

INSIDE

Sonic College 2020
Jakob Schmid

Overview

- Introduction
- Audio Engine Overview
- Animation Events
- Cloth
- Voice Sequencer
- Voice Configuration
- Wrapping Up



INSIDE

PLAYDEAD

Xbox One, PS4, PC, iOS, Switch

Developed over 6 years

Game Developers Choice Awards 2016

Best Audio, Best Visual Art

Game Critics Awards 2016

Best Independent Game

The Game Awards 2016

Best Art Direction, Best Independent Game

DICE Awards 2016

Spirit Award, Art Direction, Game Direction

13th British Academy Games Awards

Artistic Achievement, Game Design, Narrative, Original Property

The Edge Awards 2016

Best Audio Design

INSIDE

Playdead

Released LIMBO in 2010

Copenhagen-based

Around 25 employees in 2016



INSIDE Audio Team

A dark, industrial setting with a person on the left holding a flaming torch and a person on the right wearing a headlamp and holding a megaphone. The scene is dimly lit, with the primary light sources being the torch and the headlamp. The background shows structural elements of a building, possibly a warehouse or factory.

Martin Stig Andersen

audio director, sound designer, composer

Andreas Frostholt

sound designer

Søs Gunver Ryberg

composer, sound designer

Jakob Schmid

audio programmer

Audio Engine Overview



INSIDE Technology

Unity

Audiokinetic Wwise

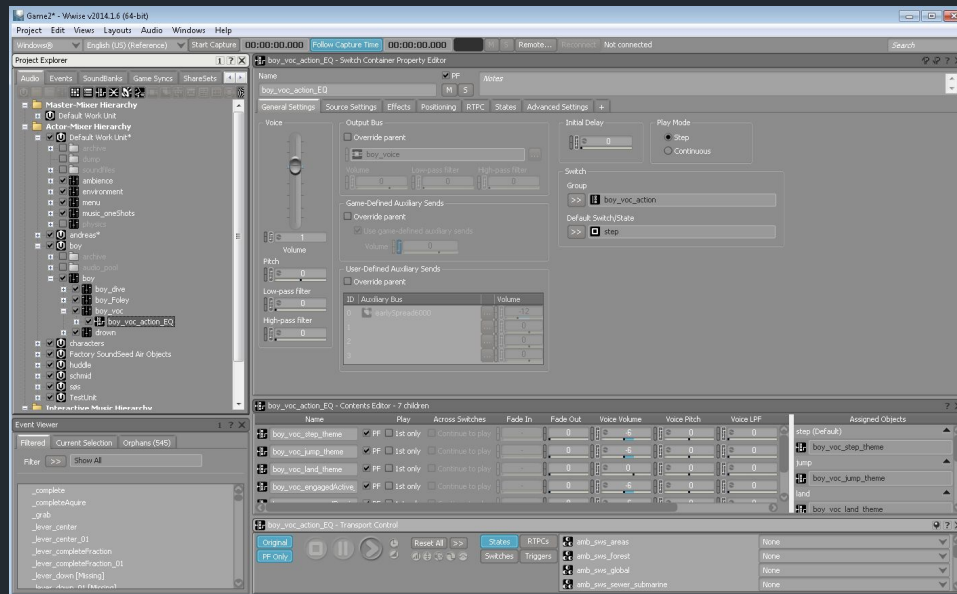
Modified Wwise-Unity plugin

PlayMaker

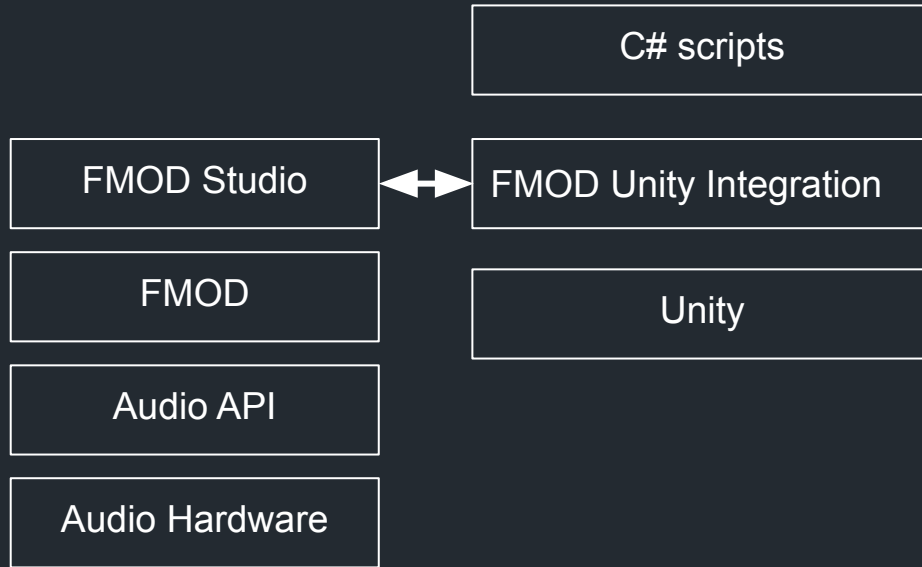
FMOD vs. Wwise

- Both leading sound engines
- FMOD: programmer flexibility
- Wwise: sound designer flexibility

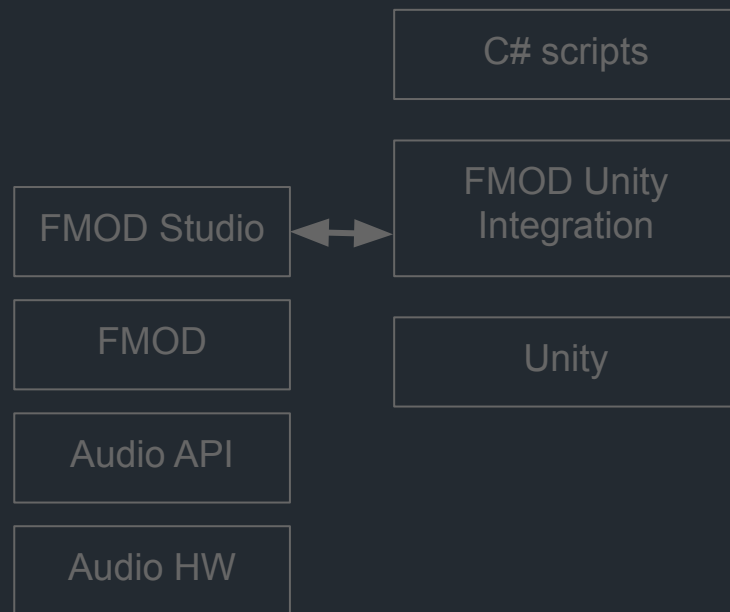
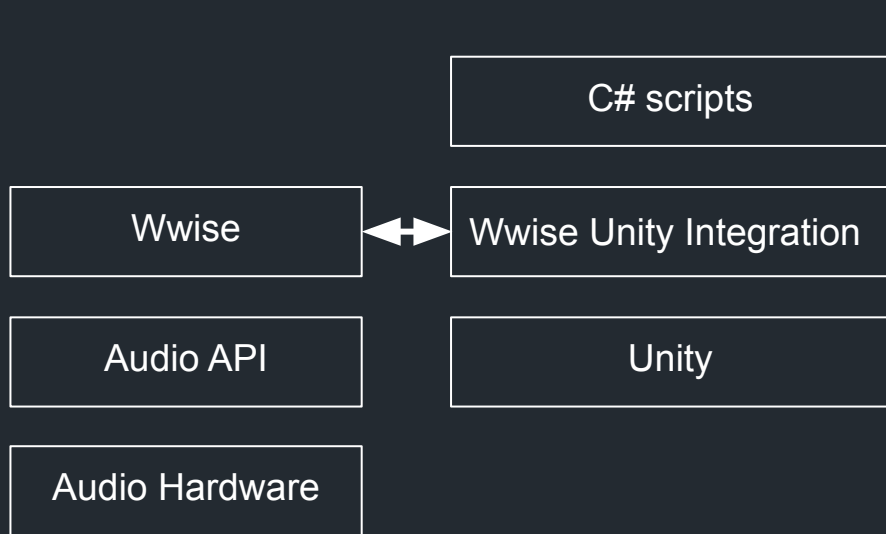
Wwise Authoring Tool (2014.1.6)



FMOD Studio / Unity Setup



Wwise / Unity Setup



FMOD Unity Integration Code

FMOD Unity Integration audio:

```
[FMODUnity.EventRef]
public string mySoundEvent;
EventInstance mySound;

fmod = FMODUnity.RuntimeManager;
mySound = fmod.CreateInstance(mySoundEvent);
mySound.start();

// modify sound while playing
float t;
mySound.setParameterValue("health", t);
```

Wwise Unity Integration Code

Wwise Unity Integration:

```
GameObject go = this.gameObject;
AkSoundEngine.PostEvent("MySound", go);

// Modify sound while playing
// - modification is defined using external tool
float t;
AkSoundEngine.SetRTPCValue("MyParameter", t, go);
```

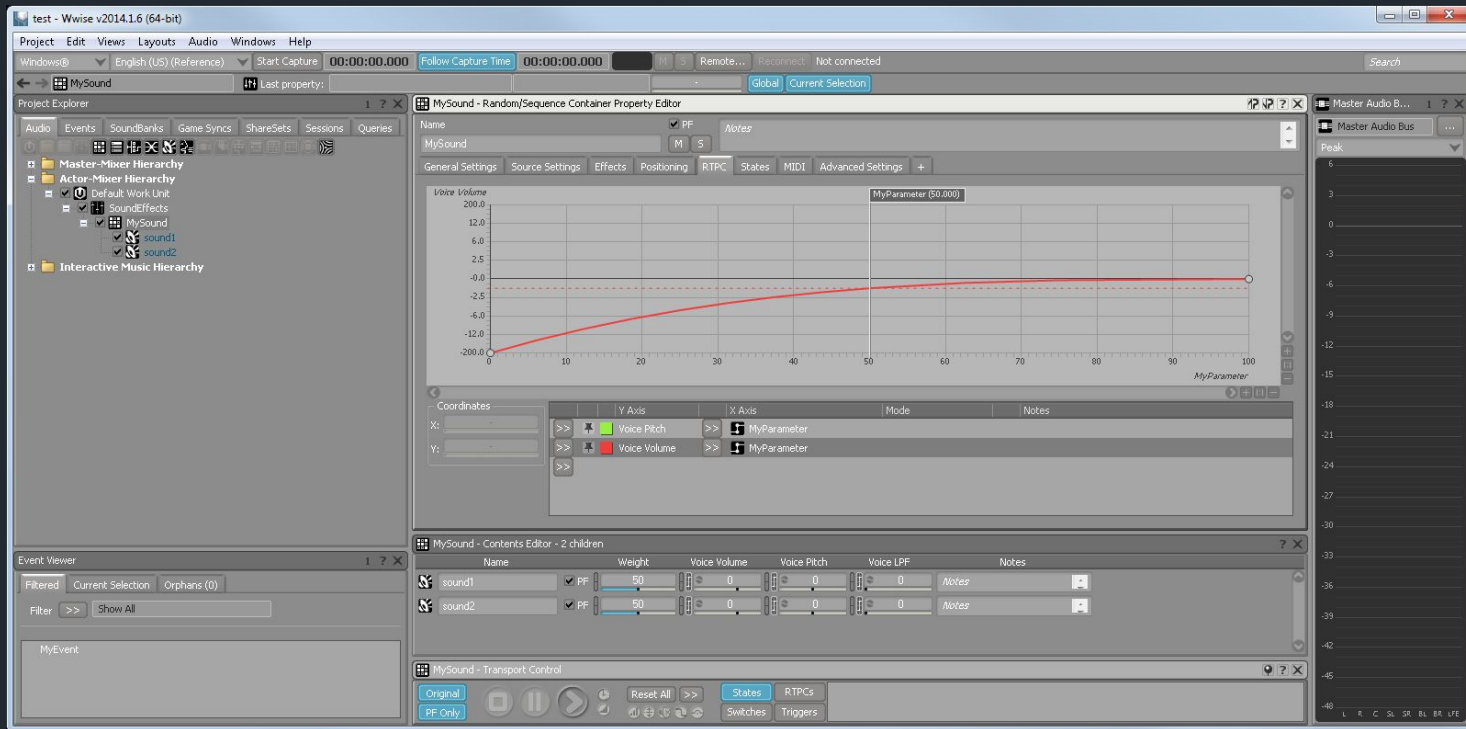
FMOD Unity Integration audio:

```
[FMODUnity.EventRef]
public string mySoundEvent;
EventInstance mySound;

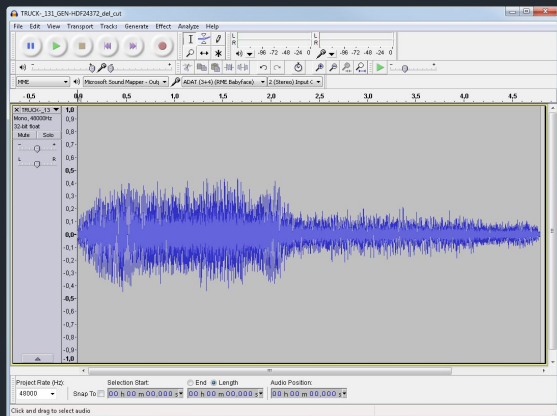
fmod = FMODUnity.RuntimeManager;
mySound =
    fmod.CreateInstance(mySoundEvent);
mySound.start();

// modify sound while playing
float t;
mySound.setParameterValue("health", t);
```

Wwise Workflow: Wwise Authoring Tool



Wwise Workflow: Defining Sounds



Project Explorer

- Audio
- Events
- SoundBanks
 - Master-Mixer Hierarchy
 - Actor-Mixer Hierarchy
 - Default Work Unit
 - SoundEffects
 - MySound
 - sound1
 - sound2
 - Interactive Music Hierarchy

MySound - Random/Sequence Container Property Editor

Name: MySound

General Settings | Source Settings | Effects | Positioning | RTPC | States | MIDI

