Character animation design document for Skunkwerks

Table of Contents

INTRODUCTION

This document attempts to clarify the details of character animation related requirements for skunkwerks, and hence the motion data requirements.

From Natural Motion Morpheme documentation

"When creating a character for pretty much any game, there are generally three main objectives:

1. *To create good looking smooth motion.*
2. *To create a responsive character that is quick to react to instructions, either from the user or the AI code.*
3. *To create a character that integrates well with its environment, including other characters.*

Unfortunately these objectives can be quite conflictive and it requires a well designed network in order to satisfy them all."

The following 5 steps are carried out to complete the design component. This is the workflow I'll be using. Morpheme:connect can be used directly in carrying out step 3.

1. Identify the requirements of the character within the game.

2. Next outline a set of individually definable actions that characters make to meet those requirements.

3. Next identify state machines

4. Then look at 'is the character responsive enough for the situation it is to be used in'?

5. Clarify how each action is to be build, and what animations are needed.

INITIAL SPEC GAMEPAD BINDINGS

Many things in skunkwerks are generic in design, but for sake of simplicity I'll take Evey as the player character here for starters. I'm going to place in Andrews control scheme spec right here for reference. It's an early design spec - we can infer some general requirements from it.

Here's the given control spec from the outset ..

Left stick \ WASD: Move character. Characters turn to face movement direction. For PC, shift toggles walk mode.

Right stick \ Mouse: Aim target reticule \ camera.

Left Trigger \ Left mouse button: Shoot equipped gadget.

Right Trigger \ Right mouse button: Swing default melee weapon.

A button \ Spacebar: Jump \ double jump. Hold for jetpack \ parachute.

B button \ 1-8: Open push-selection wheel \ equip gadget.

X button \ Ctrl \ Middle mouse button: Interact \ get in vehicle \ get out of vehicle

Y button \ Windows start +1-8: Support wheel (summon vehicle, monster, robot, air strike). For PC, override gadget list with support items.

L1 \ Alt: Hold for strafe.

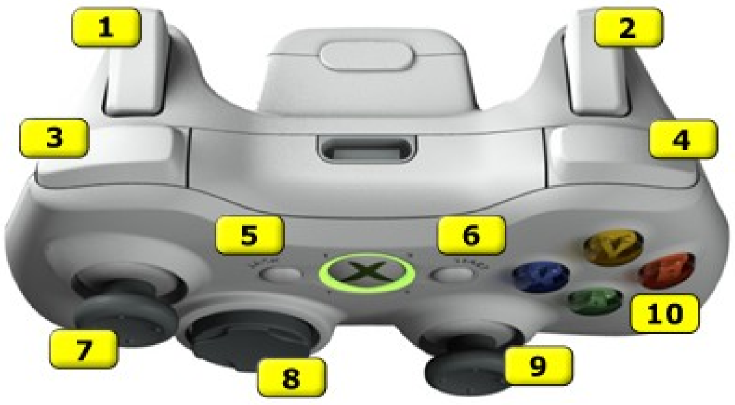
R1 \ Mouse Wheel Up: Hold for aim mode. On PC, mouse wheel up to aim, down to cancel.

Dpad: Toggle active character in squad.

Start \ Escape: Main menu

Select \ Tab: Toggle objective map on \ off. Hold to enable \ disable objective arrow

GAMEPAD CONTROLLER BINDINGS - PROTOTYPE PLAYER CHARACTER



Generally the bindings follow the way we like them to work in game.

Since this prototype deals only with character mechanics, and not inventory type things, or direct interaction with other game objects, the bindings are altered to suit running the character through the various states. Prototype controls are not meant as definitive game controls – they are partly set up to get a feel for playing the characters, but partly also to allow me to easily demo various actions.

1. Left Trigger - Blend in the pistol shooting system. This equates with equip a gun in gameplay.

2. Right Trigger - Hurt, quick reaction

3. Left Bumper - Knockback

4. Right Bumper – Dance – sort of stands in for taunt

5. Back - Not used

6. Start - Do a giant attack. Also locomotion slows down for a while

7. Left Stick - Walk / Run / Strafe. Pressing the left stick transitions to stunned.

8. D-pad - Not used

9. Right Stick - Aim gun. Only when pistol system is active. Pressing the stick shoots the gun (recoil). Right stick also controls the 3rd person camera in prototype.

10.

A - Jump

B - Sword attack. Quick taps chain

X - Pickup / Throw. (Prototype alternates between pickup and throw). Pressing X while moving does a partial body throw during locomotion.

Y - Melee. Quick taps chain

A NOTE ABOUT ANIMATION STYLE

First off lets get the issue of 'arcade', and 'style' put down on paper. Style can be thought of as anything that deviates from the realistic bipedal motion of an equivalent 'real instance of a little girl inventor-type called Evey, running about town shooting things etc.. That's - like - real, as in our world, on some street. Basically evey is a cartoon character - pretty detailed in relative terms, as far as you average cartoon representation goes, but alas, cartoon. Comic. Not real. So she needs style over and above the style naturally inherent in say, our hypothetical 'real' Evey.

For our game requirement, we have Maximo, Ratchet and Clank, Maximo, Psychonaughts as style references. Evey is quite different model and form-wise from the player characters in any of these games.

But basically we can take from the style of these games a few discerning qualities of the character motion : Exaduration of poses, Holds (holding the action in a static pose over time), relatively quick playback of motion, very quick transitioning from states, and good mapping of characteristic motions to each character type (like bunny hop runs in R&C).

So, basically we want Evey's motion to appear a bit comical, using these techniques and other bread-and butter rules of animation like balance and timing, but also aware that fitting the animation to her character as it appears visually has to be adhered to. It would probably be a mistake to apply R&C hopping run straight to her and disregarding her physique and level of detail.

SECTION1

Requirements for Evey *- plain english description of what Evey can do*

- Evey can run and walk in any direction, sometimes while shooting her pistol at the same time. Her shooting weapons include pistol and a large rifle, and also she has an ability to lob things with a tennis racquet. She only shoots and lobs forwards direction. She has one more special variation of lobbing whereby she throws her skunk. When evey is shooting with pistol and rifle, the shots can repeat quickly one after the other - she doesn't necessarily have to holster, draw, shoot for every bullet. So, the shooting actions will be simple the act of momentary aim and instant recoil, and they will be built for quick repetition.
- Evey needs to be able to turn around on the spot.
- She can walk and run at variable speeds.
- Sometimes she's normal size, sometimes she's a 50 foot tall Evey.
- Evey has various types of melee that can be divided into two types: a) Sword and b) Bare handed. Apart from area type melees, she only projects her attacks forwards. This holds for sword and barehanded with just a few exceptions (area attack and spinning sword).

  a) For sword, she can strike with her sword in one hand (right) downwards (vertical), across (horizontal), and spin in a circular 360 sweep. They are quick, cutting actions and when she moves the blade it cuts clean circular trajectory arcs through space so that particle trails look - circular. The spin is an area attack (not projected forwards) it's affected in a 360 arc. Like an ice skater performing a spin, Evey can repeat her spinning sword attack. Like a boxer throwing a jab, she can repeat her vertical and horizontal sword attacks. When she commits these attacks there is forwards driving motion in the action. So, given that there is a quick transition into the attacks, and that there's forwards driving motion, and that there's a quick transition out - there's no need for hampering the motion with partial body blending.

  b) Eveys bare-handed melee attacks are a punch and a kick. Her punch is a straight right. Her kick is a spinning right-heel kick. She performs these quickly, they are both forwards-driving attacks where her whole body moves into the enemies direction.
- Evey can get hit by enemies, via various means - projectile, melee, flying objects and debree, etc etc. On getting hit she reacts with a 'hurt' action. For explosions and other high-impact things she gets knocked over. She can get back up from there - or else she dies.
- Evey can jump while moving, or from the spot. Evey can double jump - an entirely fictitious action that always has the character being empowered with some mysterious 'momentum', and corresponding action while in mid-jump. Since she can jump while moving in any direction, this carries also to double-jumping in any direction. Evey can also jetpack. This is a variation of double jump.
- Evey can parachute – NOT CURRENTLY WORKING ON THIS
- Evey can enter and leave a vehicle. She can drive a vehicle. - THINK ADAMS DONE SOME THINGS WITH THE STANDARD ASSETS FOR THIS
- Evey can die.
- She can stand idle.

