
Motion Matching in
THE LAST OF US
PART II
Michal Mach
Maksym Zhuravlov
NAUGHTY DOG

Michal Mach
Lead Animator
@gameplayAnim

Maksym Zhuravlov
Principal Animator
@MaksZhuravlov

NAUGHTY DOG

What is Motion Matching? Quick Recap

System was first presented to the gaming industry by:

- Simon Clavet (GDC 2016) – “Motion Matching, Road to next-gen animation”
- Kristjan Zadziuk (GDC 2016) – “Motion Matching, The Future of Games Animation… Today”
- Michael Buttner (Nucl.ai 2015) – “Motion Matching - The Road to Next Gen Animation”

In general, it works as follows:

- A piece of code we call Motion Model provides desired trajectory based on move stick extrapolation or AI pathing
- Every frame - all animations in the set and each single frame in them are being searched to find the best match in terms of pose and trajectory
- The frame of animation that matched closest is picked and gets played on the character
- Procedural adjustments applied
- Next frame - everything repeats again from step 1.

Motivations

It was a bit step for us. Leaving the proven system behind and jumping straight into this novel technology that hasn't been really proven in a shipped game yet.

- We needed a higher quality and more natural looking locomotion
- Handling a massive scope at high quality
- Locomotion system stability
- It was easy to try

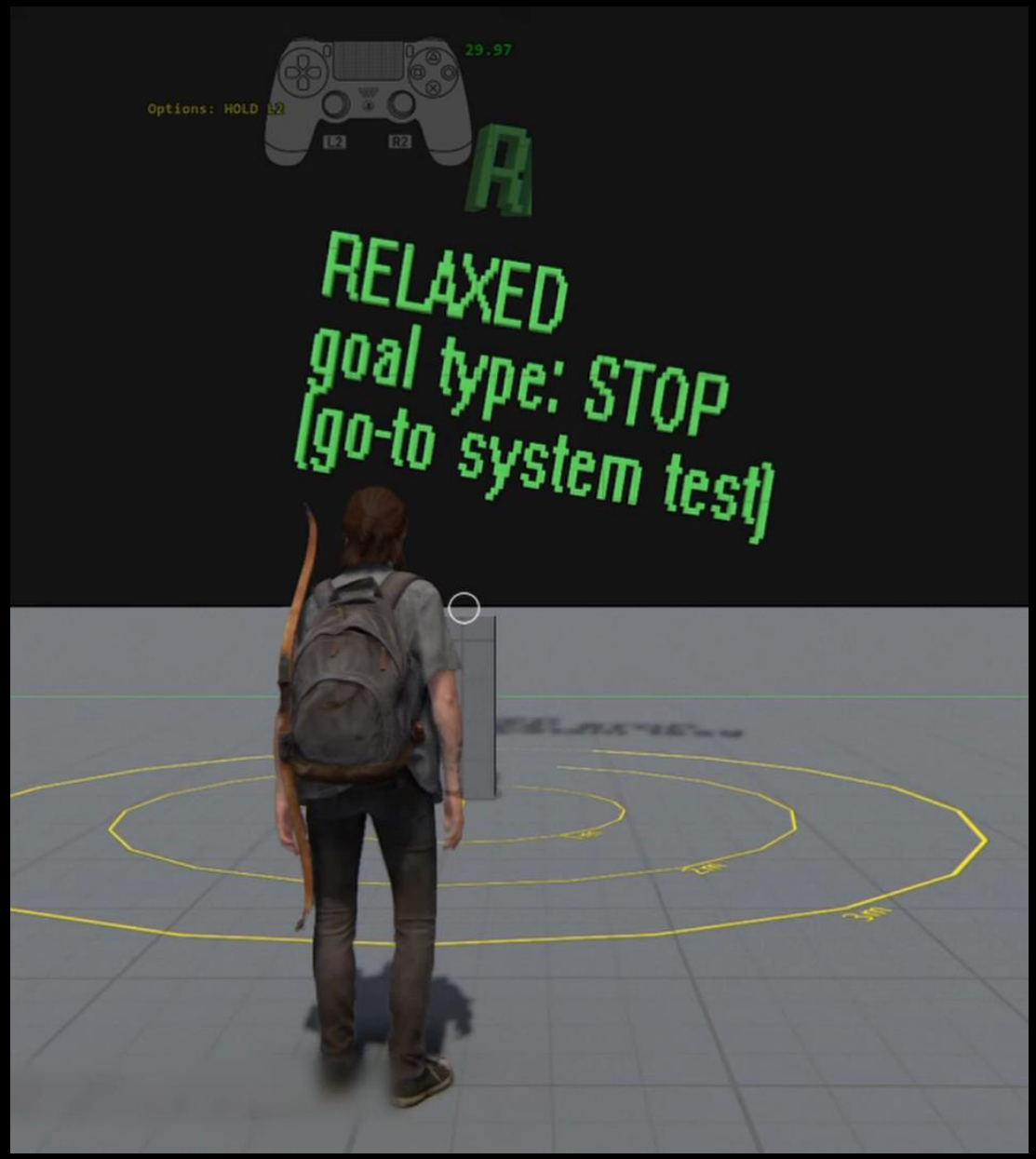
Initial Implementation Problems

- Animation playback continuity
  - Slightly unstable animation selection for authored angles
- Animations "confusion"

  Many animations in the set share segments that have very similar trajectory and poses but serve a very different purpose
- Motion Matching is not user friendly

  It works with information that is normally hidden to animators
- Transitions handover is bad.

  Transitions tend to either "stuck" or get completely "swallowed" depending on settings



Initial Implementation

Woah! It works and looks amazing!

Sort of...

Our take on Motion Matching

Root Motion Node

The core node of the character that drives the capsule and root motion.

In our engine it's called Align.

All joints are animated and evaluated in Align space, therefore its accuracy is critical!

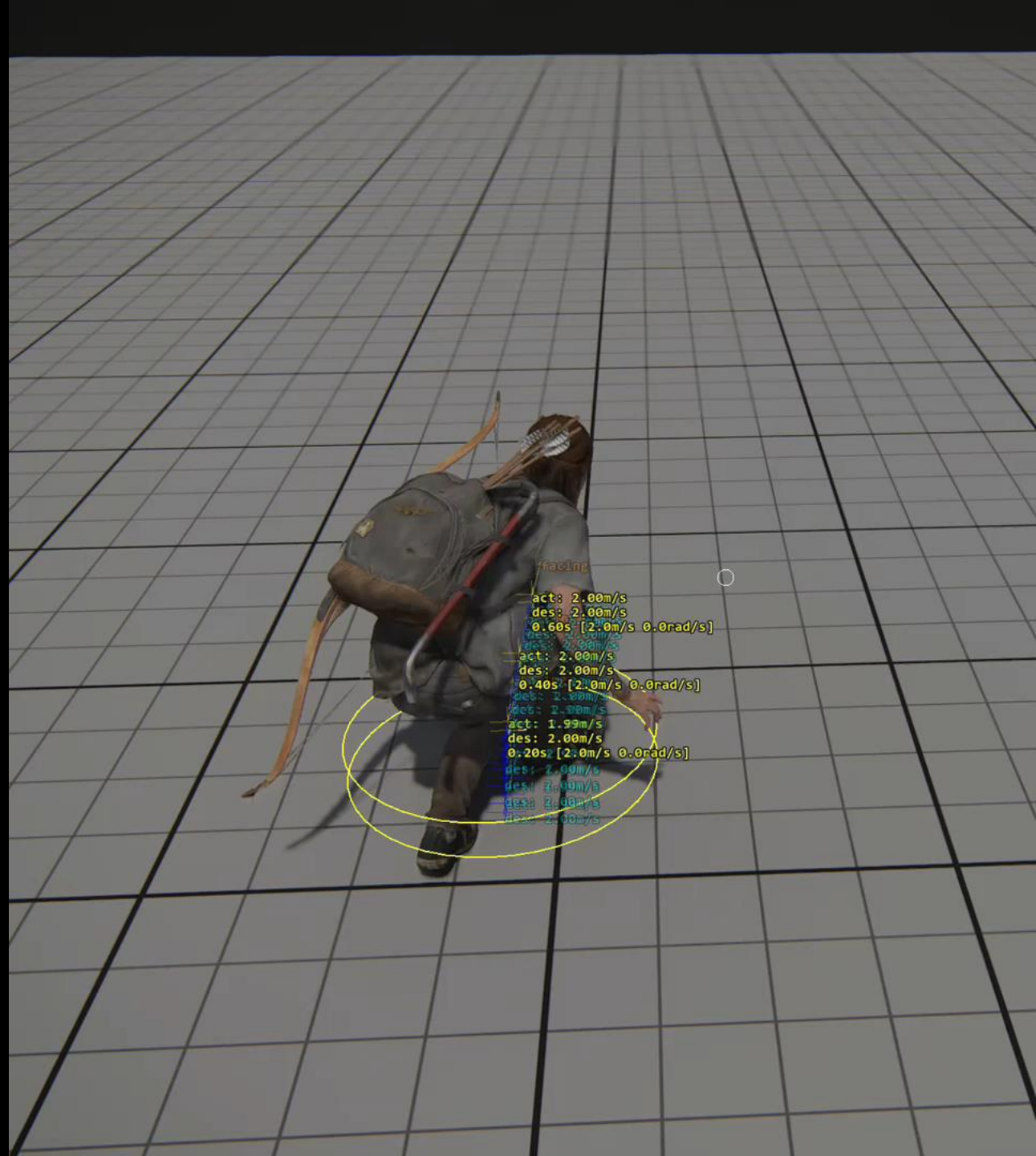
We hand animate it for every animation clip:

- Curves must be smooth (no weird tangents)
- Unnecessary lateral motion removed
- Align rotation animated in compliance to Motion Model setup
- Idle pose pasted where needed with exact align \ pose match

Motion Model

Represents how the character's Align should move in the space and is our simulation of character motion

We use a critically damped spring to simulate position and velocity and we also simulate turn speed and facing direction

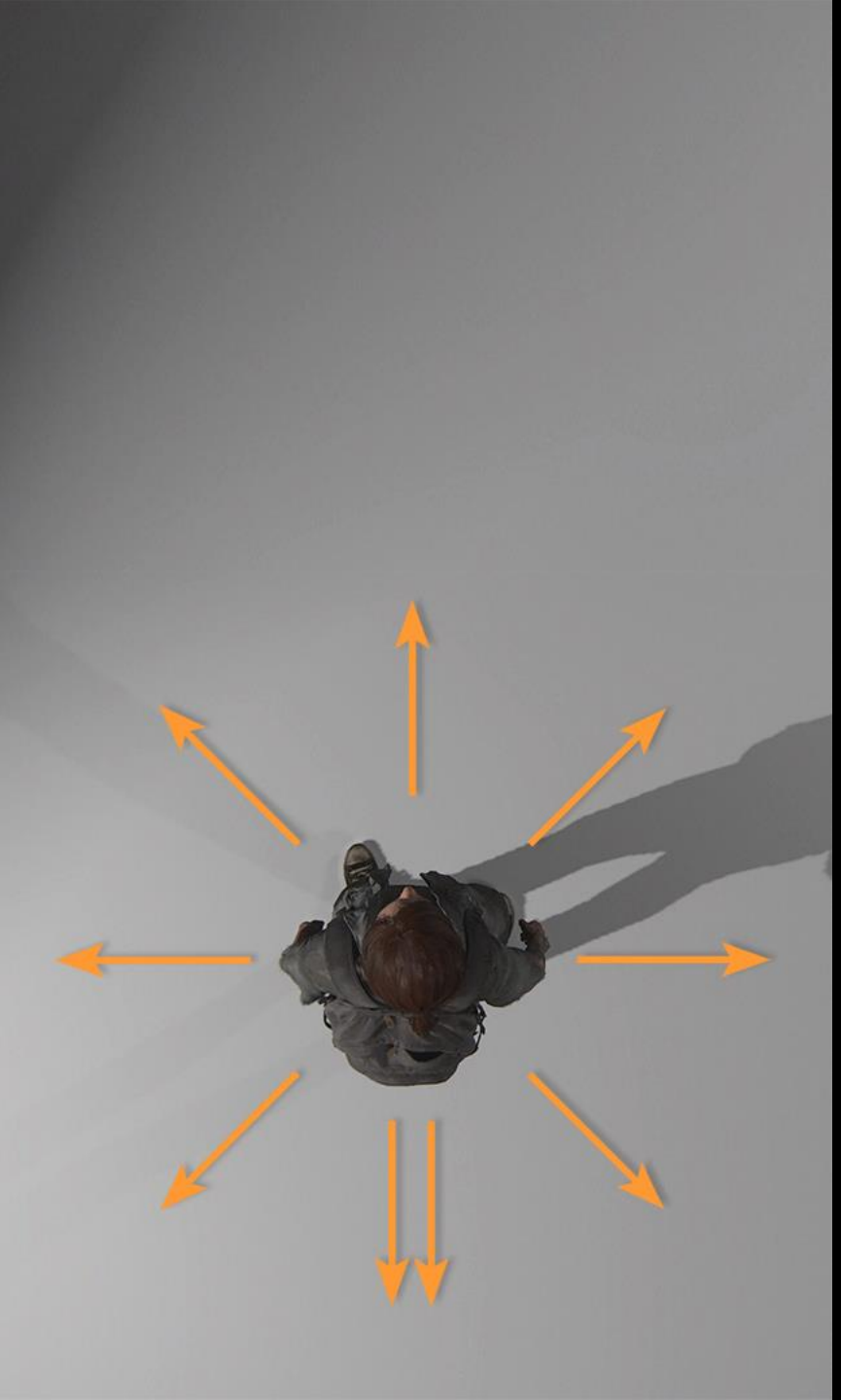


Motion Models

- Stick Input Mode: Strafing, Player
- Stick Input Mode: Non-strafing, Player
- Stick Input Mode: Cover, Player
- Horse Motion Mode
- Path Following Mode: NPCs
- Path Transition Mode: NPCs

Root Motion Allowance

- 0.3 – 0.6 m for AI
- None for Player

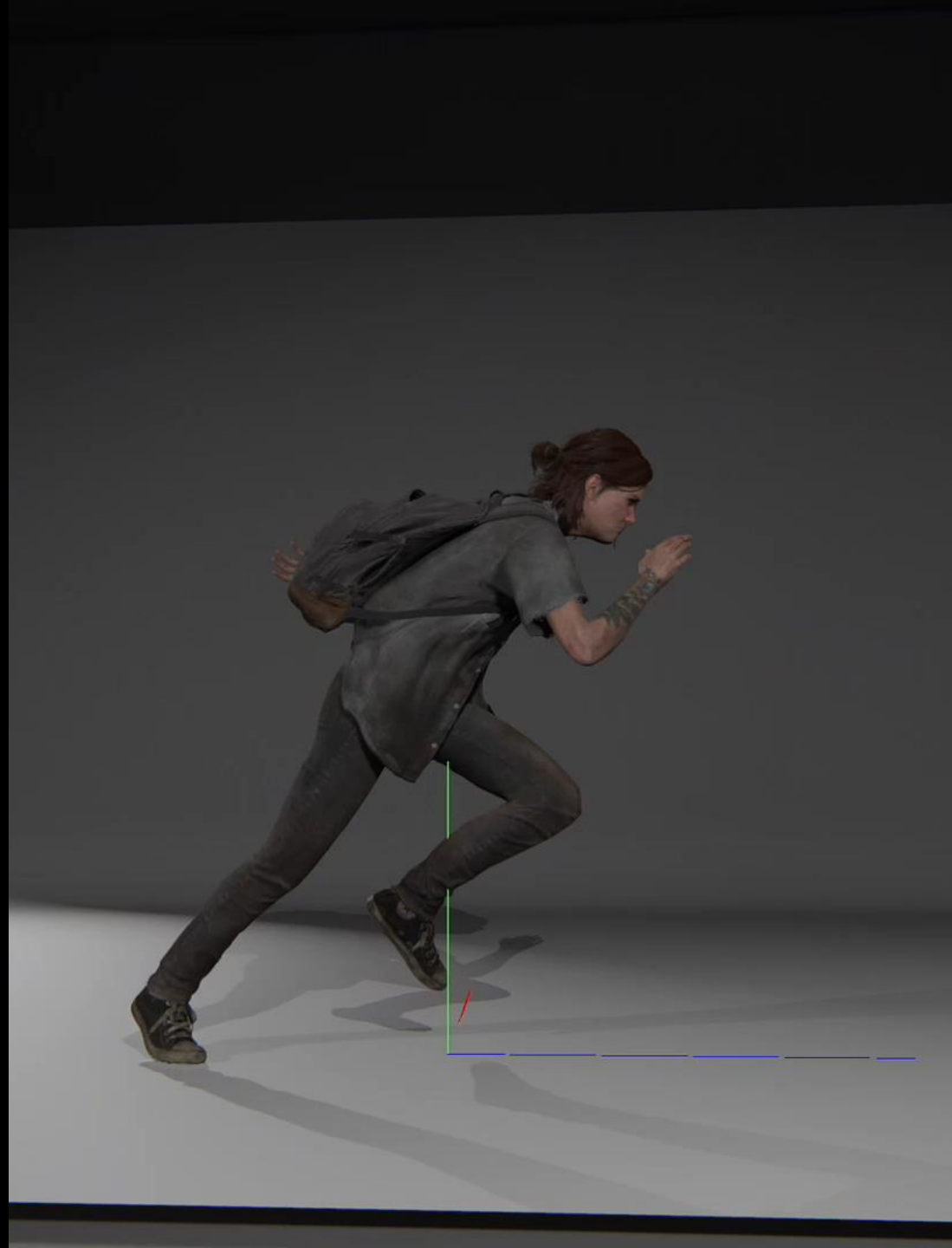


Motion Model Parameters

- Most settings can be defined as directional values
- Velocity spring constant
  - Acceleration spring
  - Deceleration spring
- Maximum speed (m/s)
- Turn rate (degrees per second)
- Translation clamp distance (root motion allowance)

Motion Matching Data Samples

- On average our individual sample vectors have around 100 floats in them
- We match only data with significant impact on the motion.
- Unnecessary joints or trajectory samples is wasted memory and performance



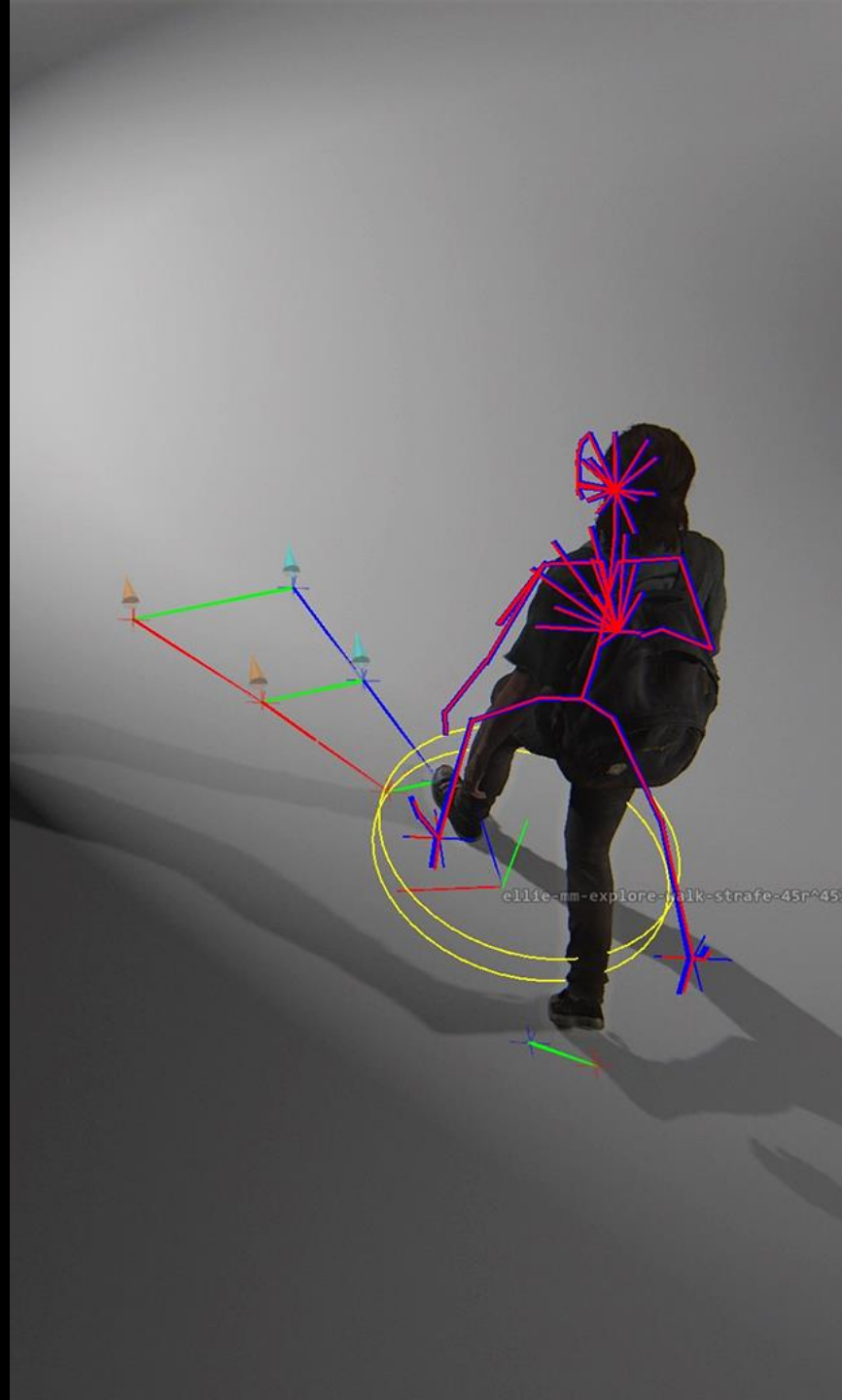
Motion Matching Trajectory Sample

Enough trajectory points to capture a simple shape

- 3 samples up to 1.8s in the future
- 1 sample up to 0.7s in the past

Sample Data

- Align Position
- Align Velocity
- Align Direction
- Align Turn Speed

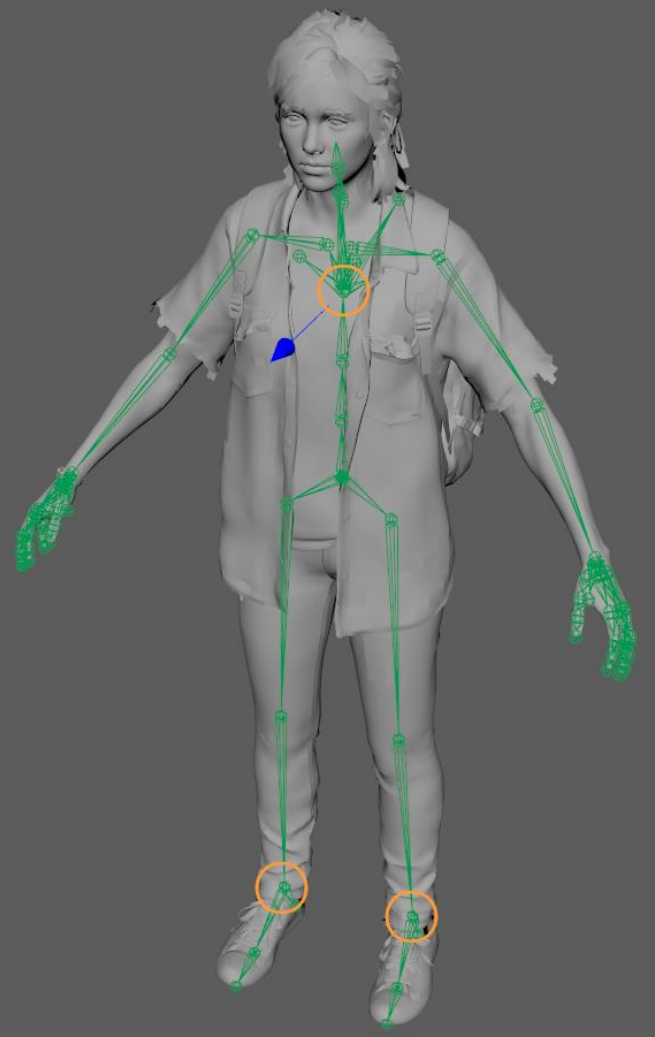


Motion Matching Pose Sample

- Ankles to keep a continuous gait cycle
- Spine to capture the body weight shifting and posture height
- Bipeds – both ankles, top spine
- Quadrupeds – all four feet, one spine joint

