



March 20-24, 2023
San Francisco, CA

Hitting the Right Notes: Tying Gameplay to Music in Soundfall

Nick Cooper
Game Director

#GDC23

My Background

- Started getting into game dev during high school
- Worked in AAA most of my career



- Went indie, co-founded Drastic Games, and began working on Soundfall in 2016

Soundfall

- Rhythmic Top-Down Looter-Shooter
 - Entire world moves in-time with the music
 - Players are rewarded for acting on-beat
 - Levels, Encounters, Enemies, and Loot are procedurally generated based on music
 - Supports player-imported tracks on PC
- Released May 11, 2022
 - Steam, Switch, PS5, PS4, Xbox Series X|S, Xbox One
 - Developed in Unreal Engine 4



SOUND FALL



Everything is Tied to the Music

- Everything moves to the beat
- Player actions are tied to the beat
- The experience of the level feels connected to the song



Overview

- Tying Gameplay To Music Outside of Traditional Rhythm Games
- Syncing Everything to the Beat
- Additional Challenges
- Using Music to Influence Procedural Generation
- Beyond Soundfall

Tying Gameplay To Music outside of Traditional Rhythm Games

Tying Gameplay to Music



Super Mario Bros. 3



New Super Mario Bros. U

Tying Gameplay to Music

Sounds in Super Mario
Odyssey Harmonize with the
Background Music

www.youtube.com/@Jalopes



Tying Gameplay to Music



The Legend of Zelda: Ocarina of Time

Tying Gameplay to Music



Fortnite



Fortnite Marshmello Concert

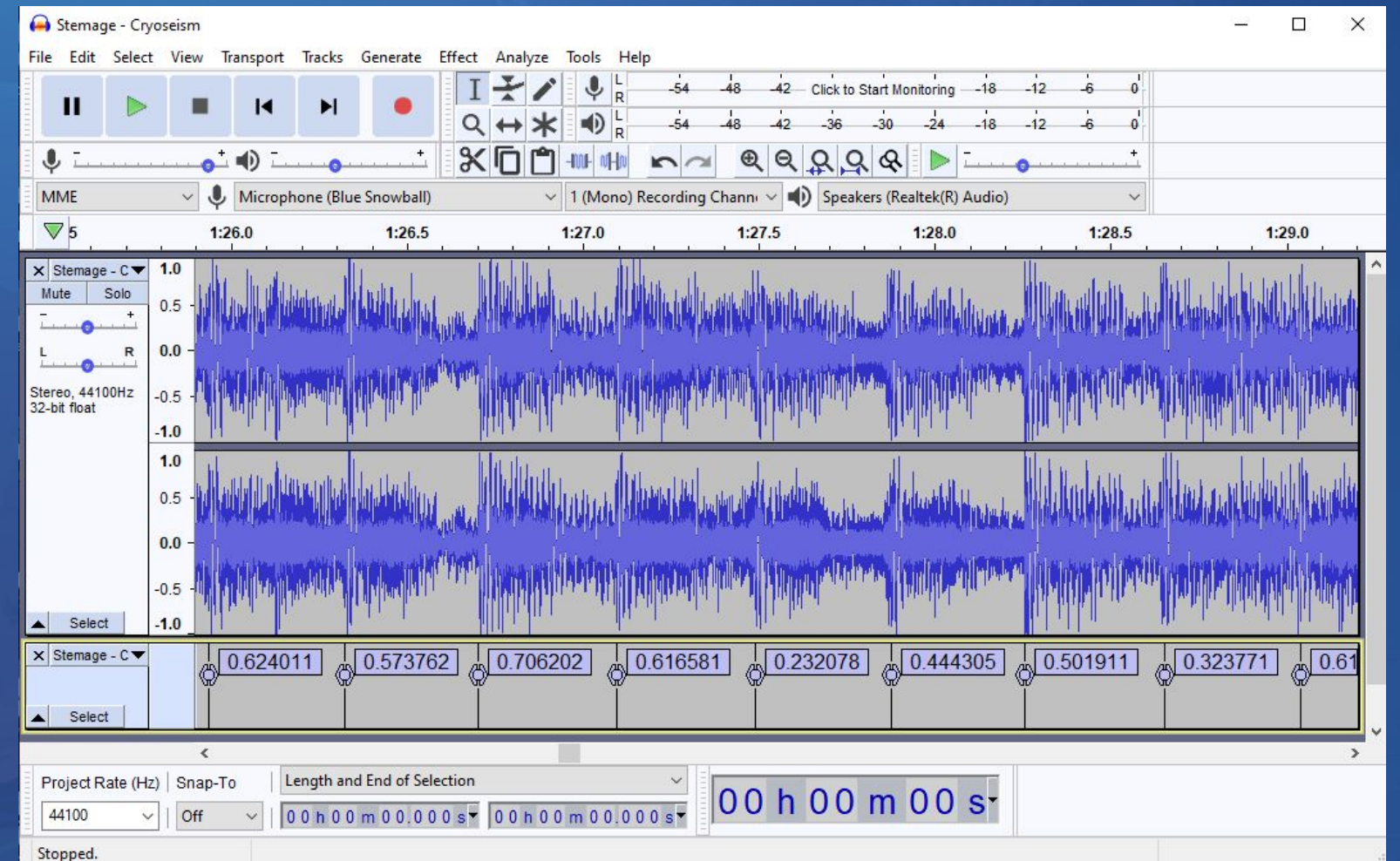
Syncing to the Beat

March 20–24, 2023 | San Francisco, CA #GDC23

GDC

Beat Detection

- Initial beat detection is algorithmic
 - Essentia open source library:
<https://essentia.upf.edu/>
- Manual fixups as necessary