- She can be stunned momentarily whereby she can't do much except stagger and 'waver around' a bit.
- She can be in pain whereby all her locomotion becomes a 'ill' - she limps a bit but can still walk / run in any direction. She's a bit slower to travel during this time too. Note: Built an example of blending in this type of motion within morpheme and one motion works pretty well blended with a multi-directional locomotion state.
- She can go into a 'combat position' ready for action. Note: adding here simple because combat idle is part of spec list.
- She can taunt monsters and enemies by gesture. Again - in spec.
- She can go into a panicked state where she runs screaming forwards with arms waving. In spec.

Requirements for Monster Additions *– append monster abilities based on the AI document requirements.*

Here are the extra requirements taken from AI behavior sets document:

- Ranged attack : EyeBlast, will shoot laser power from eyes
- Melee attack : Bite
- Area Attack: AreaAttack, will create a small dust tornado that lasts for a few seconds and moves towards the target. After performing the spin the Monster should go into a stun state, to allow for vulnerability.
- Spit: will spit ice shards as projectiles at the target
- Monster Ball Roll: , if the monster hits a building during this roll, it will go into a stun state to allow for vulnerability.
- Melee Claw
- Melee Claw Mocap

SECTION 2

Set of individually definable actions that Evey makes to meet these requirements

Here we lay out the actions that fill the requirements, and note roughly when they can take place, and what can follow from them. This step isn't a full definition of possible transitions - just some notable ones to clarify some important ways we can go on the 'mechanics' of various actions.

1.

Running and Walking Multi-directional (Velocity based strafing)

[Note: This turns out to be precisely the system utilised in Ratchet and Clank, no modification to this definition required.. so it's on with creating the state in section 3]

Running multi-directional. Class this as one action - really it's four. Clarify this action as Evey being able to face an arbitrary direction and still travel in any other direction. This is strafing. Four input blendtree either hard switching or blending based on direction. In game currently we have hard switching. Motion will cater for blending if liked. [All the locomotion animation provided is tuned technically for good blending and to provide a good clean base motion]. Evey can perform this action from standing, from walking, while shooting pistol, but not while doing melee barehanded or sword (these have forwards driving motion of their own), and not while hurt or in the process of dying or getting up, or jumping. She can't run and shoot the 4 foot rifle because it send shockwaves through her body !

Walking multi-directional. As above.

2.

In addition to the walking and running multi-directional as a strafe action (arbitrary facing direction) sometimes Evey can walk and run only in a forwards facing direction. She can change her orientation during this action but the travel action will only be forwards. [From the control scheme we can assume this is the default action for locomotion, and strafing will be instigated by a hold on the controller or keyboard.]

3.

Shooting - since the requirement is to be able to shoot from standing or while travelling, Evey performs a quick 'shot' action. Logically similar for rifle and pistol - but vastly different posture and recoil action. Pistol is sharp action and rifle propagates more through entire body. Pistol action for shooting is quicker, and both are repeatable actions, so transitioning from a shot to a shot (repeat action) is very fast. Shooting pistol can be performed while idle, turning, in travel, but not while jumping. [Note: included jumping in the prototype] Rifle shooting can be performed on the spot, and while turning. When Evey is equipped with a gun of any sort, she can aim in a forwards facing direction. This implies that she orients her facing direction to the target, and she does this when travelling or when on the spot. Aiming doesn't imply an exact projection of weapons directions to target. It does imply Evey holding the weapon and being in a combat posture with weapon 'ready' for a quick transition to the shoot action. It also implies of course that she is facing the target direction. When Evey travels with weapons equipped, the weapons are visible, carried in one hand, so any motions have to consider this. [Note: I added aiming to the prototype network].

Note it's easy to turn some decisions in this design doc - we CAN let her shoot a big rifle while running. We CAN let her shoot pistol when jumping. Allowing both of these do require just a little development of systems and tuned animation to allow it. [Note: I haven't added running with rifle to prototype network in the next stage - but I have included allowing shooting while jumping, and also extended the system with aiming].

4. Turning on spot. Evey can perform this action from idle. Turning on spot can be performed in conjunction with shooting a pistol.

5. Lobbing grenades and kitty. Two similar actions almost can be grouped as one - but require different (but similar) motion. Both actions are done from standing. Both are one continuous action done quick and neat. I see restricting throwing and lobbing grenades to a standing action as pretty reasonable - but again - the action could be developed as a top body blend. Making convincing full body motion portraying the action more difficult and complex to implement.

6. Walking and running at variable speeds - this applies to the walking / running multi-directional above, and also to the walking / running forwards default action. To do this she slows or speeds up her rate at which those actions are performed.

[Note: this is a subset of 1, and done]

7. When Evey is big, everything she does is slower. Evey employs the 4 distinct windup / action large monster melee attacks in addition - and details there can be identical to how we spec them up for Monsters - though with controller input for triggering rather than AI.

8. For Eveys 3 sword attacks, her action is to instantly 'swing into the action'. Can happen from idle or from locomotion. Can't be performed midway through another melee or jumping. The action for the sword horizontal and vertical needs to be forwards-driving. The action for spinning needs to be cyclic, so we break into 3 components - start, middle and end. The middle action can repeat like an ice skater spinning, holding momentum through the move from one cycle to the next.

9. For Eveys two barehand melee attacks requirements, the actions are basically similar to the sword attacks horizontal and vertical - quick, forwards driving actions.

10. For gettting hurt, or hit - a single 'reaction' action takes place. Can happen at any time. It's a quick action.

11. For getting hurt bad - the action is to 'fly' in one of 4 directions based on from where she gets hit. She then either gets back up or dies.

12. To fill the jumping requirement, Evey performs a single action where she springs straight up and tucks legs. Can be performed from idle or when moving in any direction. Double jump action is an extension of jumping, momentum is added upwards and she flips in the direction she is jumping.

13. Die action - happens after flying (being hit hard), or otherwise. Quick representative action not too gory or depressing.

14. Evey can stand idle and from there transition out to most other states.

15. The action that covers stunned momentarily, is much a canned motion that's looks sorta entertaining and doesn't transition out to anything. It's time for player to wait and ponder what he / she did wrong. This action has Evey travelling backwards a metre or two.

16. Her action for being in pain state can be incorporated to locomotion action - a blend of limping motion works very well in a multi-directional locomotion state.

17. Combat idle action is a combat pose with some subtle motion. Usually would transition into a melee, sword strike etc. Can be used to good effect as applying a base pose that all melee can return to. NOTE: Did that with every melee in UP project. It's nice for technical strictness but actually tends to get a bit tiresome visually after you've got 10 melees all returning to combat idle. It's my experience that it's often nicer to just be a bit 'rough' with where you end you poses and let the game system do the blending. The combat idle is also suitable as a 'ready' pose for shooting. Appropriate as an intermediate in a shooting state machine.

