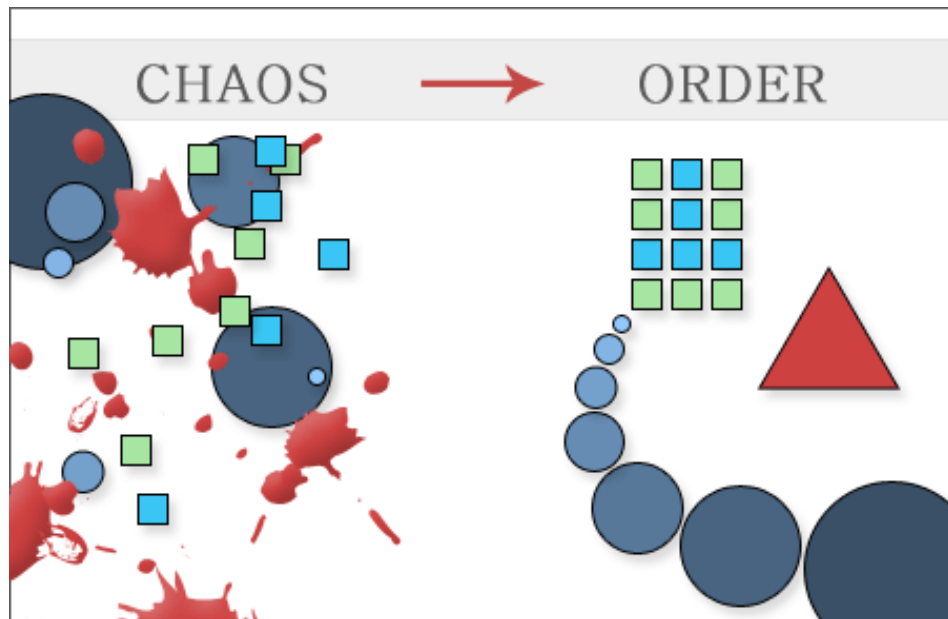
Player Movement, Unity 4

- Movement needed to be upgraded to work with Mecanim
- I had to make even more modifications to the TPC script
 - Refactor player input conditions
 - Play the correct animations using Mecanim
 - Not break everything else



Refactoring Player Input

- The Legacy TPC input system was very confusing
- I refactored all input into a single conditional block
- Each potential set of input had a direct effect on character state