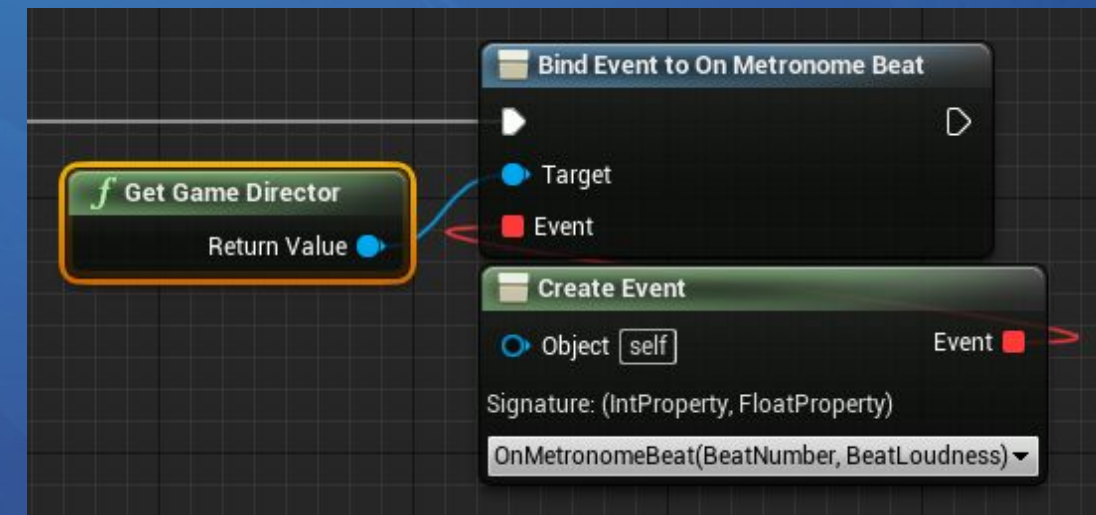


Audacity

Organizing Beat Info

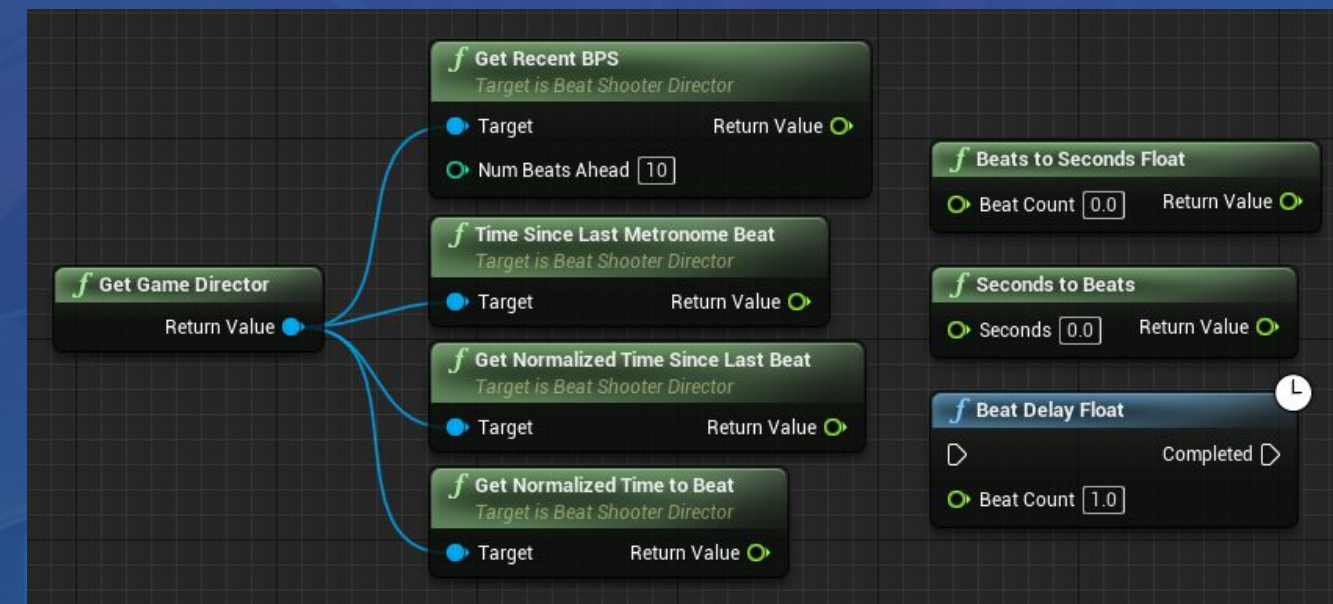
- Global singleton GameDirector

- Handles audio analysis
- Beat and frequency distribution data
- Handles checking if an input was on-beat
- OnMetronomeBeat event



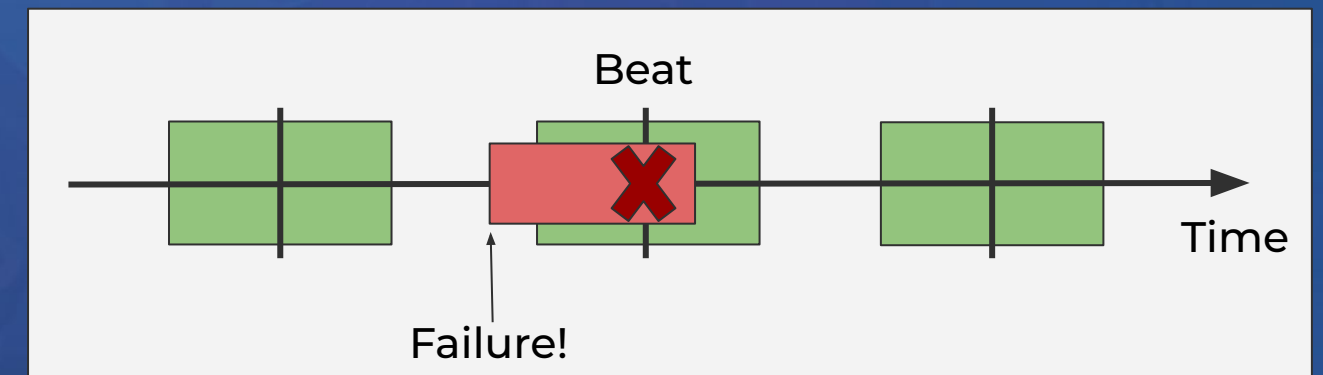
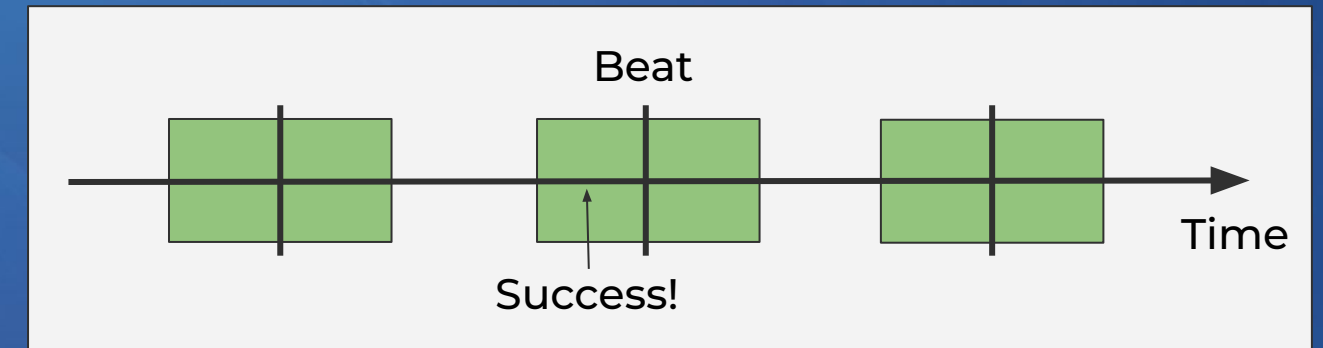
- Helper functions

- BeatsToSeconds()
- SecondsToBeats()
- TimeSinceLastBeat()
- TimeBetweenBeats()
- NormalizedTimeBetweenBeats()



On-Beat Input

- Beat input success window
 - Measured in Beats
- Beat input failure disables success on nearby upcoming beat
- Window is very generous - 28% before and after the beat!



Converting from Beats to Seconds

- Direct BPM conversion?
- Better - Upcoming BPM!
- Problem: We saw beat input could be quite early or late
- Round to start-on-beat
- Round to end on-beat

