The Boy from INSIDE

Uncompromising Character Audio Implementation

AES 2016
Jakob Schmid
PLAYDEAD

Me

Jakob Schmid

Audio programmer at PLAYDEAD

Composer and sound designer by night

Overview

- Introduction
- Animation events
- Cloth
- State, analysis, rendering
- Voice sequencer

Slides will be available online!


INSIDE
PLAYDEAD
Released soon for Xbox One
Commenced 2010
INSIDE
playdead.com/inside

Playdead Audio Team

Martin Stig Andersen

audio director, sound designer, composer

Andreas Frostholm — sound designer

Søs Gunver Ryberg — composer, sound designer

Jakob Schmid — audio programmer

Technology

Unity

Audiokinetic Wwise

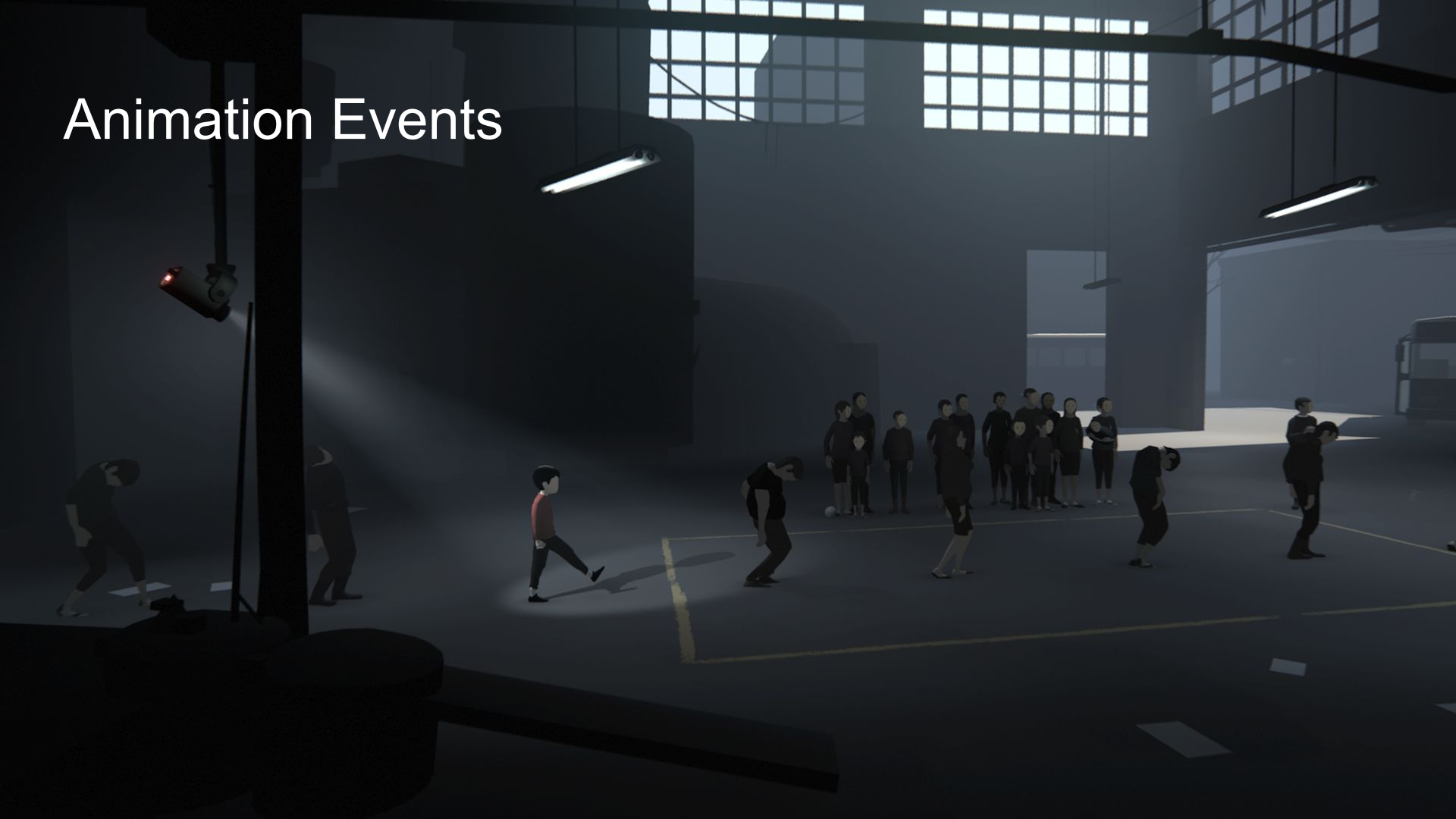
Modified Wwise-Unity plugin

PlayMaker

DEMO: Animation Events

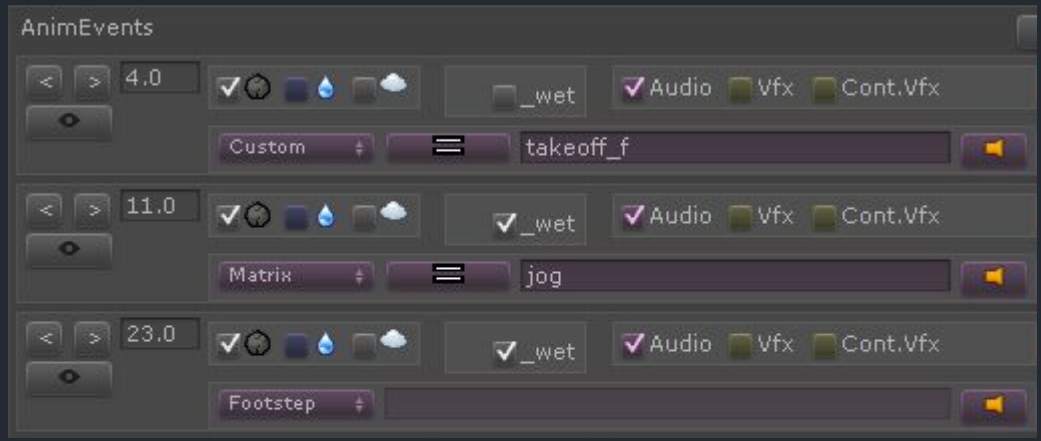
scene: #3D_crane

Animation Events



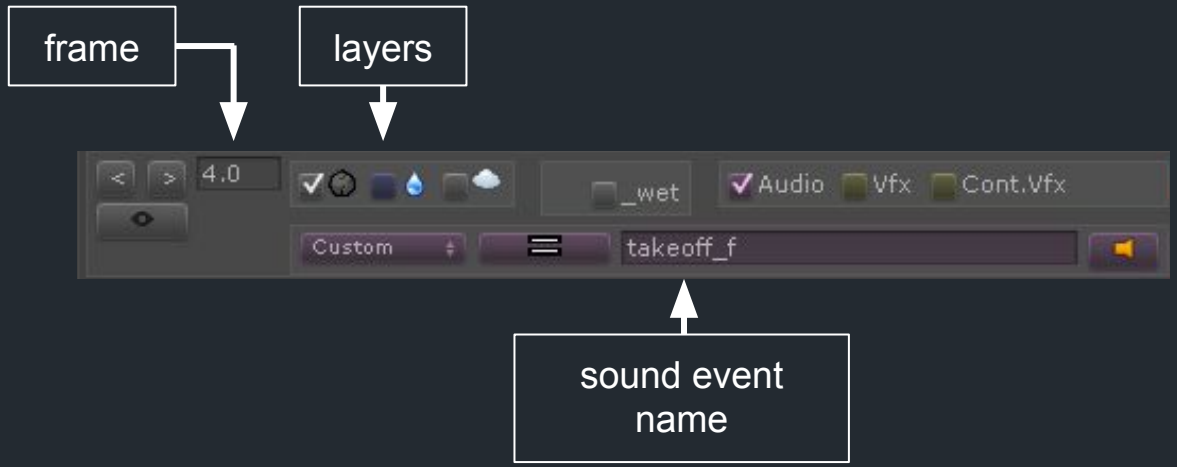
Animation Events

- Set up per animation
- Animation event types:
  - Custom
  - Matrix
  - Footstep
- Also used for VFX