Sample Data

- Joint Position
- Joint Velocity
- Facing Axis – typically a top spine joint forward axis



Animation Data

We don’t mocap unstructured data or long dance cards.

- Directing and performing long takes is uncomfortable
- A lot of redundant data
- Animation polish and editing is inconvenient on very long takes

Instead, we direct medium to short takes depending on complexity of locomotion and how well the actor is managing performance / volume of this set

Player Specific Animation Data

Premise:

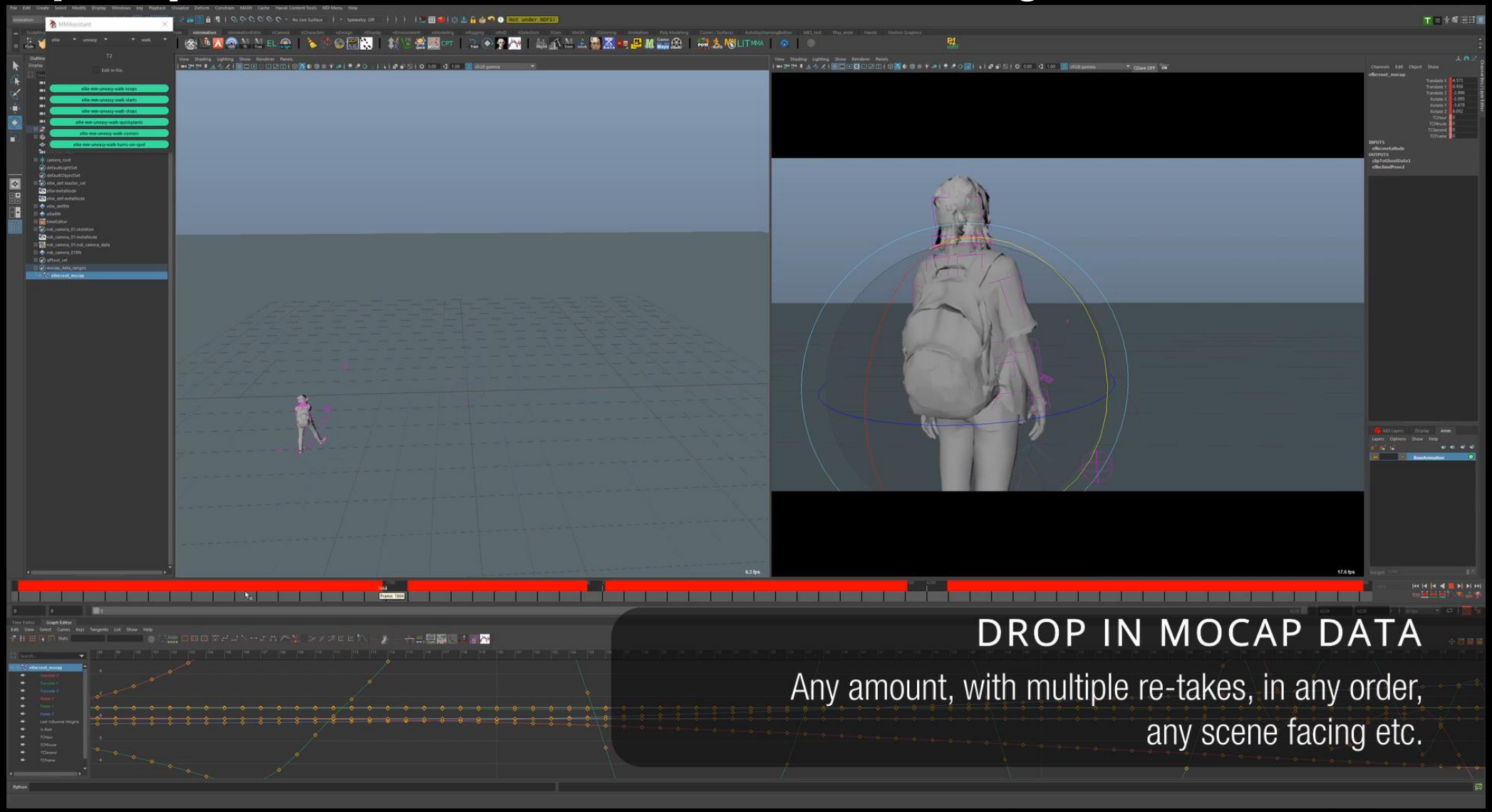
- Animations shouldn't “shred” on straight inputs
- Transitions should play out in full on straight inputs
- Animations will be heavily polished in post and can deviate from raw MoCap
- Any director’s note should be straight forward to address
- Animation subsets should be easy to reuse across similar locomotions
- Complete freedom to direct MoCap sessions as see fit

Player Specific Animation Data

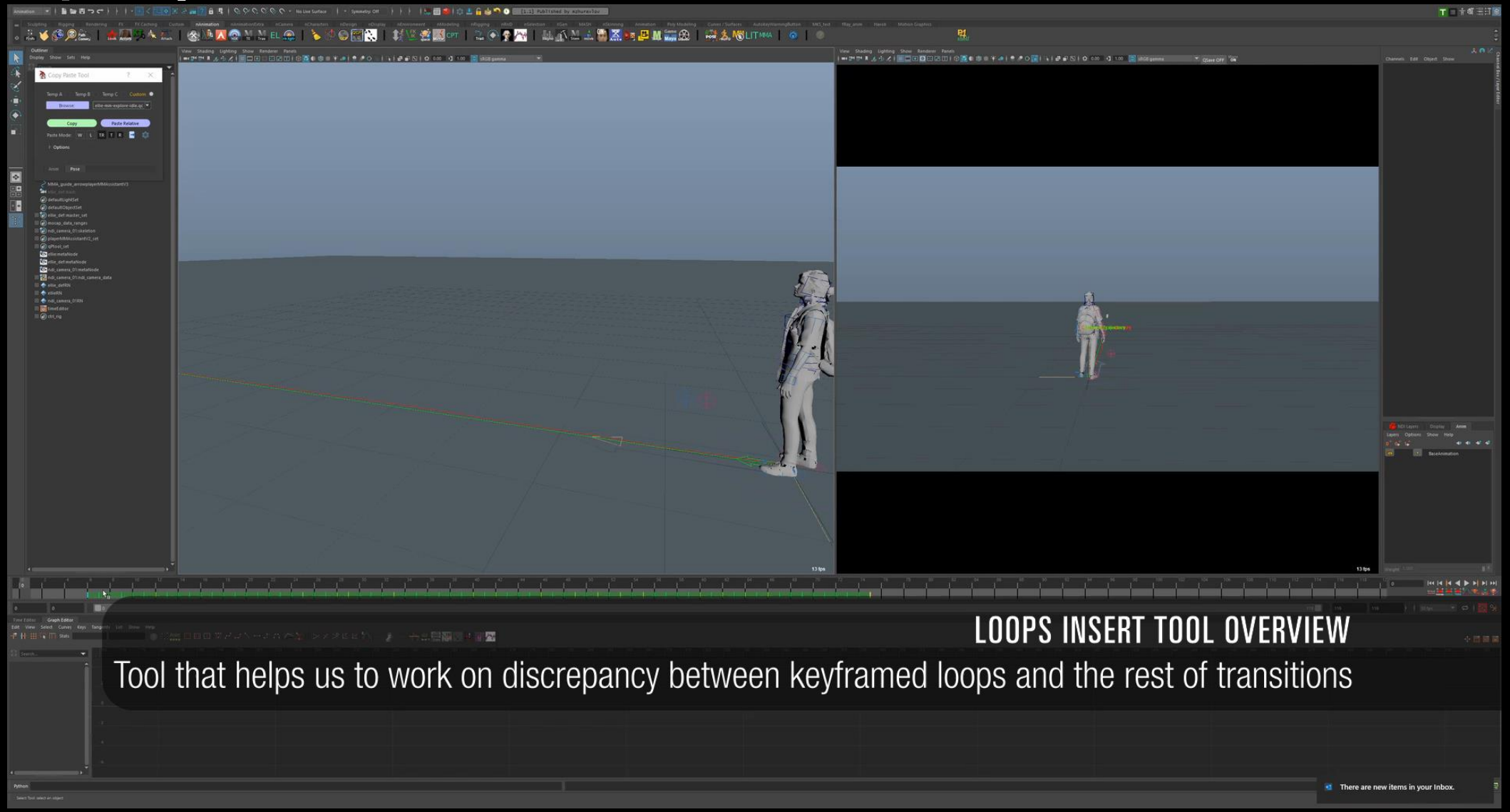
Early discoveries:

- Minimum-enough coverage is good for the player mm system and artist both
- Animation “heads” and “tails” should be of specific controlled length
- It is still vital to bracket animations with core poses
- Sets polish and editing is greatly more stable and comfortable when transitions are split per file and named according to the convention

Player Specific Animation Data (Set Authoring)



Player Specific Animation Data (Polish)



Player Locomotion Total

2627 animation clips

2 hours and 17 minutes of animation data

*Ellie and Abby Players combined

NAUGHTY DOG

NPC Specific Animation Data

Challenges:

- Huge cast of characters of different sizes, genders, species, using different demeanors
- We wanted the NPC locomotion to look as natural as possible with proper weight
- NPCs need to be able to keep up with player

Image courtesy of @IconicNephilim

NPC Specific Animation Data

Premise:

General playback continuity requirements are same as player's with addition of

- No foot sliding (or very minimal)
- Allow AI to maneuver with minimal restrictions
- Be able to navigate the space with precision
- Be able to enter cinematic sequences seamlessly

NPC Specific Animation Data

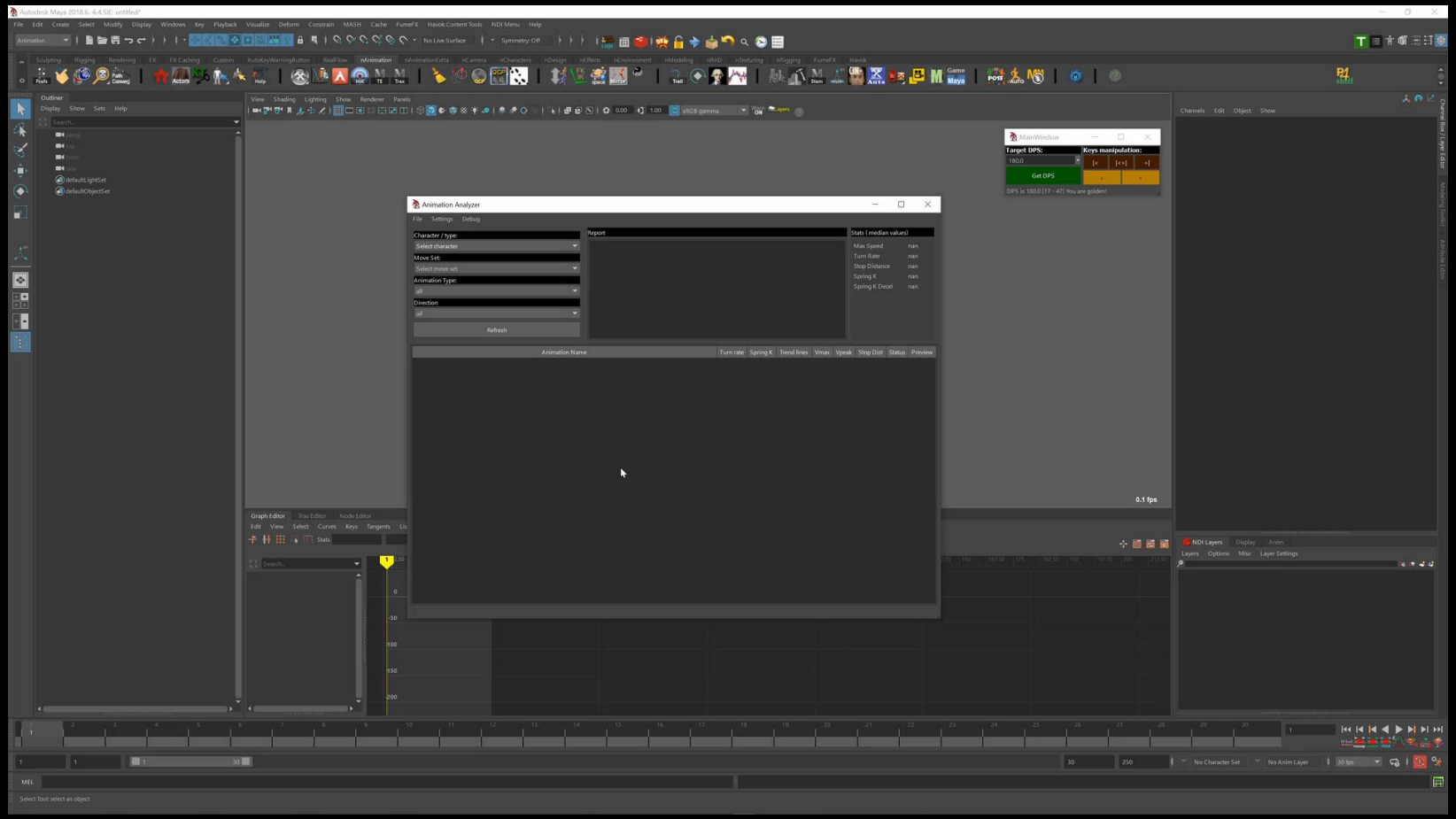
- Human NPCs share move sets to save time and memory
  - Male and female enemies use the same androgenous move sets
  - Combat move sets are shared across the board
  - Buddies have male and female specific relaxed move sets
- We've built the initial move sets using motions we were expecting the characters to do
- It's easy to add more moves later and fill in the gaps
- All moves have left foot and right foot variants
- We did a lot of post processing but not as much as player team did
  - Straightening trajectories, pasting poses relative to the Align, we didn't insert loops
  - The Maksym's tools were not available to us at the time but we're adopting them

NPC Specific Animation Data

- We can mirror animations and technically double the dataset
- Data consistency is key to Motion Matching
- We've semi-automated generating the Align trajectory
  - It still needs to be touched up because Maya's auto tangents cause velocity spikes
- In order to preserve the natural motion, we needed to marry the animation data and motion model

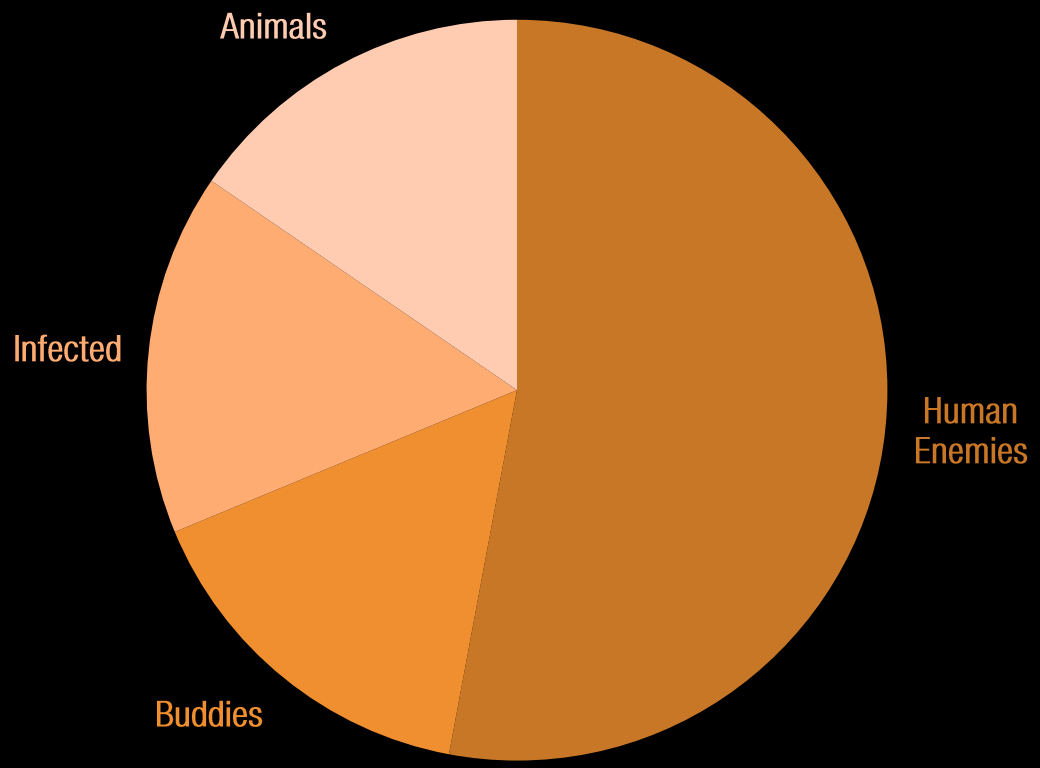
But we didn't even know what data we have…

Animation Analyzer (full screen video)