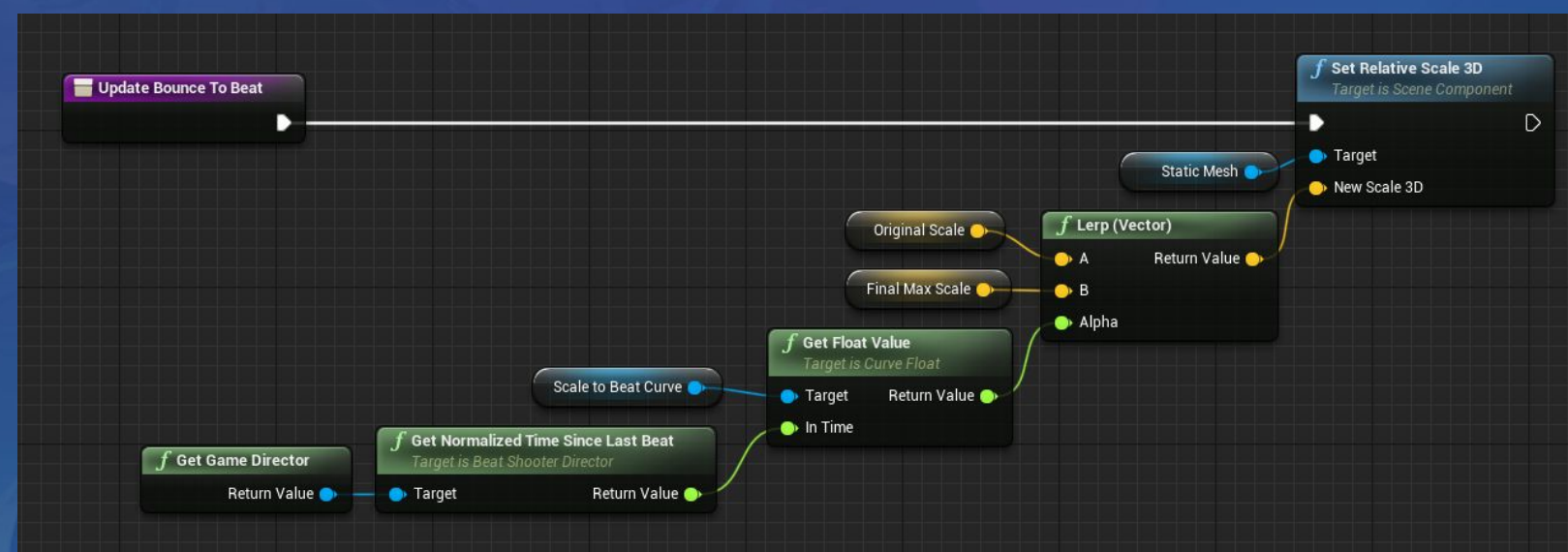
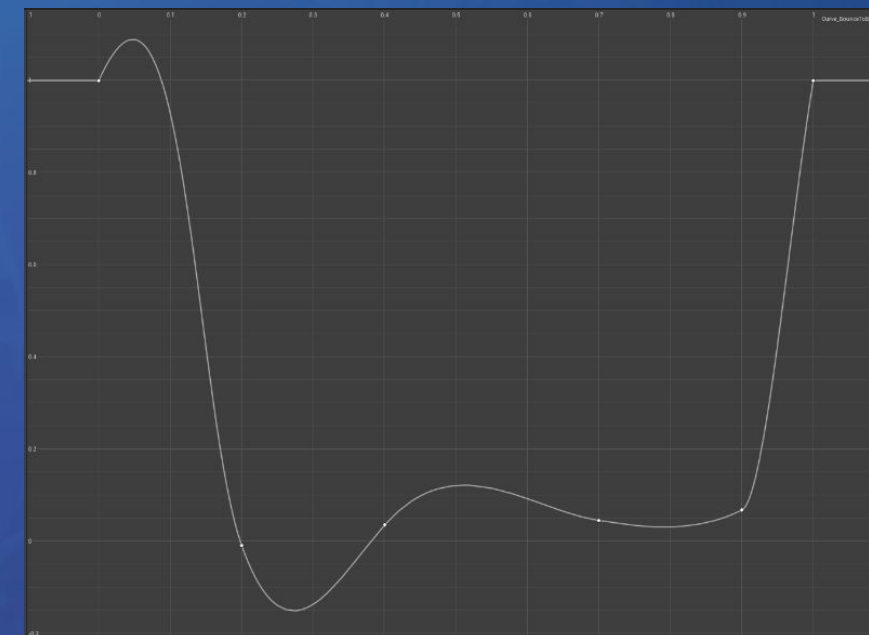
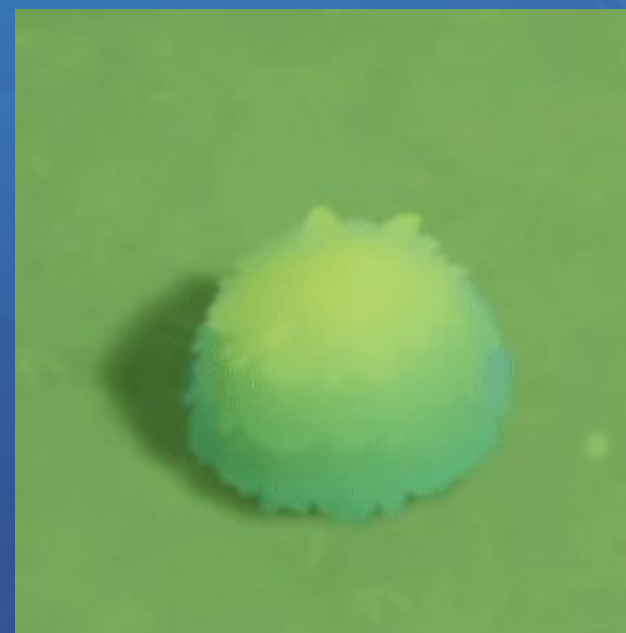


Animation



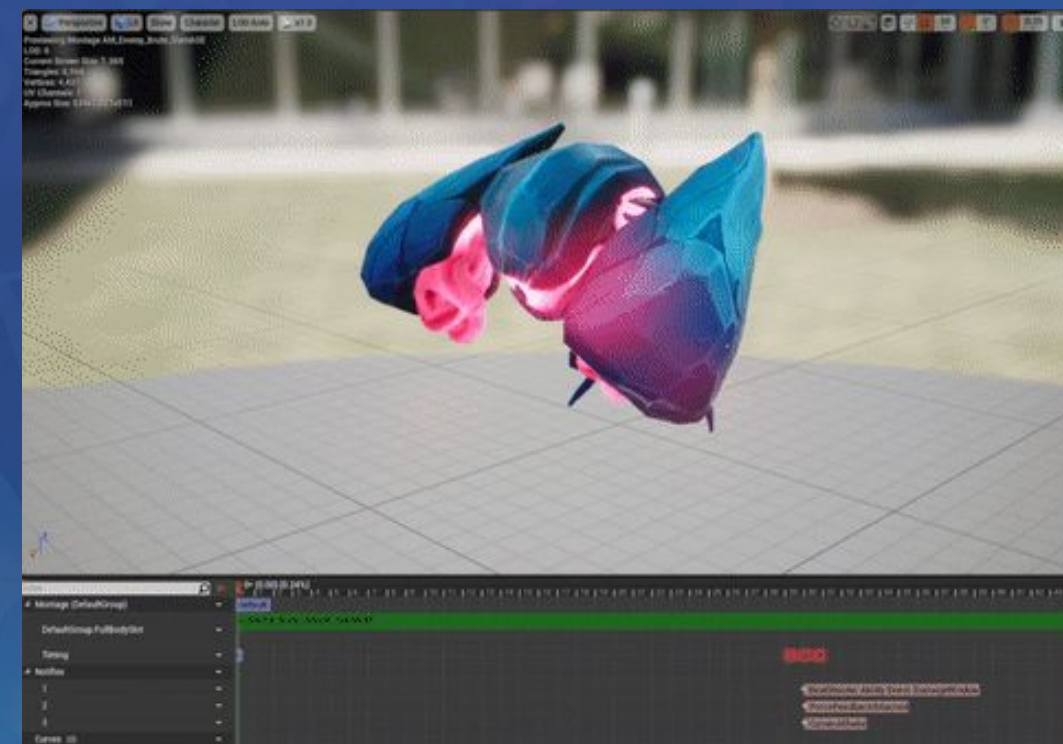
Static Mesh Bouncing

- Simpler environmental assets were not rigged, just used StaticMeshes
- StaticMeshComponents scaled via a curve in Tick()
- GetNormalizedTimeSinceLastBeat()
 - Returns a value 0.0 to 1.0