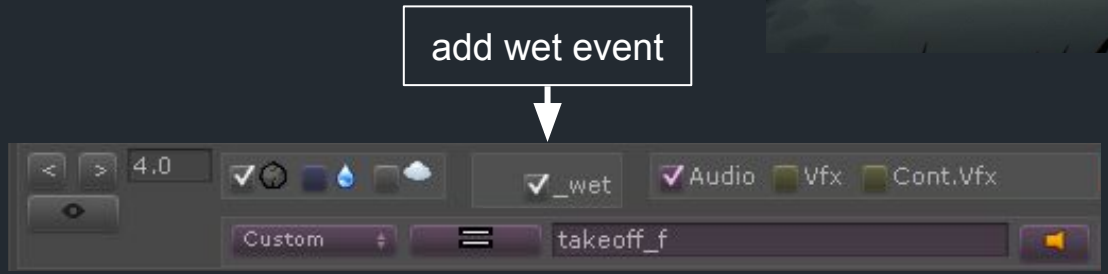
Custom Animation Events

- 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



Matrix Animation Events

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

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

Matrix Animation Events

previous key

current key

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									

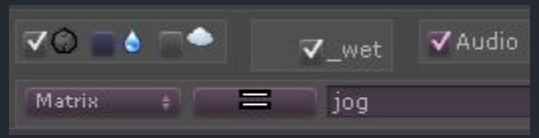
Current Matrix Key

- Current key is specified in current animation event

jog

current key: jog

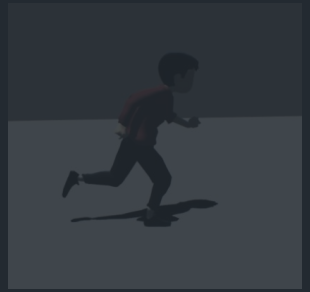
	any	idle	sprint	run	jog
any	none		sprint	run	jog
idle					takeoff_mf
sprint					
run					
jog			run		
walk			run	jog	
sneak			jog	jog	walk
JumpUp					
JumpForward					
RunTurnRun					
RunStop					



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

idle

jog

previous key: idle

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					

play sound 'takeoff_mf'

Context Sensitivity

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

sneak

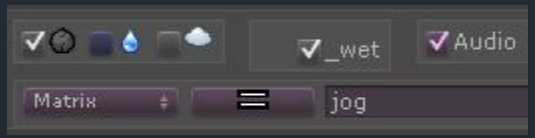
jog

previous key: sneak

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					

play sound 'walk'



Column Default

- Empty entries are replaced with column default

idle

run

previous key: idle

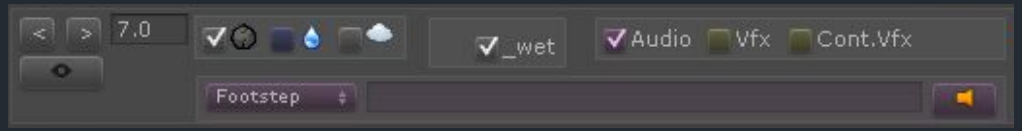
current key: run

	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					

play sound 'run'

Footstep Animation Events

- 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

- Custom events specify sounds directly
- Matrix events are used for context-sensitivity
- Footstep events are matrix events, automatically selected based on speed