18. She can taunt monsters and enemies by gesture. A simple canned action. No special requirements as yet.

19. Panicked state where she runs screaming forwards with arms waving. This action sees Evey doing the charge and presumably dishes out a bit of 'crazy power' to enemies. Still to flesh out.

[UPDATE]

Gotten the impression that panicked is more about monsters fleeing than player character going berserk. Section 3 fleshes out the detail.

Individual Actions for Monsters

20. Eye blast consists of a lead up action (hands raise to the 'project eye beams' standard posture) chaining to a looped cycle animation. The Body motion during eyeblast is full body. Donk is stationary during eye blast.

21. Bite consists of a lead up action (tense growl), chaining to a looped cycle bite sequence. The bite sequence has forwards momentum.

22. Area attack consists of a lead up action, twisting into a tensed posture, chaining to the action which is a single-sequence.

23. Spit consists of a lead up action, chaining into a standing spit action which is a single sequence.

24. Monster Ball Roll has a lead up, a middle section (the roll) and a lead out.

25. meleeClaw and meleeClawMocap are close range, single sequence attacks. Best inserted into the existing melee framework.

SECTION 3

Identify state machines

This is the 'working' part of the documentation. Each entry relates to an entry in section 2, and is the prototype implementation the subnetwork. I create all of these using morpheme:connect to build the network, and the lua script interface to provide a good level of detail on control, fine tuning, and extension on the functions available visually. This makes the prototype a little less like a prototype and more like a production design.

I'm making this section also to serve as a progress document - since I'm noting where each subnetwork gets completed, and I make video presentation and provide the network and control code, and the animation, markups etc etc, as I 'tick off' each one.

1. Running and Walking Multi-directional (Velocity based strafing)

Here's the first and probably most important animation state for player character. 8 source animations feed into two blend nodes. In data (the animation), each of these sequences is synchronised to be in phase, and each contains full root motion defining trajectory. The Lua update script gets the input direction and maps that to weight parameters in these 'directional blend' nodes. They also hold a bunch of characteristics for style - which I've also broken down into layers in some animation editing files.

[Style note: Some of these are : bounce, secondary motion, shoulder turn, extension of limbs at key poses, simple motion trajectory arcs for all the characters 'control nodes', and important is the timing adjustments that give the character quick contact phase and long suspended phase . The motion of a stylised character animation can be broken down into components pretty much like an animation network that utilises them.]

Some trivial things .. top runspeed is 2 metres per second. Top walkspeed is 1 metre per second. (This is relative though - because I have another speed multiplier node in this network which can speed up and slows down the whole state). The prototype network also has a global 'slow motion factor' which can again multiply the rate of the whole network.

The next part takes these two results (a directed walk, and a directed run), and does another blend based this time on speed (magnitude of controller input). Again the Lua update script takes care of mapping from controller to the weight parameter on the node. It also handle deadzones and provides some flexibility for tweaking and smoothing - though there's nothing fancy here currently - just a linear mapping of input to node parameter weights. One more input which is an idle, contributes at lower speeds or stationary.

One more morpheme system node does a dampening on the link between input pressure and travel weight - this gives a nice 'softness' when your quickly moving off or releasing the stick quickly. This addition makes the network slightly smoother with a negligible cost in reponse.

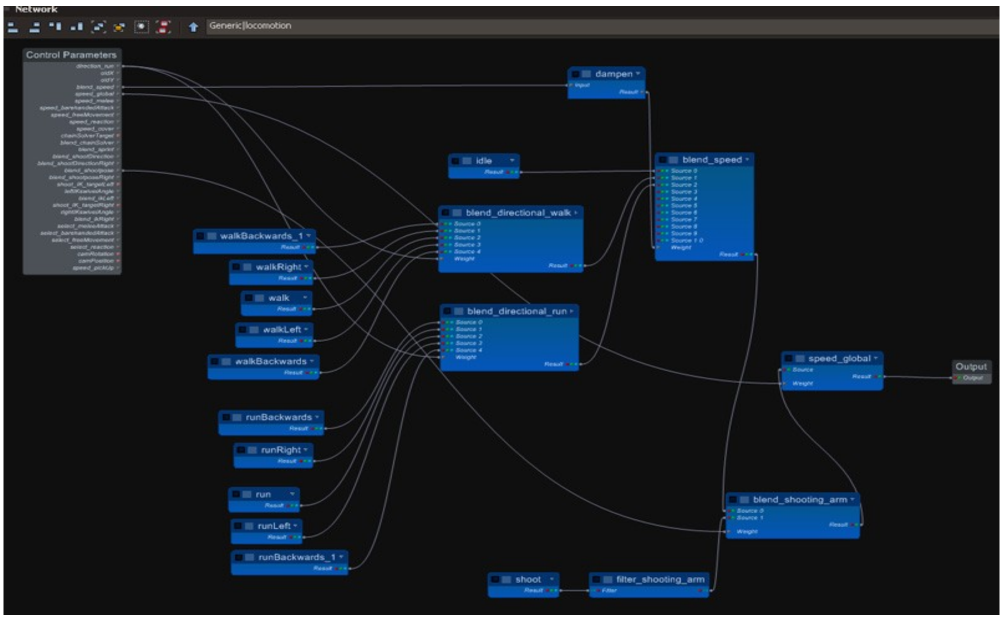


Illustration 1: Locomotion Blend Tree – early

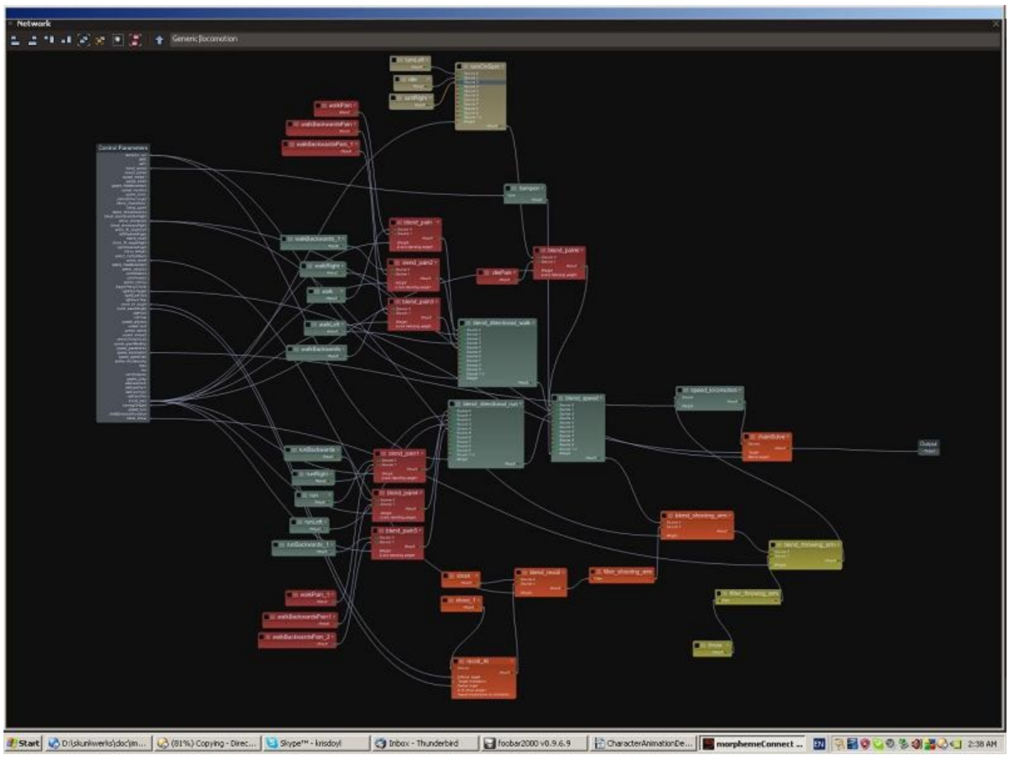


Illustration 1: Locomotion Blend Tree growing with additions for pain, throwing things, and turning on the spot.