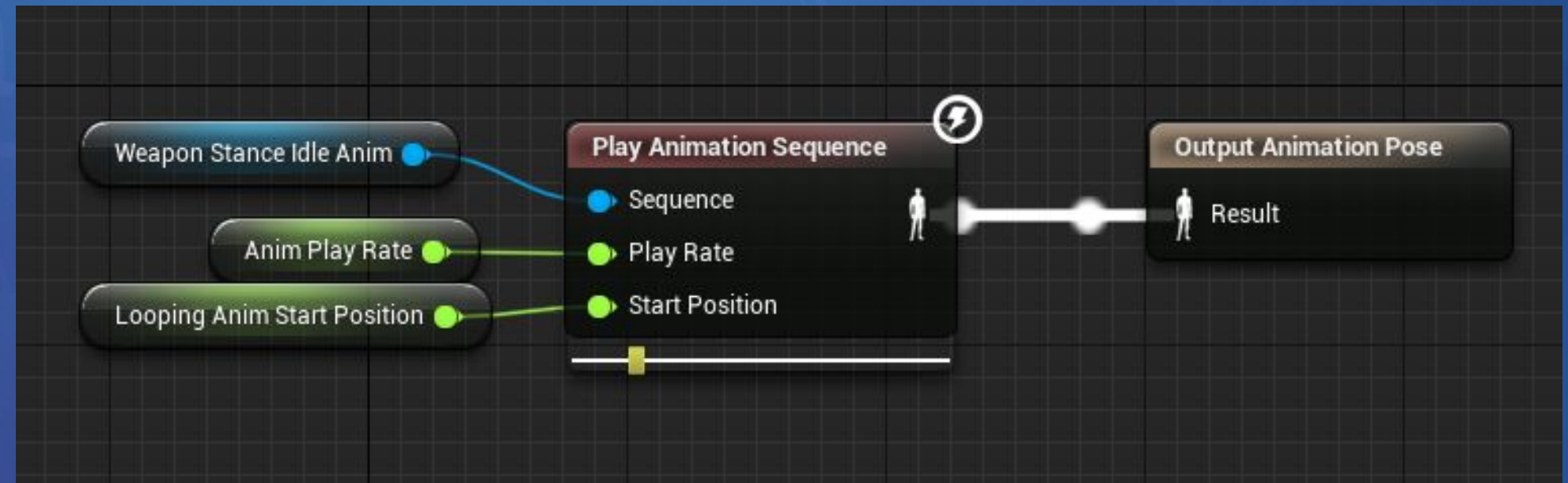


Creating Animations

- Authored assuming 120 Beats Per Minute
- 1 Beat -> 0.5 seconds
- All impactful moments of animations need to occur at a multiple of 0.5 seconds

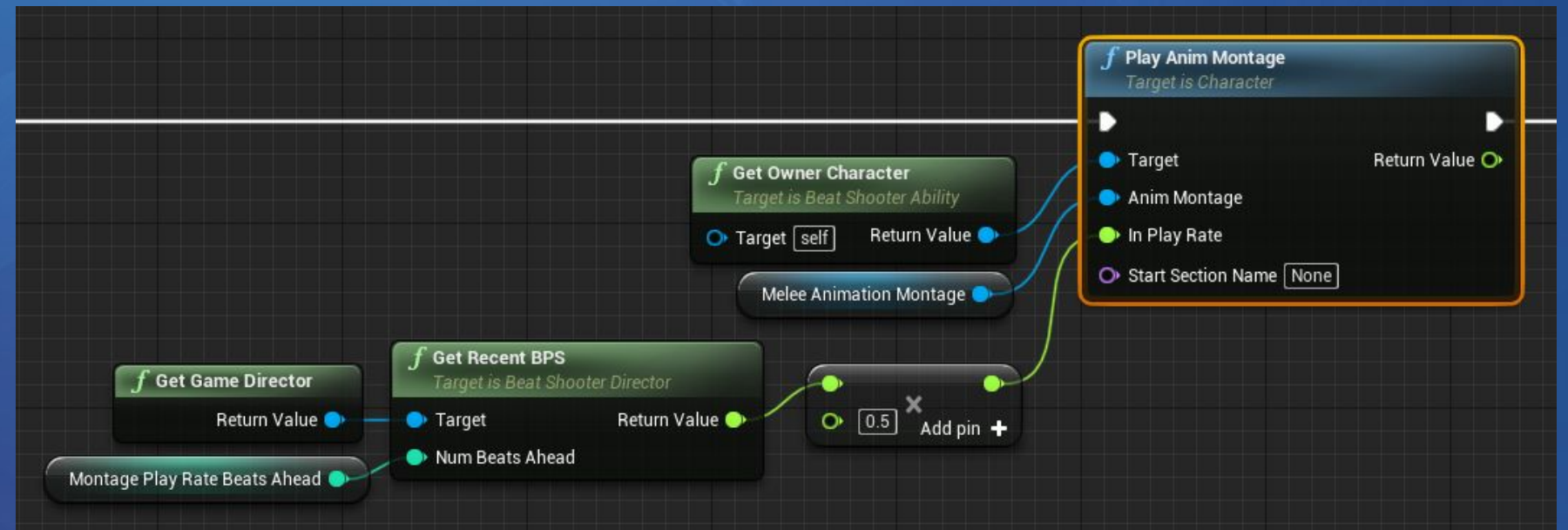


Playing Looping Animations



- Play Rate
 - UpcomingBPS = Beats per second over the upcoming 4 beats
 - Since the anim is authored at 120 BPM, 1 beat == 0.5 seconds
 - **$PlayRate = UpcomingBPS * 0.5$**
- Starting Position
 - Get time since the last beat
 - Multiply by the play rate we calculated
 - **$StartPos = TimeSinceLastBeat * PlayRate$**
- Feed these values into our anim sequence