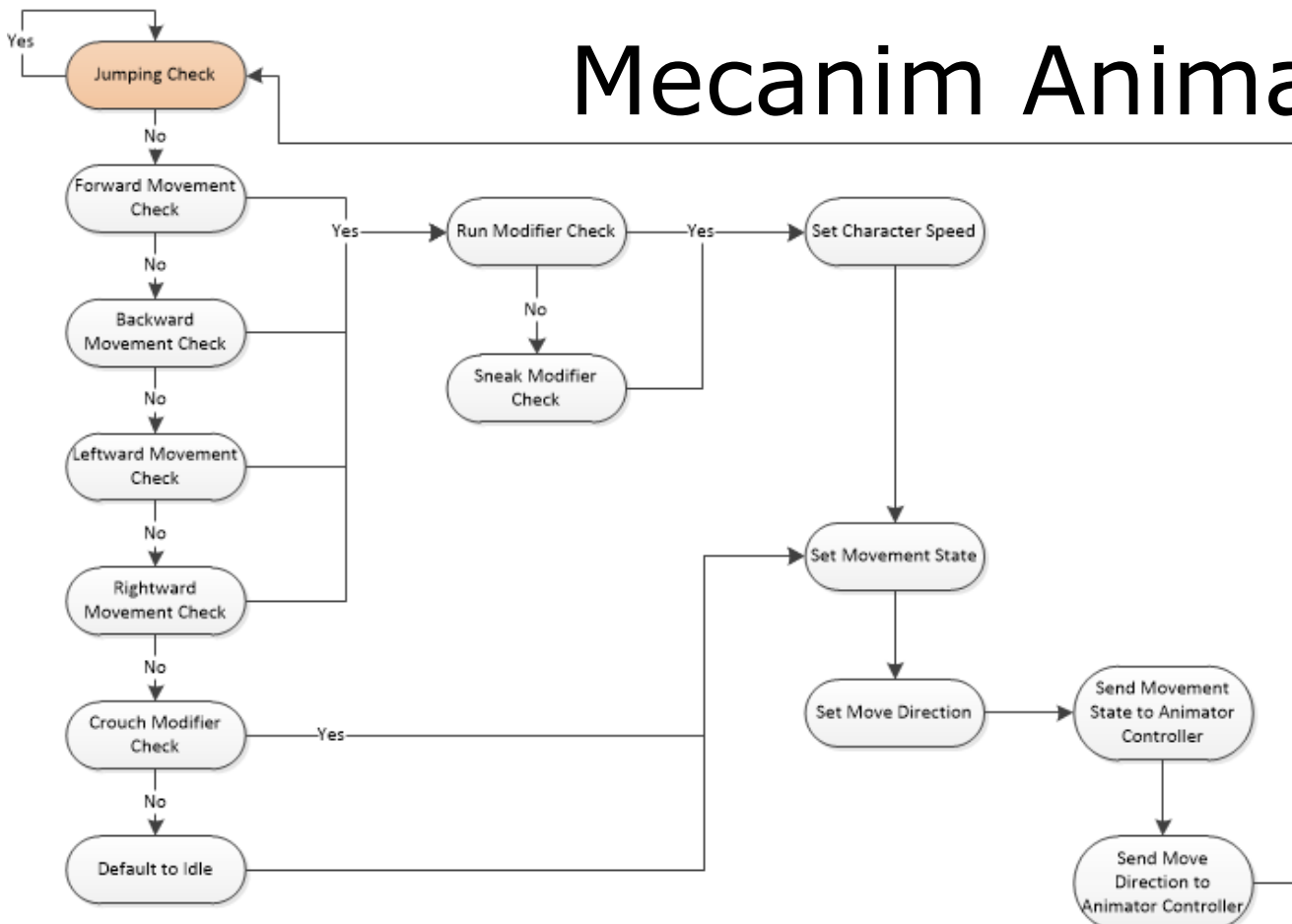
Playing Animations with Mecanim

- Character movement states acted as animation parameters
- I just had to send them to the animator controller
- Legacy code I no longer had to deal with:
 - Checking velocities
 - Setting animation speeds
 - Writing crossfades