Cloth

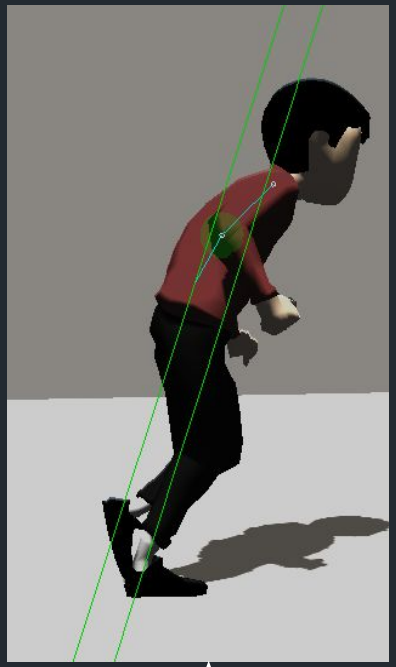
Cloth

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

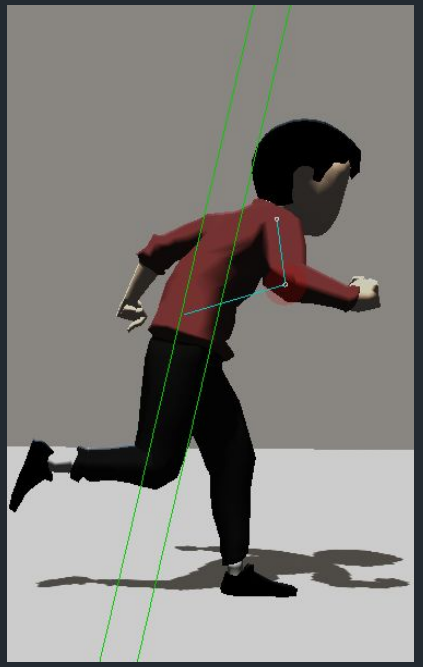
Elbow Torso Pass

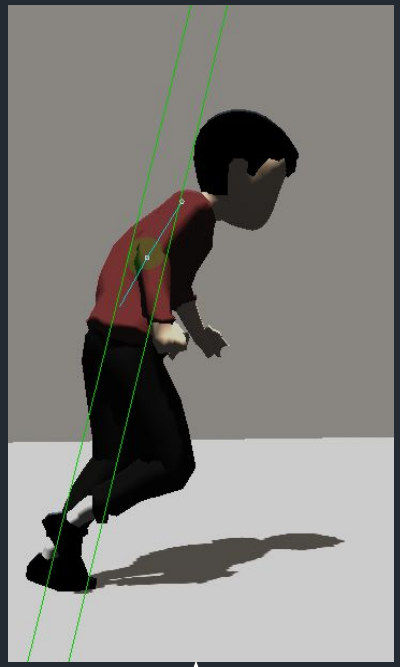
- Send elbow speed parameter
- Play sound when elbow enters 'brush zone'