Voice

Output Bus

Override parent

Master Audio Bus

Volume: 0 Low-pass filter: 0 High-pass filter: 0

Game-Defined Auxiliary Sends

Override parent

Use game-defined auxiliary sends

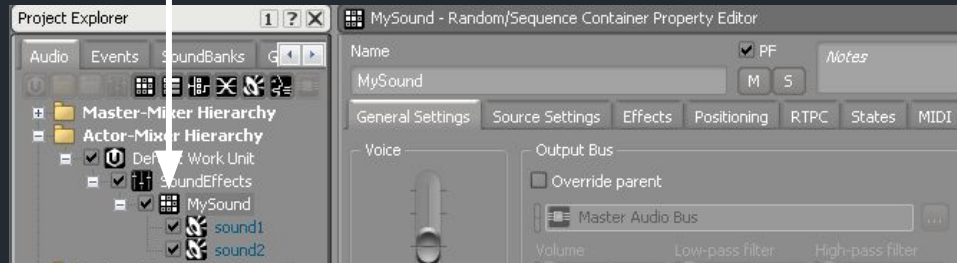
Volume: 0

User-Defined Auxiliary Sends

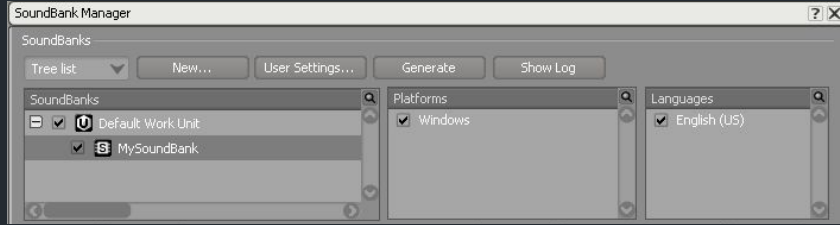
Override parent



ID	Auxiliary Bus	Volume
0		0
1		0
2		0

Wwise Workflow: Events



Wwise Workflow: Output Sound Banks





 Init.bnk	44 KB
 MySoundBank.bnk	7,330 KB

- Sound Banks contain a list of events
- ... And all the sounds used by the events

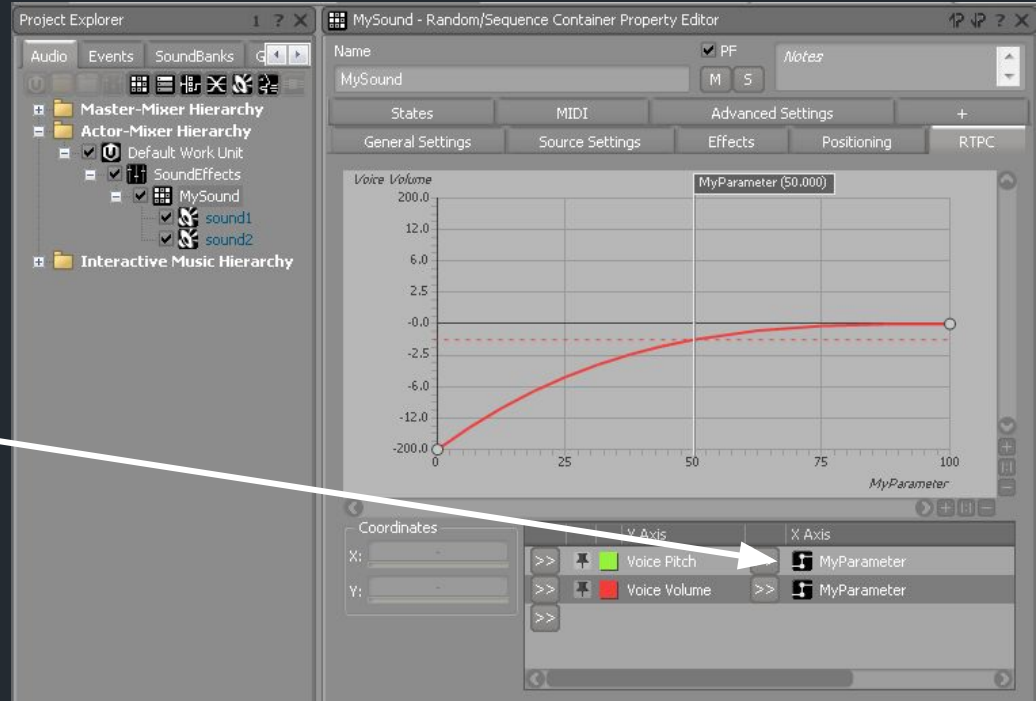
Wwise Workflow: Scripting

```
AkSoundEngine.LoadBank("Init.bnk", ... );  
AkSoundEngine.LoadBank("MySoundBank.bnk", ... );  
  
AkSoundEngine.PostEvent("MySound", go);
```



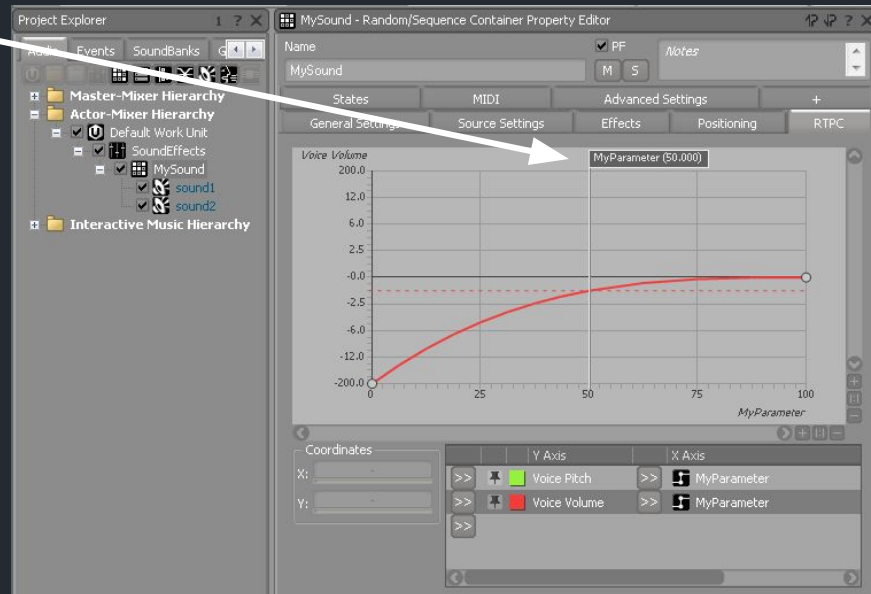
 Init.bnk	44 KB
 MySoundBank.bnk	7.330 KB

Wwise: Modify Playing Sounds



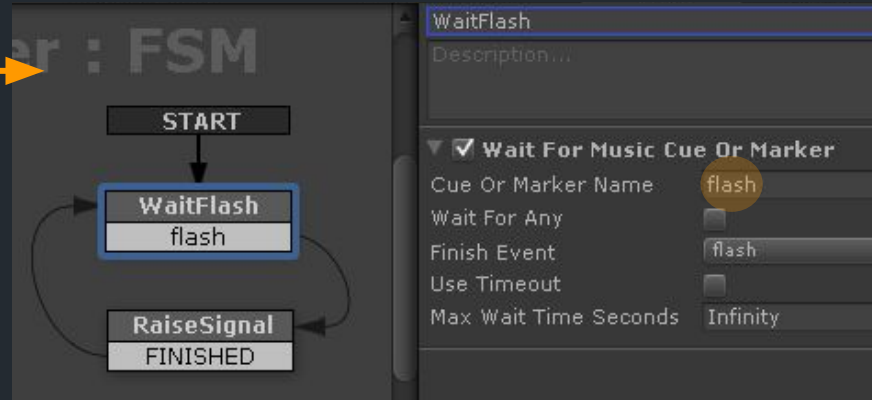
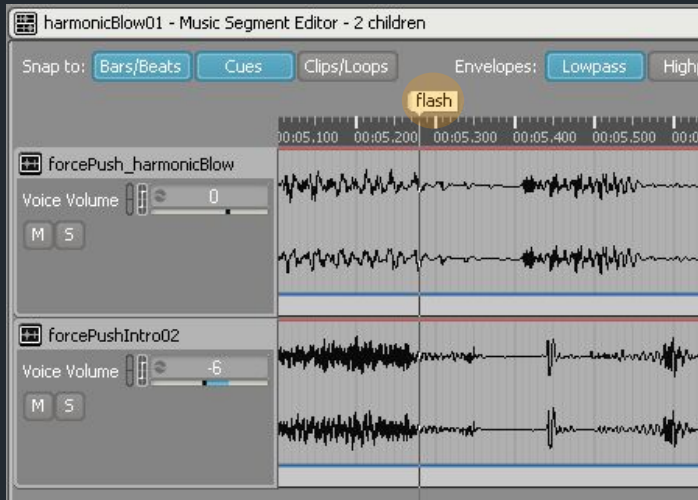
Wwise Workflow: Scripting

```
AkSoundEngine.PostEvent("MySound", go);  
float t;  
AkSoundEngine.SetRTPCValue("MyParameter", t, go);
```



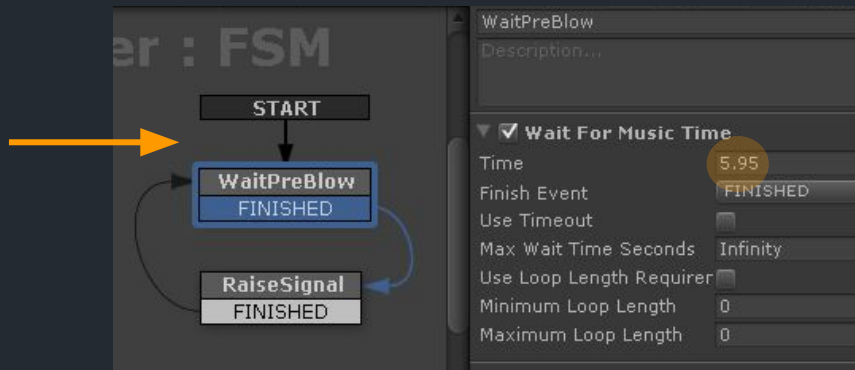
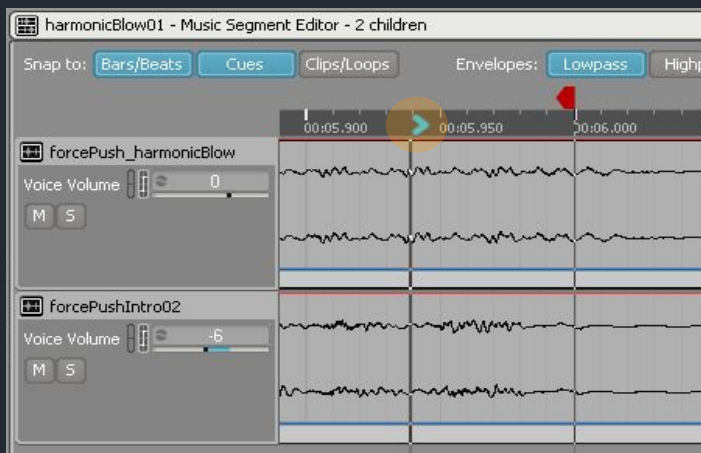
Audio-driven Gameplay: User Cues

- Music segments can have User Cues
- Received in Unity the next frame



Getting Music Time

The game can also get music time information directly from Wwise

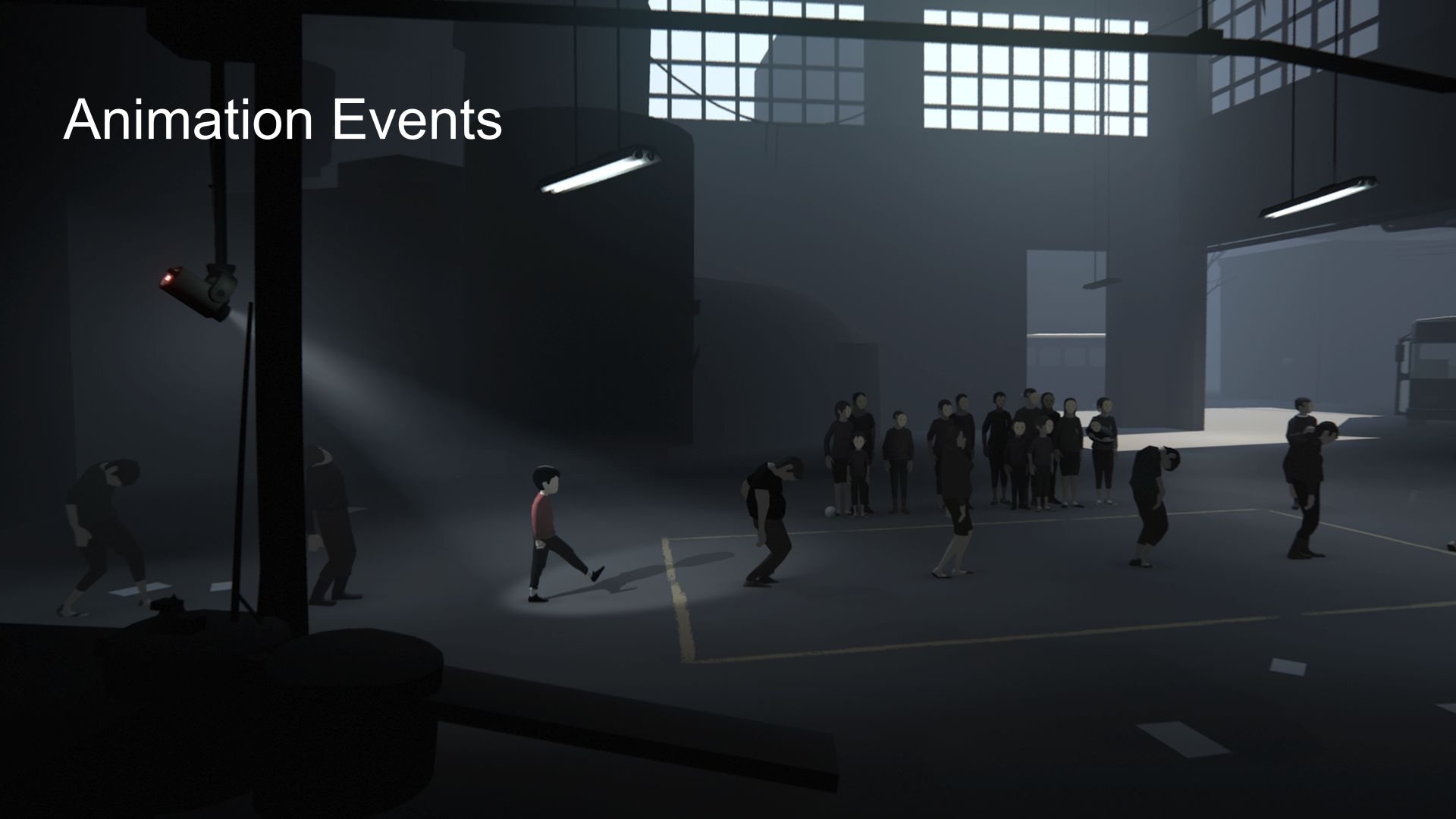


Audio Engine Summary

- INSIDE uses Wwise-Unity integration
- Wwise Authoring Tool defines events, parameters, etc.
- Wwise Authoring Tool exports soundbanks
- Wwise can be controlled from Unity
- Music User Cues and time information can be received from Wwise in Unity