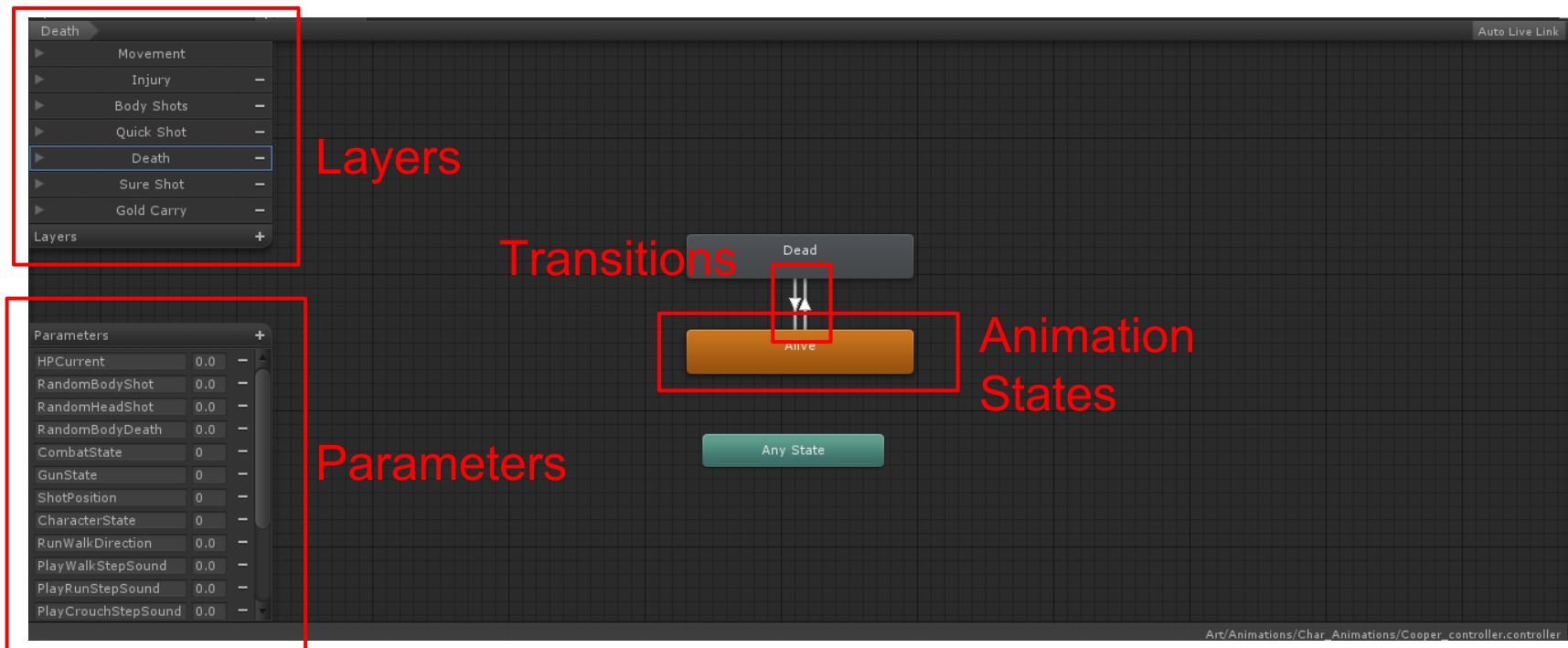
```
animController.SetInteger("CharacterState", (int)characterState);  
animController.SetFloat("RunWalkDirection", (float)runWalkDirection);
```