NPC Locomotion Statistics

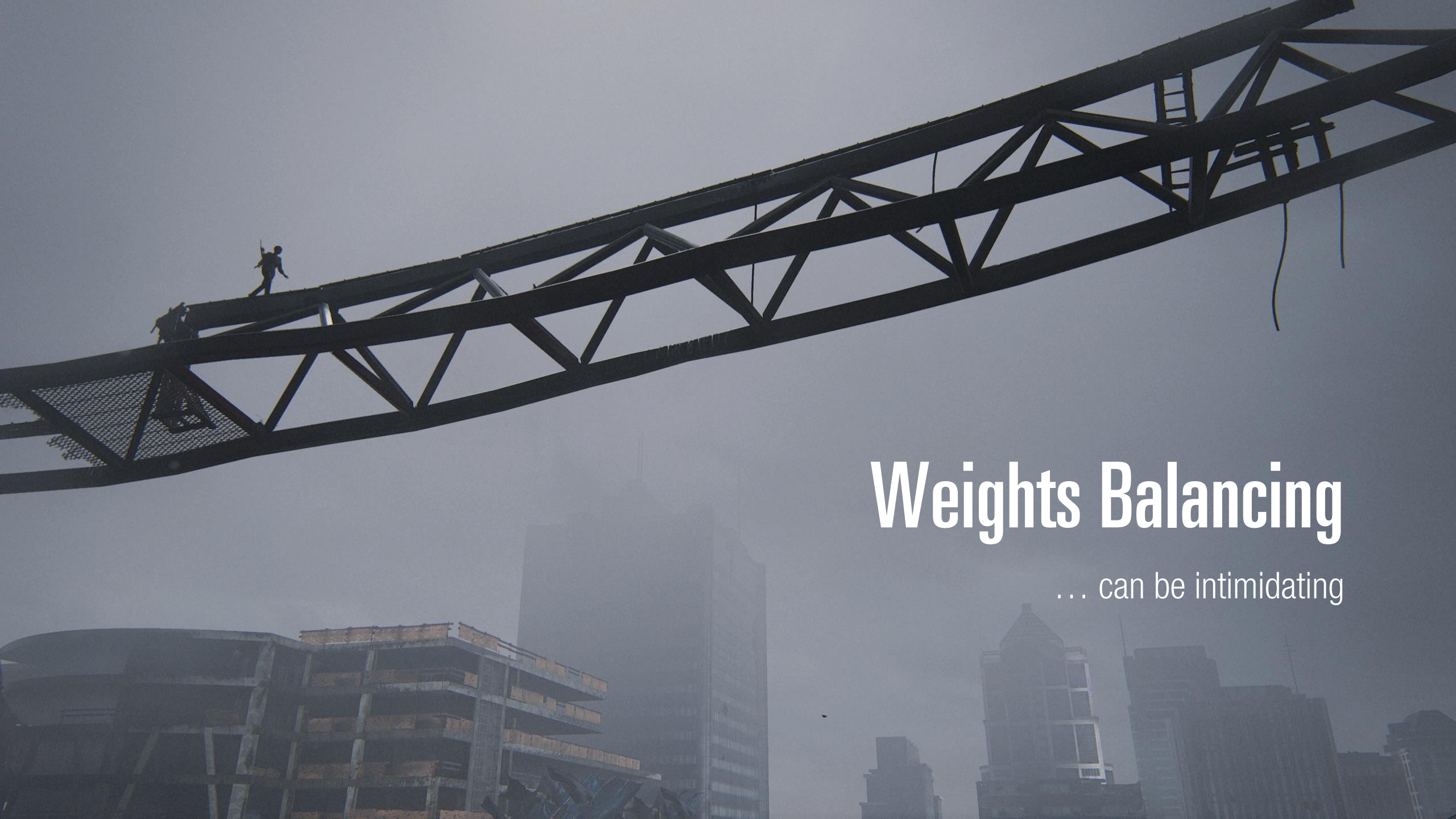
- Human Enemies
  - 2387 animation clips
  - 2 hours and 24 minutes
- Buddies
  - 678 animation clips
  - 43 minutes
- Infected
  - 768 animation clips
  - 43 minutes
- Animals
  - 431 animation clips
  - 42 minutes



NPC Locomotion Total

6050 animation clips

6 hours and 34 minutes of animation data



Weights Balancing

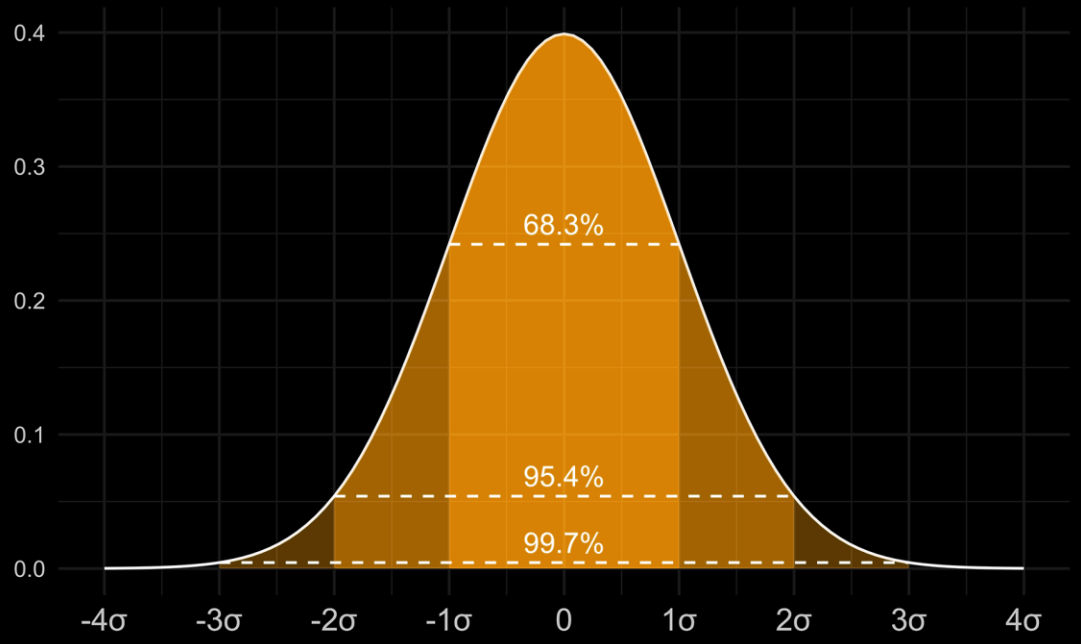
… can be intimidating

Weights Balancing

- Balancing weights can be an intimidating process
  - Too many numbers
  - The numbers change when the data change
- We group individual weights into Pose or Goals groups
- Master multiplier for each group
- Having a structure allows for faster iteration

Data Normalization

- We normalize our data ranges
- Let's us compare features with significantly different value ranges
- Gives us a normal distribution where $\sigma = 1$
- Typically, velocity values are much larger than positional values
- Much easier to understand when comparing costs



Grouping & Biasing

In general Motion Matching struggles with differentiating moves of a similar trajectory.

To assist it we introduced ways to help the system differentiate animations within the set.

- Natural Bias
- Core Loops Group
- Moving Group
- Custom User Layers

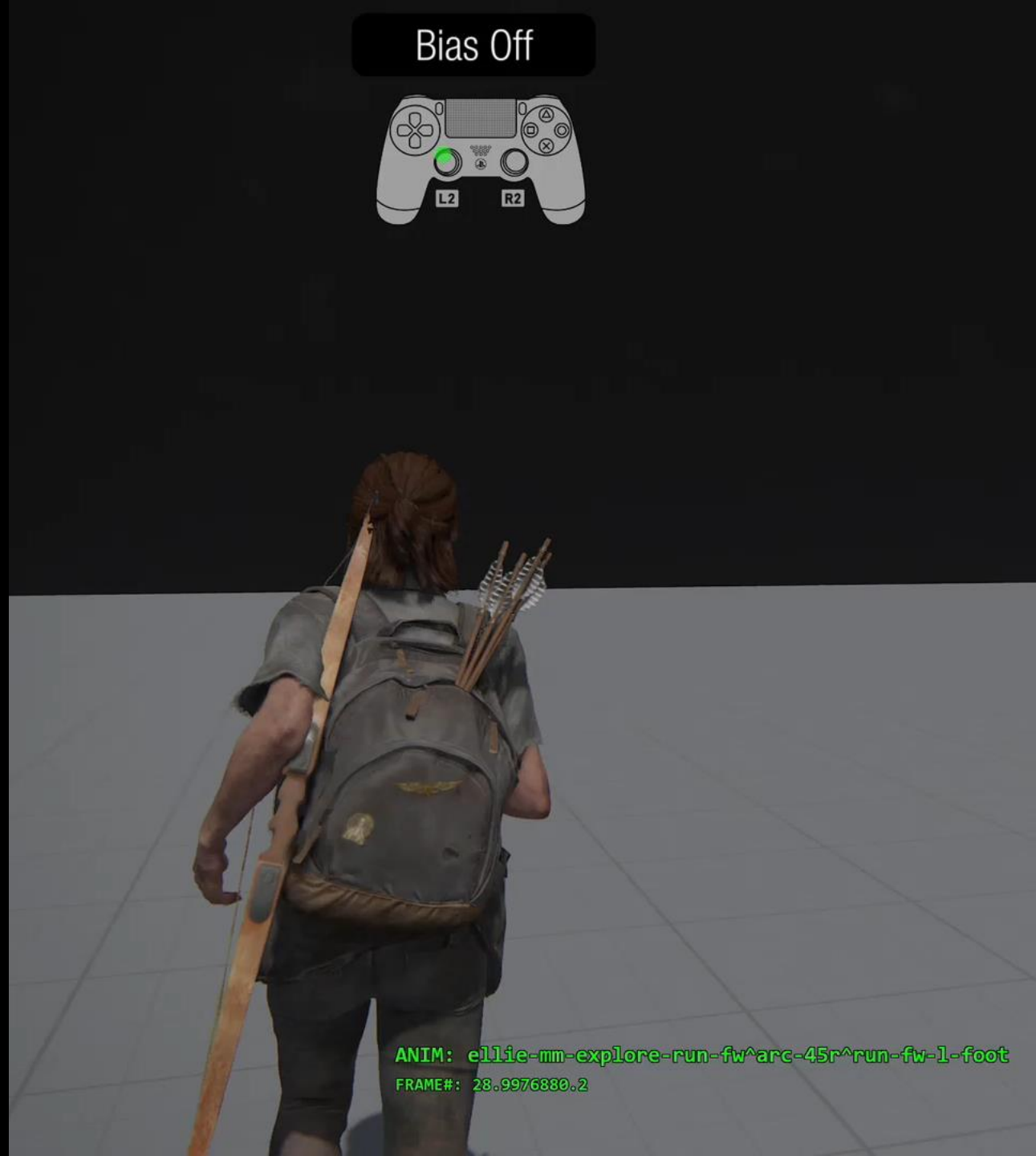
Natural Bias

There's a high chance we want to keep playing what we're currently playing

- Biases currently playing animation and frame
- Helps reduce animation skipping
- Cannot be too high otherwise animations feel too "sticky"

Core Loops Group

- A list of animations that we prefer to play when input is steady
  - Loops
  - Idle
- Prevents character from getting stuck in a stopping animation
- Prevents system from stitching pieces of animation together for off angles (non-45 degrees)

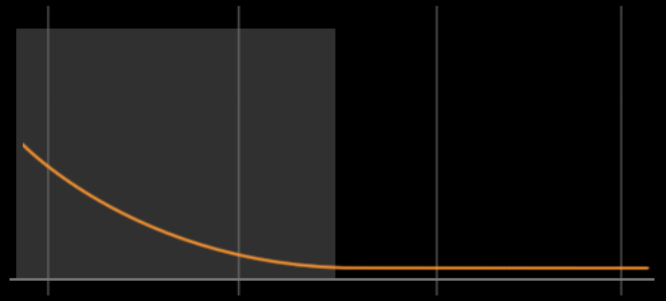


Moving Group

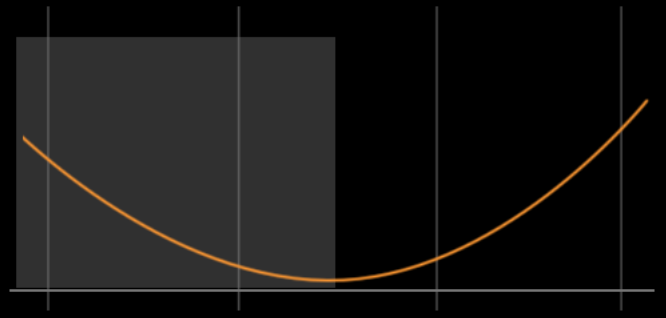
Our system kept confusing stopping and direction changing animations because piece of their trajectory is very similar.

- Added a list of animations where character “wants to be (keep) moving”
- Other animations are in the default group
- We switch between groups
  - Player - based on Motion Model speed that is further sprung to filter out temporal speed changes
  - NPCs – higher logic based on locomotion state

Stopping Animation

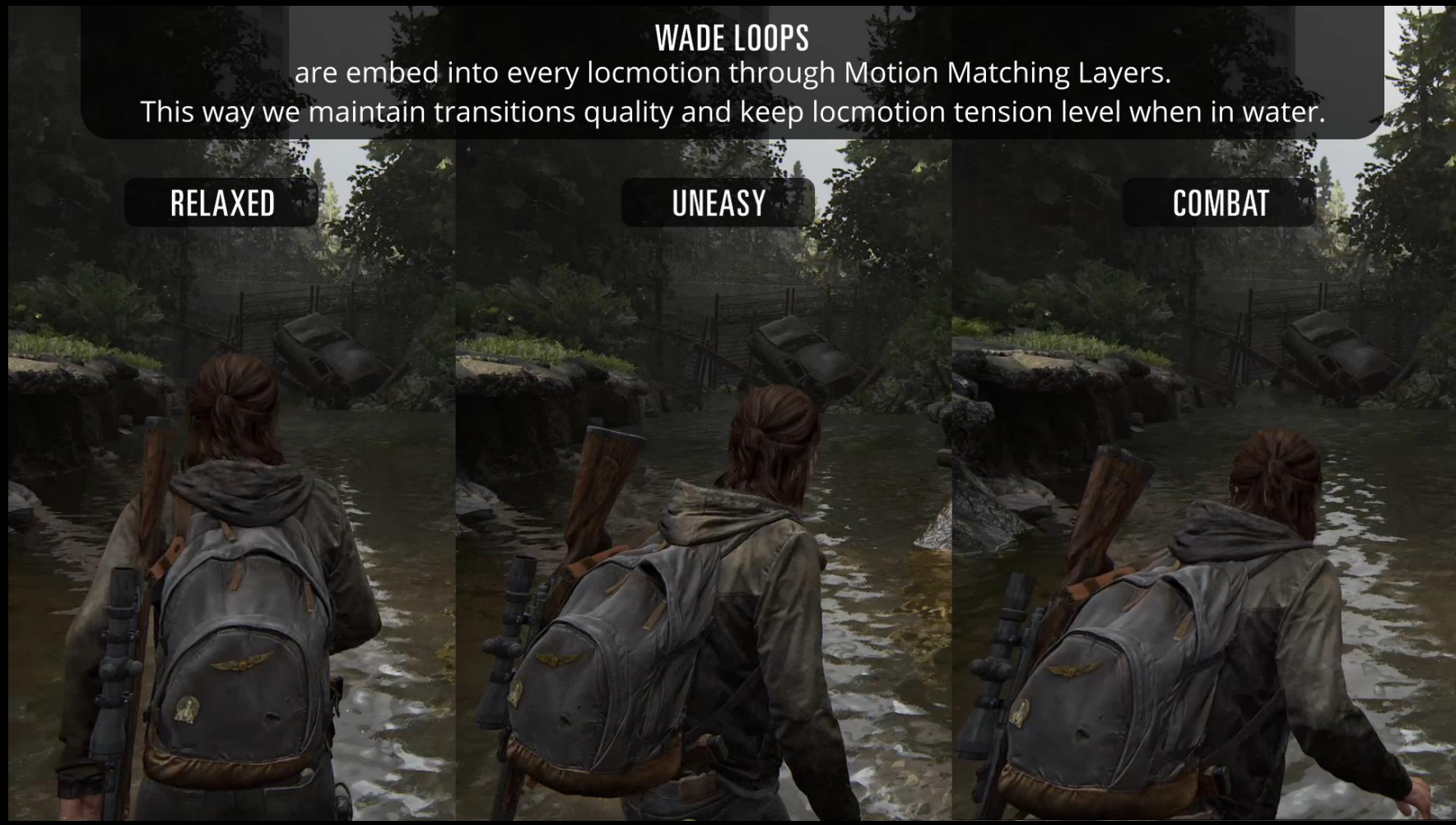


Foot Plant Animation



Custom User Layers

Custom User Layers



Weighting Strategy: Player

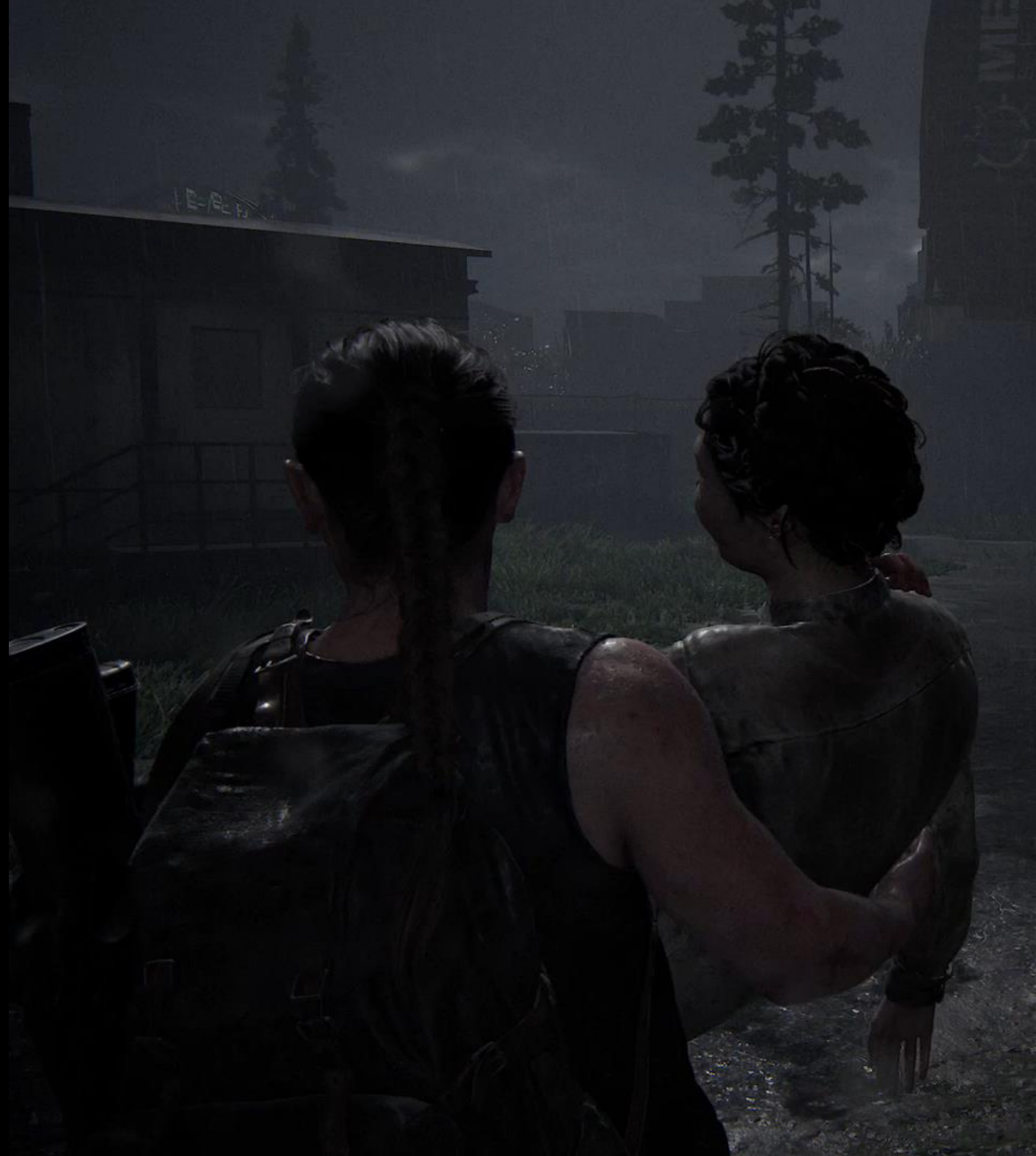
Player specific challenges:

- Stiff Motion Model Springs by design (acceleration and deceleration happens faster than in real life)
- Luck of future trajectory prediction
- No root motion offset allowed
- Close camera placement demands really solid connectivity between animations

Weighting Strategy: Player

Strategy:

- High pose cost (1.3-1.5 times on trajectory cost)
- Compulsory use of previous trajectory goal (0.4-0.7 sec)
- Medium-far future trajectory (1.2-1.8 sec)
- Low current velocity cost
- Low grouping bias, medium continuous playback bias
- Transitions get dropped by absence of future samples, rather than by the cost (that's why we cut them to a specific length earlier in presentation)



Weighting Strategy: NPCs

Challenges:

- Generated paths are series of straight segments therefore quite different from smooth animated trajectories
- We wanted to keep the motion as realistic as possible
- We needed the locomotion to be precise, but we couldn't slide the characters around much
- Players can clearly see the whole character therefore any motion discontinuities or foot sliding are obvious

Weighting Strategy: NPCs

Strategy:

- Short to medium trajectory length (0.5 – 1.2 sec)
- High trajectory cost (1.2 - 1.5 times the pose cost)
- High directional weight so NPCs face the direction we need
- Low Natural Bias to not hamper the movement precision
- Introduced a few new features
  - Yaw Speed Weight – to differentiate between turning and done turning trajectory goals
  - Interim Directional Weigh – to further increase directional stopping precision
  - Stopping Face Distance – when to start rotating the facing vector before reaching the goal

Animation Blending

Animation blends tend to wash out the character motion therefore trajectory starts falling behind quickly