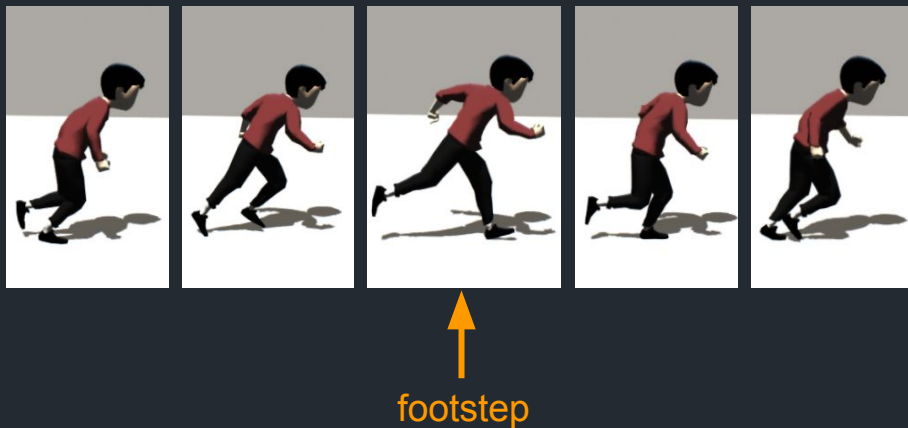


Animation Events



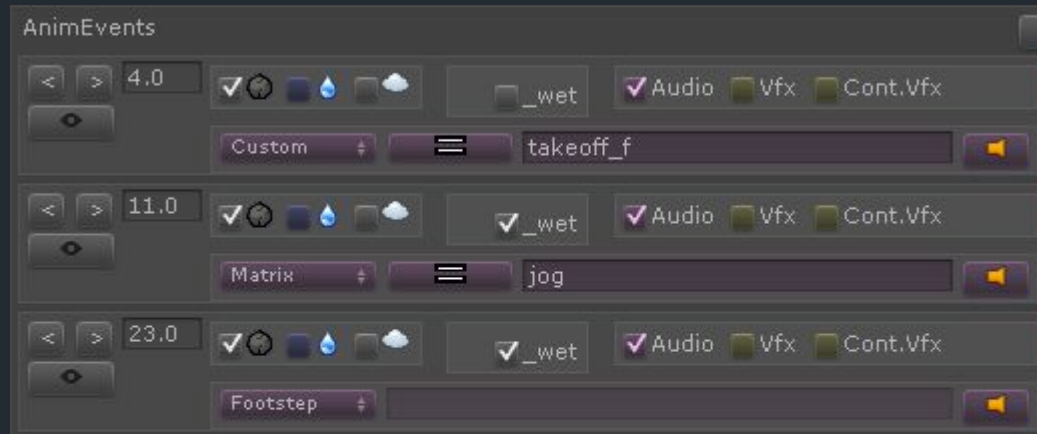
Animation Events

- Associated with a specific animation
- Occur at a specific animation frame
- Can trigger sounds or visual effects



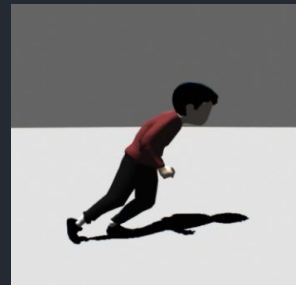
Animation Events

- Animation event types:
 - Custom
 - Matrix
 - Footstep
- Also used for VFX



Custom Animation Events

- Name of sound event specified directly
- Fires when animation frame has been passed
- Checks layers: ground, water, air

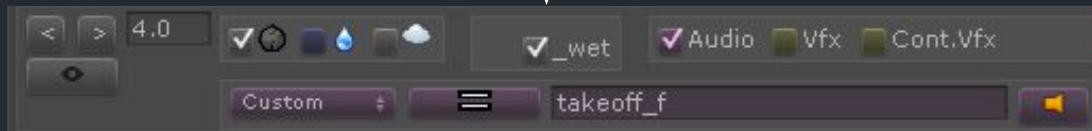


Wet Animation Events

- Optionally plays additional wet sound event
- Current wetness is sent as a parameter
 - Is set high when in water or on a wet surface
 - When dry, approaches 0 over time



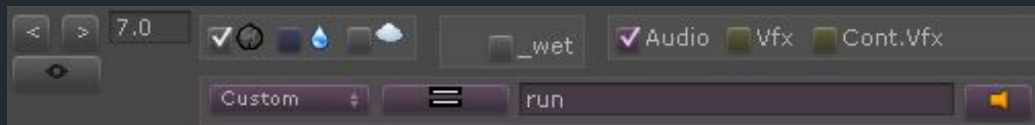
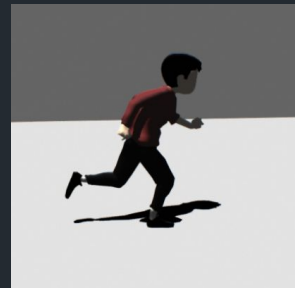
add wet event



Matrix Animation Events

- Matrix key instead of sound event name
- Context-dependent sounds

e.g. from 'run' to 'stop' yields 'brake'



matrix key

Matrix Animation Events

previous key

current key



	any	idle	sprint	run	jog	walk	sneak	JumpUp	JumpForward
any	none		sprint	run	jog	walk	sneak	jump_2feet_f	jump_1foot_mf
idle					takeoff_mf	takeoff_mp	takeoff_p		
sprint									jump_1foot_f
run									jump_1foot_mf
jog			run						jump_1foot_mp
walk			run	jog					jump_1foot_p
sneak			jog	jog	walk				jump_1foot_p
JumpUp									
JumpForward									
RunTurnRun									
RunStop									

Current Matrix Key

- **Current key** is specified in current animation event

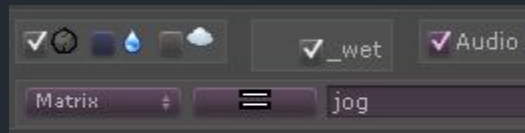


jog

current key: jog



	any	idle	sprint	run	jog
any	none		sprint	run	jog
idle					takeoff_mf
sprint					
run					
jog			run		
walk			run	jog	
sneak			jog	jog	walk
JumpUp					
JumpForward					
RunTurnRun					
RunStop					



Previous Matrix Key

- **Previous key** was specified in previous animation event



idle



previous key: idle



	any	idle	sprint	run	jog
any	none		sprint	run	jog
idle					takeoff_mf
sprint					
run					
jog			run		
walk			run	jog	
sneak			jog	jog	walk
JumpUp					
JumpForward					
RunTurnRun					
RunStop					

Play Sound

previous key: idle

current key: jog



idle



jog

	any	idle	sprint	run	jog
any	none		sprint	run	jog
idle					takeoff_mf
sprint					
run					
jog			run		
walk			run	jog	
sneak			jog	jog	walk
JumpUp					
JumpForward					
RunTurnRun					
RunStop					

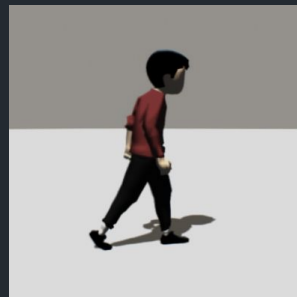
play sound 'takeoff_mf'

Context Sensitivity

- If previous matrix key was 'sneak', a different sound is played

previous key: sneak

current key: jog



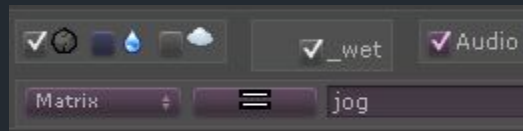
sneak



jog

	any	idle	sprint	run	jog
any	none		sprint	run	jog
idle					takeoff_mf
sprint					
run					
jog			run		
walk			run	jog	
sneak			jog	jog	walk
JumpUp					
JumpForward					
RunTurnRun					
RunStop					

play sound 'walk'



Column Default

- Empty entries are replaced with column default

previous key: idle

current key: run

	any	idle	sprint	run	jog
any	none		sprint	run	jog
idle					takeoff_mf
sprint					
run					
jog			run		
walk			run	jog	
sneak			jog	jog	walk
JumpUp					
JumpForward					
RunTurnRun					
RunStop					



idle

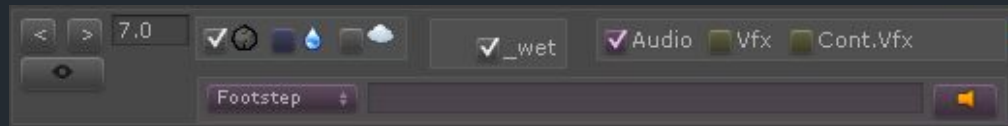


run

play sound 'run'

Footstep Animation Events

- Footsteps = automatic matrix events
- Key is determined from movement speed:
 - idle : speed < 0.07
 - sneak : speed < 0.37
 - walk : speed < 0.70
 - jog : speed < 0.92
 - run : speed < 1.30
 - sprint : speed >= 1.30
- Robust with smooth animation blending
- Simple to define



Animation Events Summary

- Animation events occur in a specific frame in an animation
- Animation events trigger sounds and visual effects
- Custom events specify sounds directly
- Matrix events are used for context-sensitivity
- Footstep events are matrix events, automatically selected based on speed



Cloth



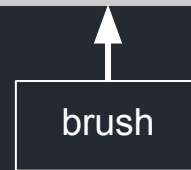
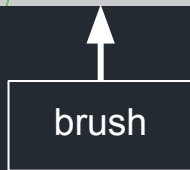
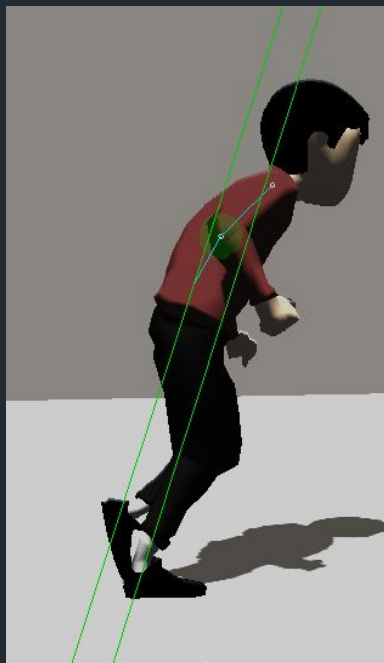
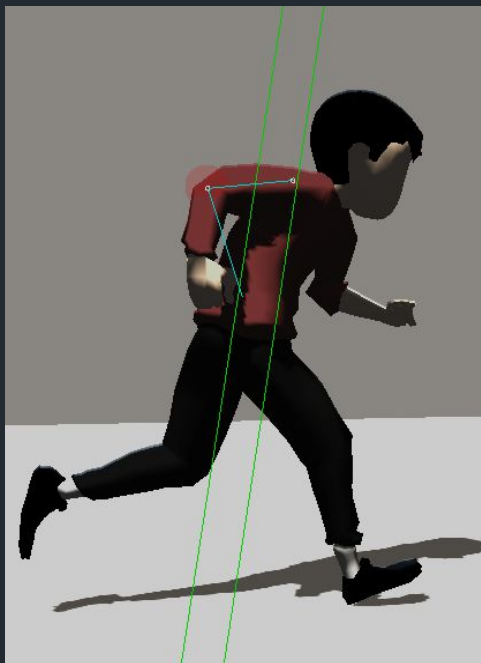
Cloth

- Sound of clothes rubbing against itself
- Generated at runtime from character geometry
- Sounds are selected based on movement speed

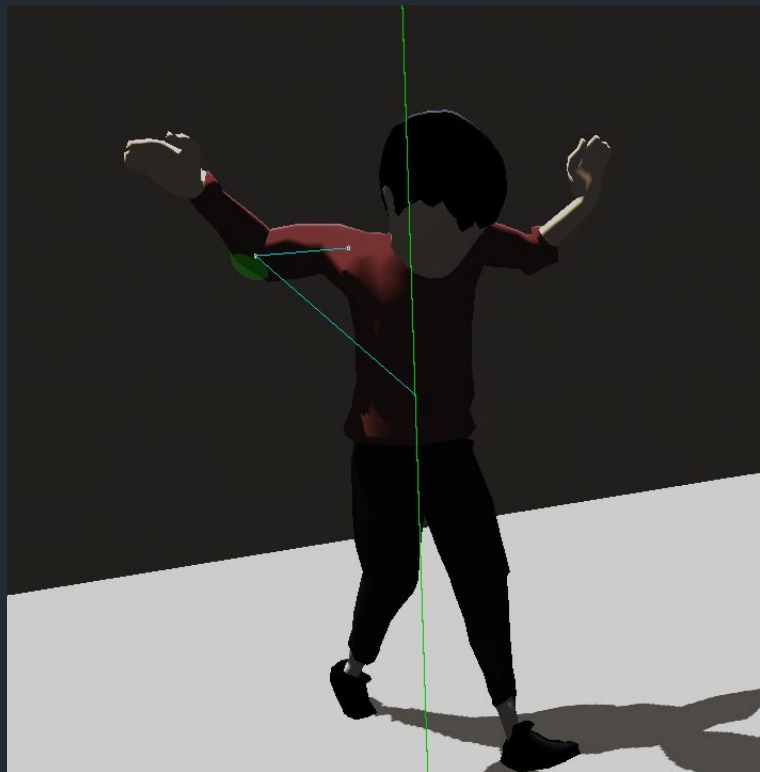
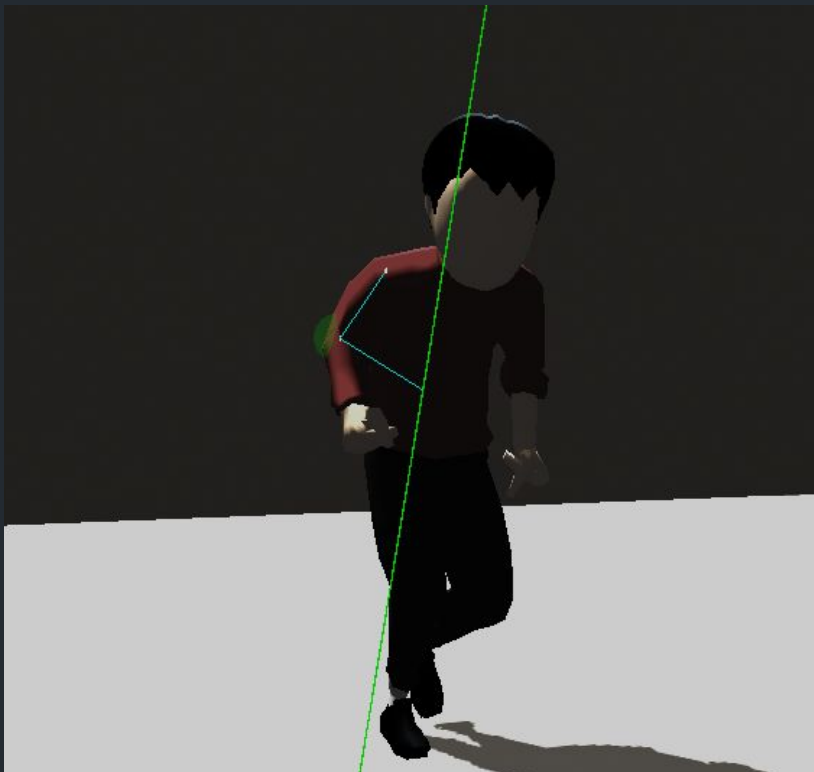
Elbow Torso Pass