Mecanim Animation Flow

**Get Input
&&
Play Animations**

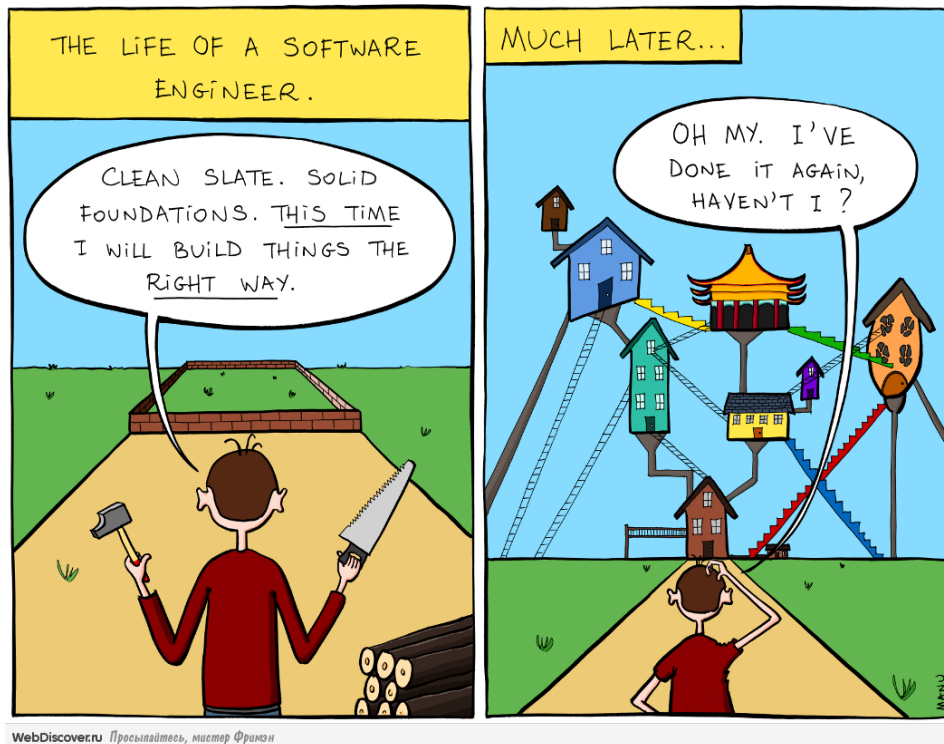


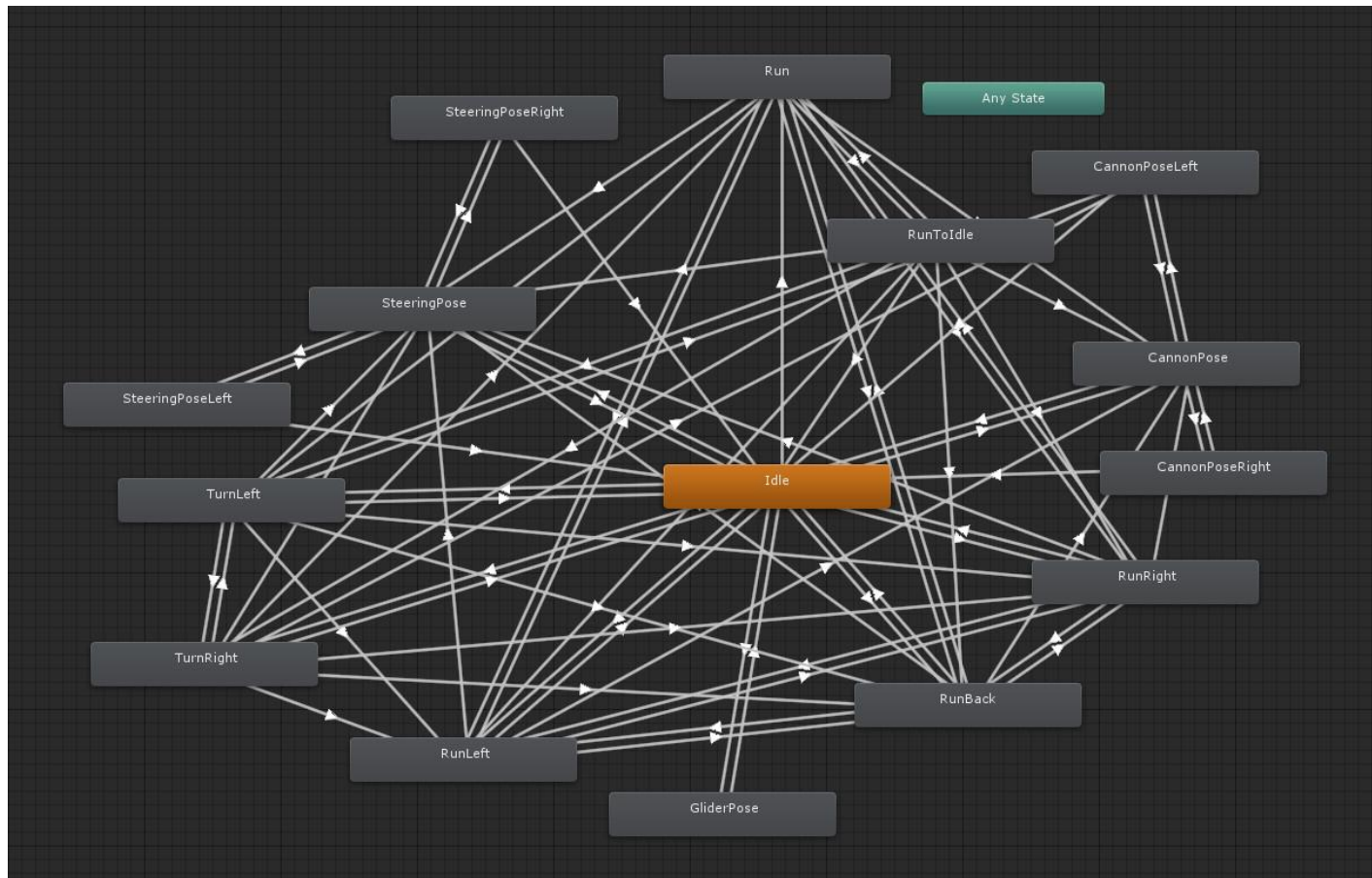