Playing Animations from Abilities

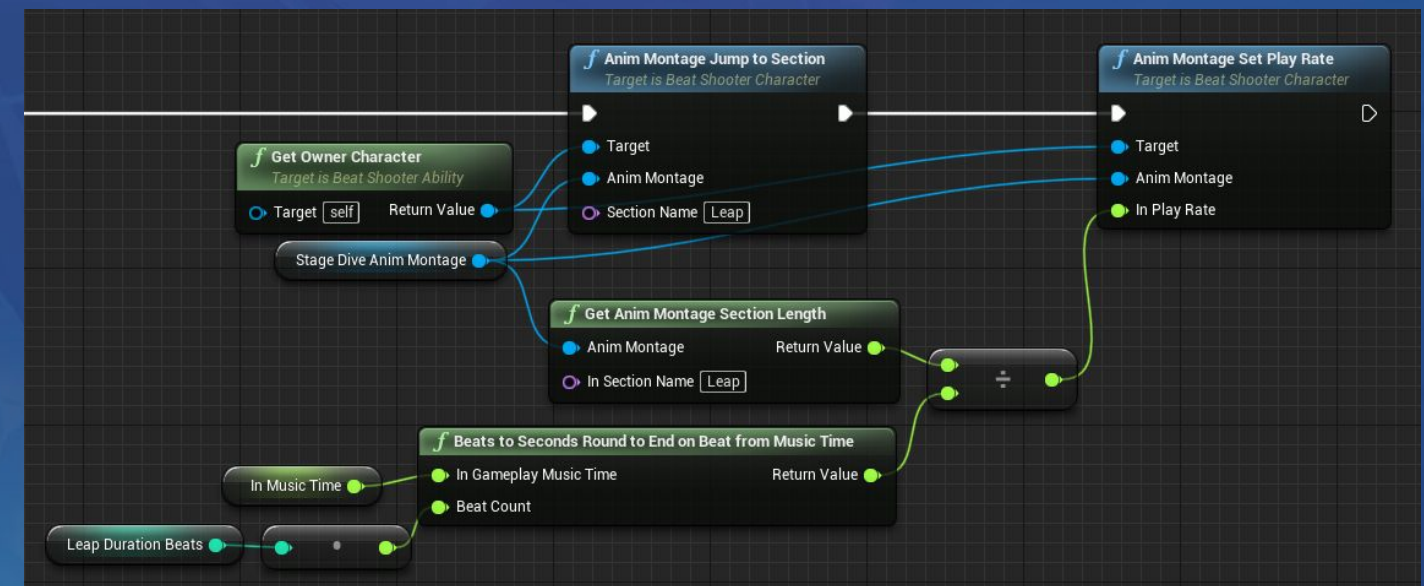


AnimMontages

- Can adjust play rate in the same way, using upcoming beats-per-second
- In this case though, always want to start at the beginning of the anim

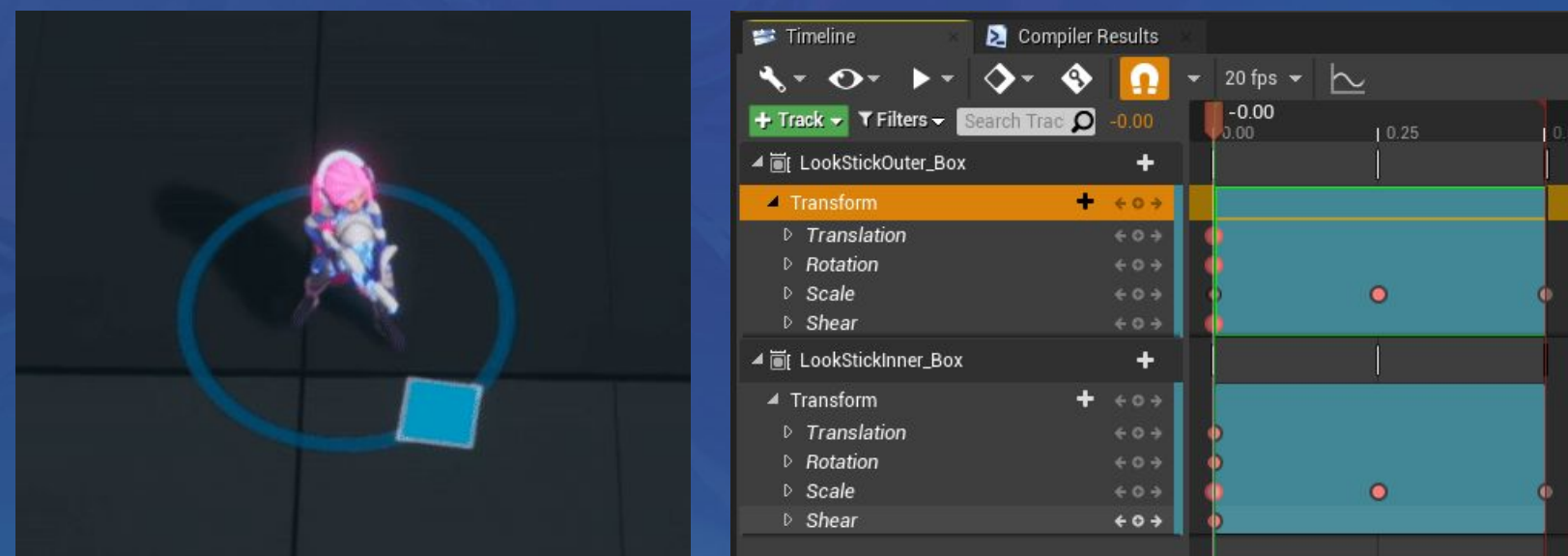
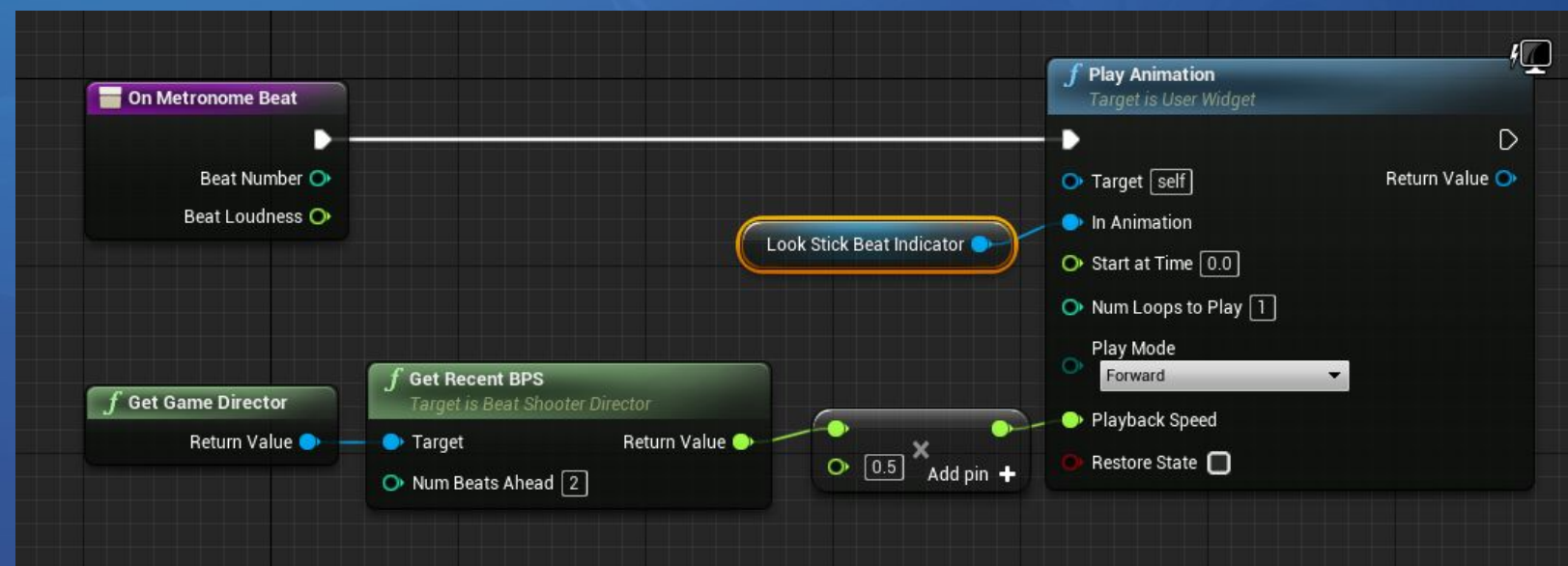
Playing Animations from Abilities