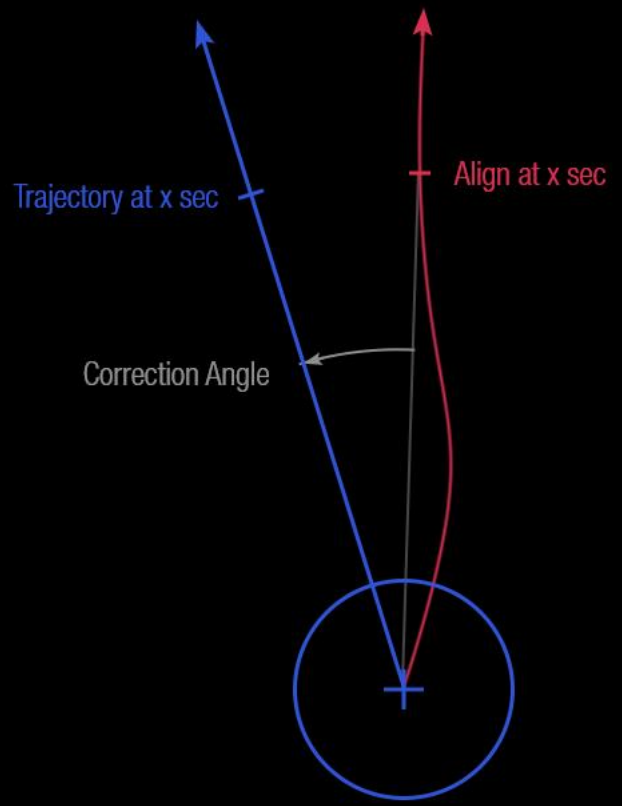
- We can blend root motion and skeletal animation at different rates
- Motion blend 0.2 - 0.3 sec vs 0.4 - 0.6 sec animation blend
- We can specify custom blending settings for every pair of MM sets
- We can setup custom blend time for any animation clips pair on top of that

Procedural Pass

Filling in the gaps for free

Procedural Pass: Steering

- Compares future trajectory and Align positions
- Rotates Align to put future Align on trajectory
- Rotation error distributed over time
- Configurable per move set

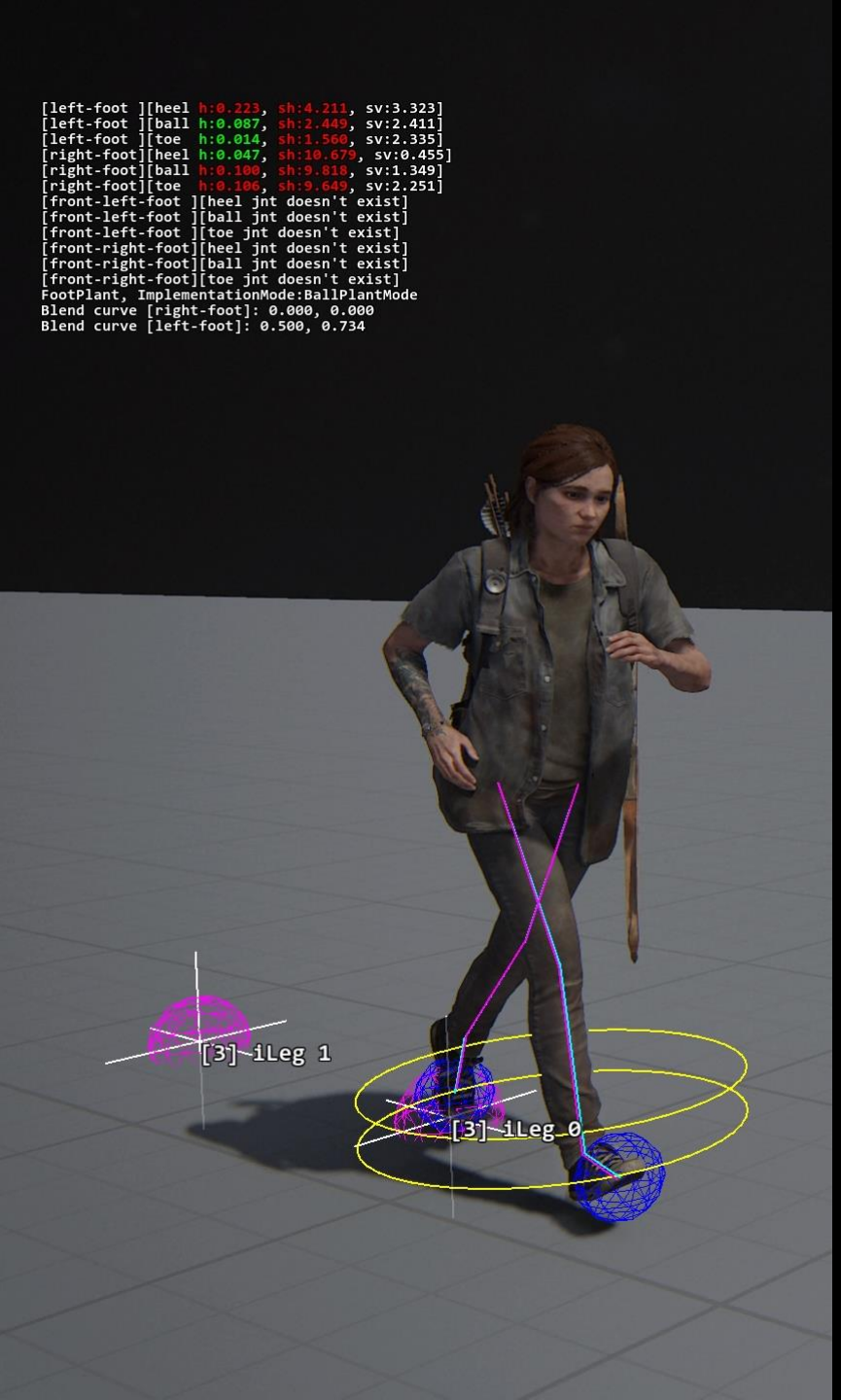


Procedural Pass: Time Scaling

- We time scale animation clips to compensate for trajectory differences
- Normally ± 10 - 15%
- Time scale coefficient is on a spring which prevents instantaneous changes

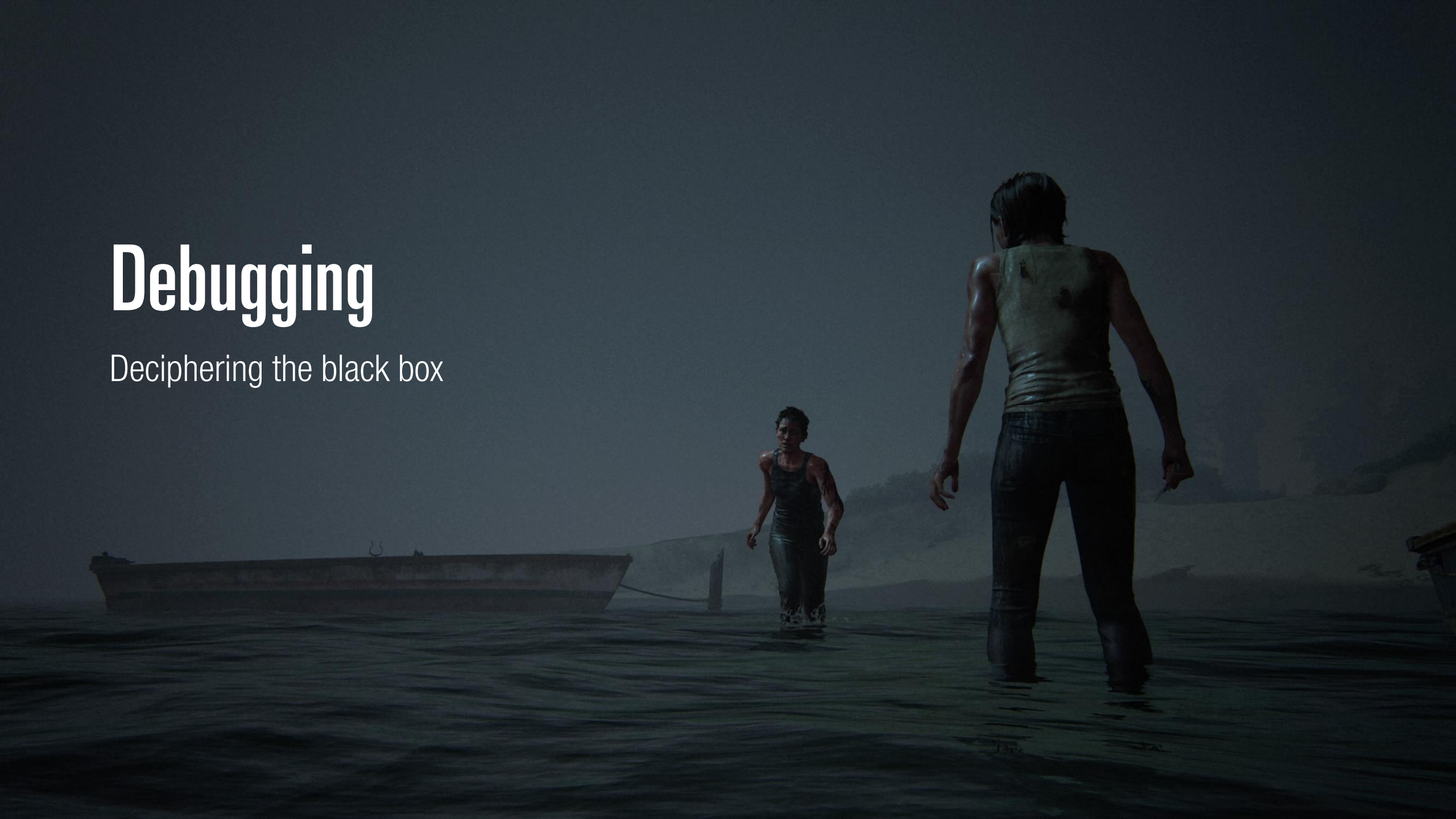
Procedural Pass: Clamping

- Locomotion is animation driven
- We allow the Align to drift away from Motion Model
- Align is constrained to the clamp distance radial constraint
- Basic constraint has a hard edge
- Soft clamp variant tracks towards center using a low pass filter



Procedural Pass: Foot Plant IK

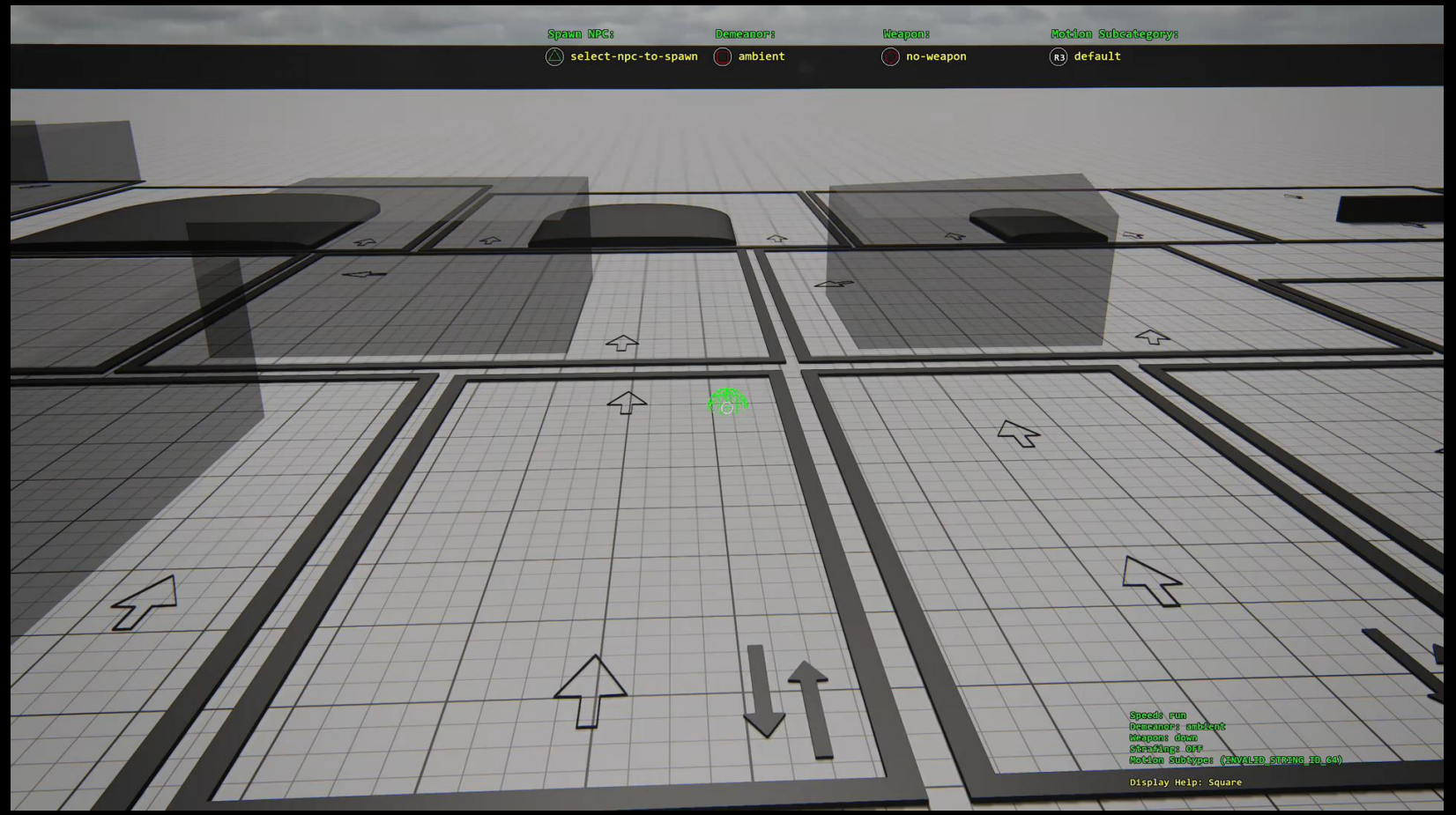
- Post process running all the time when locomoting
- Game tracks ankle joint positions every frame
- We pin the foot to its stationary position when below a certain threshold
- Jacobian IK solver
- Work for bipeds and quadrupeds



Debugging

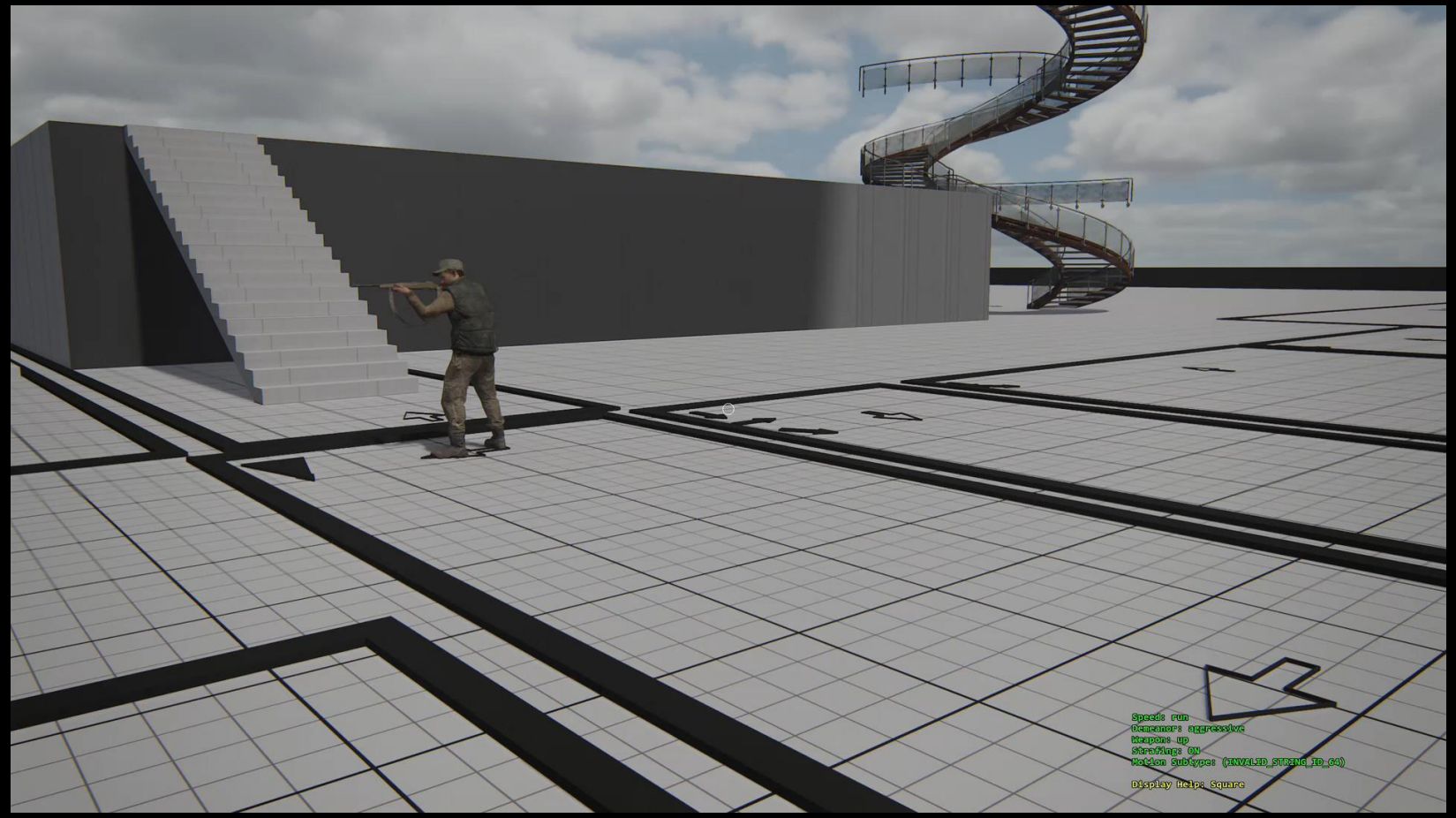
Deciphering the black box

Motion Matching Test Cases (full screen video)



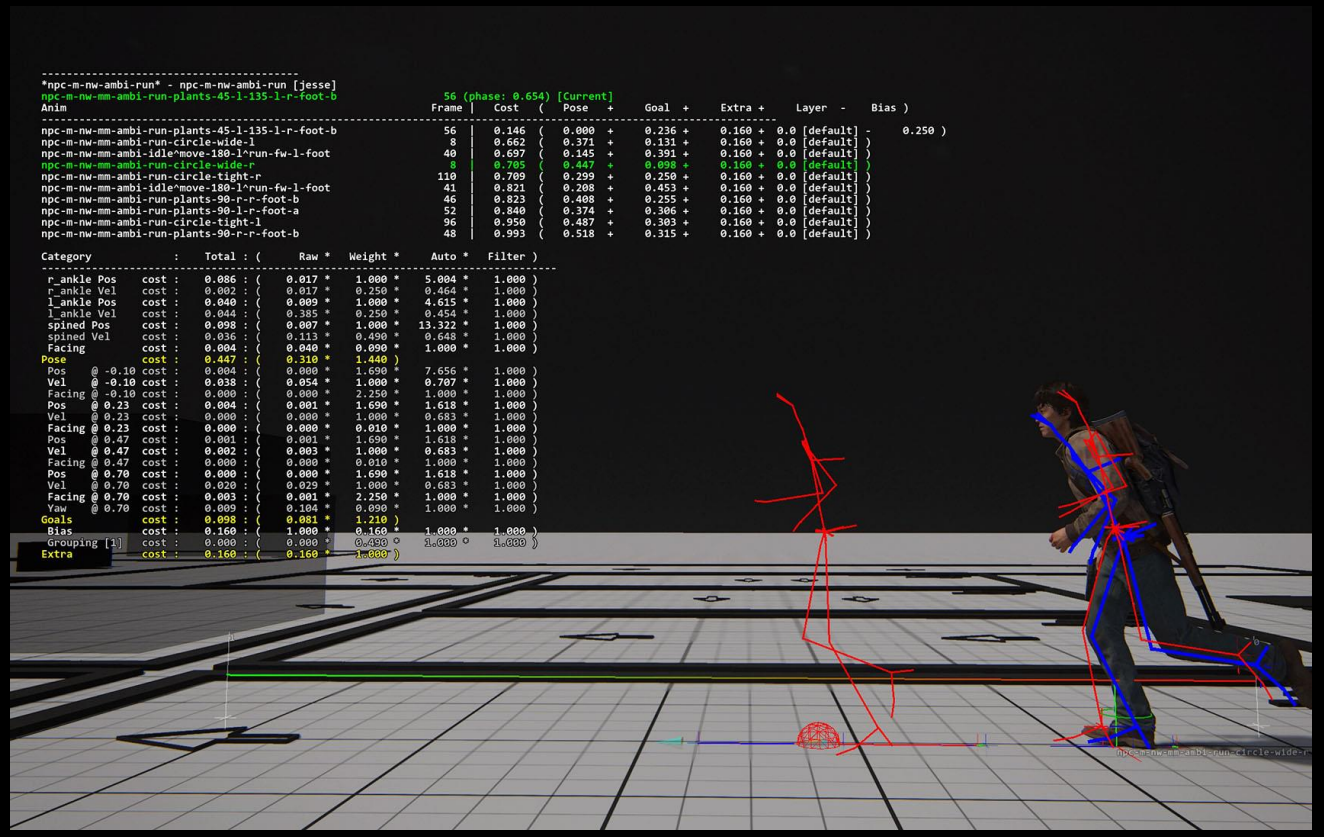
NAUGHTY DOG

Motion Matching Test Cases Strafing (full screen video)