Developing Our Animation Trees

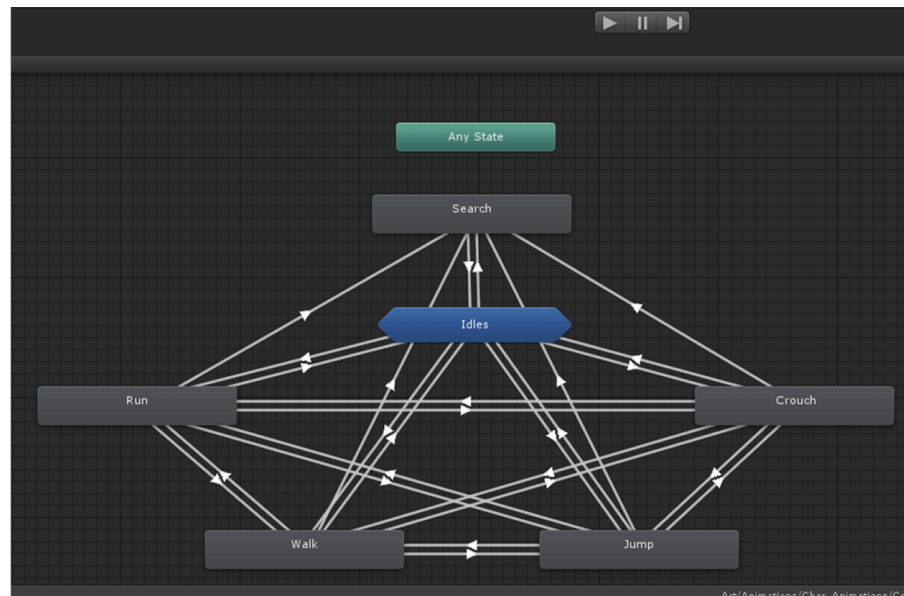
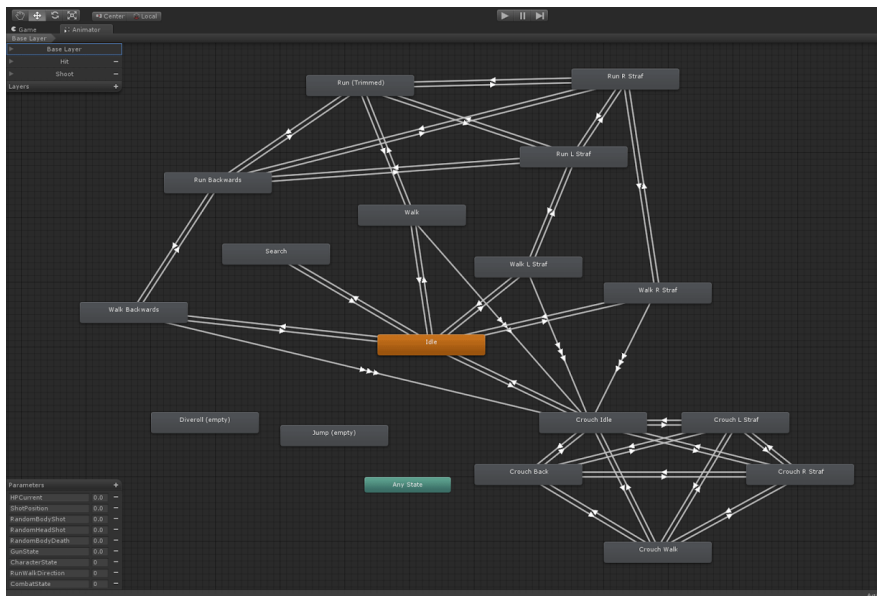