Elbow Torso Pass



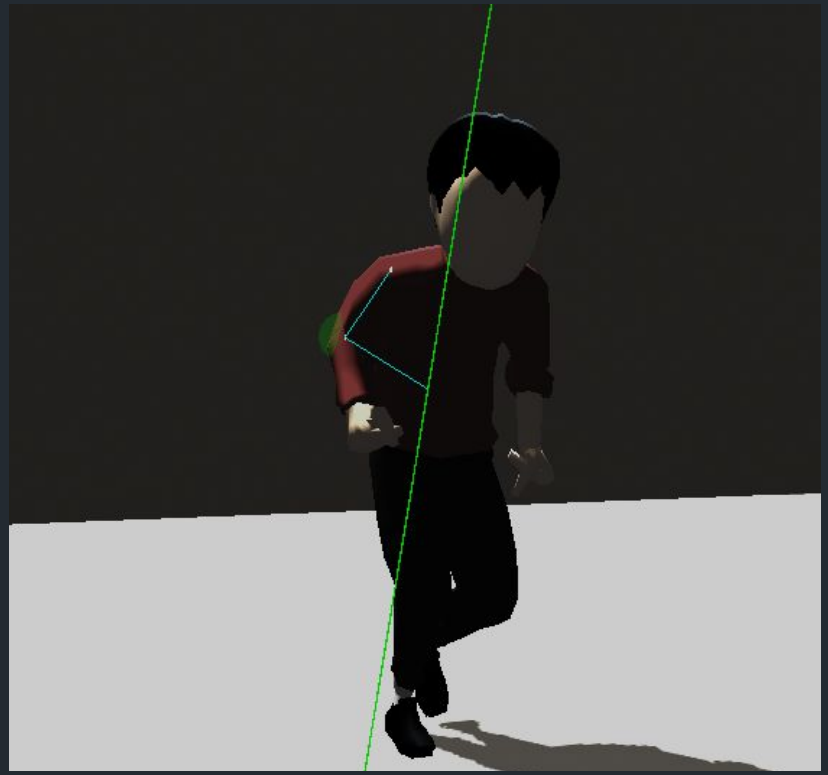
brush

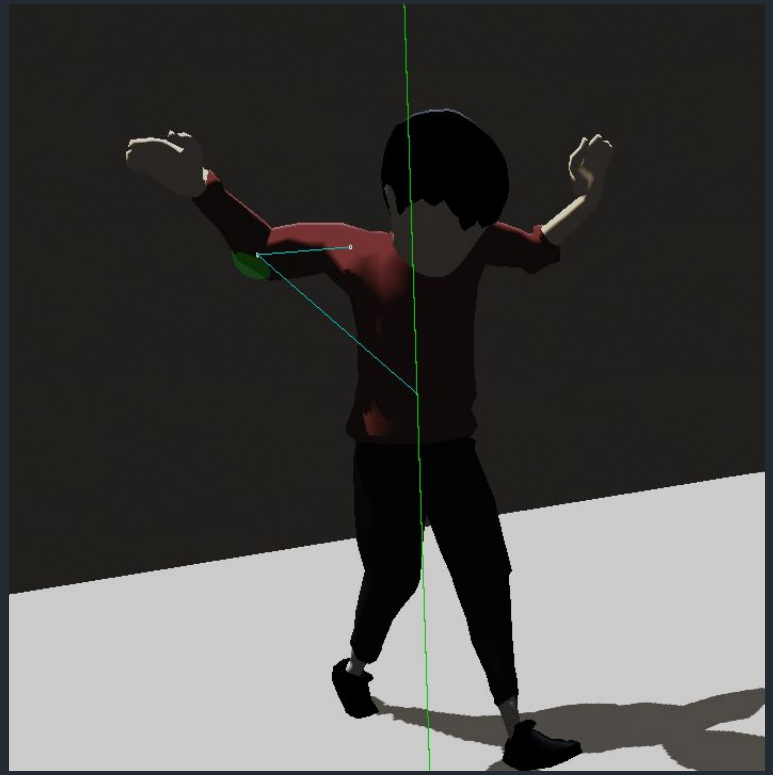




brush

Arm Angle

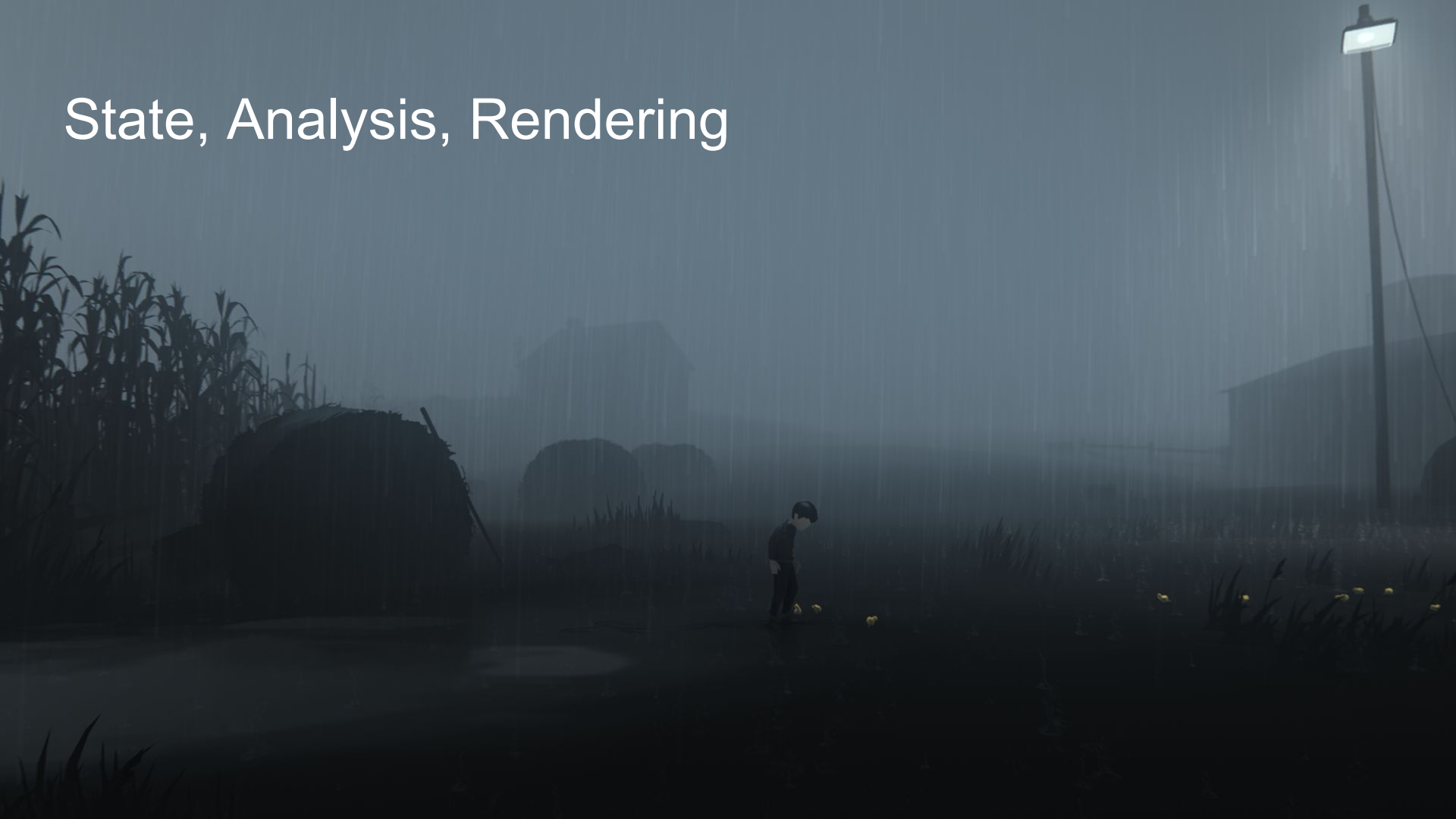




Cloth Summary

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