- Send elbow speed parameter
- Play sound when elbow enters 'brush zone'

Elbow Torso Pass



Arm Angle



Cloth Summary

- Sound events generated from geometry
- Tracking a single elbow was enough
- Creates coherence between discrete foley sounds



Voice Sequencer



Voice Concept

- Natural and adaptable audio playback
- Integration of physical and emotional states

Voice Demo



Voice Sequencer

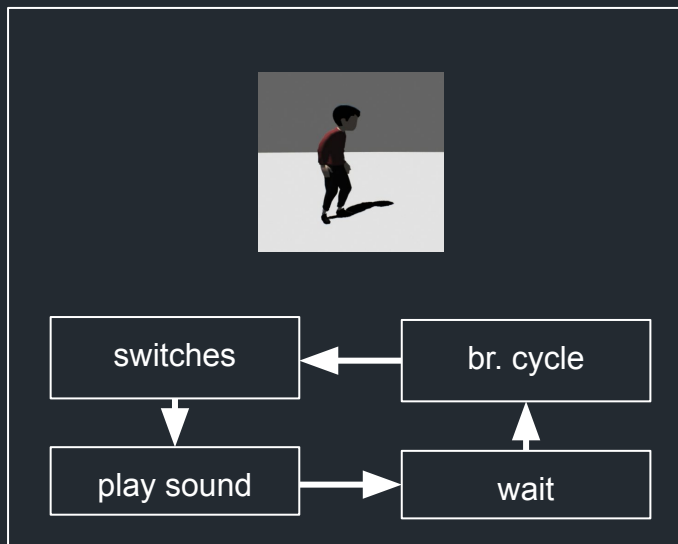
- Sequencer implemented in C# using Wwise callbacks
- Sequences voice sound events, alternating between inhale and exhale

Voice Sound Events

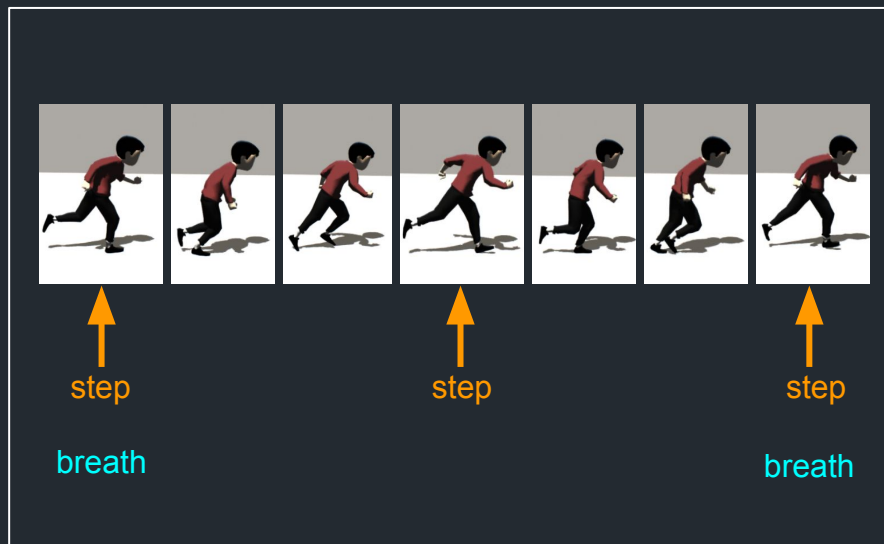
- Which sound to play is defined by switches:
 - Action
 - Emotion
 - Intensity
 - Etc.
- Intensity is a numeric value:
 - Increases with physical exertion
 - Decreases when idle

Voice Sequencer Modes

Continuous Mode

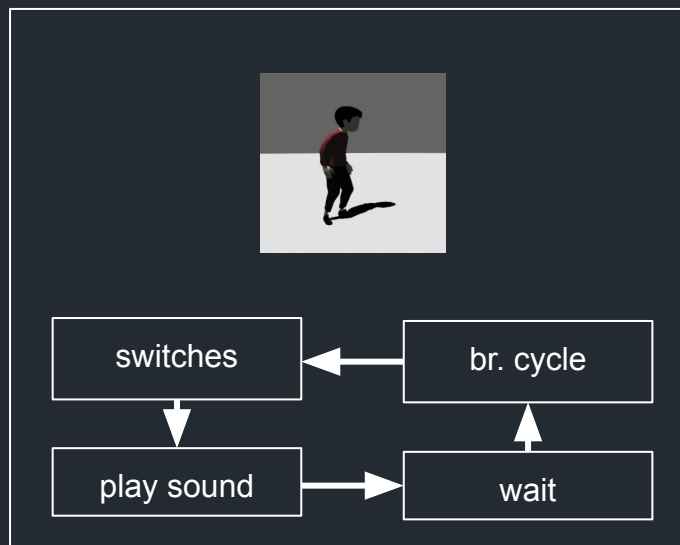


Rhythmic Breathing

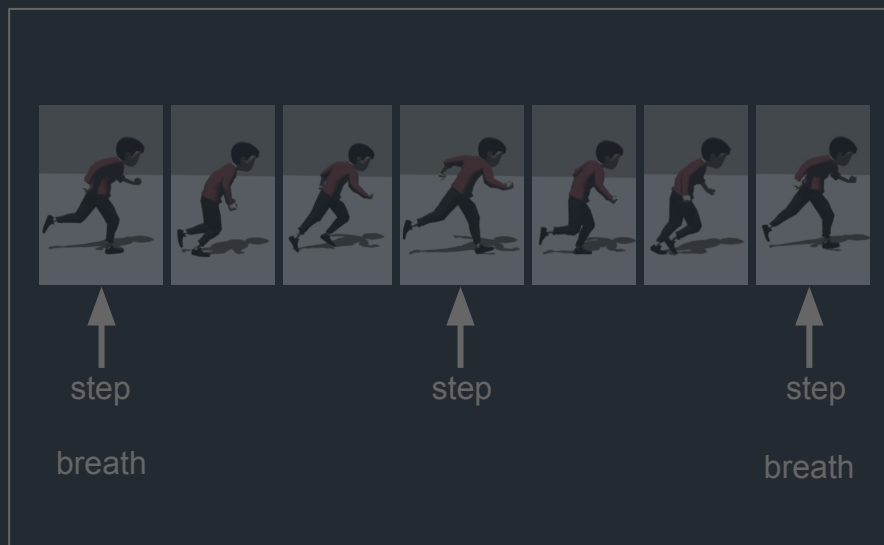


Voice Sequencer Modes

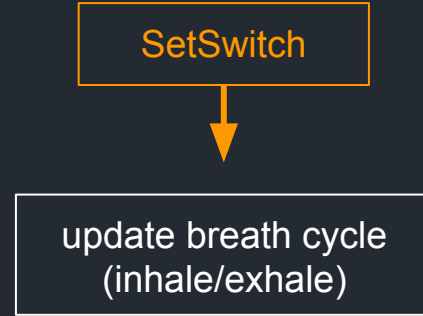
Continuous Mode



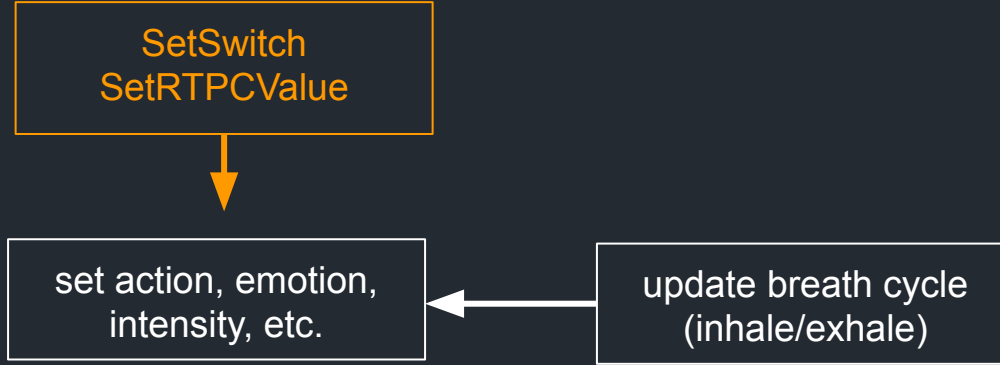
Rhythmic Breathing



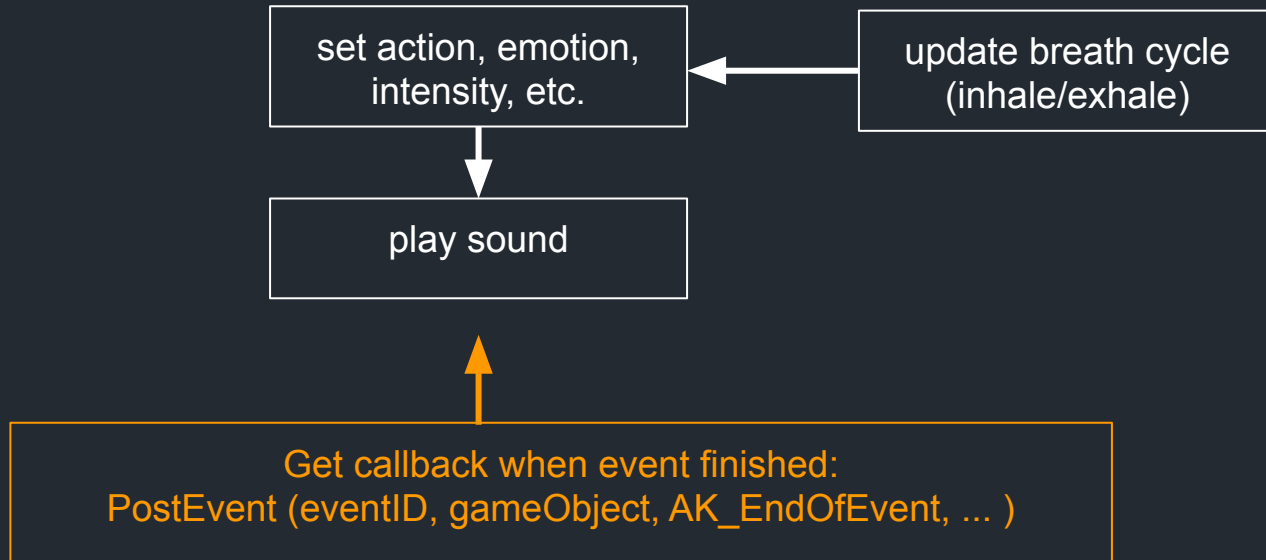
Voice Sequencer: Continuous Mode



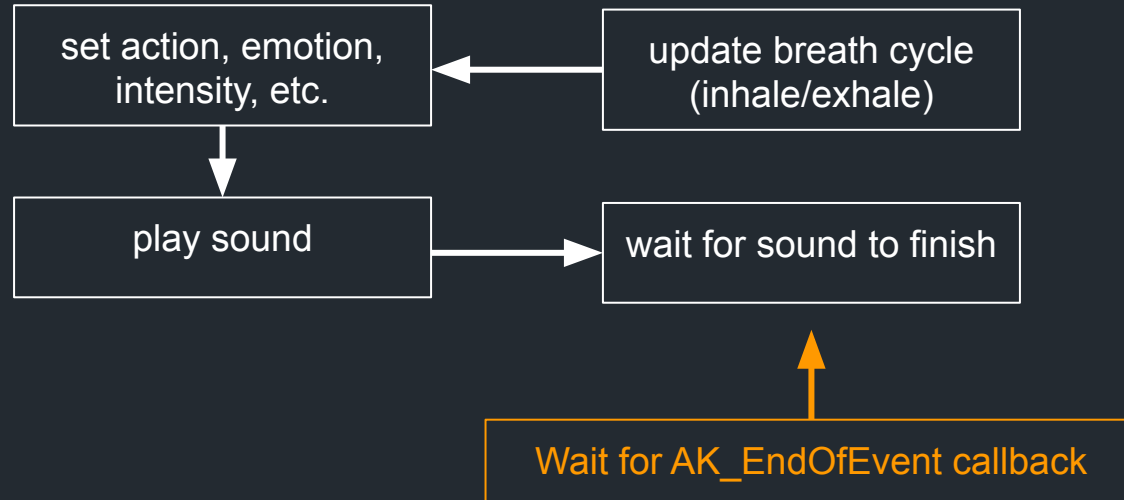
Voice Sequencer: Continuous Mode



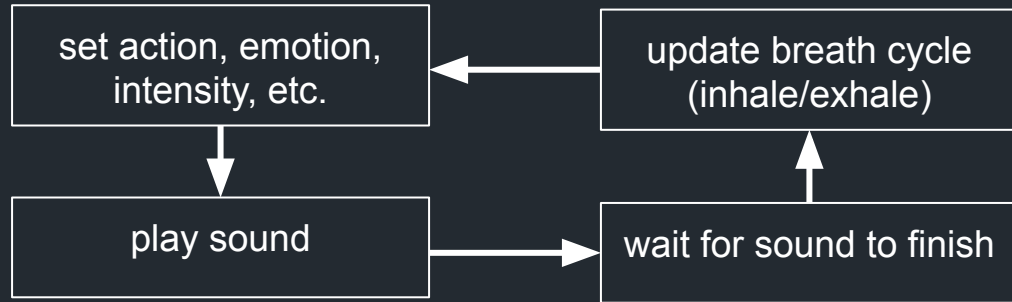
Voice Sequencer: Continuous Mode



Voice Sequencer: Continuous Mode

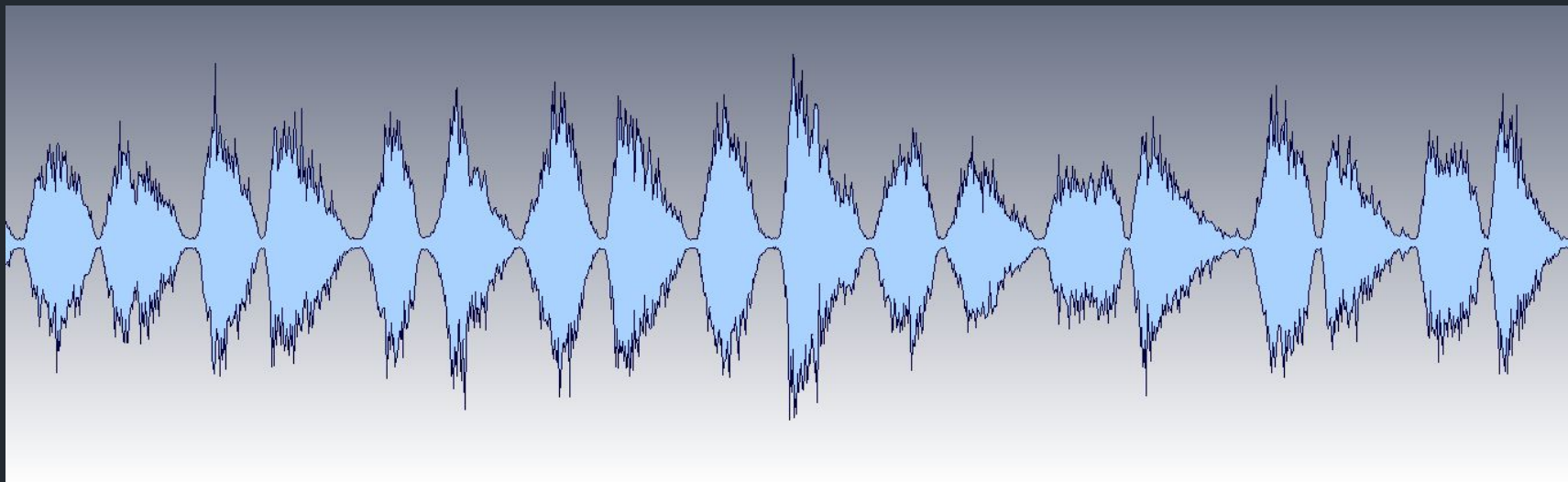


Voice Sequencer: Continuous Mode



Natural Breathing

- Recorded breath sounds have varying durations
- Continuous sequencing results in natural, uneven breathing pattern