2. Walk and run only in a forwards facing direction

As far as animation network goes, walking and running in a forwards only direction is a subset of the strafing network - with the addition of one rotational transform on the root of the character, around one axis, giving the direction of travel.

3. Shooting

Shooting is really the next most important character state in the animation network. Shooting is an integral part of the character play in Ratchet and Clank - and it's mostly implemented with systems over animation data - and that's what I've done here. However - that said - those systems have to work well with the other animation data that contributes during the shooting. Namely, the locomotion and the jumping.

One of the things that makes R&C fun, is being able to quickly equip the combustor, have the character run along with the gun level and ready, and quickly fire off shots either from the 'usual' far-behind view, or the closeup 'shooting' mode - and not think too much about which mode your in while doing it. I've completed the underlying character system to provide the motion for this - and also tuned it to work and look nice. When running with pistol I've put some things in place that hold the 'gun' part level and steady, while the rest of the locomotion runs as usual underneath. This is far different in visual result than simply blending a 'shooting motion' onto top body, or arm. A good analogy is what a dead parrot looks like stuck on a pirates shoulder - he just sways around like one object stuck rigidly on another with no consideration of the internal motor-drive of the parrot reacting to his connection with the world - the pirates shoulder.

Here's a basic description of the network (it's what's done in Ratchet and Clank combustor weapon) ..

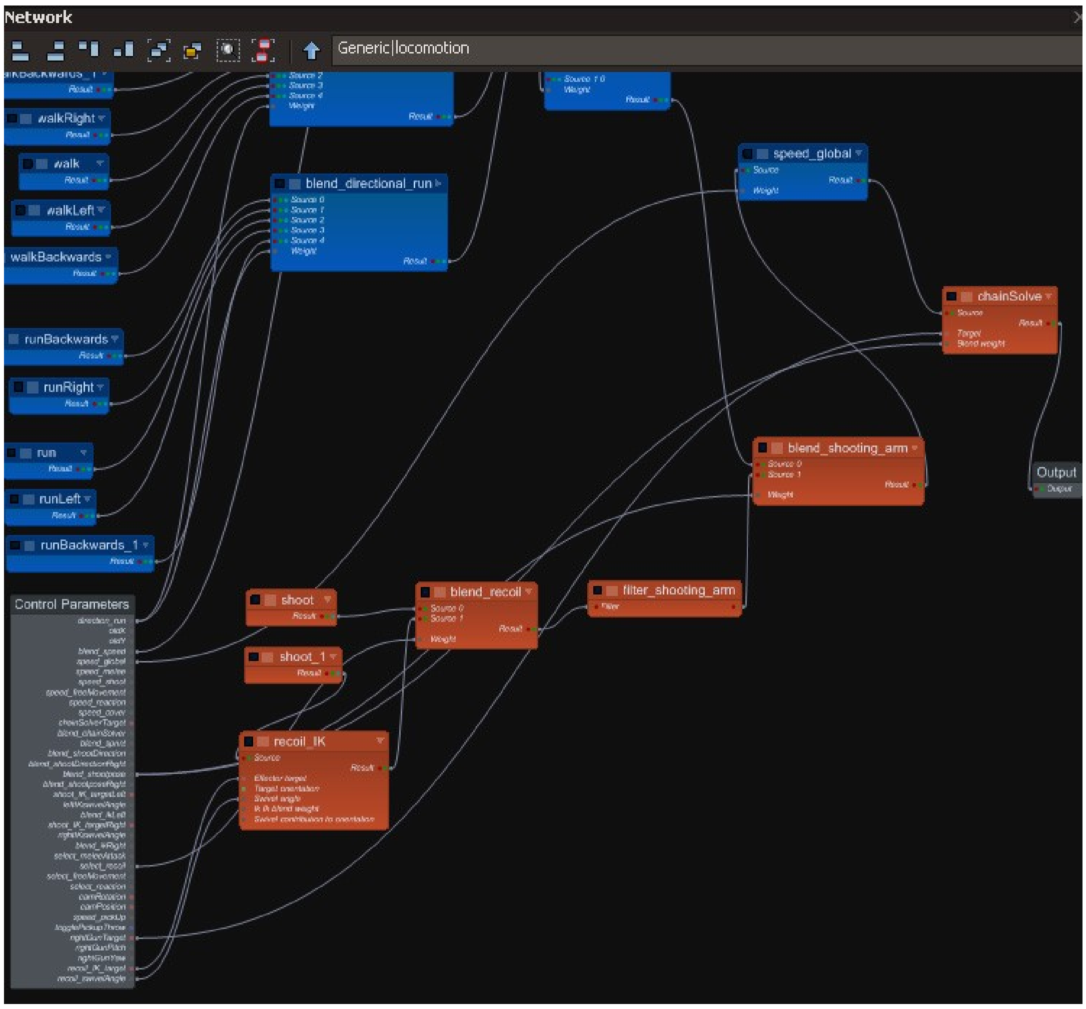


Illustration 2: Shooting Blend Tree

The shooting sub-network is in red above. It all sits inside the locomotion state, since it's tied closely with the strafing (which is actually a big part of the closeup shooting system in R&C).

From upstream to downstream. There's one, 1-frame animation at the begining of the stream, a very simple shoot pose, but with the character standing in idle, 'shoulders forwards', right gun aiming forwards.

This pose gets modified with a 2-bone IK node (basically pull the target in towards the body a bit, what happens on recoil of gun).

There's a node to basically blend that on and off, and that's essentially the 'shoot' bit.

Then the arm gets filtered, and soft blended into the main stream, at the 'blend_shooting_arm' node.

Then the important bit - the right arm is modified with a chain solver . This does two really important things. First, it provides a stabilisation to the hand, or gun, such that is constrained in position and rotation while the character is running. Looking at R&C, this is what happens there as well. The other thing - is that this node provides an ideal spot to add a further function to the system - which R&C has in the closeup mode - actually aiming (the stabilised) weapon at a target - in pitch and yaw. The chain solver is ideal for this because, unlike simple 2-bone IK solving - it solves a whole chain from clavicle to hand, and solves it based on an aimpoint rather that positional target of the hand.

In Ratchet and Clank the aiming in closeup is really very, very rough - we had a good look at this on playing .. though this prototype system gets the aim spot-on, no matter what the underlying locomotion.

So, in this prototype, a target point gets projected onto a distant sphere, by way of a pitch and yaw control (which I've hooked up to the right thumbstick for now for playtesting) - but that positional target can be any game target, deriving from target reticle projected into the world, as in R&C, or based on an auto-aiming system. Anyhow - the player character prototype can now run in any direction, any speed, point the gun in any direction in a wide spherical arc in front (and slightly behind if we want), and also a recoil action occures correctly jolting the gun back no matter which direction the gun is pointing.