State, Analysis, Rendering

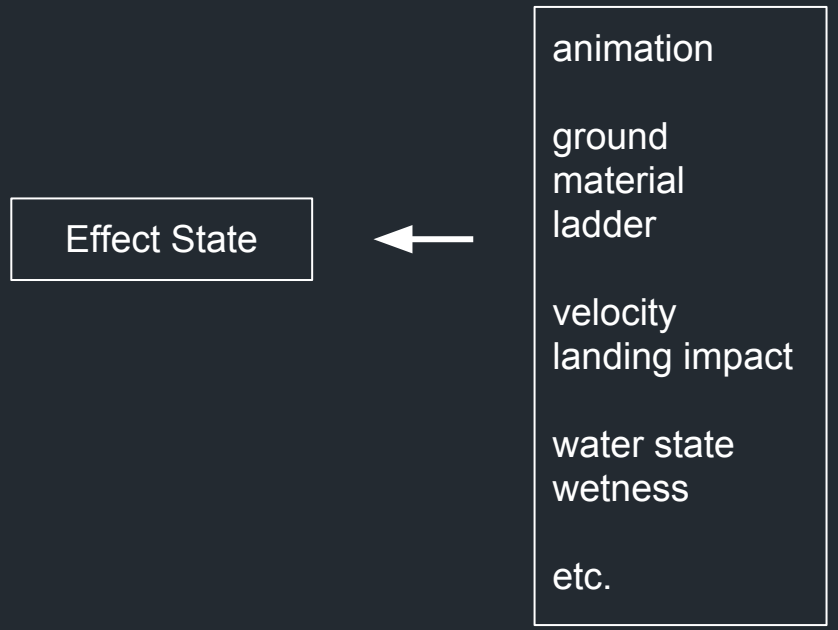


Boy Audio

- Boy audio feature set grew a lot during development
- Debugging was tricky
- Called for a well-defined and easy-to-debug audio state