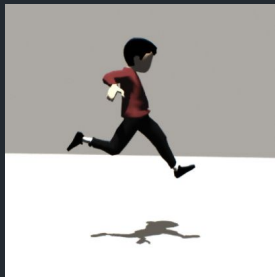


Animation Feedback

- Every breath results in a callback to the game
- Callback controls additive breathing animation, affecting boy pose



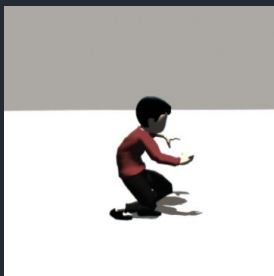
Holding Breath



On jump:

if currently inhaling, stop afterwards

if currently exhaling, do a quick inhale, then stop



On land:

restart breathing with exhale (action = land)

soft impact: normal exhale, hard impact: grunt

Engagement Actions

Special actions indicate performing work, uses different set of sounds



not engaged



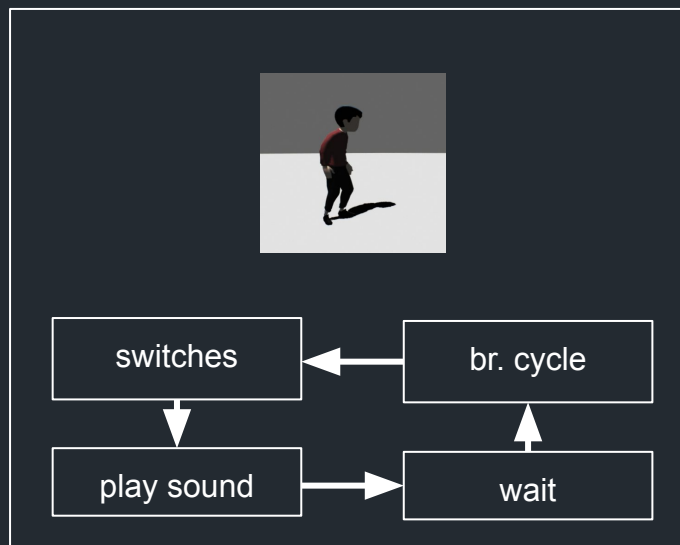
engaged passive



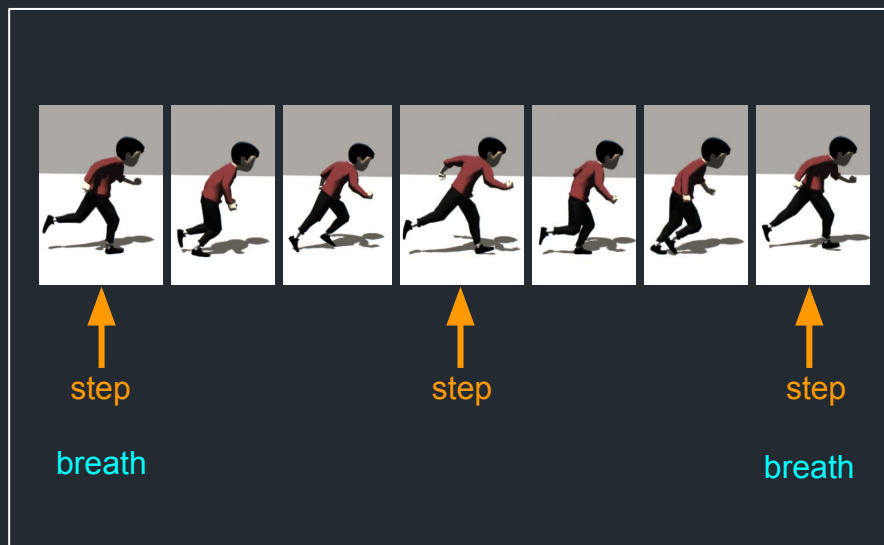
engaged active

Voice Sequencer Modes

Continuous Mode

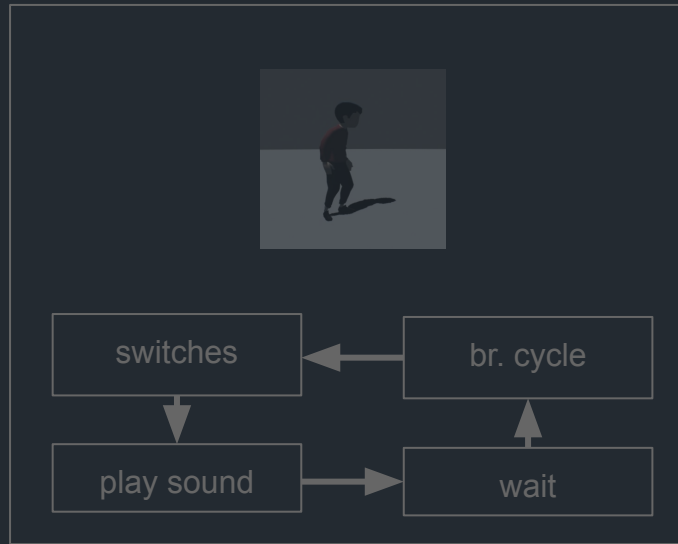


Rhythmic Breathing



Voice Sequencer Modes

Continuous Mode



Rhythmic Breathing

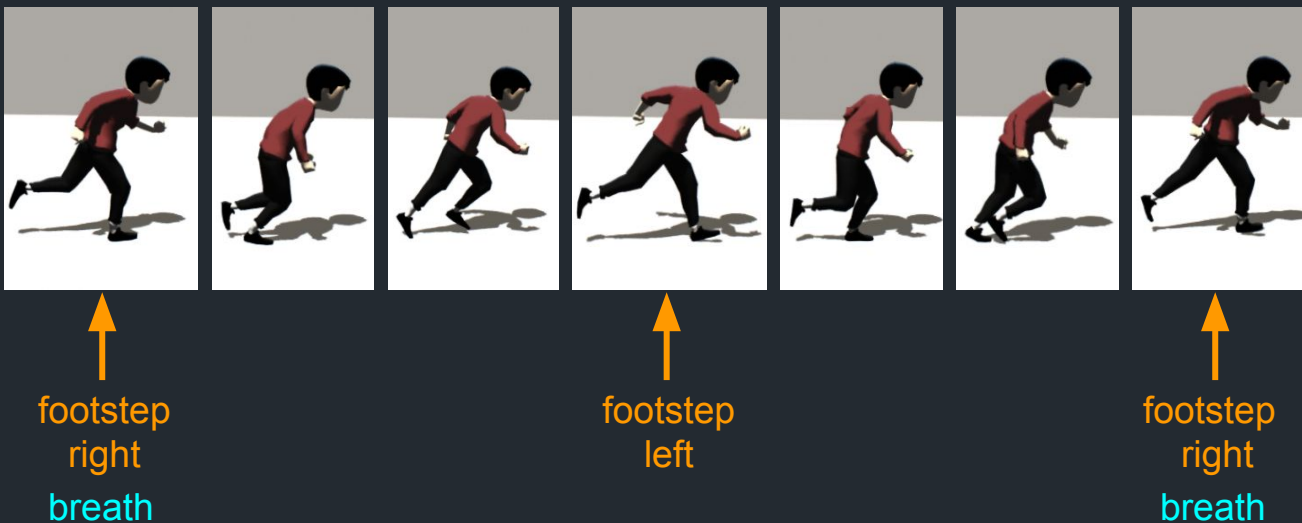


Rhythmic Breathing

- Goal: breath should align with footsteps when running
- Non-continuous sequencing

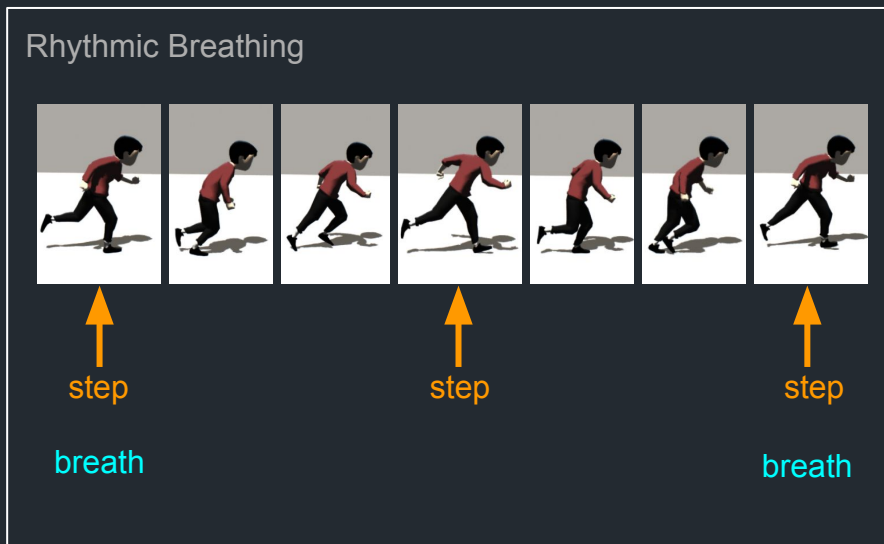
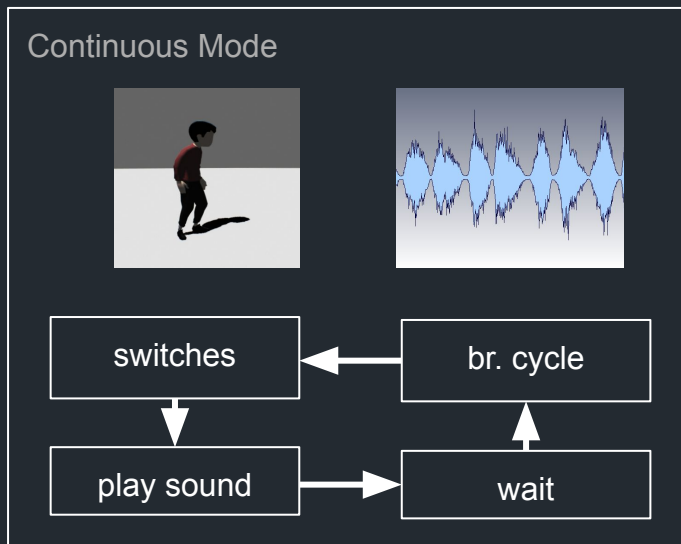
Rhythmic Breathing

- Goal: breath should align with footsteps when running
- Non-continuous sequencing
- 1 **breath** for every 2 **steps**

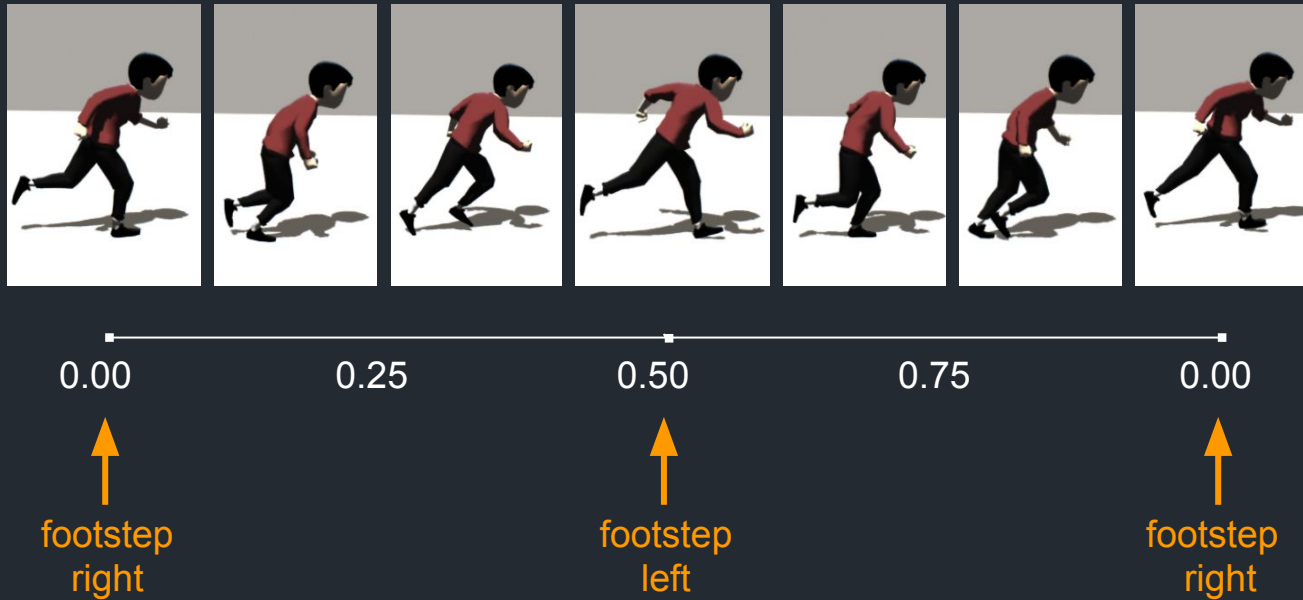


Rhythmic Breathing Transition

- When not running, breath runs continuously
- When starting to run, gradually transition from continuous rhythm to footstep rhythm