When jumping between sections, we need to recalculate PlayRate

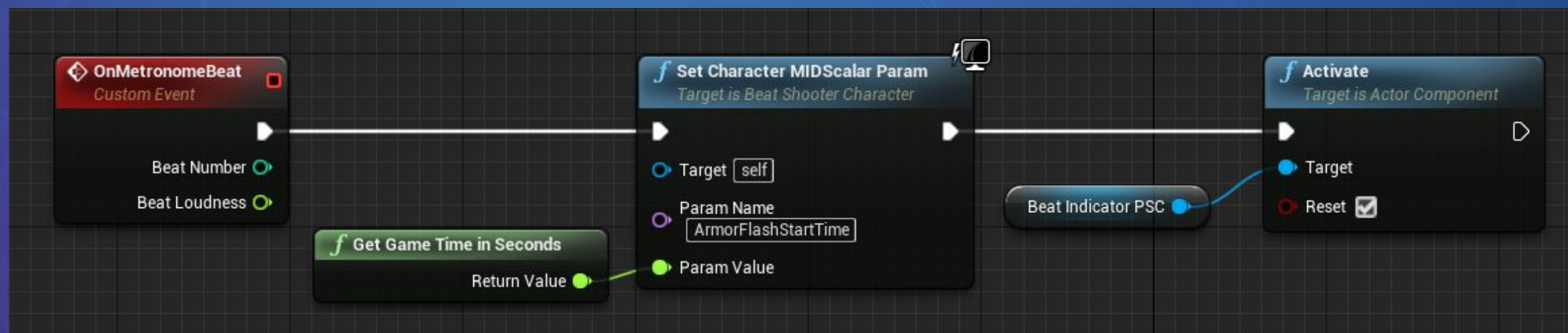


UI Animations

- Also authored assuming 120 BPM, with impactful moments occurring at multiples of 0.5 seconds
- Triggered as one-shots on metronome beat
- Or looping with StartAtTime using NormalizedTimeSinceLastBeat()



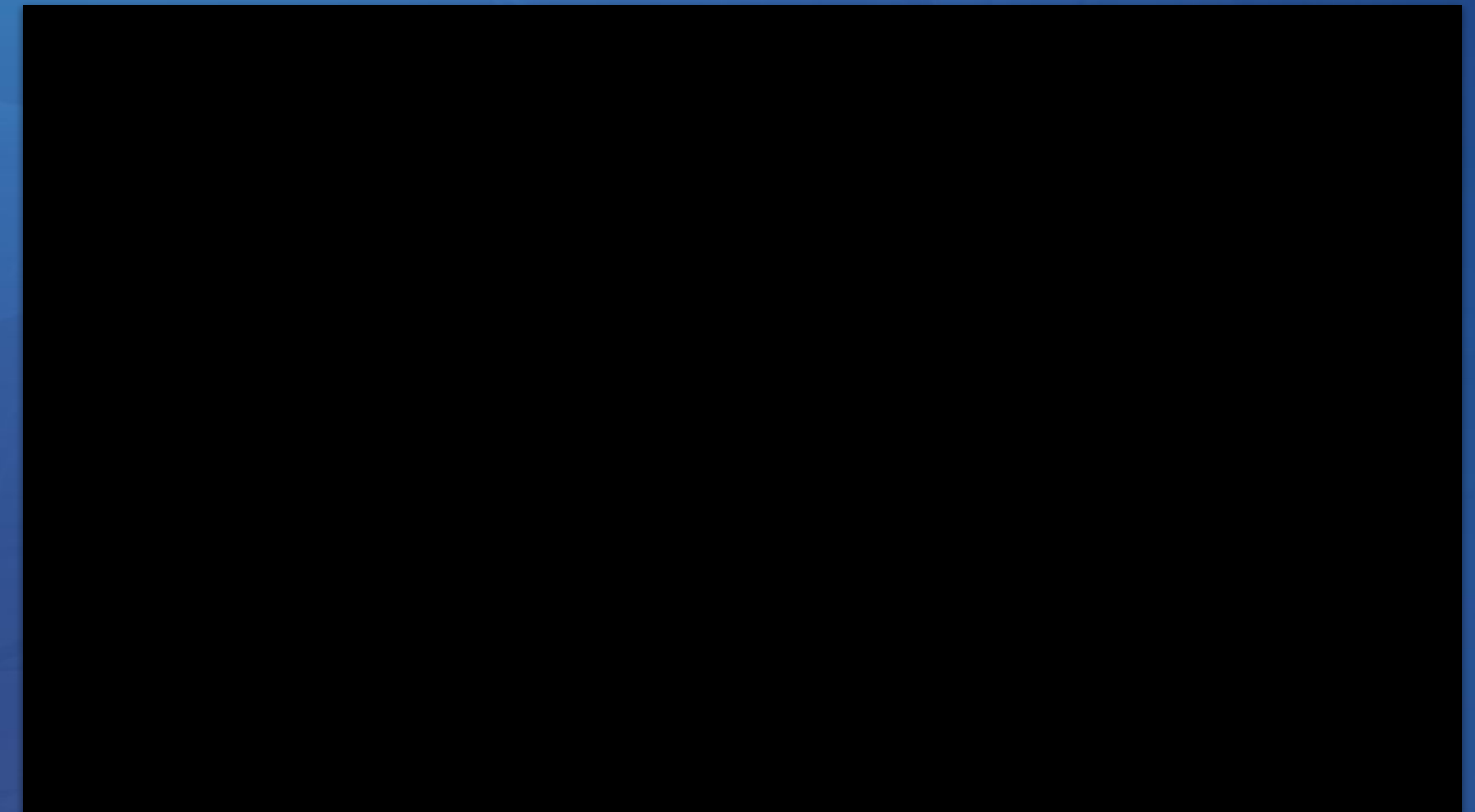
VFX and Materials



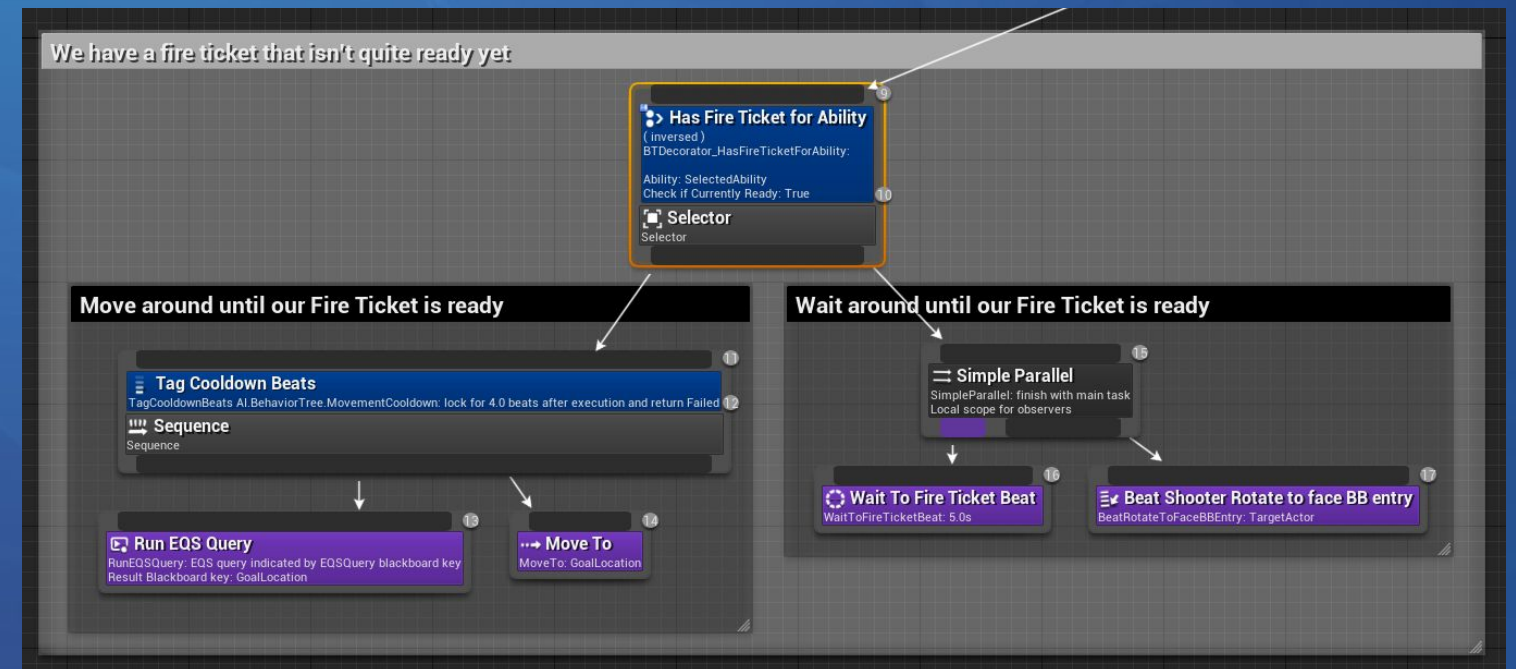
On-beat VFX & material parameters were either tied to animations and abilities which we've already synced, or triggered using OnMetronomeBeat() events