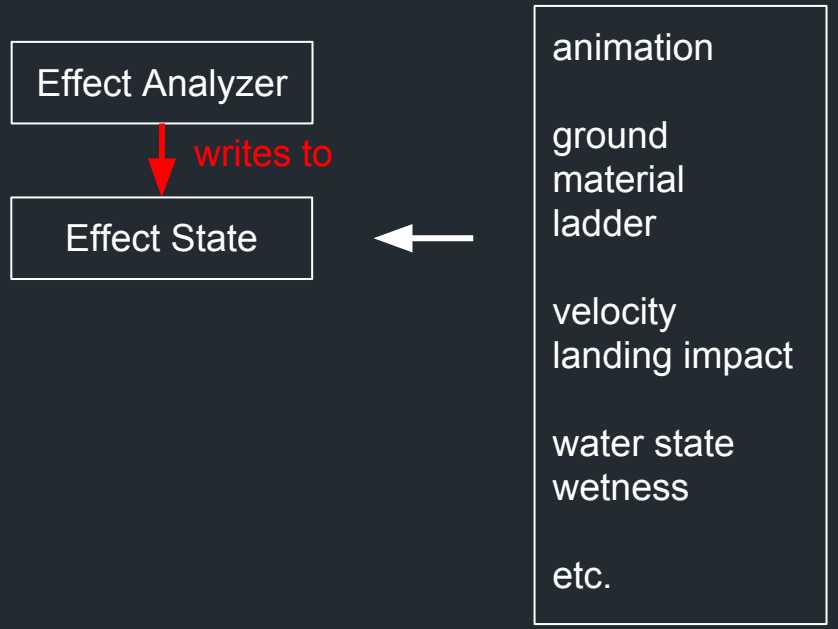
Effect State

- General audio state for boy
- Used by several components
- Well-defined state for every frame



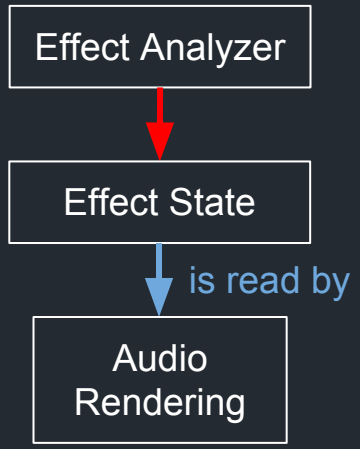
Effect Analyzer

- Aggregates state from analysis of
  - Animation system
  - Physics data, e.g. velocity
  - Collider metadata, e.g. material
- Responsible for updating Effect State



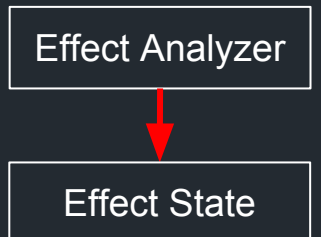
Audio Rendering Components

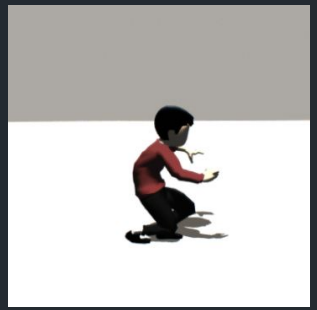
- Set parameters and play sounds exclusively based on Effect State
- Never modify Effect State



Structural Benefits

- Effect State can be shown as debug information
- Bugs are divided into two groups:
  - State is wrong: buggy analysis
  - State is right, sound is wrong: buggy rendering
- Ordering of data sent to sound engine is explicit





debug information

isLanding	true
landImpact	0

etc.

Double-buffered State

- State should be double-buffered
- Sounds are often a response to a state change
- Certain data from previous frame needed

Effect State	
previous	current
isLanding false landImpact 70 etc.	isLanding true landImpact 0 etc.

Example: Land Impact

isLanding	false
landImpact	70

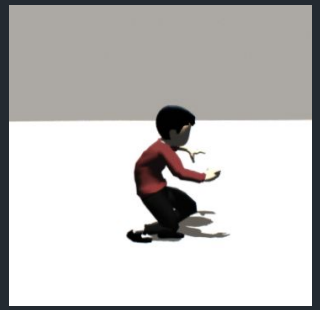
land animation
is not playing

Example: Land Impact

isLanding	false
landImpact	70

based on velocity in the direction of surface normal

Example: Land Impact

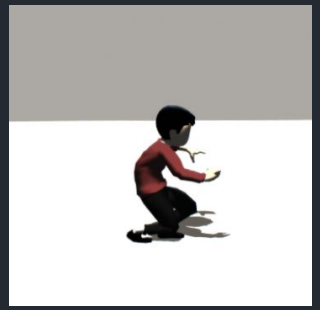


previous	
isLanding	false
landImpact	70

current	
isLanding	true
landImpact	0

isLanding has changed to true: just landed!