Run Cycle Phase



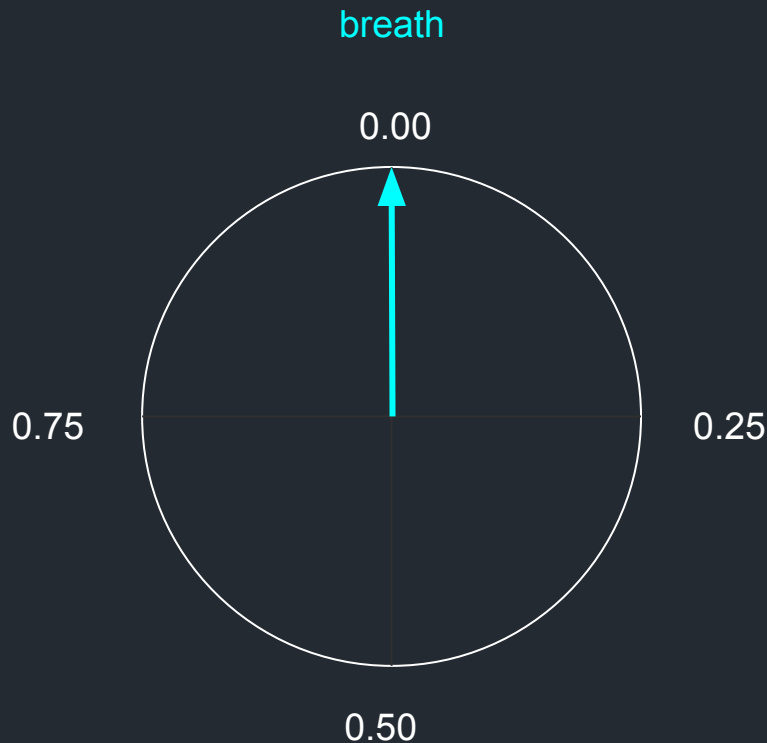
Run Cycle Phase

- Full cycle is 2 steps
- Right footstep on 0.0
- Left footstep on 0.5



Breath Phase

- Breathe when phase is 0
- Full cycle is 1 breath
- When switching from continuous to rhythmic breathing:
 - Compute frequency from last 2 breaths
 - Compute phase from frequency and last breath time



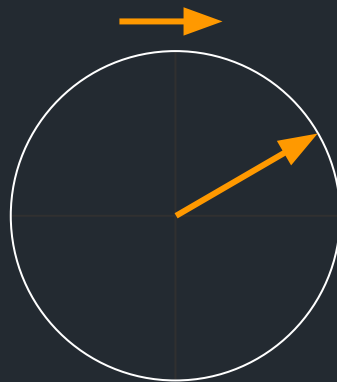
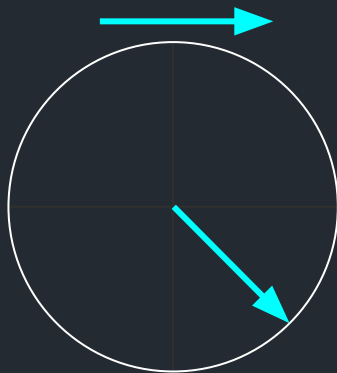
Gradual Alignment

- Gradually align breath rhythm to run cycle rhythm
- Align two **frequency, phase** pairs

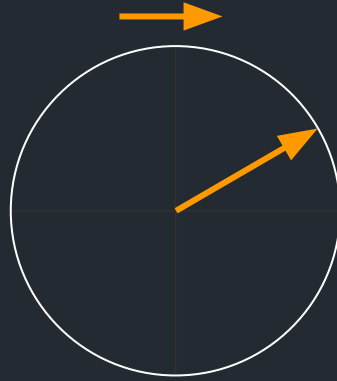
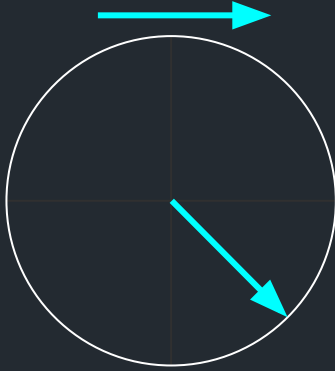


Gradual Alignment Problem

- General problem: aligning two **frequency, phase** pairs



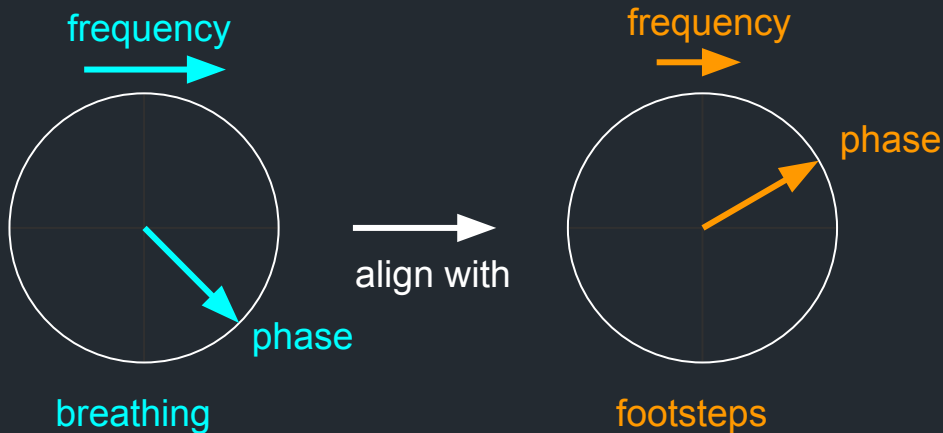
Solution: Beat Matching



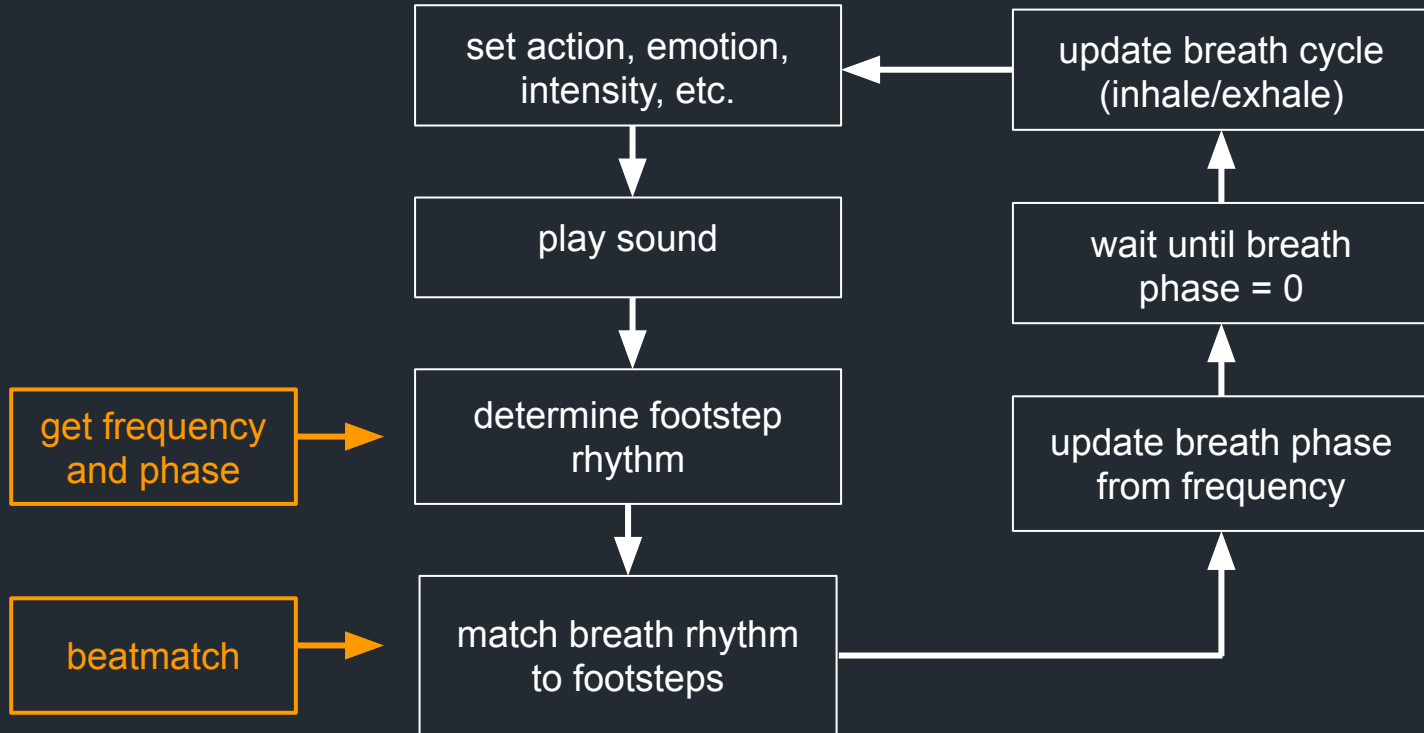
Solution: Beat Matching

- Gradually interpolate breath frequency towards run cycle frequency
- Compensate breath frequency for phase offset

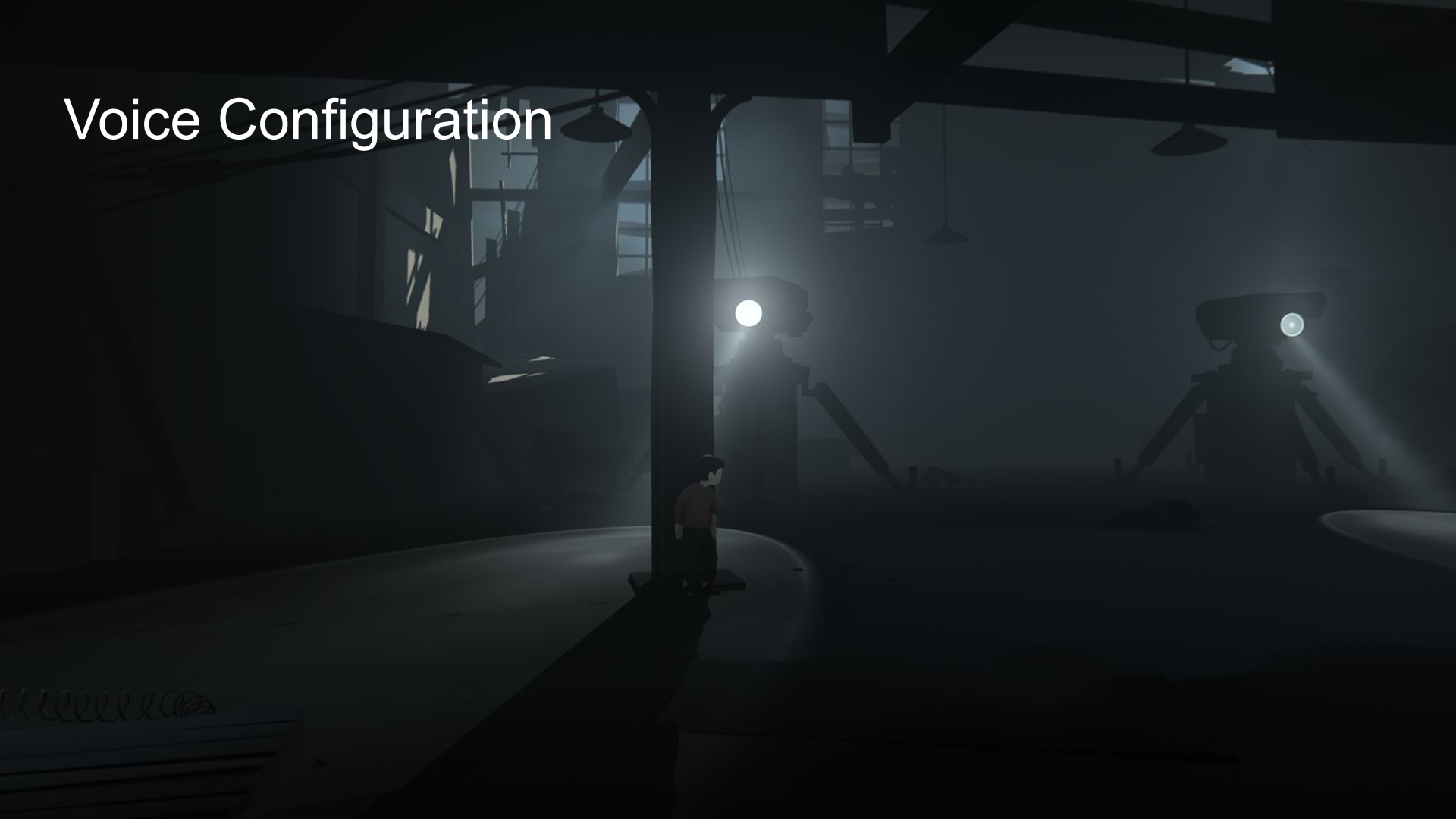
- Like a DJ that uses pitch adjust without nudging the record



Voice Sequencer: Rhythmic Breathing



Voice Configuration



Voice Direction

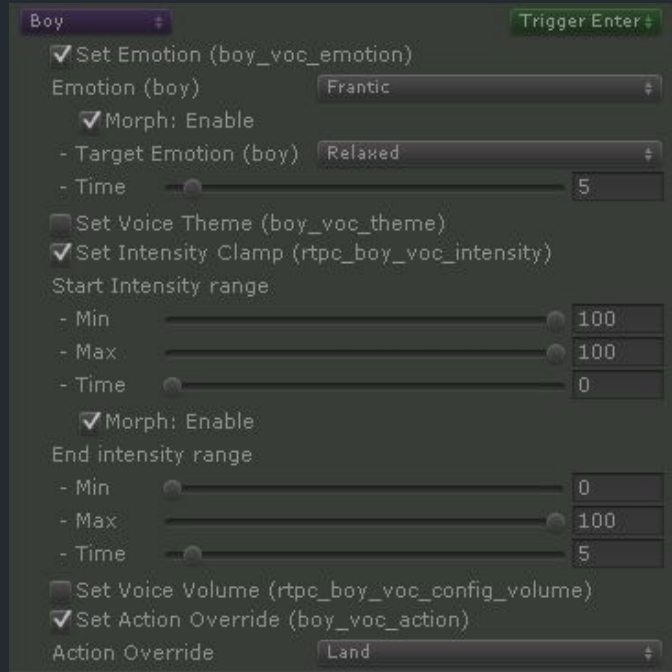
- Voice direction is accomplished using our voice configuration system
- The director (Martin) instructs the actor (voice sequencer) how to emote:
 - based on location or
 - based on reacting to events



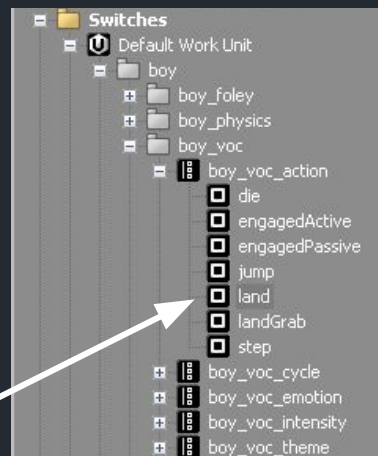
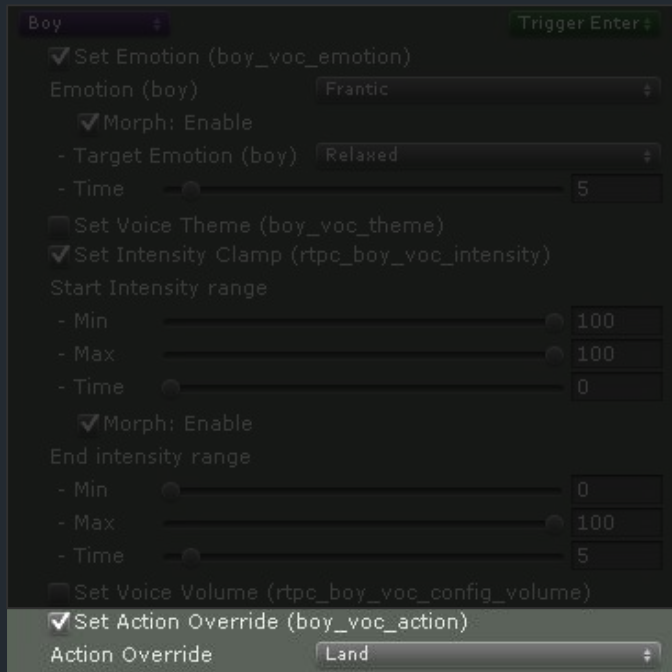
Voice Configuration