Also, I extended to include the system inside the jump state. So, whatever applies above, also to jumping. This is also a feature of R&C.

I've added also a separate state which implements a full-body type shooting (so logically only appropriate during stationary).

The way I've set up the controls for prototyping are:

I've put in the left trigger to blend in the system (this would be 'equip pistol weapon' in a game system)

I've bound the aiming to the right thumbstick (this can be anything in the game system - R&C uses raycast from reticle through camera, and only uses the pitch and yaw during closeup shooting).

When shooting without the system applied (just pressing B without left trigger), this prototype trasitions to the separate, simple, full body shooting motion.

4. Turning on spot

Turning on spot is now included in the network.

Note: I think turning on spot only suitable for monsters – this looks really nice from a fixed camera, observing the character, but for player – it's better just to 'rotate the pawn' I think.

I've provided two alternative additions – One being state-based, the other being an inclusion in the locomotion state. Both use a set of animations turnLeft and turnRight, for which I've set up a customised trajectory constraint in Maya, so the trajectory is still on the ground, and it still stays directly below the hips, but it is rotationally constrained in Y (around vertical axis) to the Hips node. It provides the ideal data for animation-based root motion.

The state-based addition consists of two states, turnLeft and turnRight, which can be transitioned to like any other state. This provides a nice simple way for the game to turn a character on the spot – just transition to either (or from one to the other), and the character turns, smoothly blending in and out, and with a constant rotational velocity.

The locomotion option is a blending of turning on the spot, with idle. A control parameter governs a blend node, with inputs, turnLeft, idle, turnRight. Adjusting the control parameter (range 0..1) away from 0.5, turns the character at variable rate. Looks best though when eased in to the limit values. Because this addition is part of locomotion state – turning on spot can happen at the same time as shooting, or throwing (which are also part of locomotion state).

5. Lobbing grenades and kitty

Included the throw and pickup states - currently done as a full body motion. Can definitely provide a quick partial body action for throwing while in locomotion (similar to pistol shooting) - but currently not.

[Update: the prototype also has included the tennis lobbing action. The prototype alternates between lobbing and throwing]

Update: Included throwing while moving now. Done as a partial body blend within locomotion. For the prototype network, I've set things up so that when the character is moving, the 'moving while throwing sub-network does the throwing. When the character is stationary, the character does a full body throw. Both Use the same base animation. I tidied up the animation so it loops exactly.

6. Walking and running at variable speeds

Subset of locomotion state - done

7. When Evey is big, everything she does is slower. Evey employs the 4 distinct windup / action large monster melee attacks

I have a set of monster attacks which should look good on a scaled up and slowed down player character, so can use that as an animation base.

Done in prototype, to a degree - a chain of states to wind up and punch ground, and also a control put in to slow down the locomotion briefly.

8. Eveys 3 sword attacks

Completed two sword attack states, and setup the control to chain them.

9. Eveys two barehand melee attacks

Completed the states for two barehanded melee attacks - punch and kick. Also provided control and transitions for chaining them together. This also corresponds to Ratchet and Clank (Ie: The second melee is only available when chained quickly after the first). Have tested out how the action feels with Physics additions (punching around dynamic blocks in a prototype gameworld environment).

10. For getting hurt, or hit - a single 'reaction' action takes place

Done. Would be nice to extend with some partial body physics.

UPDATE: Done some reviewing of how to set up the 'soft keyframing' on partial body.

UPDATE: So, hurt is a state containing a single animation. It's an overriding action and switching to the state directly is required to initiate the action.

11. For getting hurt bad - the action is to 'fly' in one of 4 directions based on from where she gets hit (Knockback). She then either gets back up or dies

Included a state for this, and also getting back up. One direction currently.

Dying can be simulated by requesting the ragdoll physics state set up in the network.

There's no setup in there for blending naturally from physics back to animation - but can be set up quite easily.

UPDATE: Included a state to get knocked back in one of four directions. Animation nice and tidy – removed the wiggling feet from the knocked back animation, and knocked back always transitions into getting back up. Knocked back transition links set up pretty fully – so that knockback gels well with other main states – melee, sword, locomotion, jumping.

12. Jumping

Done. Could be refined in terms of true directional blending.

UPDATE: Multi-directional jumping is in. Which ever direction character travelling in, that's the exact jump direction. Also there is a blending of the actual animations happening – but at this time haven't varied them much (only the backwards jump varies in look a bit) Extra control put in which takes the direction of travel at the instigation of the jump, and maintains that direction through the jump. Took the R&C approach here.

UPDATE : Double jumping is out.

13. Die action

Prototyped as physX ragdoll

14. Evey can stand idle and from there transition out to most other states

Idle is a subset of locomotion and has transition network to other states. DONE.

15. The action that covers stunned momentarily (Stunned)

Done – separate state, like hurt. Because its' an overriding state a direct instruction to make the state current is how to initiate that network action. The network plays the animation and returns to locomotion. The character travels backwards 1 meter during this action. The animation is non-looping and the network transitions back to locomotion on finishing the sequence.

16. Her action for being in pain state can be incorporated to locomotion action (Pain)

Not done yet but probably a quick one to incorporate

DONE. Added Pain to locomotion state – consists of 3 animations, a painIdle, pain walk forwards, and pain walk backwards. These blend in at various locations within the locomotion tree. I've added an extra control which limits the locomotion speed to walking when in pain. Controlled with one parameter, float. Looks best when eased in to max value, and eased out to zero (turned off).

17. Combat idle action is a combat pose with some subtle motion

To be completed. Have done some tests in Unity engine on this one and there's a blog page for it.

UPDATE: We don't need this.

18. She can taunt monsters and enemies by gesture (Taunt)

I've put in a state which transitions out to a little dance, taken from capture session. Other taunt animation are available in Unity animation set, and can include them quickly into prototype

19. Panicked state where she runs screaming forwards with arms waving (Panicked)

Done as an inclusion to the locomotion state. Panicked animation blends in to be the source run animation. One float control parameter, blend_panicked, turns on the blend. So, panicked, in terms of network, is a replacement on the running forwards animation.

State machines for Monster additions

20. Eye Blast

A Chain of two states. First state is eyeBlastLeadUp. The request to it is called eyeBlast.

Currently, for the generic network, I've put in minimal transition linkages to and from eyeBlastLeadUp and eyeBlast, to other states across the network. This is for simplicity currently, also there are two other factors that contribute to the final transition graph. One is finalising the states themselves – if the states arrangement is altered – obviously the transition network linkages connecting them has to be altered. This shouldn't need to be changed though. The second factor is – how many states are we going to call 'overriding' states. Currently in the prototype there's ONE 'overriding' state – hurt. The assumption for hurt is for the controlling code to switch to hurt explicitly. If a state is an 'overriding' state, there's no requirement to provide transition links TO it from other states. The code just does a 'hard cut' to it.

The main transition network is dividing itself up into two categories – states which area connected heavily with other states, and states which are connected minimally. All of these new 'monster states' are of the minimal variety, for now. I plan to connect them up more fully – for example – there really should be a transition path from the eyeBlast group of states and neighboring transitions, to, say, the knockback state.

The network will flow then to the eyeBlast state – which is a continuous looping, standing animation. When EyeBlast state receives locomotion request, network returns to locomotion.

The eyeBlast state contains a sequence with markup events for the usual meleeDamage and meleeComplete tracks. meleeDamage events are exactly at the start and end of the sequence.

meleeComplete is just there in case it's useful. Probably the events markups aren't important for eyeBlast i'd say.