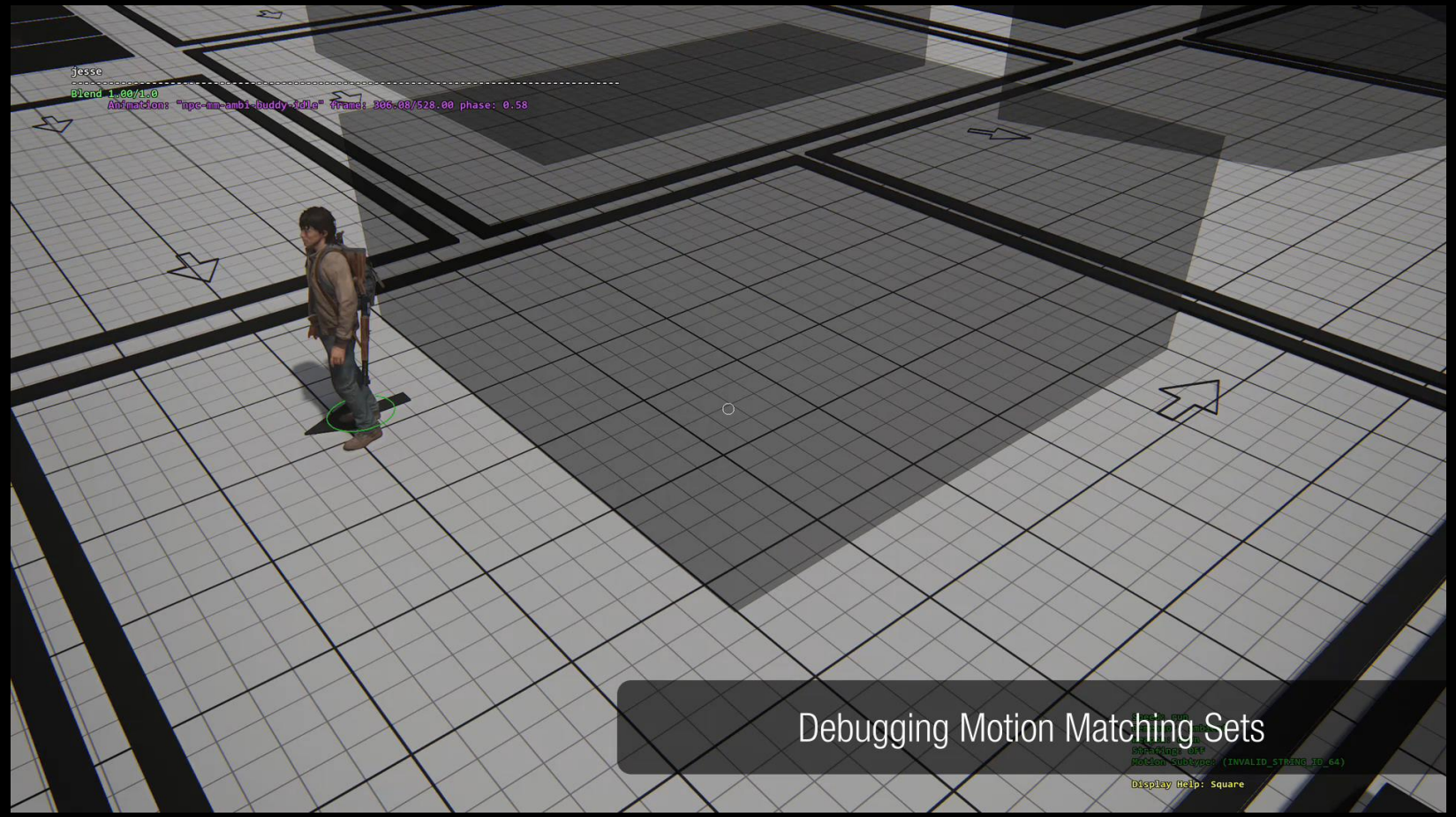
In-Game Motion Matching Debug Draw

- List of best animation samples with their costs
- Cost breakdown for the selected sample
- Current character pose (blue stick figure)
- Stick figure playing selected animation sample (red)
- Animation Trajectory (red)
- Motion Model trajectory (blue)



NAUGHTY DOG

Debugging Motion Matching Sets (full screen video)

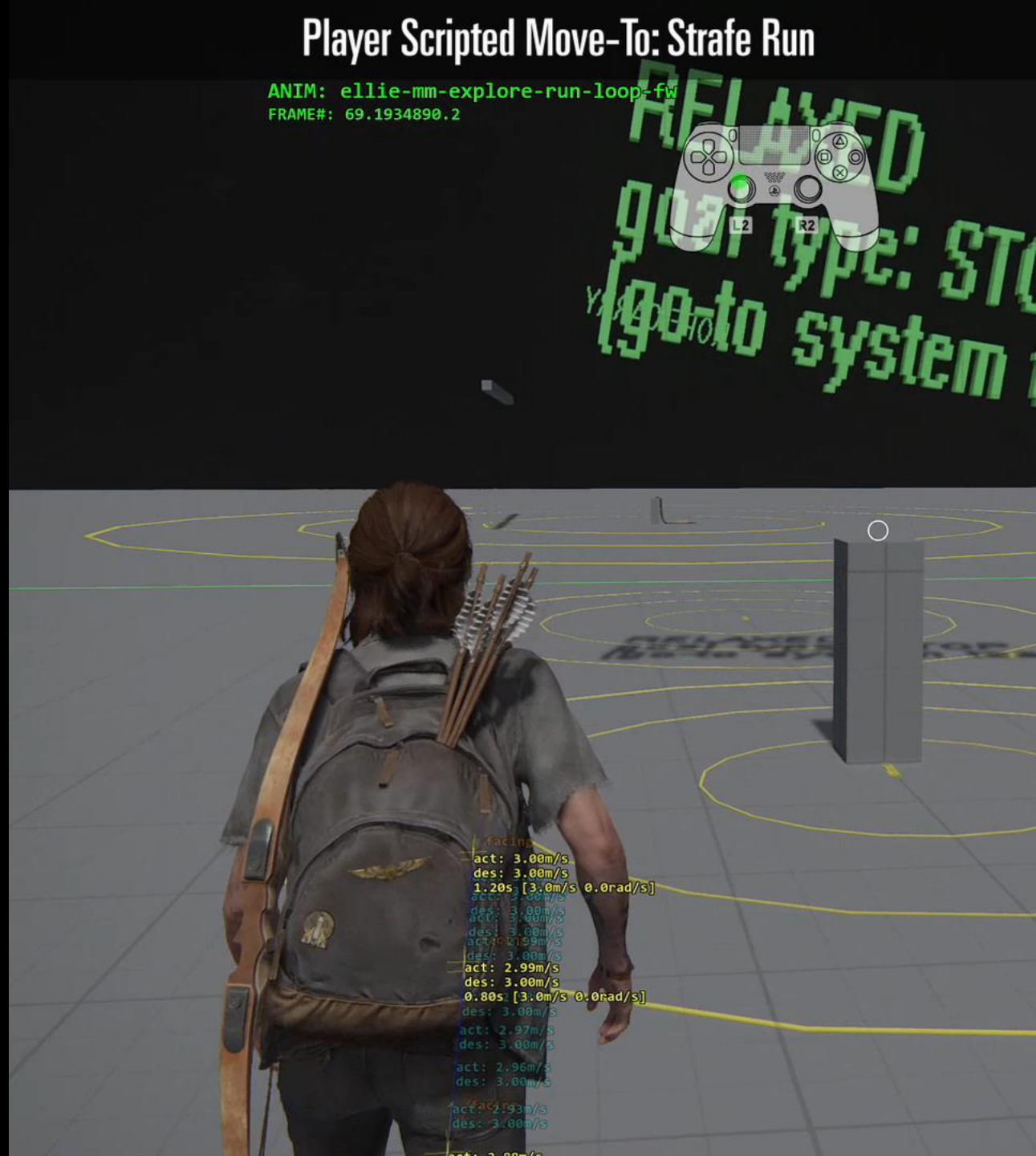


Connecting With Other Systems

MM Idle Recovery

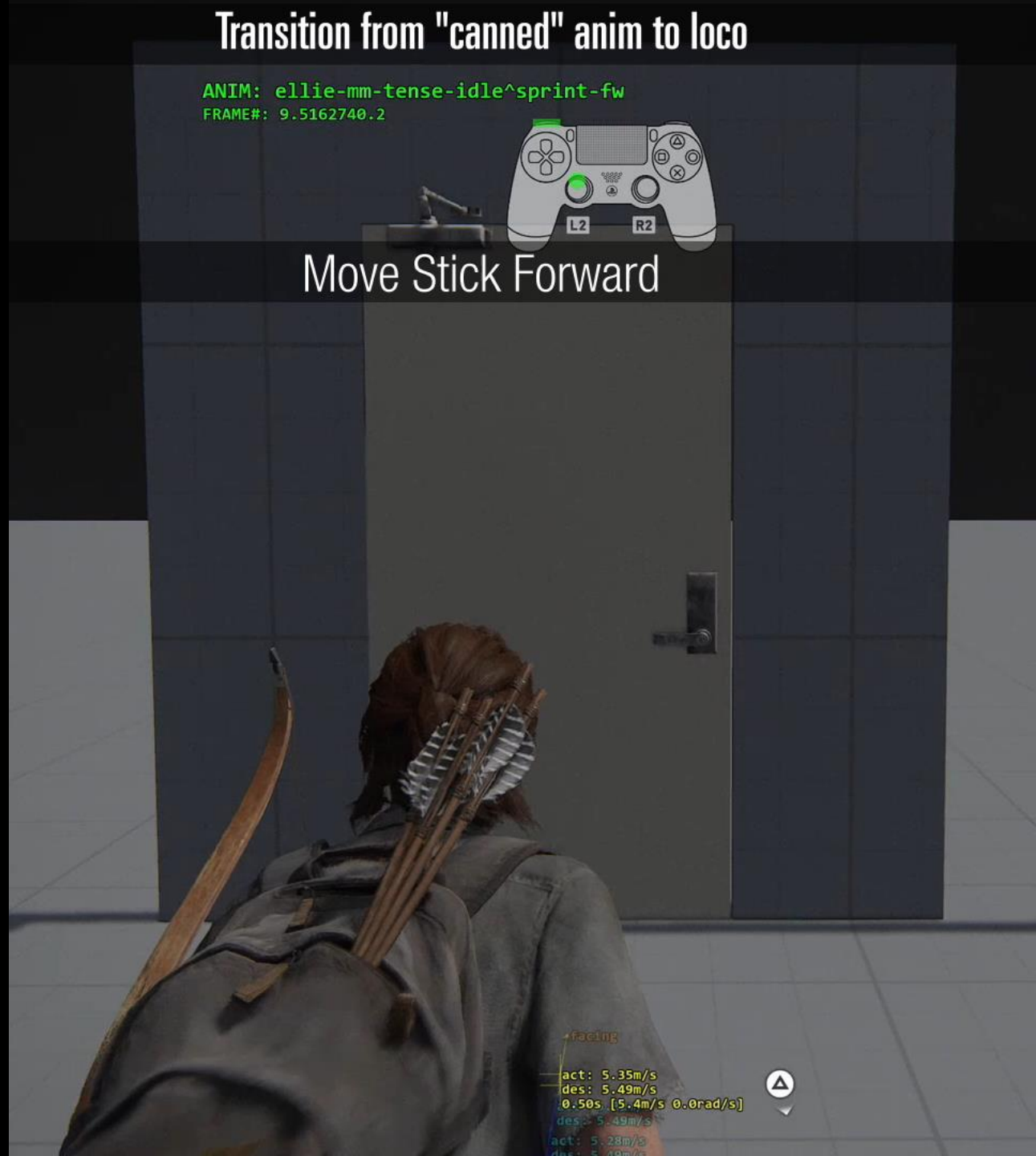
Player Scripted Move-to

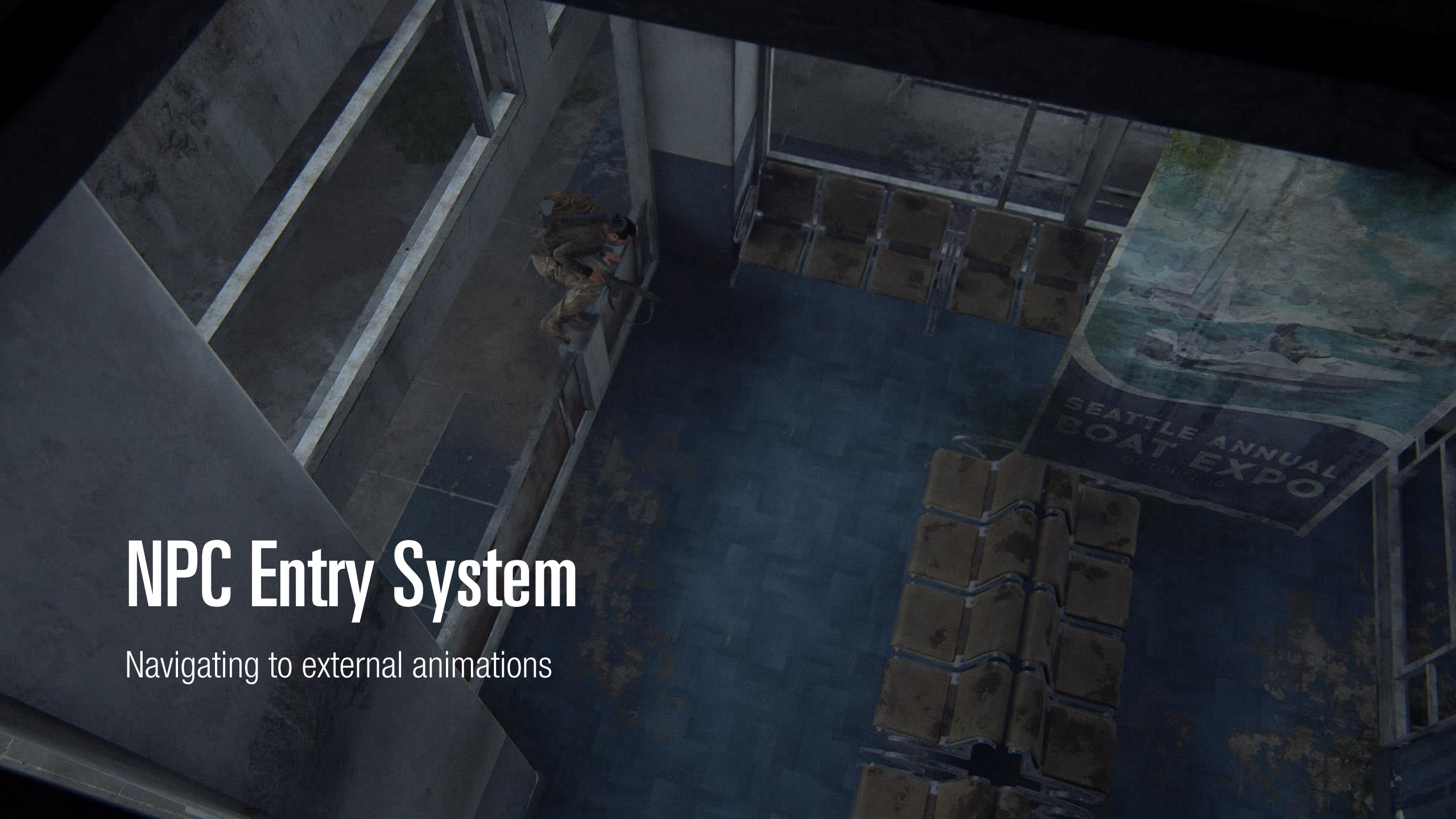
- We have a strafing with free idle camera locomotion scheme design. It poses quite of a challenge for a move-to
- We wanted approaching interaction target to look natural and have humanly appropriate deceleration
- We wanted approaching interaction to look intentional in terms of body language
- We still slightly pull the player into the target as distance to the goal becomes short



Exiting “Canned” Animations

- There is no additional setup on the door animation, except of abort frames
- MM does a pretty good job of capturing pose, speed and current stick request and shuffles in the best animation available in the set
- This behavior allowed us to use full body animations more confidently in the context of responsive gameplay without worrying to much about exiting part of the systems





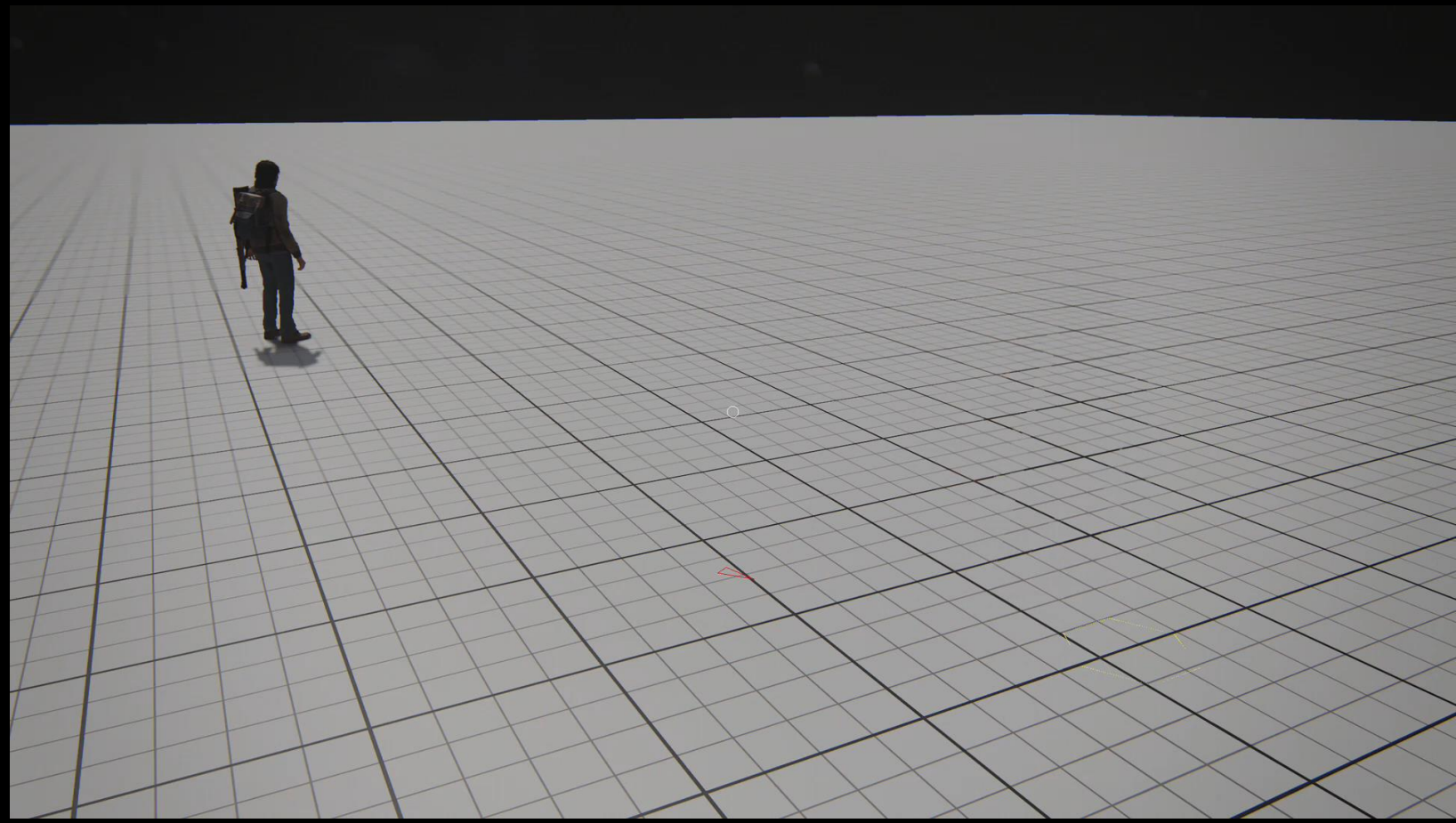
NPC Entry System

Navigating to external animations

NPCs Entry System

- Walking used to be simple – one walk loop animation
- We used to be able to scale the stride to arrive with a correct foot phase
- Motion Matching is a soup of animation data
- There's no guarantee that the walk loop is even going to play
- We HAD to solve this problem to reach the Naughty Dog quality