- Trigger boxes
- State machines
- Scripts
- Gives full control over voice parameters
 - action
 - emotion
 - intensity

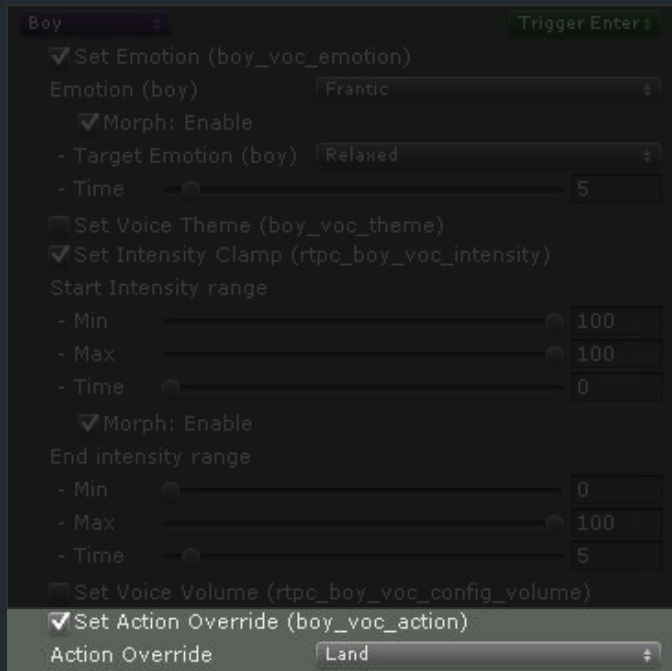
Voice Configuration: Trigger box



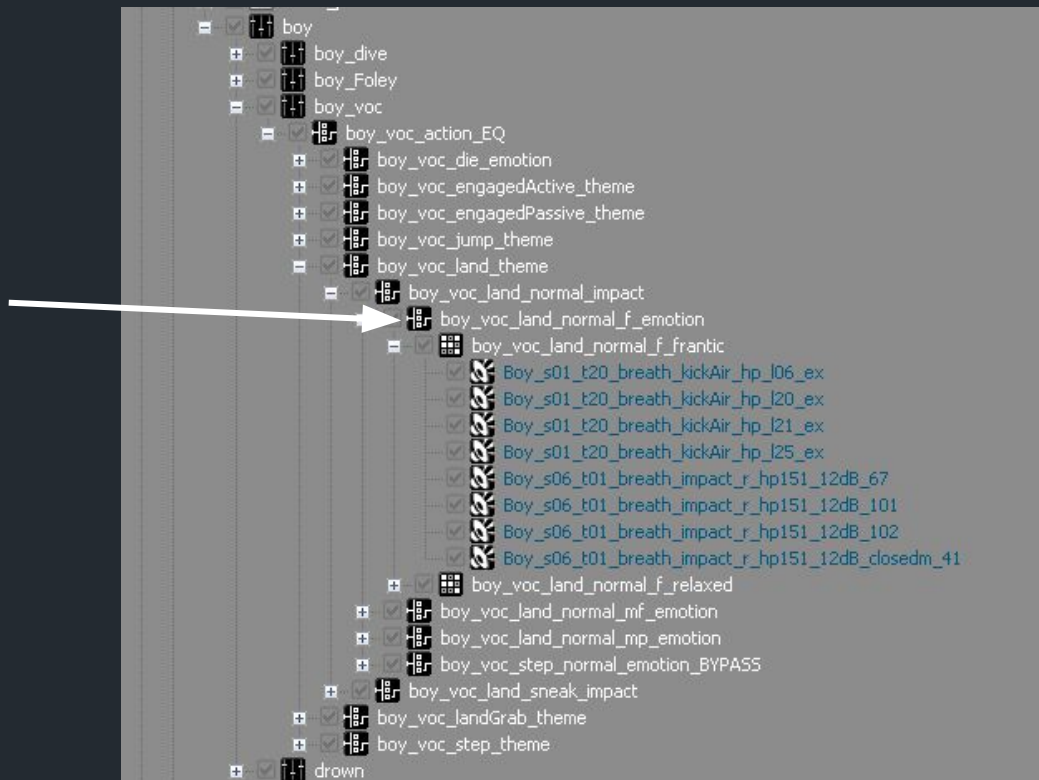
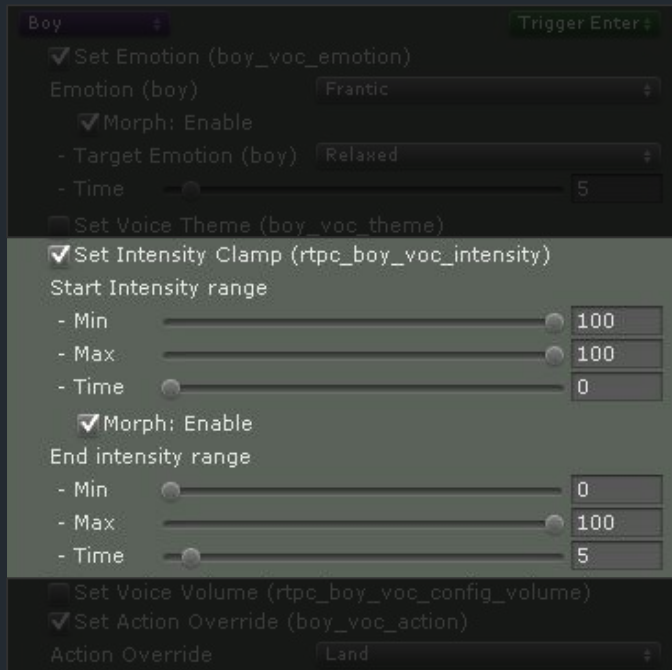
Switch



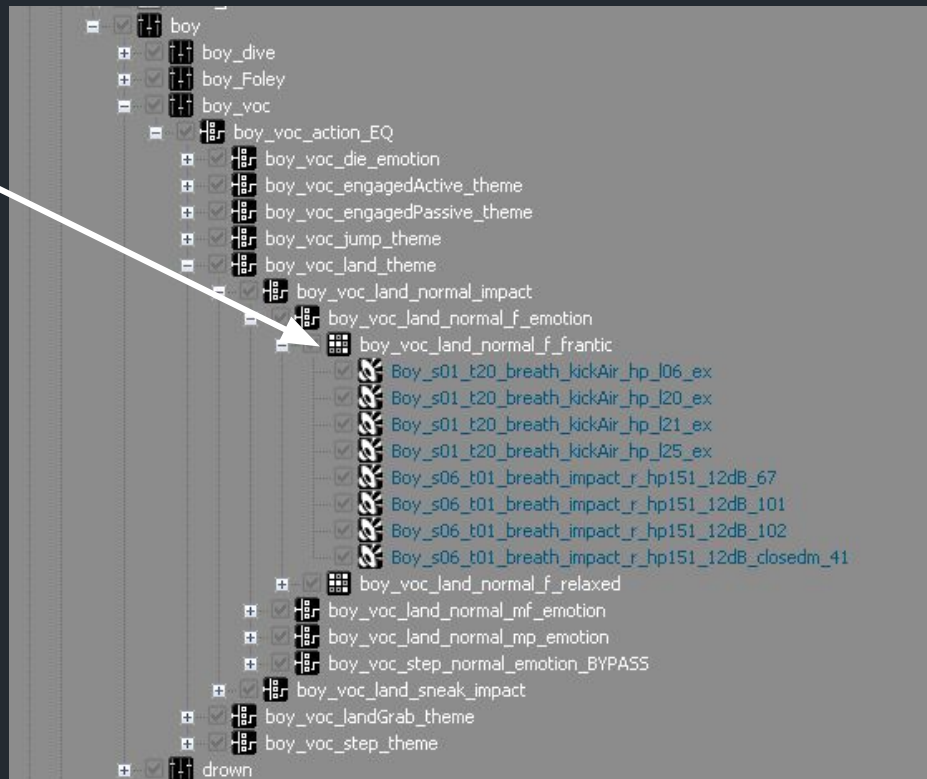
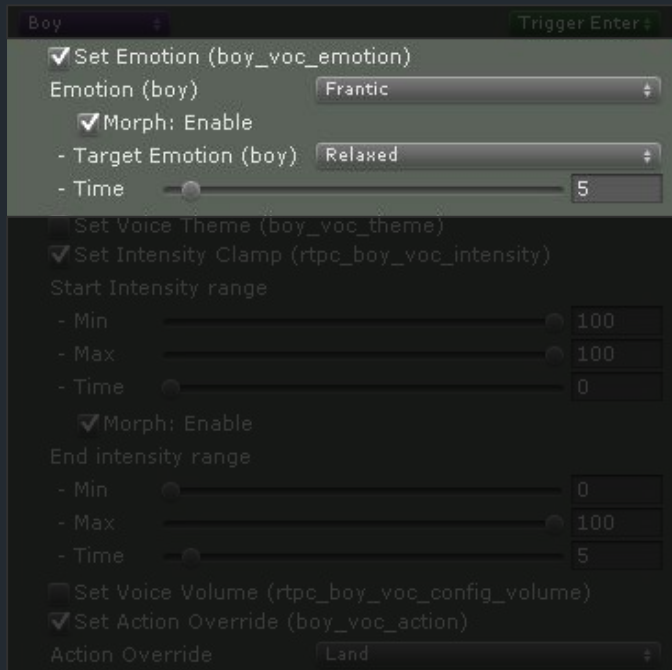
Switch Container: Action



Switch Container: Intensity

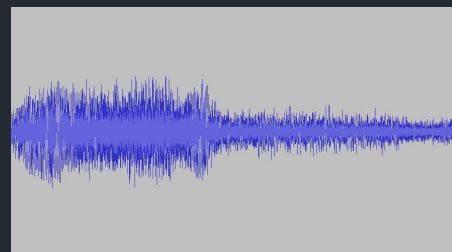
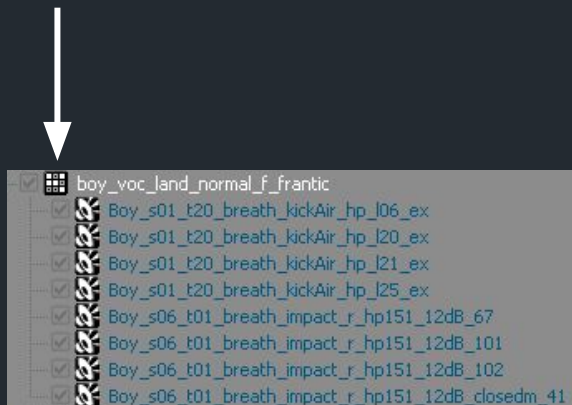


Switch Container: Emotion

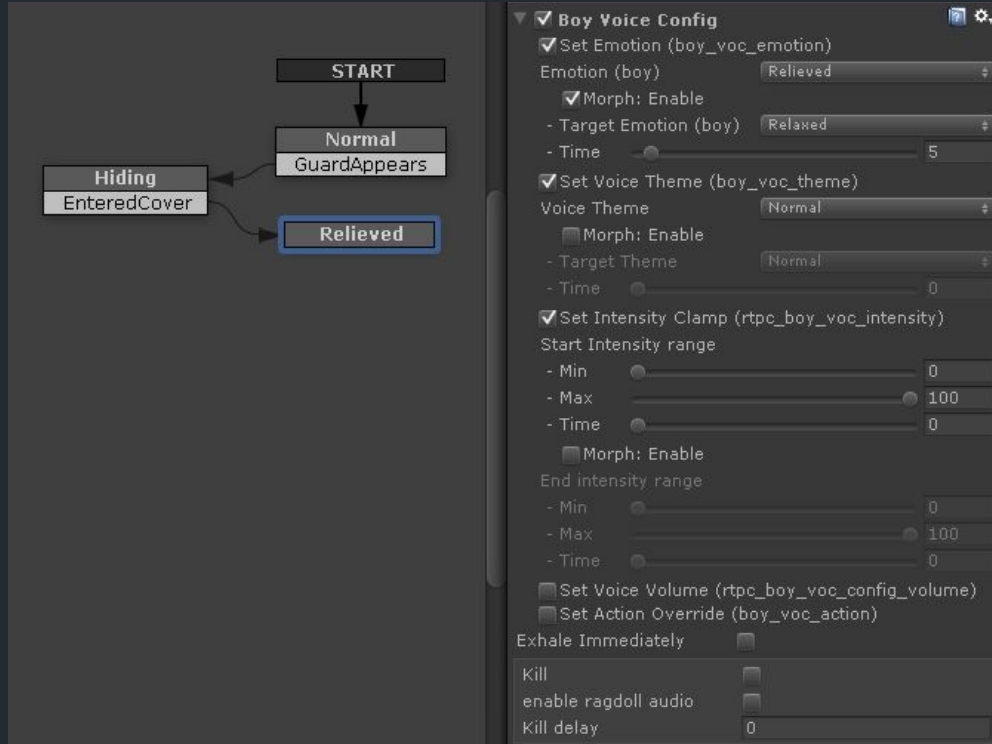


Random Container

Randomly selects and plays one of its children sounds

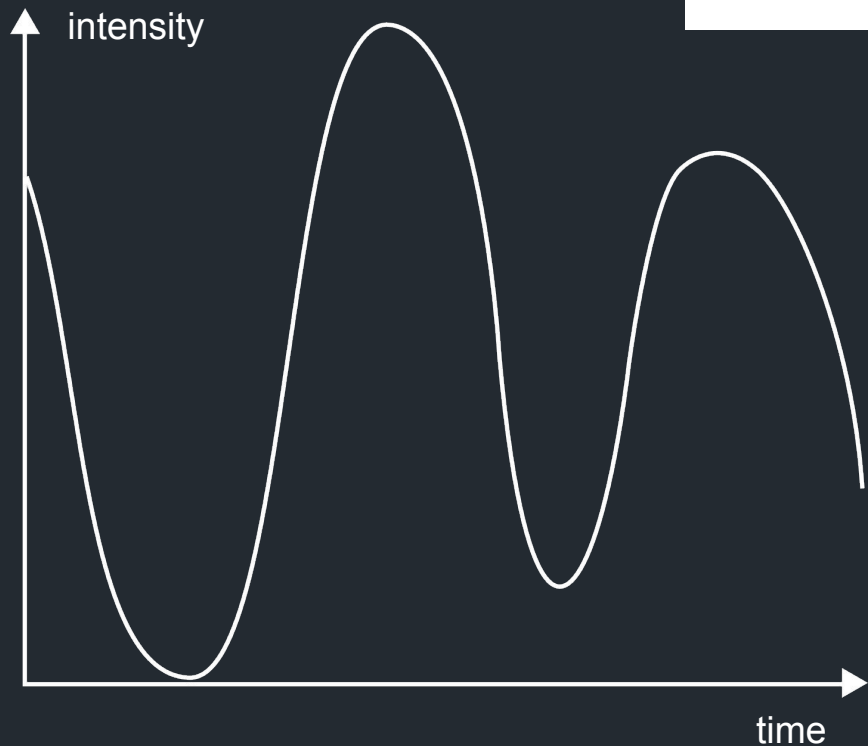
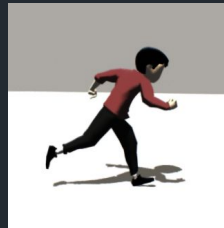