[UPDATE] I've done two 'optional enhancements' to eyeBlast. First is the ability for the character to twist body and hence aim the eyeblast. A chain solver node inside eyeBlast state handles this. It takes the same rightGunTarget input parameter as shooting for it's aim vector input, and switching it on is done with control parameter blend_eyeBlastChainSolve.

The second enhancement is allowing the character to move and jump while eyeBlasting (and aiming). This is all a combination of additions that are really the same sort of setup as shooting – so it was easy to pop in and prototype without doing anything 'new'. I made a video demo of this, it's in the network as additions to locomotion and jump states, and to turn it on, one control parameter blend_eyeBlastLocomotion.

21. Bite

Bite is similar setup to the above (without the extra partial body blending additions). It consists of two states. First state is biteLeadUp. Network will then flow to bite, a continuous looping forwards moving animation. Likewise with above, when bite state receives locomotion request, network returns to locomotion.

Bite state markup events also include meleeComplete and meleeDamage. meleeDamage is a series of user type events 0 1 0 1, marking two short intervals (two bites).

22. Area Attack

Again a simple two-state action. Request 'areaAttack' from locomotion sends the current state to areaAttackLeadUp. From there the network flows to areaAttack, the sequence plays once and network returns to locomotion.

AreaAttack state has event markups for meleeDamage and meleeComplete.

23. Spit

In all respects apart from the motion, spit is the same as Area Attack above. Request 'spit' from locomotion sends the network to the lead up, then will transition straight into the action, and back to locomotion.

Spit state has markups for meleeDamage and meleeComplete.

24. Monster Ball Roll

A simple 3-state chain of states does the monster ball roll. MonsterBallStart, monsterBallMiddle and monsterBallEnd. A request (while in locomotion) for monsterBall , transitions to monsterBallStart, then network travels to monsterBallMiddle. monsterBallMiddle, for the prototype, contains a single animation of a looping rolling monster. The trajectory travels forwards during the sequence, but the trajectory doesn't actually rotate. The rotation is done more internally to the character data (at the Hips node). This provides easy transitioning basically. In prototype, monster rolls forwards indefinitely until the request monsterBallEnd is broadcast. Network then transitions (aesthetically nicely) to monsterBallEnd state, then back to locomotion. There are some extra transition links from these states (and their close transition links), to the 'reaction' state (knockback). Did a video demo showing how things look running into a wall (or close to it), ending the monster roll, and transitioning quickly to the knocked backwards reaction.

The monsterBallMiddle state has sequence with events meleeDamage and meleeComplete.

25. Melee Claw and Melee Claw Mocap (Adding 'responsive' types of melee)

The generic network has a nice melee framework designed for responsiveness for PC's. meleeClaw and meleeClawMocap attacks go nicely into that framework. AI characters should be responsive too.

So, to drive these actions in the network, just do the same as if 'doing a melee'. melee1 state is made use of here – there's no need for touching melee2 (melee1 defaults to transitioning back to locomotion on completion).

The request for melee1 state is 'melee1'.

I've added a 'switchWithEvents' node inside the melee1 state to make available the appropriate animations for these monster melee actions. SwitchWithEvents node takes a weight parameter as input. The weight parameter input is called switch_melee. Since there are three variations currently (punch, from generic), meleeClawMocap, and meleeClaw, the respective values for switch_melee are 0, 0.5 or 1.

So, if you want to meleeClawMocap, set switch_melee to 0.5, and request melee1. Note that melee1 state has very good transition network to other main states (locomotion, jumping, knockback ..)

meleeClaw and meleeClawMocap sequences have events markups for meleeDamage and meleeComplete. meleeClawMocap has two intervals of damage (he strikes twice in the sequence).

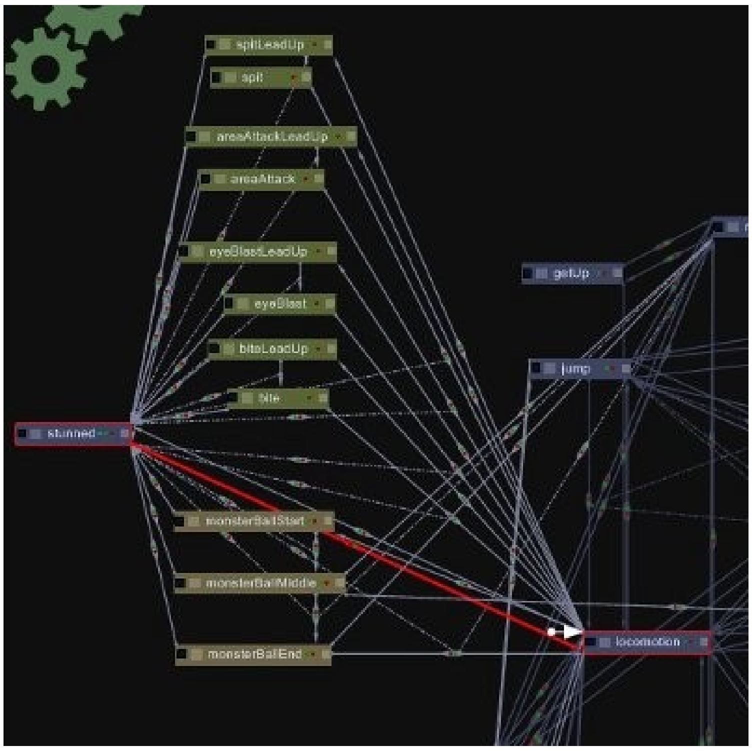
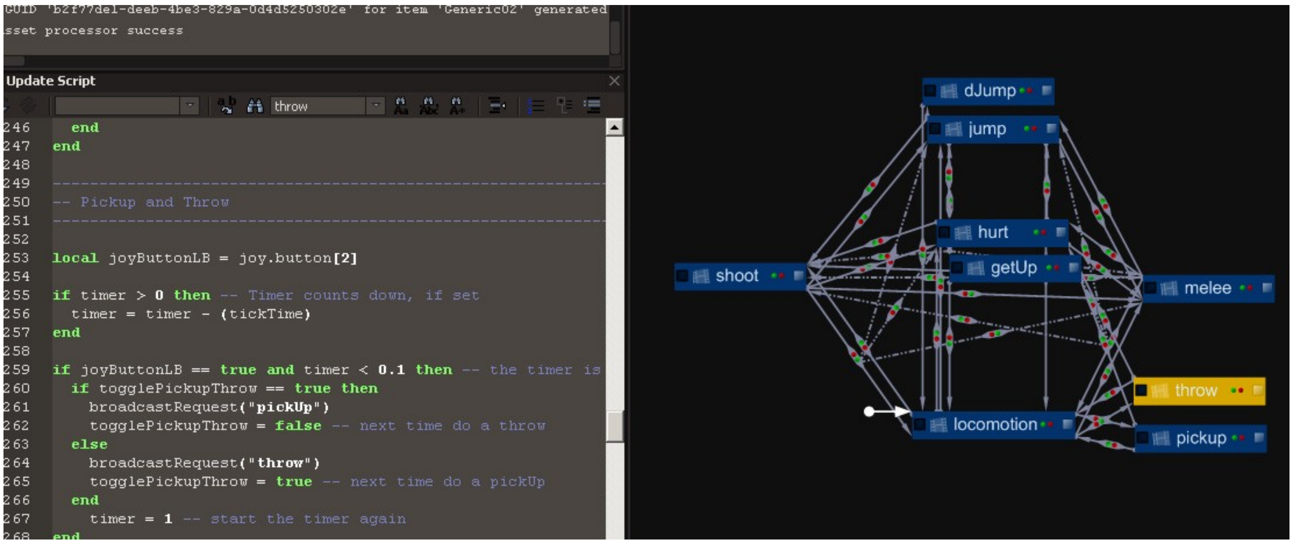


Illustration 3: Monster attack additions

SECTION 4
'Is the character responsive enough for the situation it is to be used in'?