Sound Effects

- Looping SFX were authored at 110 BPM
 - Sounded better after scaling than 120 BPM
- Created a BPM Fmod parameter, used for scaling the play rate



Enemy Behaviors



- AI Manager and Fire Ticket system
 - Fire tickets have a beat index for when the AI should begin the intended ability
- Custom Behavior Tree nodes for waiting for Fire Ticket beats, or waiting for a specified beat count
- Once an enemy ability starts, its execution will continue to be on-beat due to animation timing we discussed earlier

Additional Challenges

Performance

Problem: A **lot** of stuff happens on-beat, so that makes beat ticks prone to hitches

- Moved anything that didn't absolutely need to be on-beat, to an off-beat tick
 - PreMetronomeBeat and PostMetronomeBeat events for actions that don't need to be perfect
 - Disable ticking on all actors outside of current and adjacent chunks
- 
- Moved enemy spawning to off-beat, but hidden with collision disabled

Networking

We somehow need to keep this all synced, and feeling good with network latency

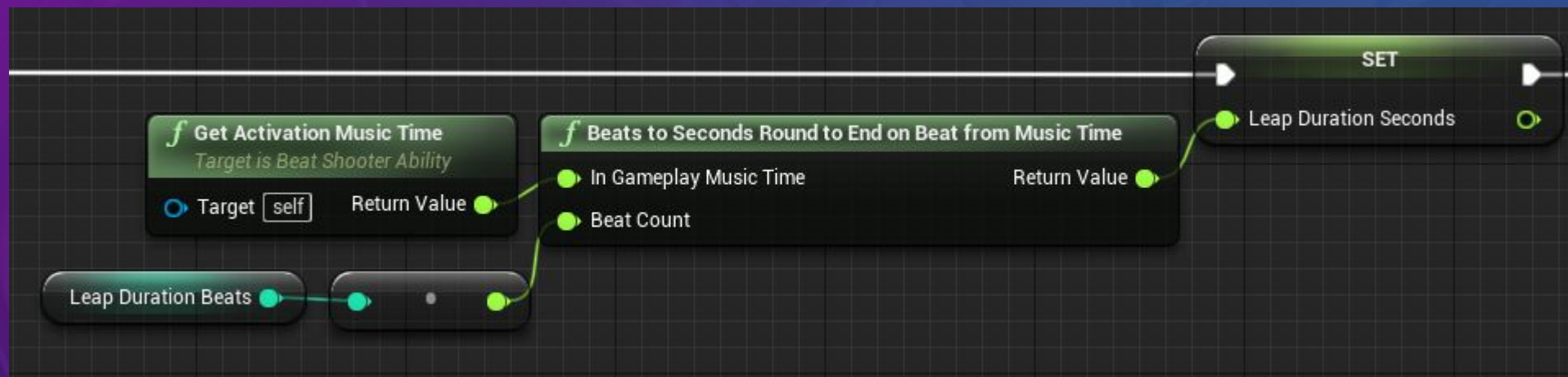
- Priority: **You** need to feel awesome and on-beat
- Would be nice for friends you're playing with to appear on-beat from your perspective - *but this is nowhere near as important*, as long as they feel on-beat **from their perspective**



Networking

Solution: Trust the client way more than most games do!

- A client says they did something on-beat, and they reasonably could have? OK!
- Non-gameplay anims are always based on the local music time
- Gameplay actions pass a MusicTime parameter on RPCs
- Dampen all sounds instigated by remote players



Using Music to Influence Procedural Generation

Music Categorization

We want the experience of the level to feel connected to the song

Genre

Classified into 12 different genres, which we then mapped into 10 different biomes:



Pop - Skyland



Latin - Beach



Classical - Forest



Rock / Metal - Volcano



EDM - Cityscape



Jazz - Depths



Acoustic / Country - Desert



Chiptune - Crypt



Hip Hop - Jungle



R&B - Peaks

Music Categorization

Mood

- Came up with “Family” categorization - Orchestral, Digital, Primal
- Mapped to alteration types on loot and enemies