Voice Configuration: State Machine



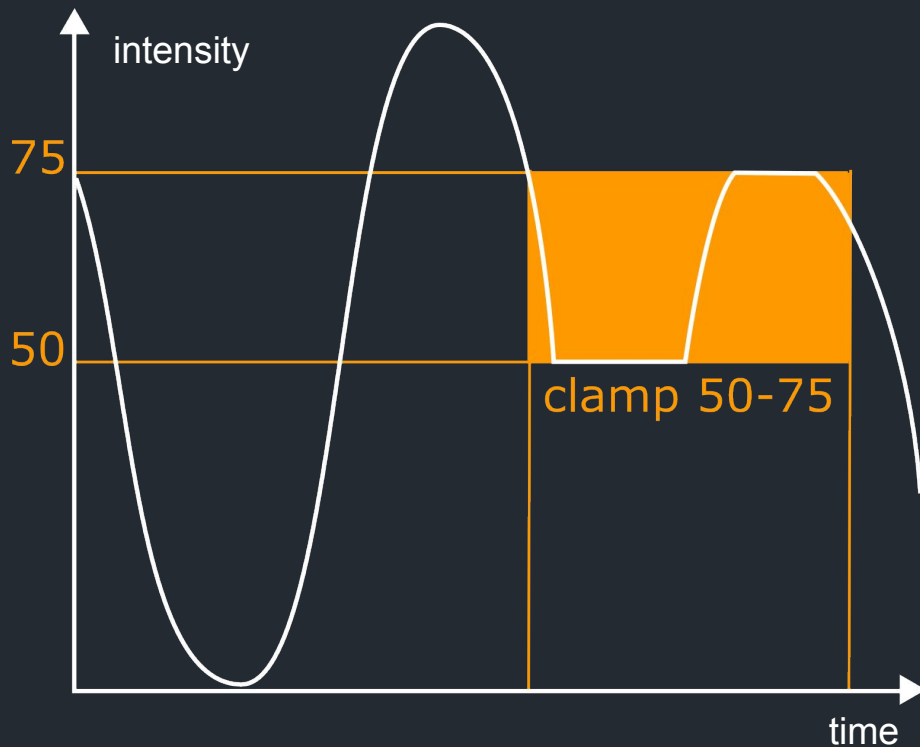
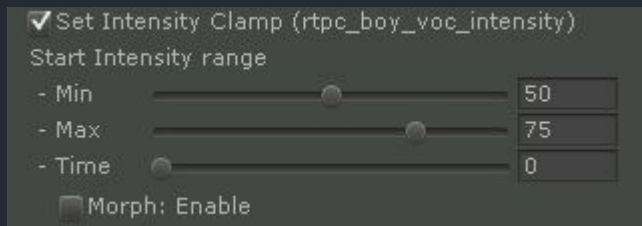
Voice Intensity

- Boy movement generates exhaustion
- Voice intensity = lowpass filtered exhaustion
- Voice Intensity selects depth and force of breathing
- Depending on the emotion, intensity defines:
 - Physical exertion level
 - Intensity of character emotion



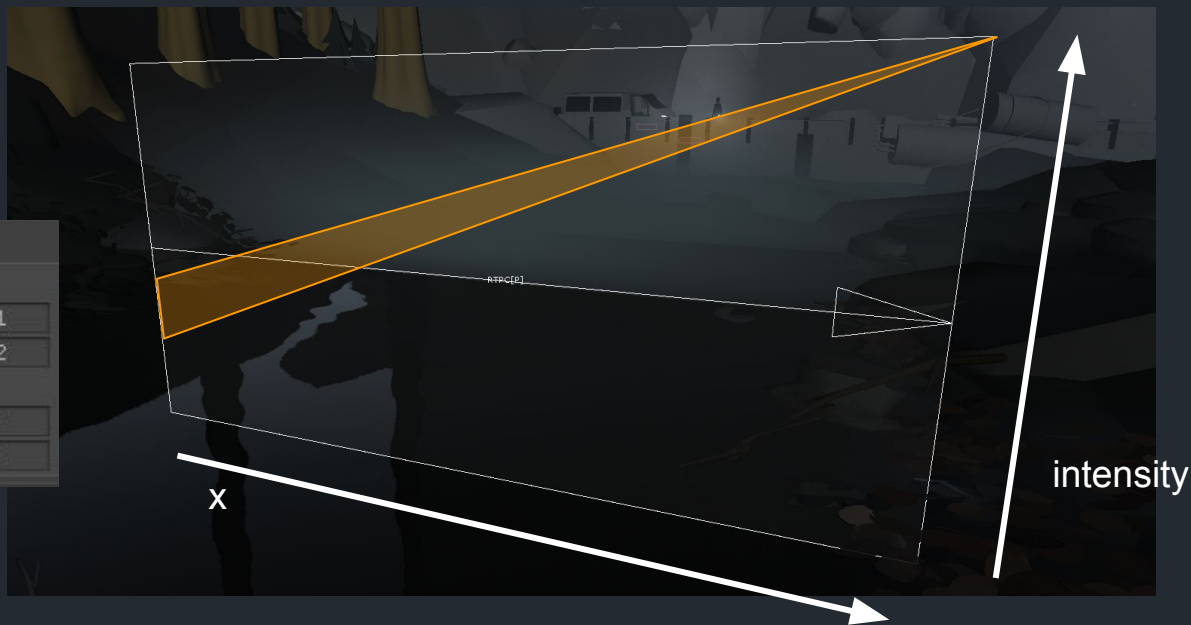
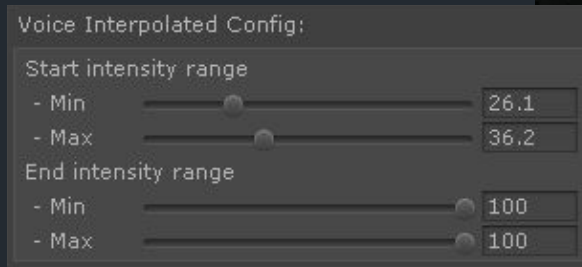
Voice Intensity Clamping

- Clamping constrains intensity to a given range



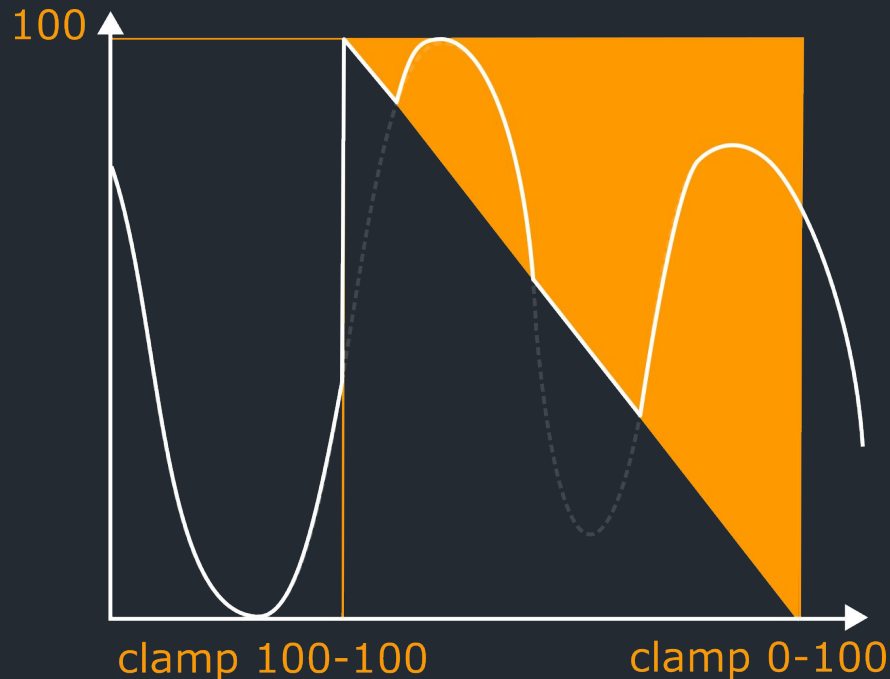
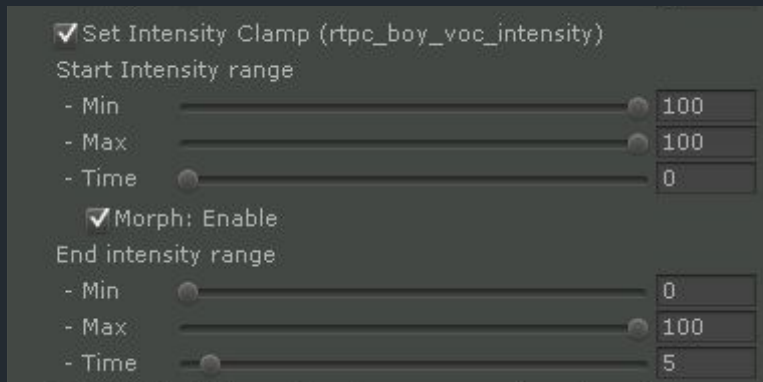
Voice Intensity Interpolation in Space

- Useful for indicating proximity to danger



Voice Intensity Interpolation over Time

- Useful for creating reactions to game events, and relaxing over time.



Voice Summary

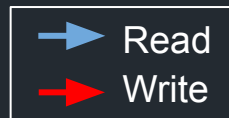
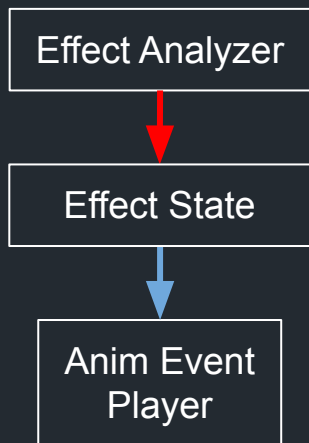
- Single event, switch hierarchy determines sound
- Continuous sequencing using callbacks
- Rhythmic breathing uses beatmatching to align breath to footsteps
- Voice direction with trigger boxes and state machines
- Voice Intensity can be clamped
- Clamping can be interpolated in space and time



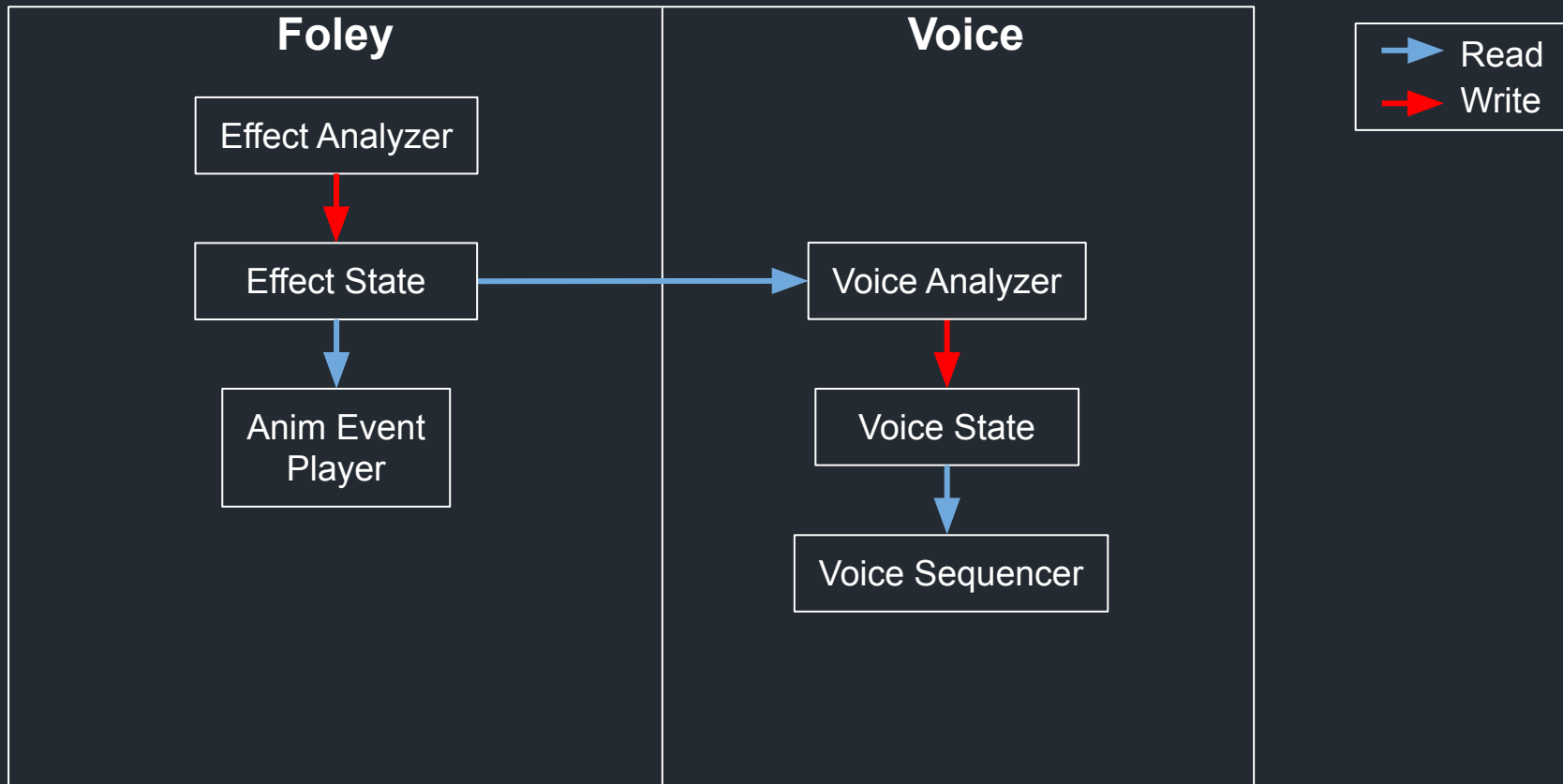
Wrapping Up



Animation Events



Voice Sequencer



Full Audio Architecture for the Boy