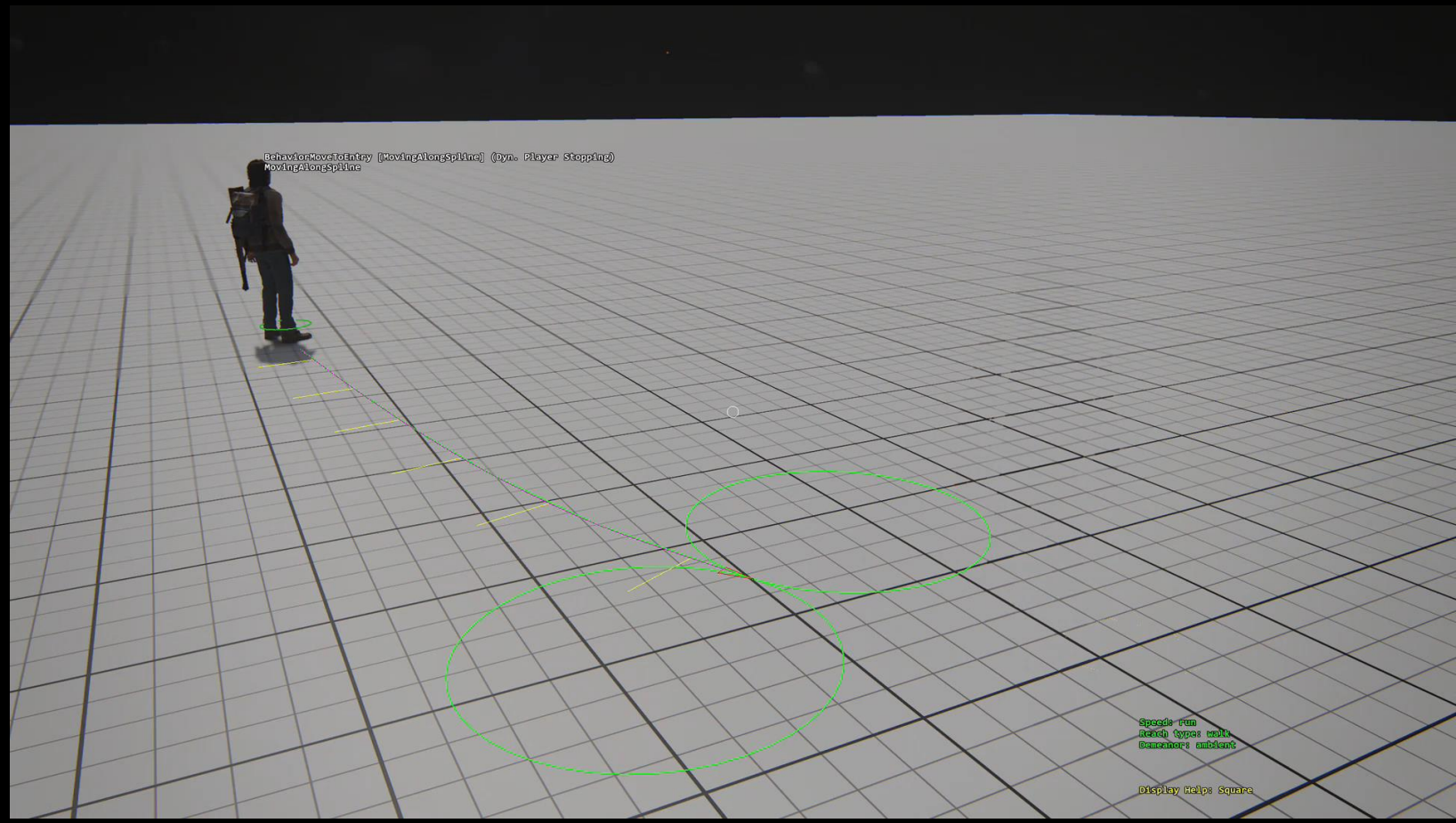
Entering External Animations (full screen video)



NPC Entry System

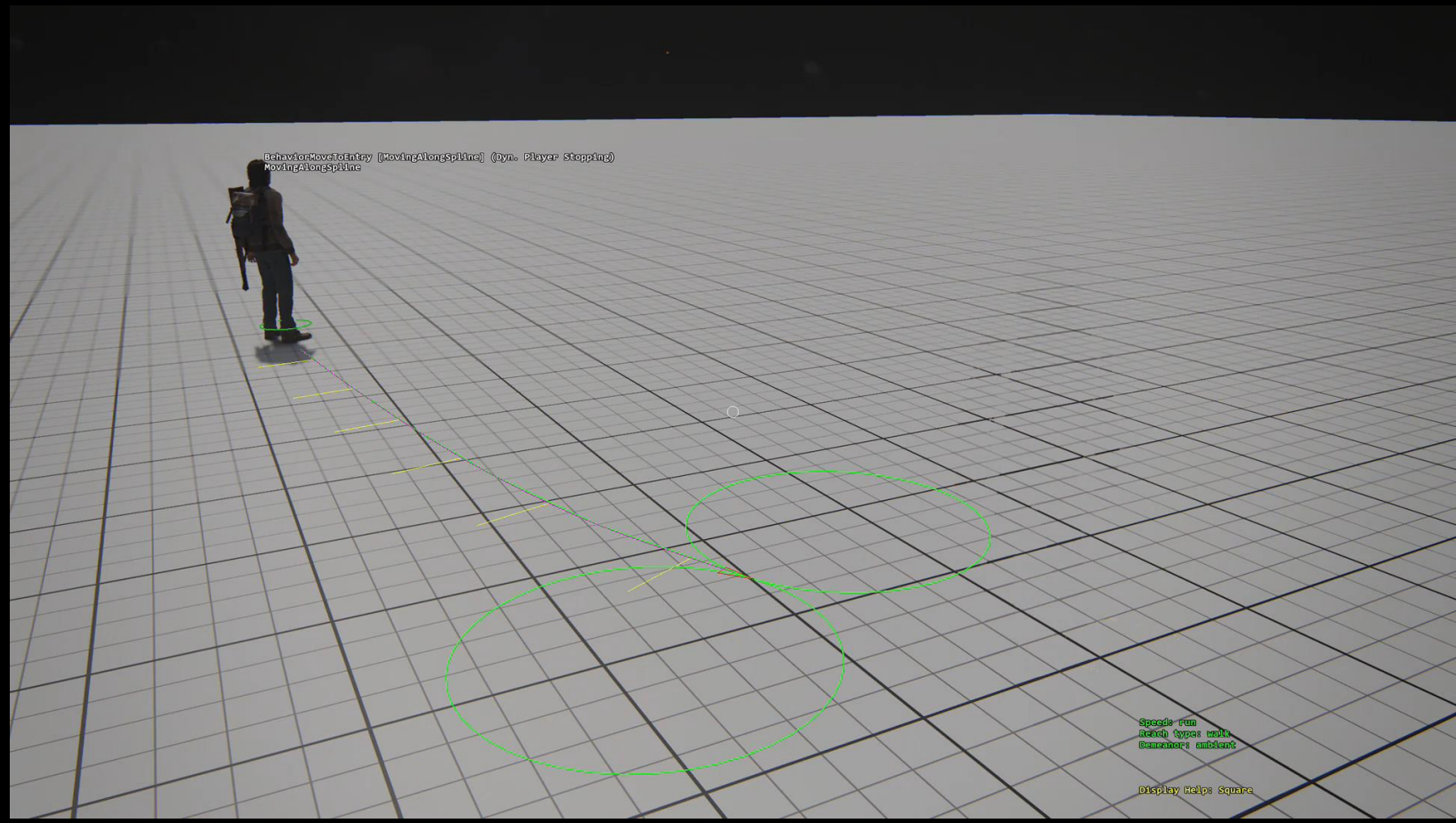
Solution

- Spline trajectory from entry vector to NPC
- Movement speed transition distance
- Match the Motion Model speed to the entry Align velocity
- Mark up “matching loop” animation clips in all move sets



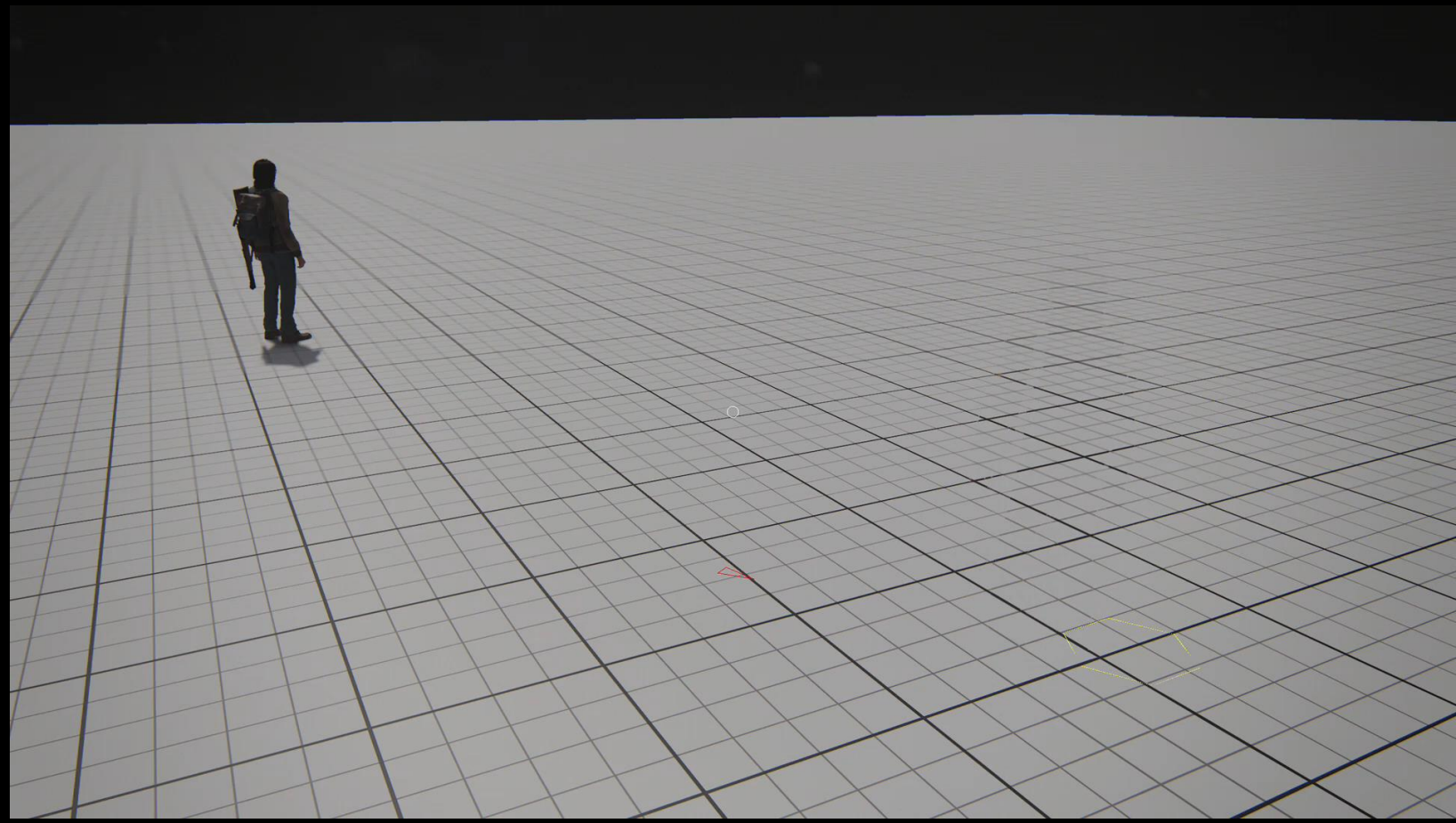
NPC Entry System

- Compute the closest matching phase of a loop animation
- Compute translation error from the phase delta
- Move the Align closer to the ideal distance from entry
- Repeat every frame



NPC Entry System

And this is the result



We’re using this system when entering

- Cinematics/IGCs
- Cinematic Action Packs
- Traversal Action Packs
- Search Corner Checks

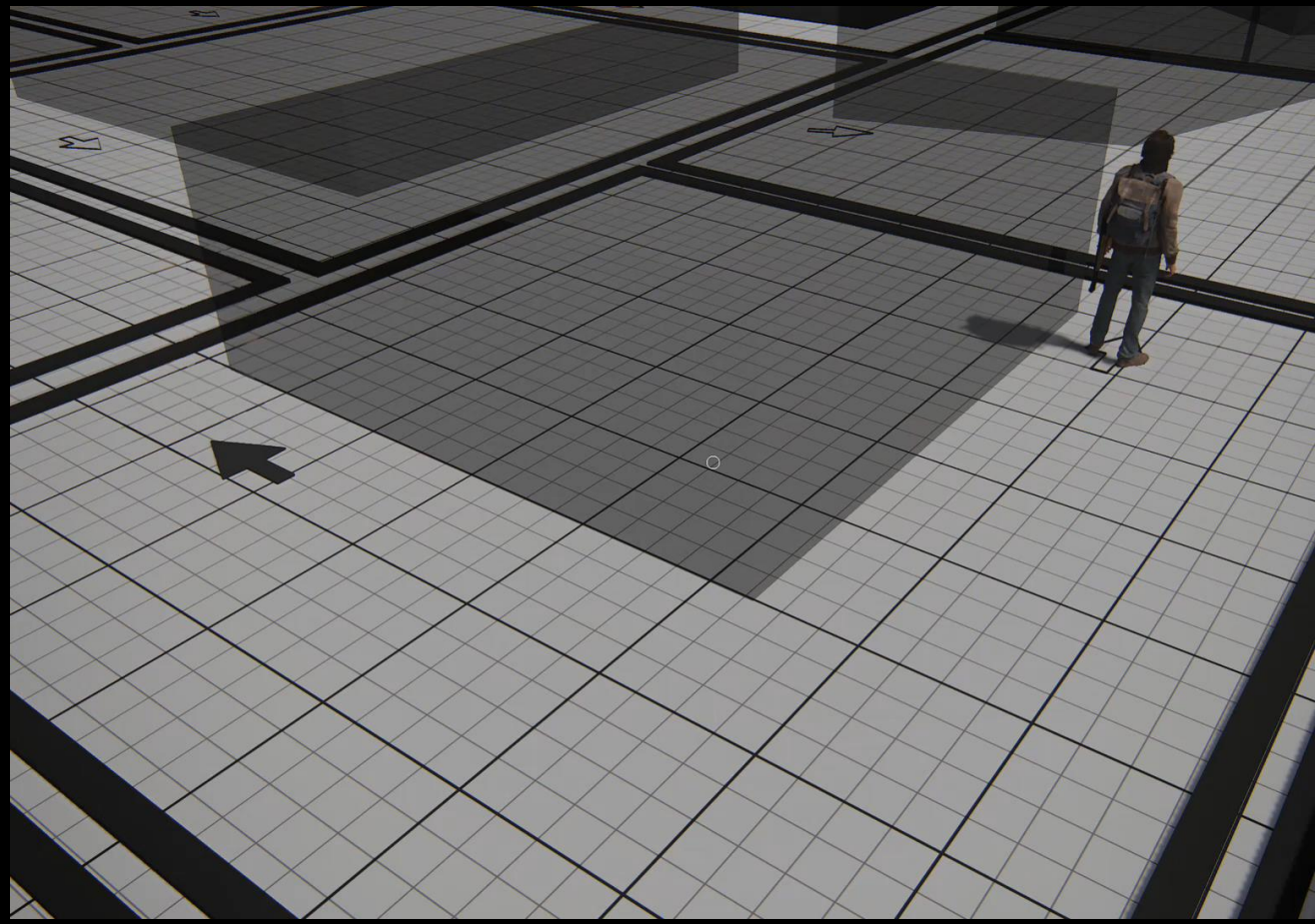
Preserving Character Weight

Slowing down at corners

Slowing Down At Corners

Problems

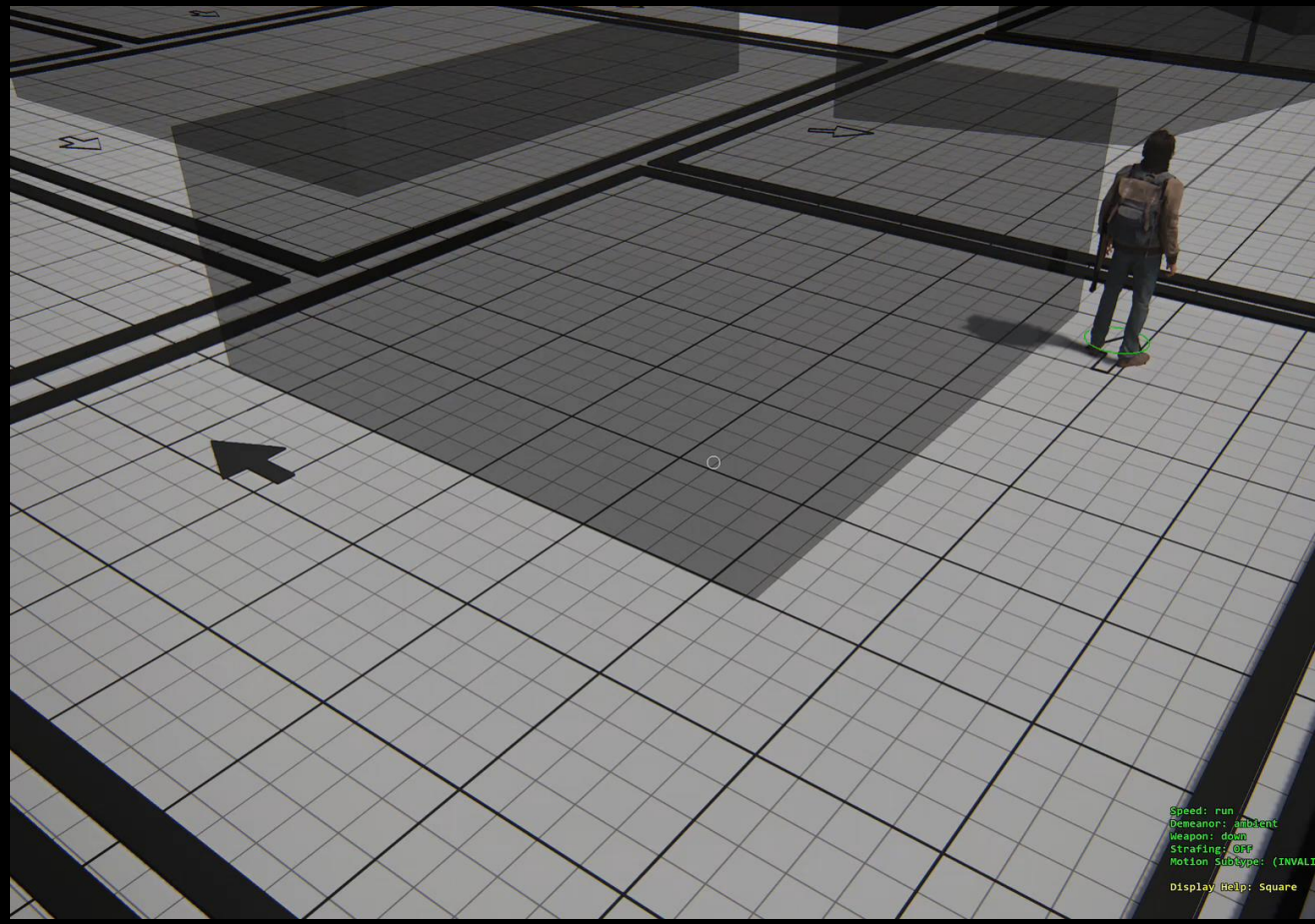
- The NPC whips around corner without slowing down
- Weightless feel
- Increases foot sliding
- The Motion Model deviates from the animation data



Slowing Down At Corners

Solution

- Motion Matching Settings
  - Define the threshold angle and corresponding speed values
  - Define the slowed down duration
  - Values read from Animation Analyzer
- Runtime
  - Collect path corners within a radius
  - Calculate the aggregated angle value
  - Calculate the slow down distance based on deceleration spring



NPC Traversal Action Packs

Vaulting, climbing, jumping and balance beams

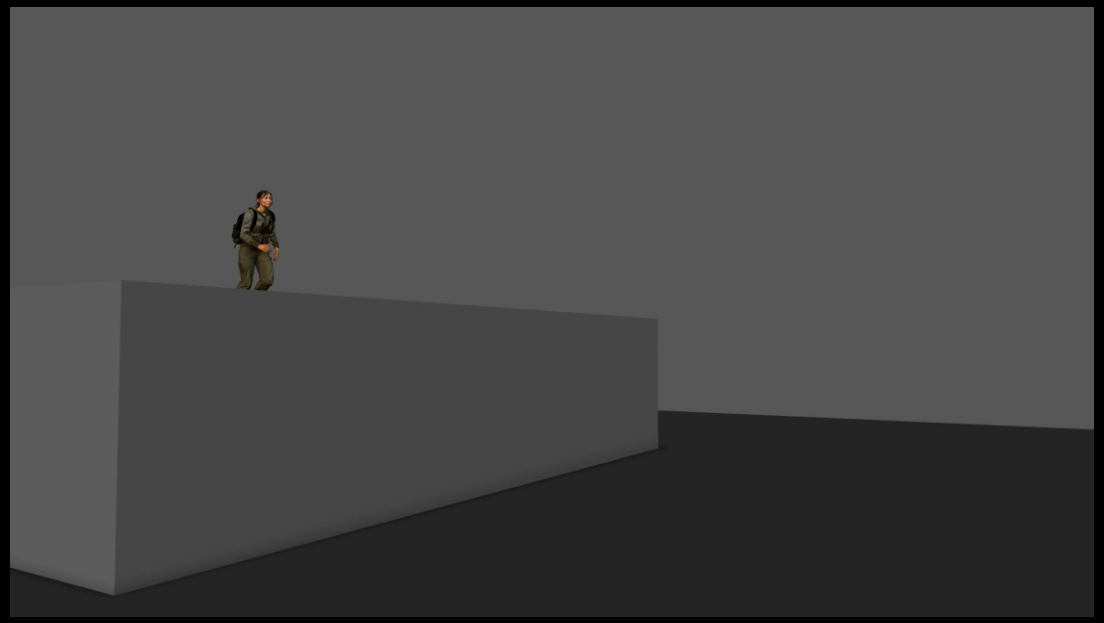
Traversal Action Packs v1.0

Pre TLOU2 System

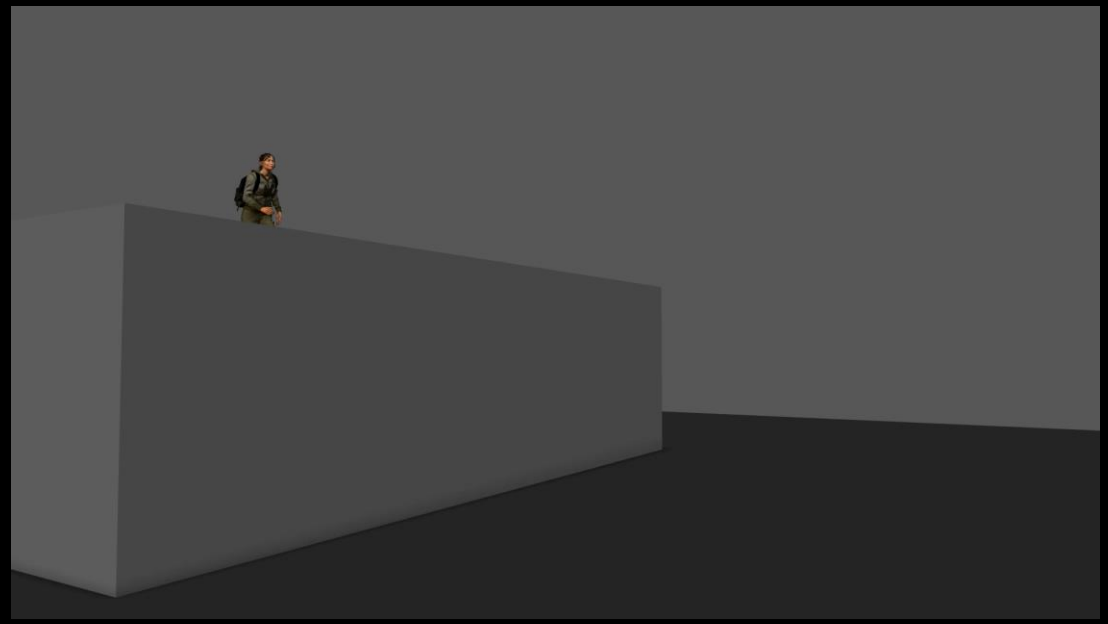
- Set of Enter, Middle and Exit Animations
- Strict branching points
- Due to short blend times, poses at branching points had to match closely
- Limited height or length flexibility
- Solved by crossfading two instances of the middle animation, each anchored to either the enter or exit point