No Alteration



Orchestral



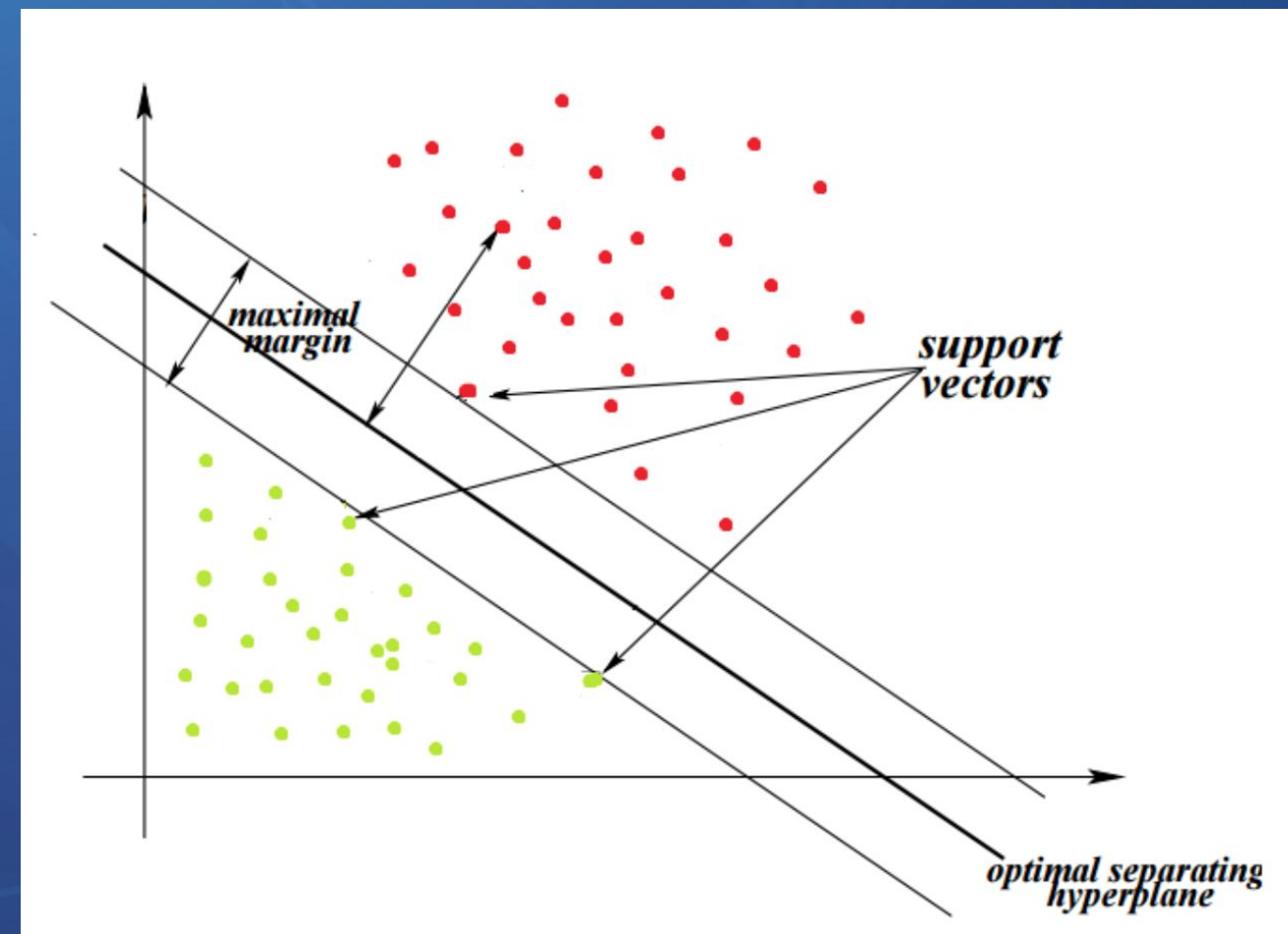
Digital



Primal

Music Categorization

- Support Vector Machines
 - Supervised Learning model useful for classification
 - Calculates a hyperplane that separates data into classes, maximizing the separation between the classes
- Used LIBSVM open source library



Source: stackexchange.com

Music Categorization

- Built a large training set by running a large set of audio analysis algorithms on 867 songs that covered each of the 12 genres fairly evenly
- Labeled each by “Genre” and “Family” manually - fairly subjective in many cases
- Initial set of audio analysis algorithms generated 670 parameters per song

Music Categorization

- At runtime, for imported songs, all of these algorithms would need to be run, and then classified.
- Accuracy was around ~70% for each axis (genre and family)
- Running all of these algorithms was *slow*!
 - With trial and error, reduced algorithm count
 - Ended up with 88 parameters while maintaining ~70% accuracy

Music Categorization

~70% accuracy is obviously far from perfect, but,

- Music Categorization is inherently subjective!
- The player doesn't see the claim about either genre or family directly
- Getting surprised by an unexpected result for a song can be fun!

Audio Analysis -> Level Generation

- Biome selected based on Genre
 - Biome = Persistent Level + Chunks + Subchunks + Skybox Chunks
- Chunk - rectangular sublevel with one or more “chunk connectors”
- Subchunk - rectangular sublevel that slots onto a Subchunk Marker within a Chunk
- Skybox Chunk - rectangular sublevel that fills area outside of the playable space



Audio Analysis -> Level Generation

Level Generation:

- Build a main path out of Chunks
- Add Chunks for branches off main path
- Fill in all Subchunk Markers with appropriate Subchunks
- Fill in Skybox under and around the completed playable space



Audio Analysis -> Level Generation

- Level generation tries to make the time to complete the level match the duration of the song. Sum of:
 - Encounter Estimate - Based on the number of “encounter points” spent in generating an encounter
 - Traversal Estimate - Based on the XY dimensions of the chunk
- Additional parameters:

Audio Analysis Result	Level Generation Parameter	What it influences in Level Generation
Mean Beat Loudness	EncounterRating	Proportion of chunks that are encounter-focused
Mean Spectral Energy	TravelRating	Walking distance travelled to complete level
Mean Spectral Complexity	BranchingRating	Number of branches generated
Number of Tempo Changes	TwistynessRating	How straight vs. bendy the level is
Onset Rate	PrimaryLevelDirection	General direction from start of the level to end
Danceability	TreasureChestsPerMinute	Number of treasure chests to generate per minute of music

Audio Analysis -> Probabilities

Defining probability weighting

- Default curve mapping probability weight to level
 - “Level” being Mission Intensity or Item Level depending on context
- GameplayTags associated with the Mission can modify this
 - For instance, tags could be *Music.Family.Orchestral* or *Environment.Volcano*
 - Tags can cause a multiplier to be applied onto the default curve, or cause an override curve to be used

Audio Analysis -> Encounters and Enemies

- Genre / Biome
 - Biases Encounters toward certain enemy types
 - Allows / Disallows / Biases elemental enemy types
- Family
 - Biases other enemy alterations and enemy weapon types



No Alteration



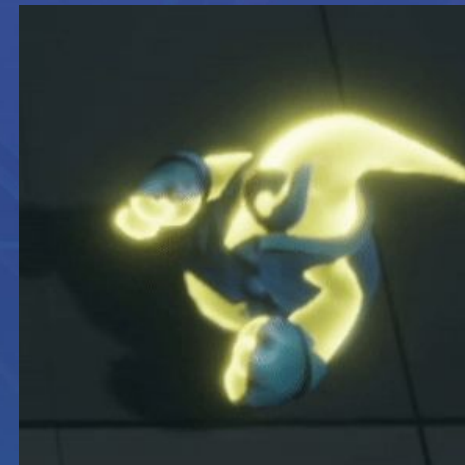
Bolero
Volcano / Jungle



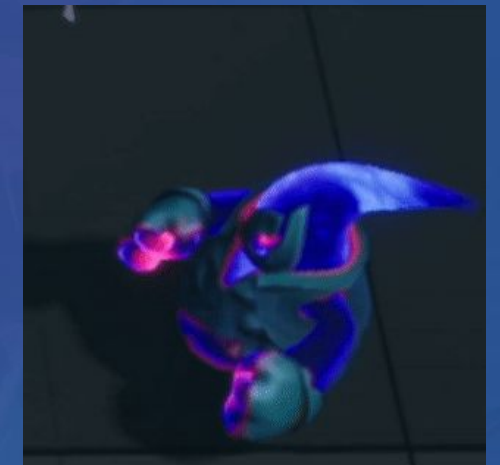
Serenade
Skyland / Peaks



Minuet
Forest / Desert



Requiem
Beach / Underwater



Nocturne
Cityscape / Crypts

Audio Analysis -> Loot

- Genre / Biome - Biases elemental type
- Family - Biases all other alteration types



Orchestral



Digital



Primal

Everything is Tied to the Music

- Everything moves to the beat
 - Global “metronome” heartbeat
 - All timing is in beats, not seconds
- Player actions are tied to the beat
 - Round and fudge Beat time to encourage and enhance this
- The experience of the level feels connected to the song
 - Aspects of the music drive every part of our procedural systems



Beyond Soundfall

March 20–24, 2023 | San Francisco, CA #GDC23

GDC

Other Things We Tried

- Key / Chord Detection
- Player generated music
- On-the-fly beat detection

Applications for Other Projects

- Syncing background actors to music
- Better emotes - synced to background music
- Syncing actors to player-generated music
- Syncing actors to things like combos or rapid-fire weapons which have an inherent “beat”

Thank you!



Nick Cooper

drasticnick@gmail.com

[linkedin.com/in/drasticnick/](https://www.linkedin.com/in/drasticnick/)

[@DrasticNick](https://twitter.com/DrasticNick)

[@SoundfallGame](https://twitter.com/SoundfallGame)

[soundfallgame.com](https://www.soundfallgame.com)