Here's a screengrab of the top level of the prototype network (Early stage, end of week 2 prototype work). Here is the transition network between the main states. Each arrow is a possible transition in the network, and each transition has an individual set of parameters, the most important being the transition time, together with the basic layout of the connections themselves (which go off the design in the above sections). When prototyping though - the transition network can be refined. Some of these transitions are transitions from transitions. For example, say the character has just meleed, and is currently transitioning back to locomotion state, but then the player presses the shoot button, or the character just gets shot. There are two arrows branching out not from the states, but from the transition arrow itself, to cover these events. One of these arrows goes to the shoot state, and another to the hurt state.

Some of these arrows are request type transitions, and some are event type transitions. Events are authored into the animation data as markup events, within connect. Say - a kick melee - has a user event near the end of the sequence (I use the number 9 to indicate the point where animations should start to transition out). So each animation can have a custom time at which it looks good transitioning out - and I put in this for every animation.


```
GUID 'b2f77de1-deeb-4be3-829a-0d4d5250302e' for item 'Generic02' generated
sset processor success
```

Update Script

```
  end
end

------------------------------------------------------------------------
-- Pickup and Throw
------------------------------------------------------------------------

local joyButtonLB = joy.button[2]

if timer > 0 then -- Timer counts down, if set
  timer = timer - (tickTime)
end

if joyButtonLB == true and timer < 0.1 then -- the timer is
  if togglePickupThrow == true then
    broadcastRequest("pickUp")
    togglePickupThrow = false -- next time do a throw
  else
    broadcastRequest("throw")
    togglePickupThrow = true -- next time do a pickUp
  end
    timer = 1 -- start the timer again
end
```

Illustration 4: Transition Network

The request type transitions look for a broadcast request, and these are handled in the Lua update script, and they in turn are usually direct results of controller input, or some timer or toggle. All processing on the controller input is also done in the update script - there are no unknown factors from input to resulting motion on the character.

Here's the state network at end of week 3. A characters responsiveness is determined by the connection links between states. Example, the melee group and and sword attack group to the right of the network .. The links between them allow quick chaining between melee and sword attack. It might look a bit messy and rough but each connection has a definite purpose, and each has a specific transition duration. Slight adjustments to layout and timing of the transitions have a great effect of the playability of the character. The more the prototype is playtested, the web between various states grows and the mechanical action of the character becomes more fluid.

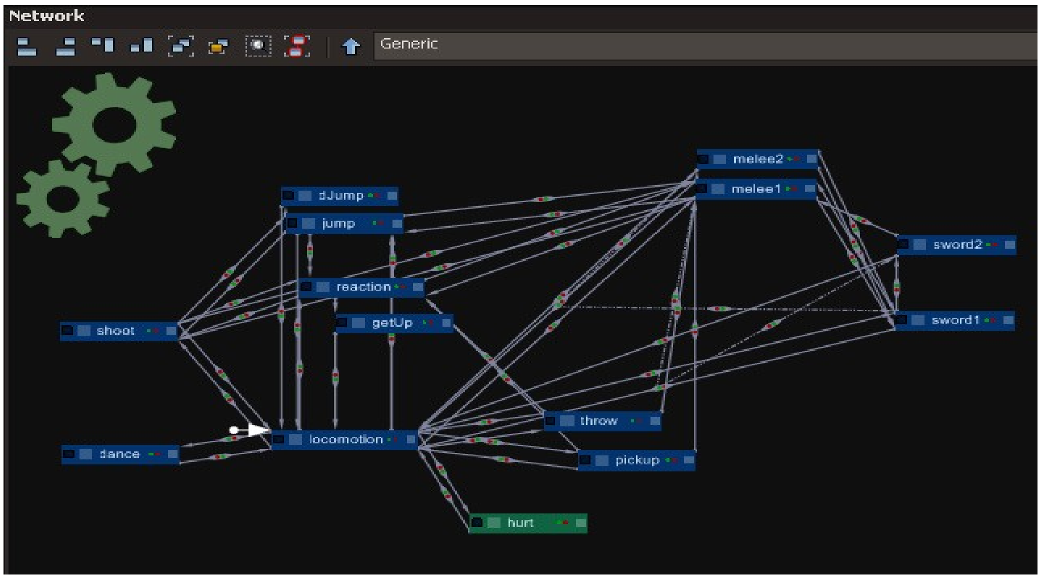


Illustration 5: Transition Network Growing

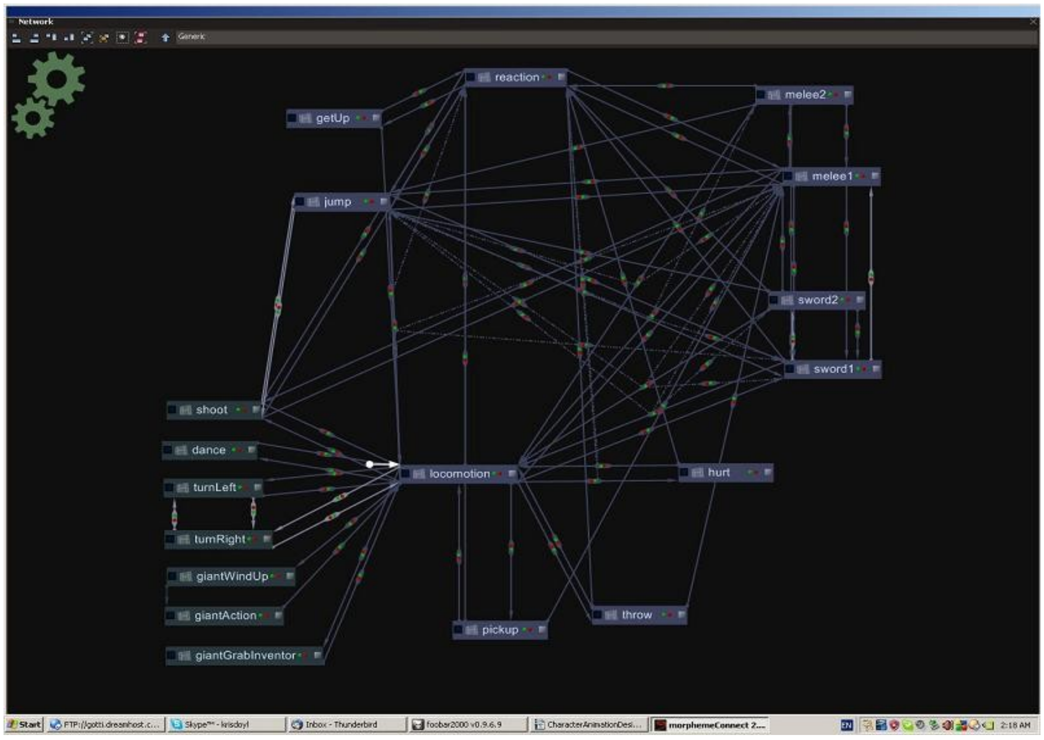


Illustration 6: Transition network filling out .. more connectivity for melee, sword, locomotion, reaction, jumping.

SECTION 5
What animations are needed?