Traversal Action Packs v1.0

Authored height 2m



Adjusted height 2.4m

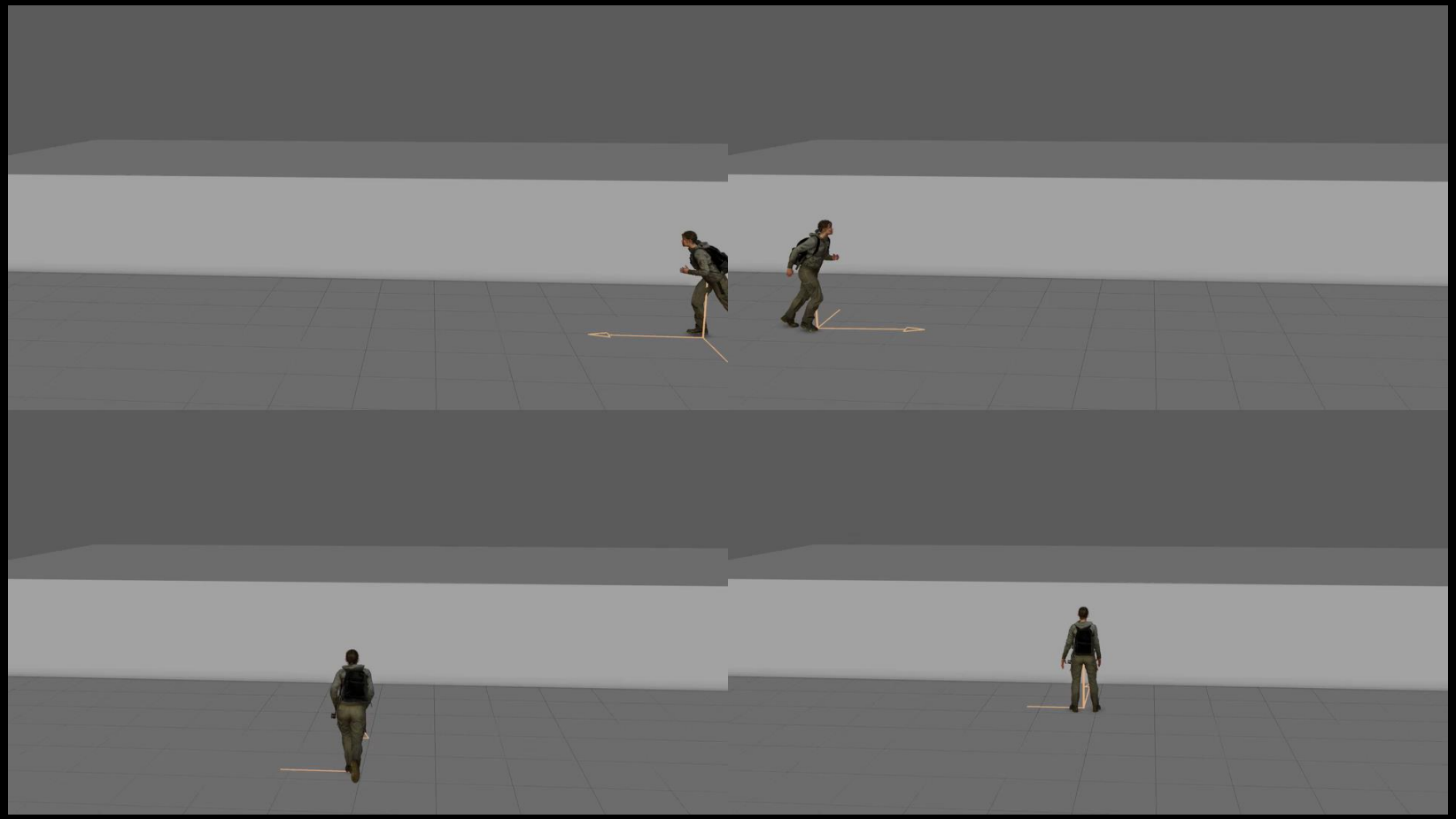


Traversal Action Packs v2.0

Improvements

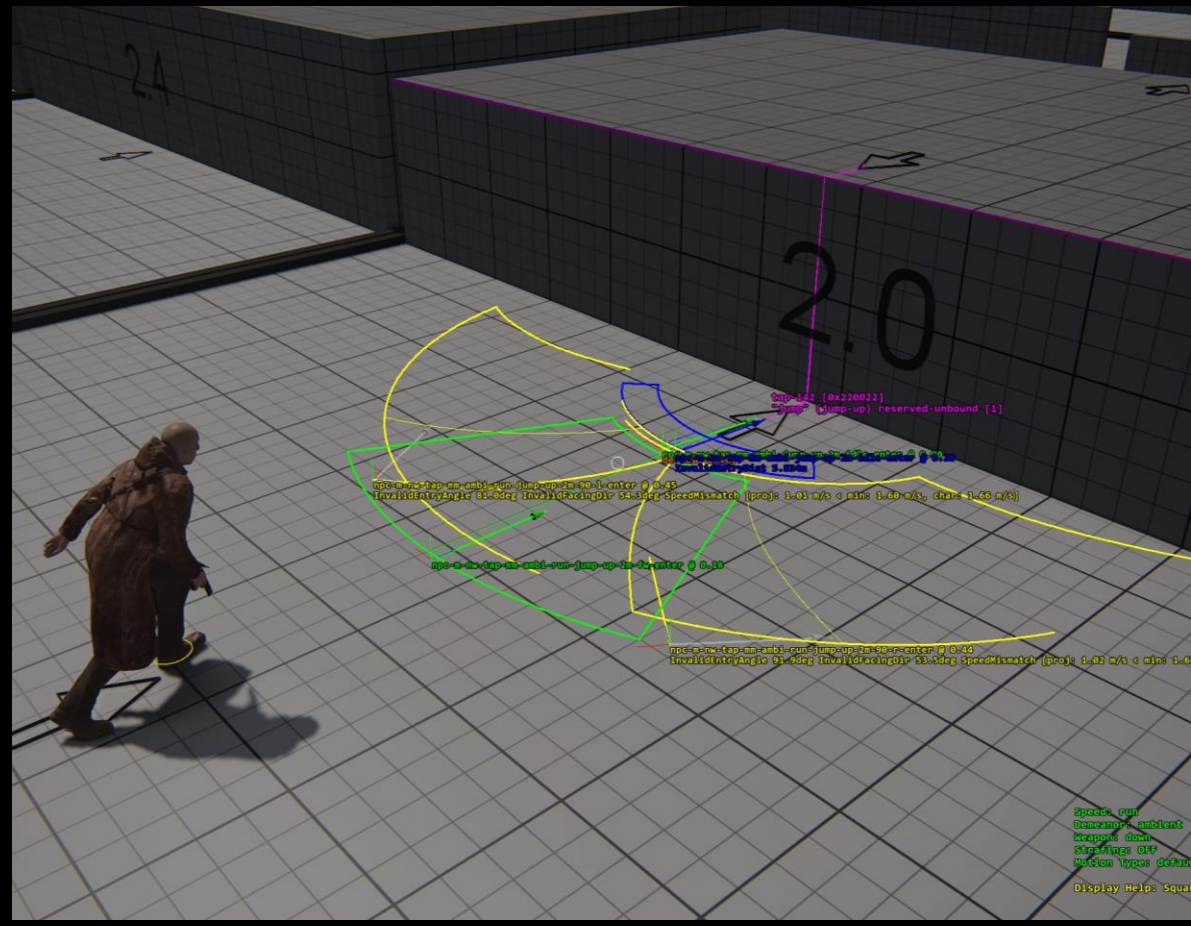
- Leverage Motion Matching
- Animations used for generating Motion Model trajectories
- Dynamic trajectory shape preserving momentum
- Trajectory intersect points improvements

Traversal Action Packs v2.0: Animations (full screen video)



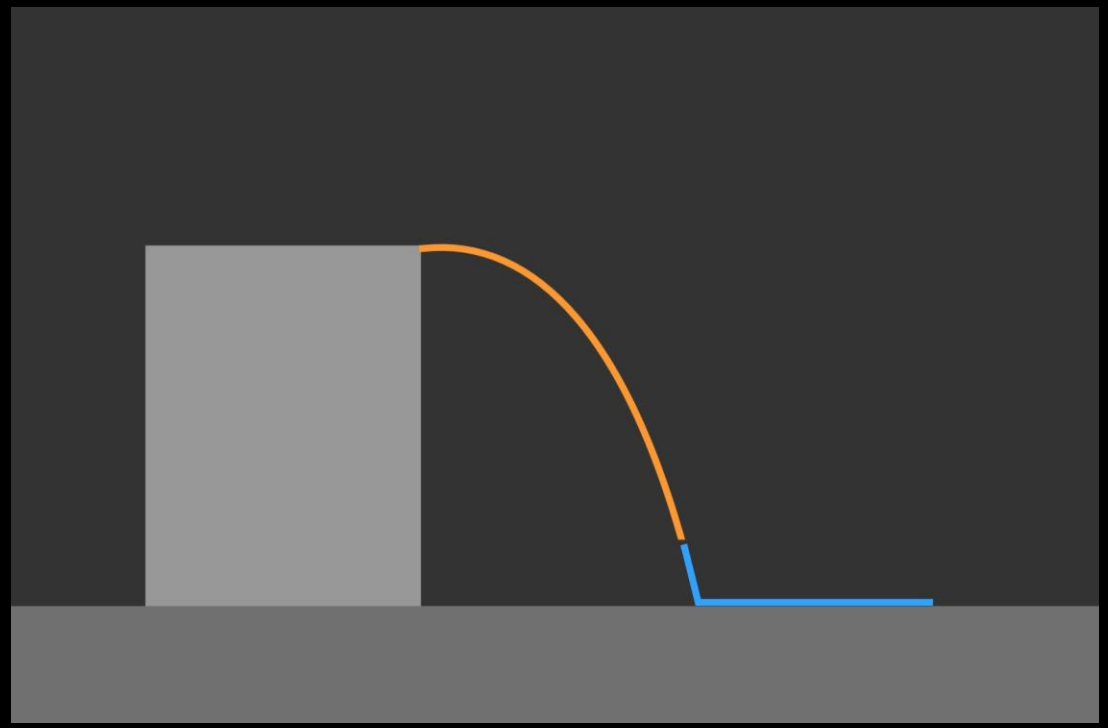
Traversal Action Packs v2.0: Trajectories

- Constructed using:
  - Idle, Forward, Left, Right Enter Animations
  - 1 middle animation (optionally more)
  - Idle, Forward, Left, Right Exit Animations
- Enter and Exit animations can be procedurally rotated
- After the procedural rotation is done, we find ideal intersection points with minimal distortions
- Generated trajectory is used as source for Motion Model



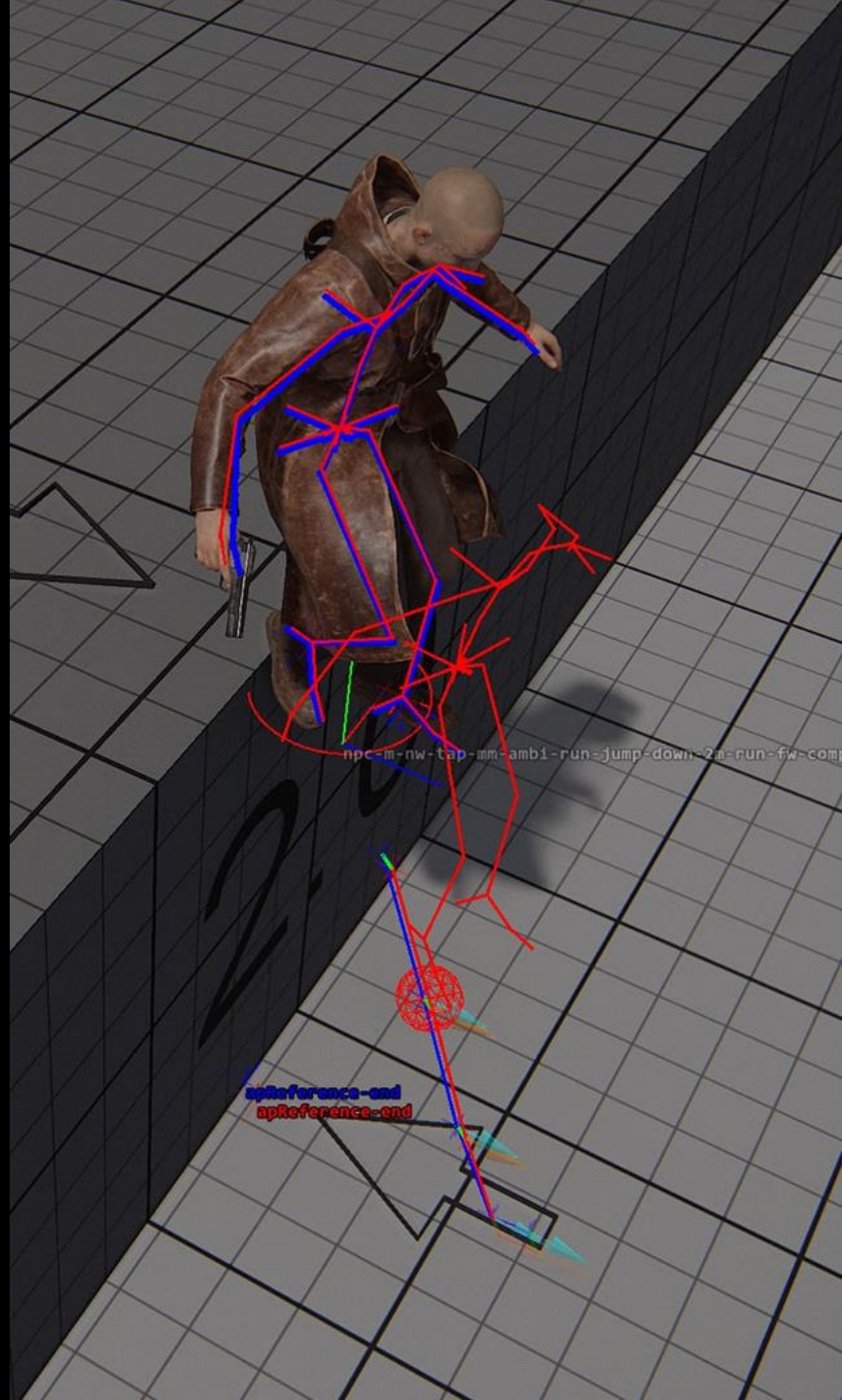
Traversal Action Packs v2.0: Trajectories

- The trajectory can self adjust to the shape of a TAP
- We trim the middle animation when the distance is shorter than authored distance
- We extrapolate the trajectory if the distance is longer than authored distance
- Doing that automatically makes the fall last longer
- Exit point moves so it perfectly connects with the trajectory



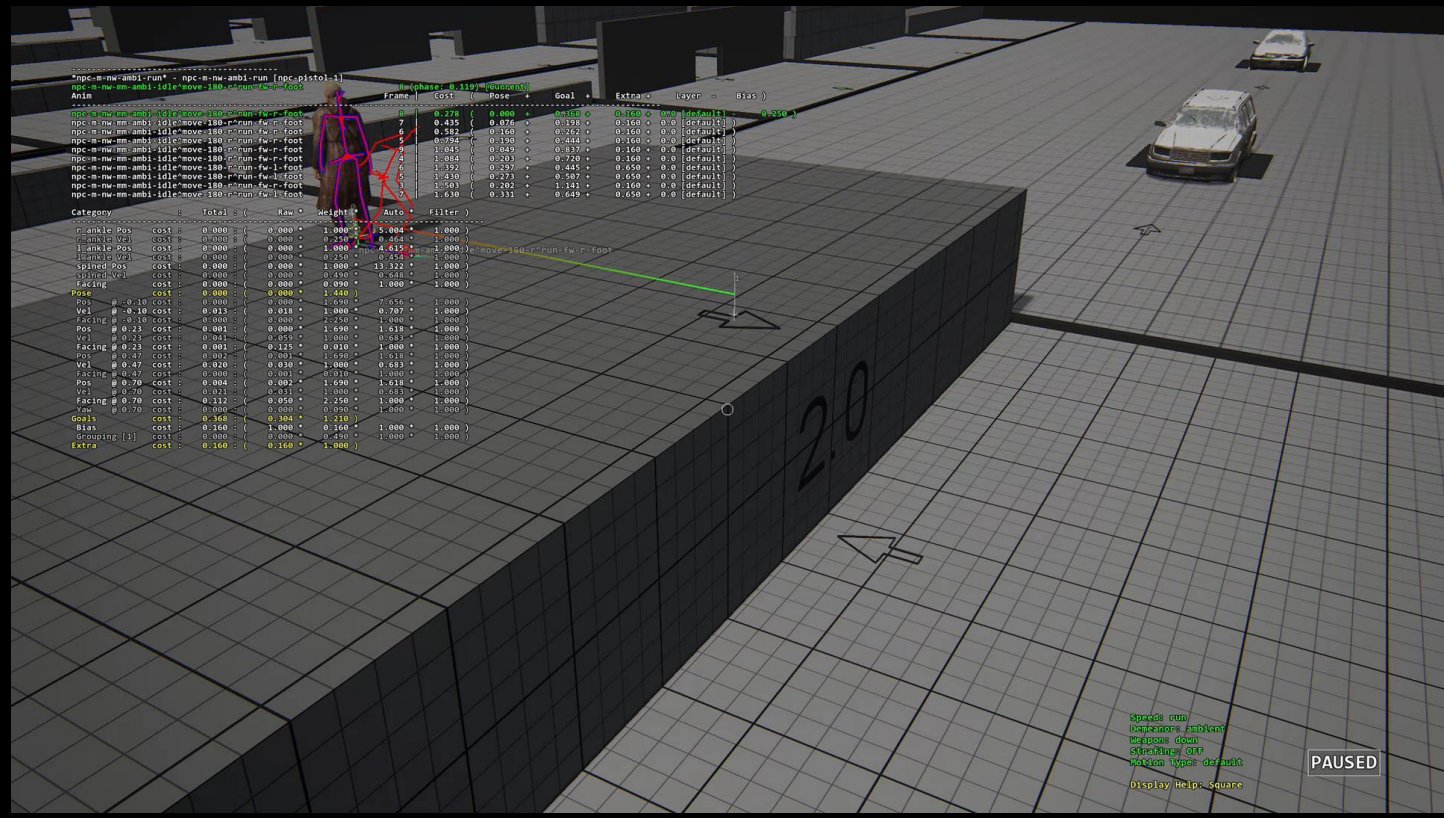
Traversal Action Packs v2.0: New Features

- External Goals in Motion Matching
  - First foray having world registration represented in our search logic
  - AP Locators as query data (position only)
  - Like AP refs, gameplay code supplies the runtime value
- Constraints
  - Handle External Goals switching
  - Move character in space over time to minimize distance error between External Goal and AP ref locator in the animation clip



Traversal Action Packs v2.0: In-Game

- NPC navigates to an entry point
- TAP is entered
- We're matching the first apReference as an external goal
- Constraint to apReference is active
- Mid-fall we start matching the second external goal at the base
- Constraint to apReference-end is active
- Exiting the TAP