Developing Our Animation Trees

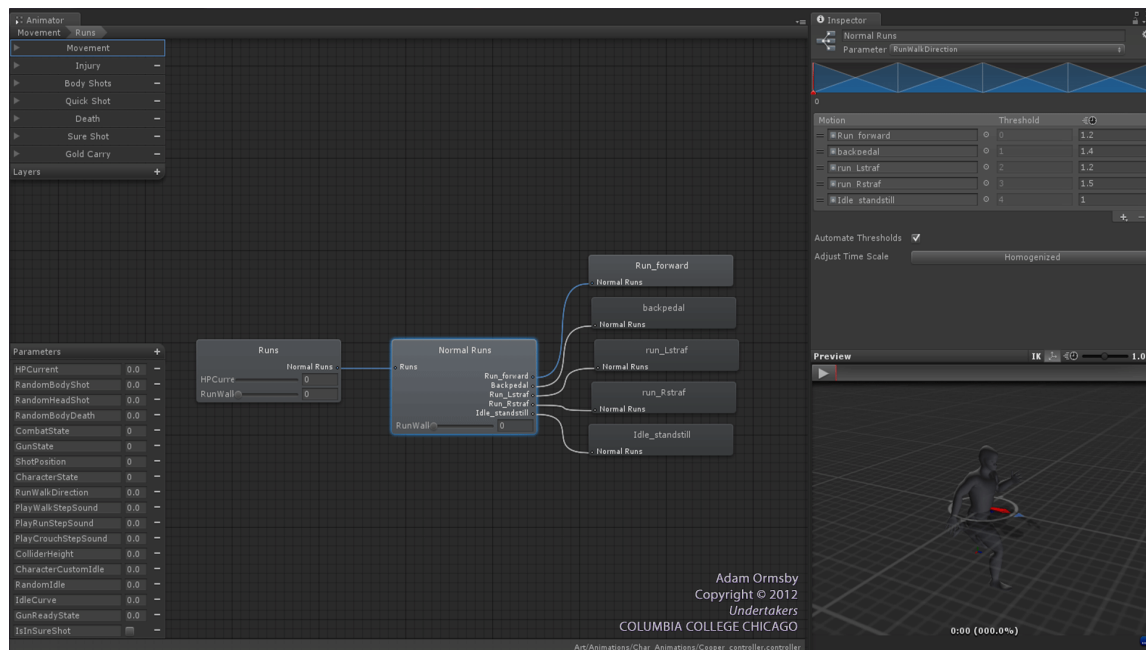
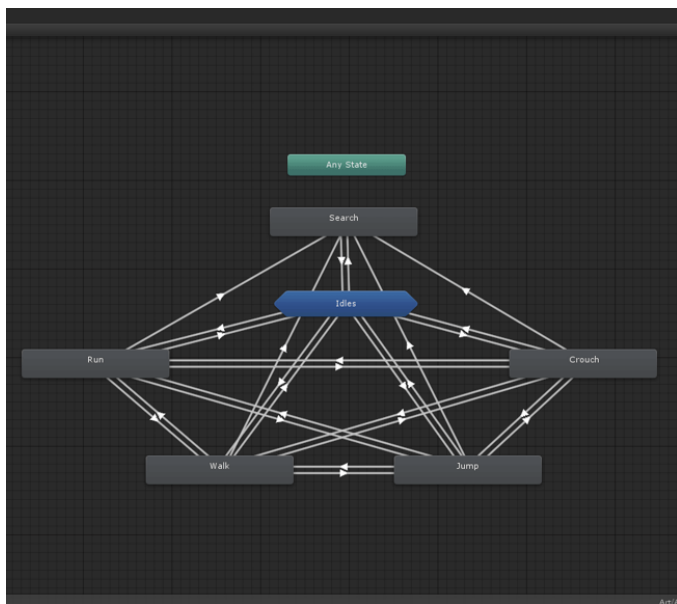
- Animation trees need regular gardening
- Two basic rules:
 - Complex systems are just a combination of simple systems
 - If it looks simple, it is simple