Currently, at the end of the third week of working on this prototype network, I've taken 34 animation sequences from our Skunkwerks set, and authored trajectory root motion for each. Some of these sequences have animation editing applied over what's currently in game. The animations that the prototype is using so are listed in morpheme's attribute editor, and listed below in the morpheme screenshot.

These represent most of the core animation - idle, walking, running, jumping, basic melee, swordfight, hurt, shooting, and just a sample of the large character group and a touch of the windup / action attack animations. Roughly speaking there are probably a few more weeks worth of animations in the current set that I could 'feed off', building up the prototype a little more fully into the 20 or so networks that I've identified in section 3.

[UPDATE] .. After completion of generic, and the extensions to generic to include requirements from the monster AI doc, there are 54 animation sequences that the generic network sources, not including the rig and Tpose entries.

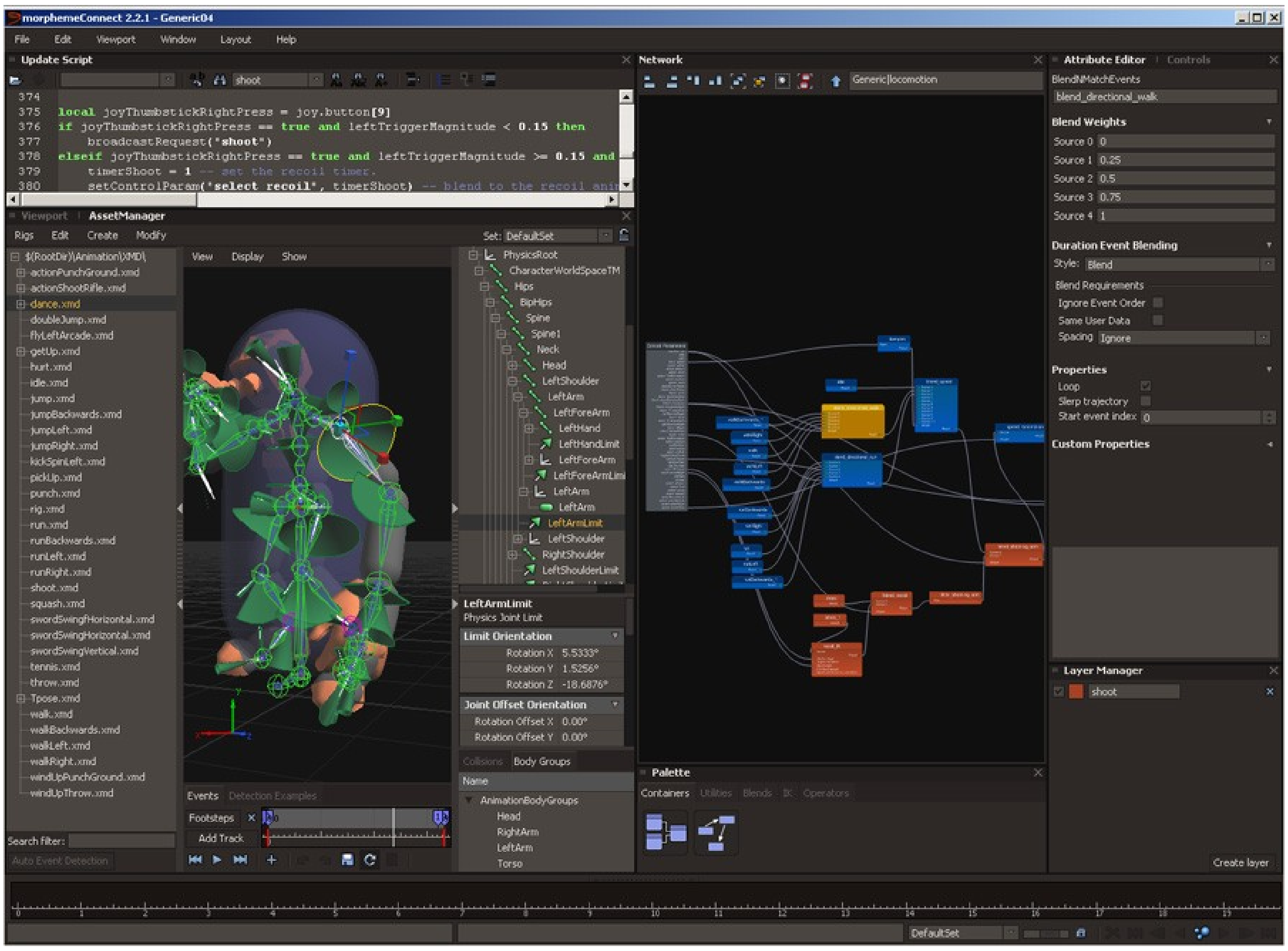
Tpose.xmd
actionAreaAttack.xmd
actionBite.xmd
actionEyeBlast.xmd
actionPunchGround.xmd
actionShootRifle.xmd
actionSpit.xmd
dance.xmd
doubleJump.xmd
flyBackArcade.xmd
flyForwardsArcade.xmd
flyLeftArcade.xmd
flyRightArcade.xmd
getUp.xmd
giantGrabInventor.xmd
hurt.xmd
idle.xmd
idlePain.xmd
jump.xmd
jumpBackwards.xmd

jumpLeft.xmd
jumpRight.xmd
kickSpinLeft.xmd
meleeClaw.xmd
meleeClawMocap.xmd
pickUp.xmd
punch.xmd
rig.xmd
run.xmd
runBackwards.xmd
runLeft.xmd
runRight.xmd
runScreaming.xmd
shoot.xmd
specialMonsterBallEnd.xmd
specialMonsterBallMiddle.xmd
specialMonsterBallStart.xmd
stunned.xmd
swordSwingHorizontal.xmd
swordSwingVertical.xmd
tennis.xmd
throw.xmd
turnLeft.xmd
turnRight.xmd
walk.xmd
walkBackwards.xmd
walkBackwardsPain.xmd
walkLeft.xmd
walkPain.xmd
walkRight.xmd
wield.xmd
windUpAreaAttack.xmd
windUpBite.xmd
windUpEyeBlast.xmd
windUpPunchGround.xmd
windUpSpit.xmd

Appendix

1. Morpheme PhysX Rig

Inside morpheme:connect, a duplicate rig is created which carries the collision bodies and joint limits, which PhysX uses to simulate the character is a physical world. The collision bodies are geometry shapes like capsules, which roughly approximate the mesh. The joint limits define the swing and twist angle maximums, for each joint - and there are other per-joint attributes here as well - Spring, Damping, Restitution.



```
local joyThumbstickRightPress = joy.button[9]
if joyThumbstickRightPress == true and leftTriggerMagnitude < 0.15 then
    broadcastRequest("shoot")
elseif joyThumbstickRightPress == true and leftTriggerMagnitude >= 0.15 and
    timerShoot = 1 -- set the recoil timer.
    setControlParam("select recoil", timerShoot) -- blend to the recoil ani
```

BlendNMatchEvents
blend_directional_walk

Blend Weights

Source 0	0
Source 1	0.25
Source 2	0.5
Source 3	0.75
Source 4	1

Duration Event Blending
Style: Blend
Spacing: Ignore
Start event index: 0

LeftArmLimit
Physics Joint Limit

Limit Orientation

Rotation X	5.5333°
Rotation Y	1.5256°
Rotation Z	-18.6876°

Joint Offset Orientation

Rotation Offset X	0.00°
Rotation Offset Y	0.00°

Illustration 7: Morpheme Screenshot