Quadrupeds and Motion Matching

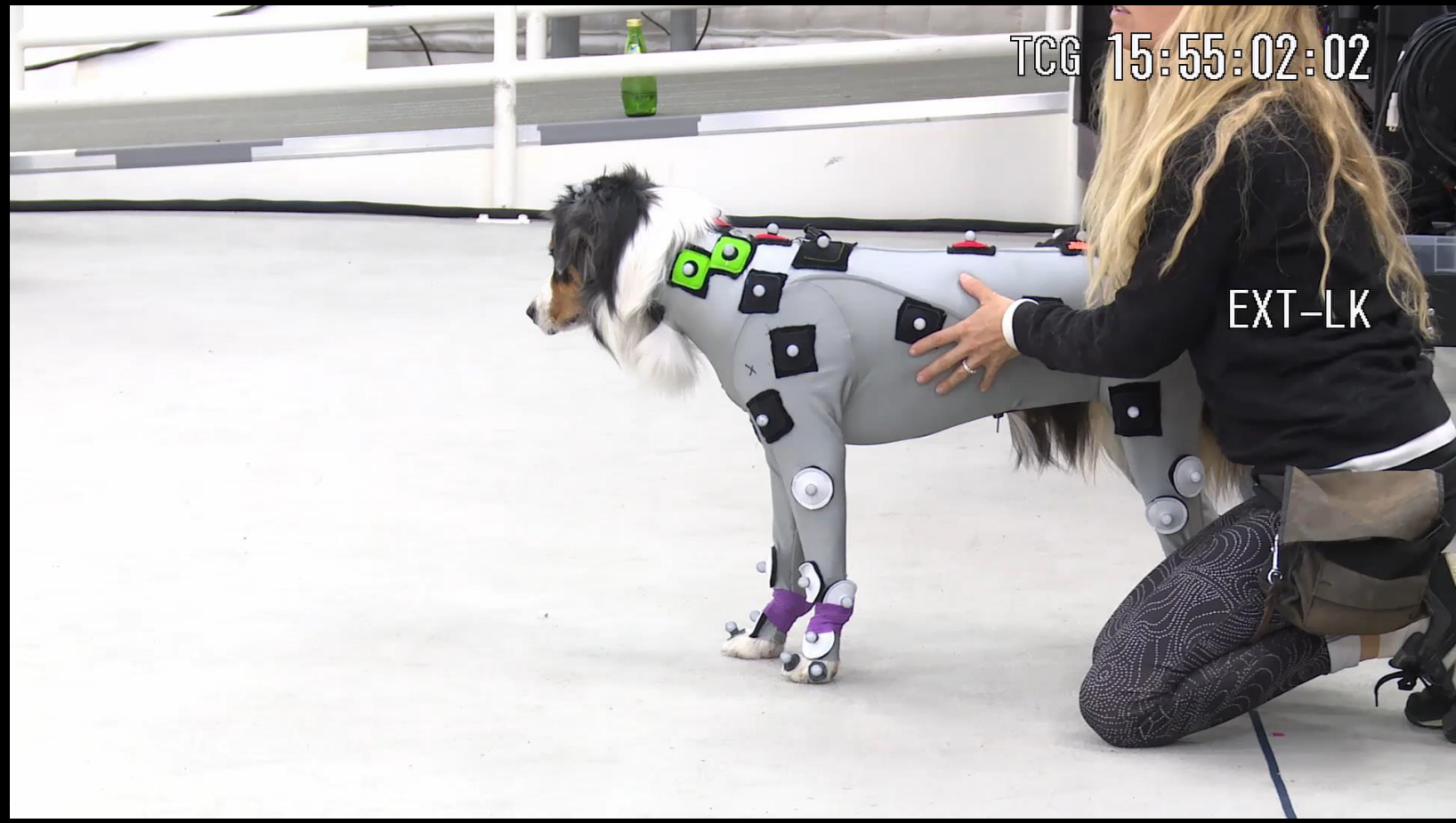
Dogs & Horses

Dogs Animation Data

- Used same approach as for humans
- Structured clips
- Tried to get as close to the desired trajectories as possible
- We've supplied the list of moves ahead of time so the dogs could be trained to do them

Dogs Animation Data

- Tracking move set was impossible to shoot as structured data
- We've decided to go the unstructured route
- It worked well enough for our purposes, but it's hard to get full coverage that way



Dogs: Lessons Learned

- Ambient run and walk worked well
- We were not able to record any aggressive move sets for the dog
- We used a regular run and heavily keyframed over it
- It's incredibly difficult and the results reflect that

Horse: Motion Model

Horses are large animals and the human model pivoting around its center wasn't suitable

Additional features for player horse

- Delayed facing to simulate realistic turning
- Simulation of understeering / oversteering
- Custom settings for riding in circles
- Facing momentum when stopping

Horses Animation Data

- Horses are much easier to control than dogs
- We were able to obtain decent data for walk, trot and canter
- Gallop was impossible due to limited space and unsuitable floor surface
- Structured vs unstructured data 50:50

Horses Animation Data

- Horse was the only character having three locomotion speeds (walk, trot, canter) mixed in one motion matching set
- Gallop was still a separate set because the riding style is very different
- Motion Matching allows for flawless transitions if the set has enough coverage

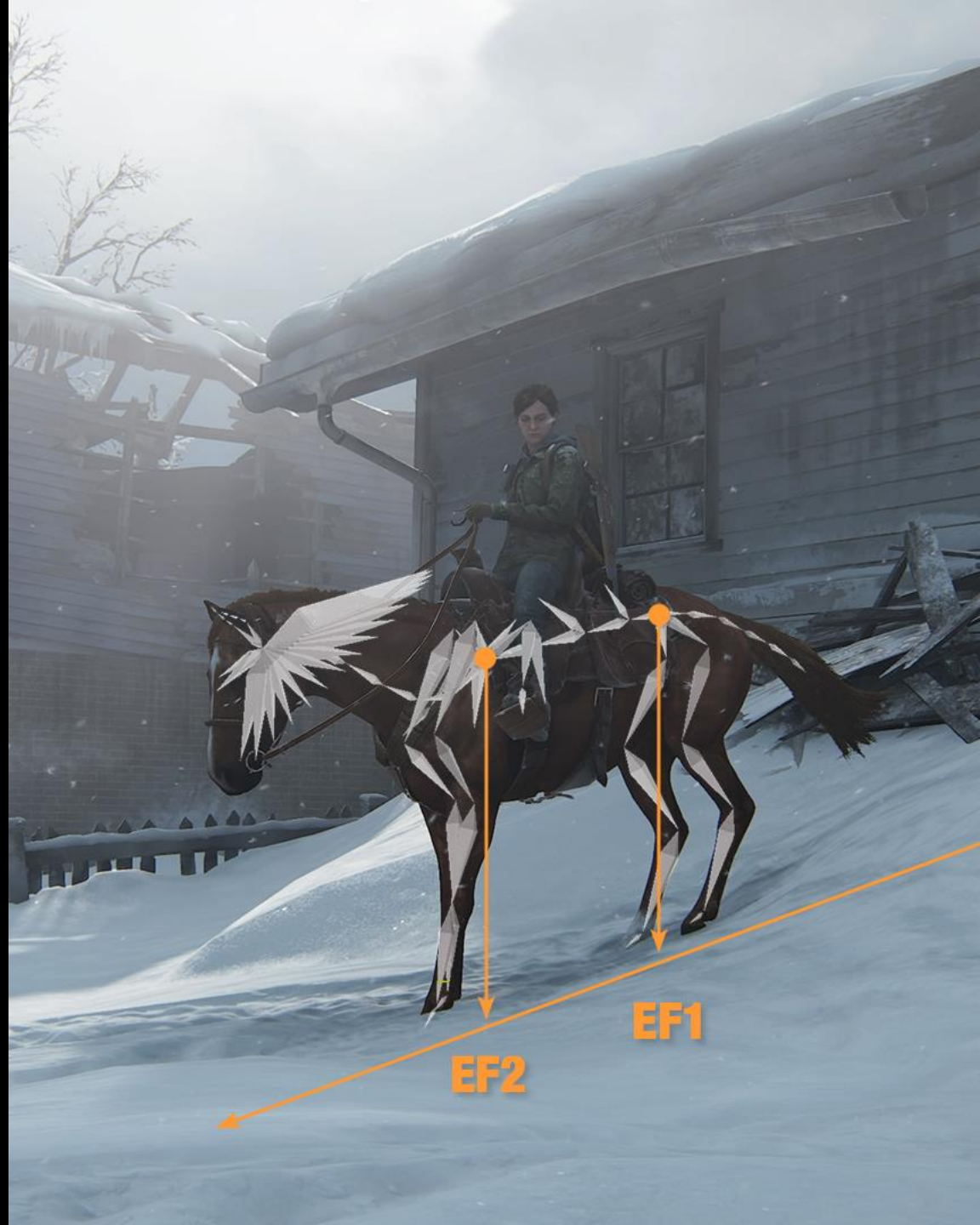
Horses On Slopes

Procedural Animation For The Win!

Horse On Uneven Terrain Video (full screen)

Full Body Jacobian IK

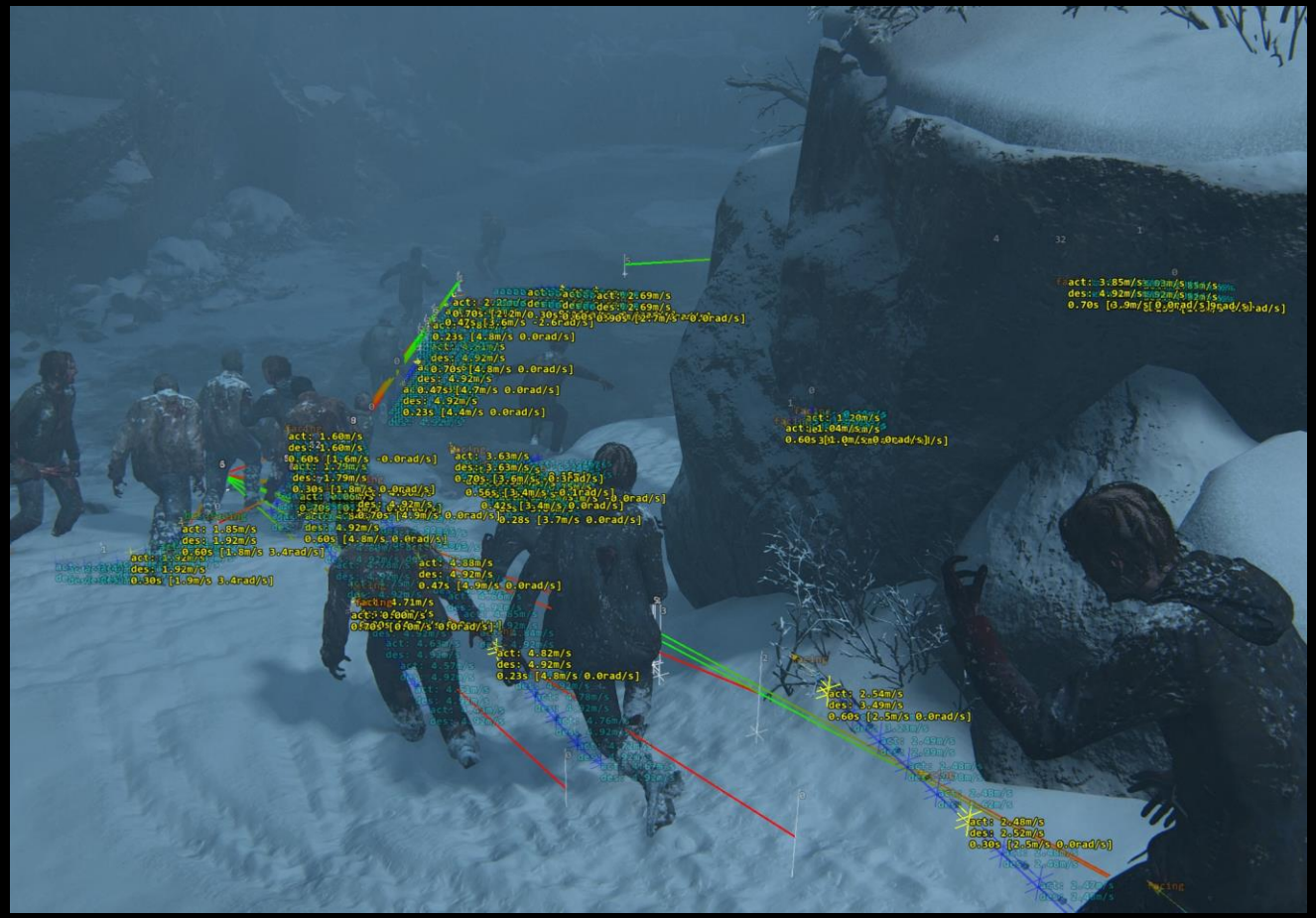
- Main IK Chain
  - From root joint to chest joint
  - Each end effector tries to stay at an animated height from the ground
  - The vertical position is sprung using critically dampened spring
- Leg IK Chains
  - Each leg solved independently to the ground



Performance

Performance

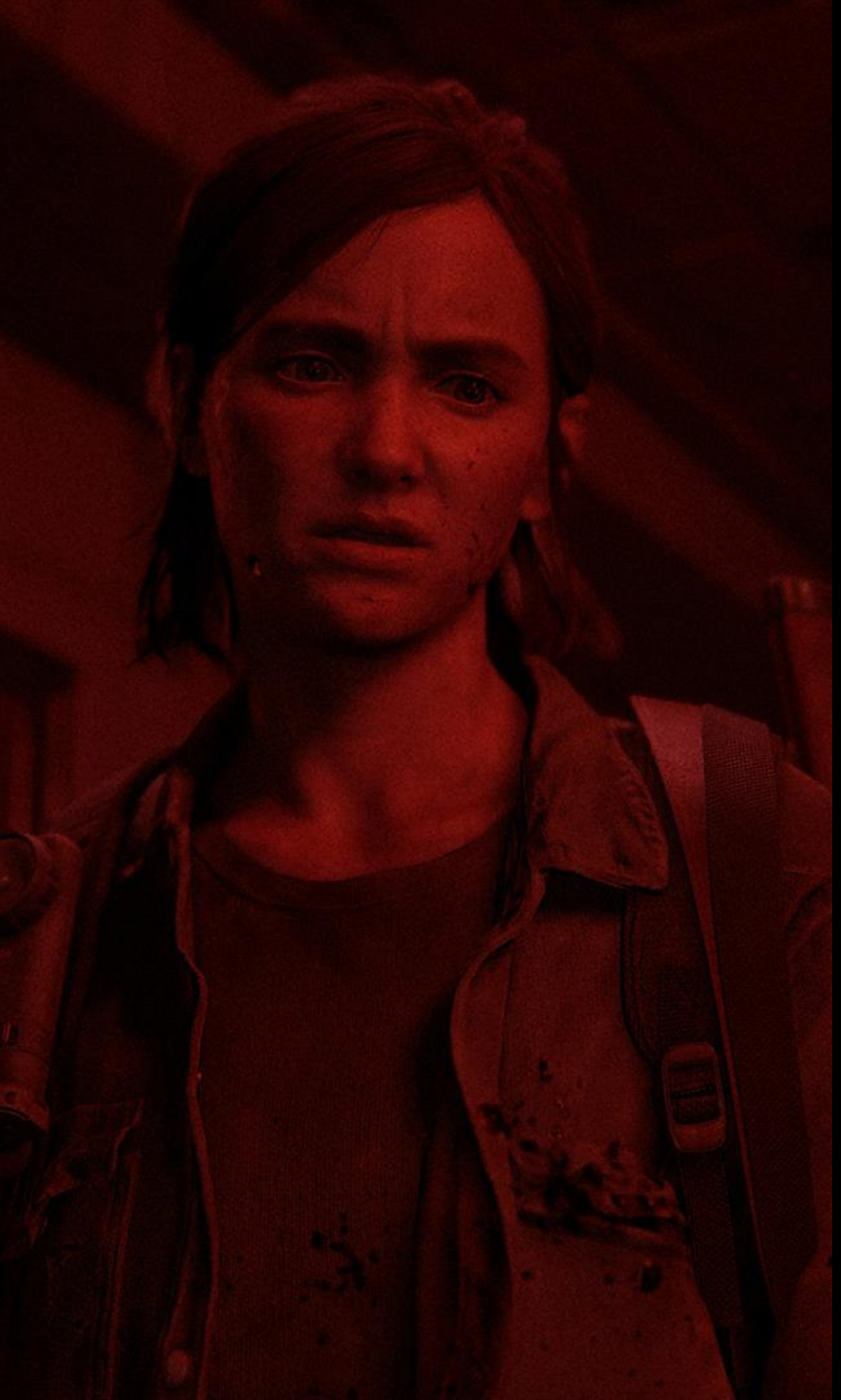
- Index search ~ 0.1ms
- Complete update 0.5–1.0 ms depending on the complexity of the model and trajectory
- The old locomotion system took 2–3 ms a frame
- Can be parallelized
- The infected horde NPCs used a move set with limited coverage
- We could easily afford more if we didn't update every frame



Summary: Pros

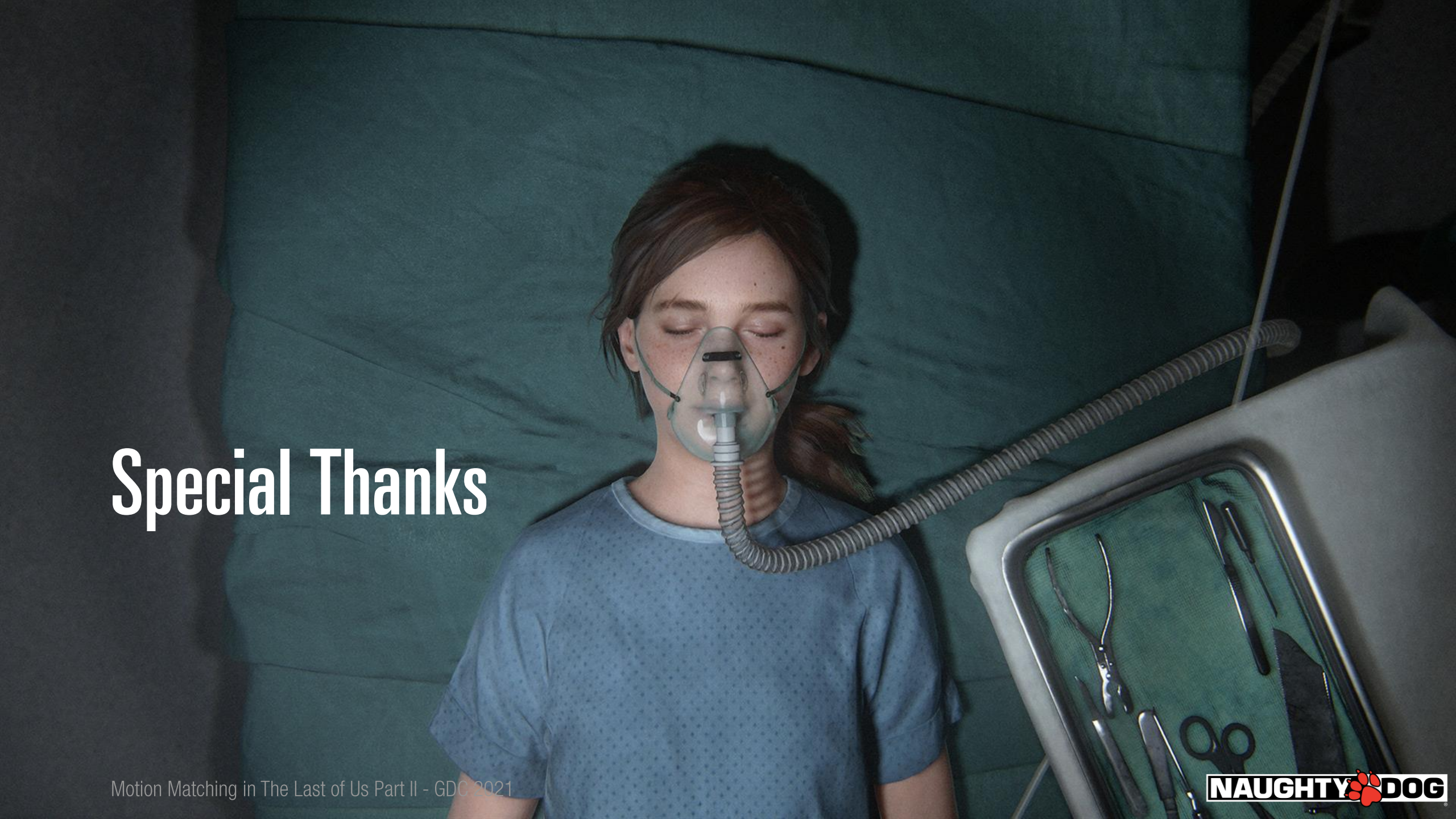
- Very high animation connectivity within a single locomotion and between different sets
- Much more organic feel thanks to basically unlimited number of transition options
- Transitioning from “canned” animations into a locomotion set comes for free
- Higher stability of locomotion sets
- Faster in runtime
- Once all-cycle pipeline is established and tutorialized creating new* high-quality sets is quite straight forward

*new sets that are mechanically-similar to the existing



Summary: Cons

- Requires technically minded animators
- Rather memory hungry
- High upfront cost of full-cycle technology & pipeline
- Motion Model creation can easily become a bottleneck
- Relies on large MoCap stage access bandwidth
- Many stunts struggle with left foot and right foot versions of all moves while keeping the performance natural



Special Thanks

John Bellomy
Principal Programmer
@cowbs

Programmers

Eli Omernick

Ian Jones

Animators

Troy Slough

Ari Flesch

Laura Swartz

Morgan Earl

Jason Lei


We are hiring!
https://www.naughtydog.com/careers
NAUGHTY DOG