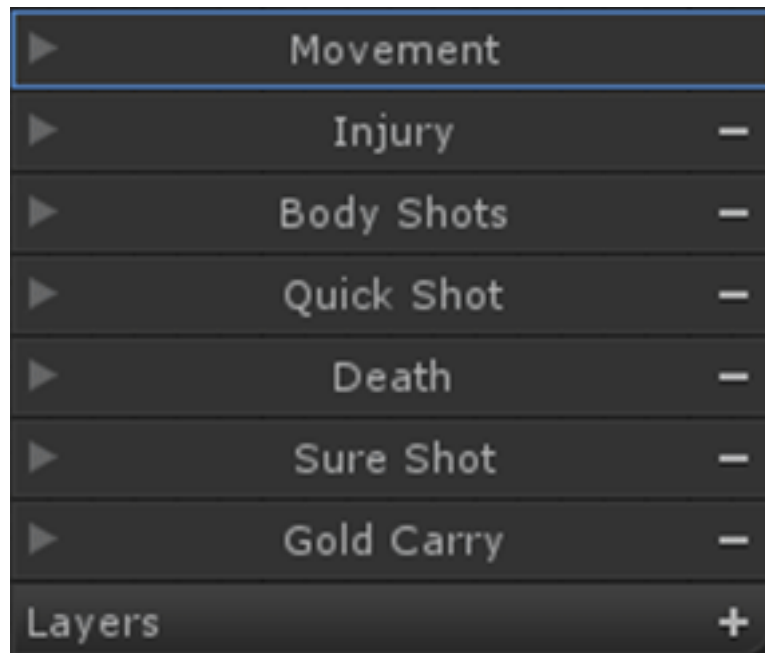


Minimizing Clutter w/ Blend Trees



Grouping Animations In Layers

- Animation layers were great organizers
- Motions were separated according to functionality
- Override/additive settings were used to create a pseudo-state machine

A screenshot of a dark-themed user interface menu for animation layers. The menu is a vertical list of items, each preceded by a right-pointing triangle icon. The first item, 'Movement', is highlighted with a blue border. The other items are 'Injury', 'Body Shots', 'Quick Shot', 'Death', 'Sure Shot', and 'Gold Carry'. At the bottom of the list is a 'Layers' item. To the right of each item is a small icon: a minus sign for the first six items and a plus sign for the 'Layers' item.

▶	Movement	
▶	Injury	—
▶	Body Shots	—
▶	Quick Shot	—
▶	Death	—
▶	Sure Shot	—
▶	Gold Carry	—
	Layers	+

Blending Between Layers

- There is no blending between layers
- I hacked in some basic blend code
- Not perfect, but it worked

```
if(currentPlayerGroupState == PlayerGroupState.GoldCarrier)    //if player has gold
{
    if(animController.GetLayerWeight(6) < 1)    //if Gold Carry layer weight is below 1
    {
        //increment layer weight every frame until it is 1
        animController.SetLayerWeight(6, animController.GetLayerWeight(6) + 0.2f);
    }
}
else    //if player does not have gold
{
    if(animController.GetLayerWeight(6) > 0)    //if Gold Carry layer weight is above 0
    {
        //decrement layer weight every frame until it is 0
        animController.SetLayerWeight(6, animController.GetLayerWeight(6) - 0.2f);
    }
}
```

Blending Between Layers



No Blending



Blending

Looking Back

The Good

- Using Mecanim
- Improved animation pipeline
- Streamlined movement
- Boosted team morale

The Bad

- Going in blind
- Late MoCapping
- Animation curves

The Ugly

- “Exit Time”



THE MECWARRIORS



www.mecwarriors.com

Ashley Egan

Animation Artist

Email: animsbyashley@gmail.com

Twitter: [@haiaashleyy](https://twitter.com/haiaashleyy)

Web: www.ashleyegan.com

Adam Ormsby

Technical Designer

Email: ormsbyadam@gmail.com

Twitter: [@TrainerGary](https://twitter.com/TrainerGary)

Web: about.me/adamormsby