Example: Land Impact



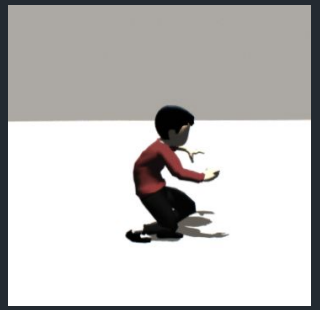
previous	
isLanding	false
landImpact	70

current	
isLanding	true
landImpact	0

use previous frame impact

Result

- Set land impact parameter to 70
- Play land sound



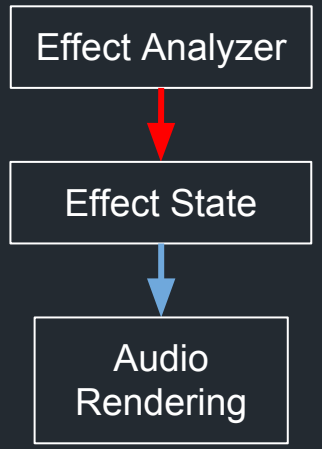
Effect State	
previous	current
isLanding false landImpact 70 etc.	isLanding true landImpact 0 etc.

DEMO: State History

scene: #3D_crane

State Update

- Effect Analyzer
  - Aggregate data
  - Write to Effect State (current)
- Render audio based on Effect State
  - Set values based on previous or current frame state
  - Play sounds as reaction to changes from previous to current
- Copy current state to previous



Order Matters!

Update Effect State

...

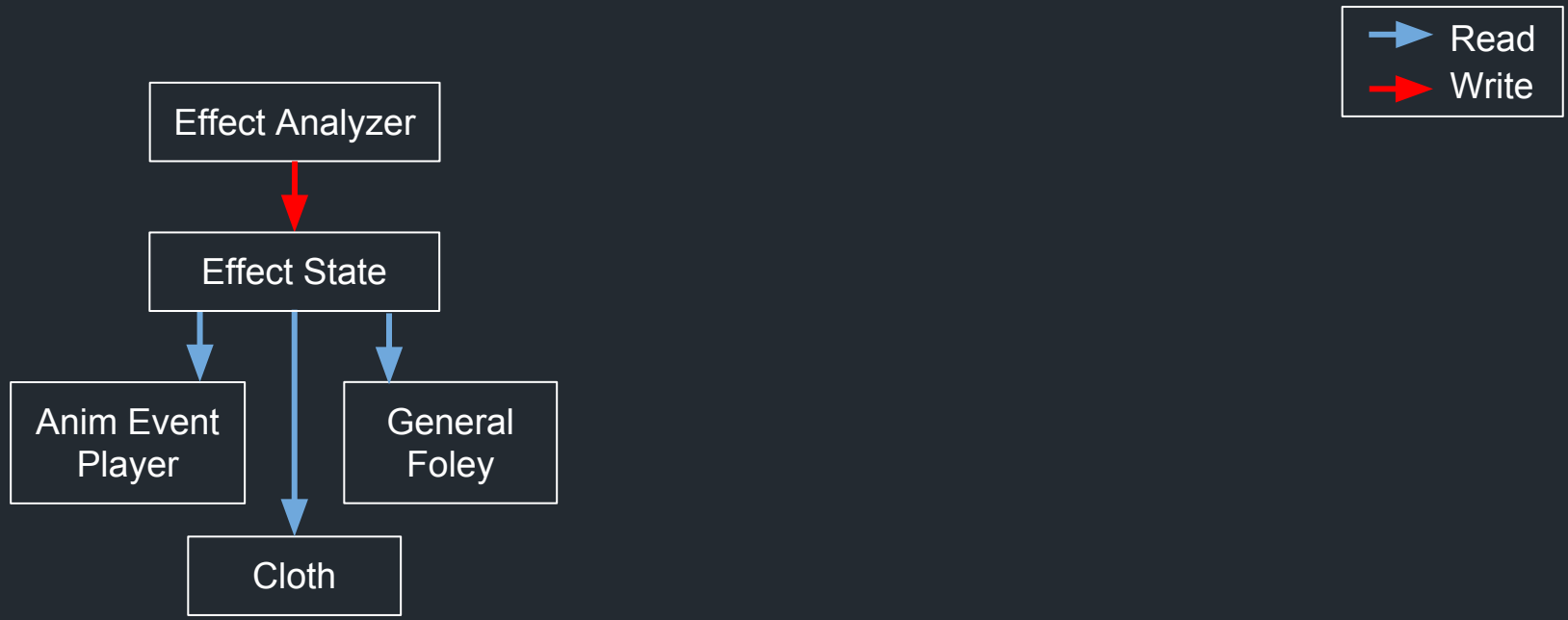
1. Render audio based on Effect State
2. Copy current state to previous

keep these two together to avoid missing any changes

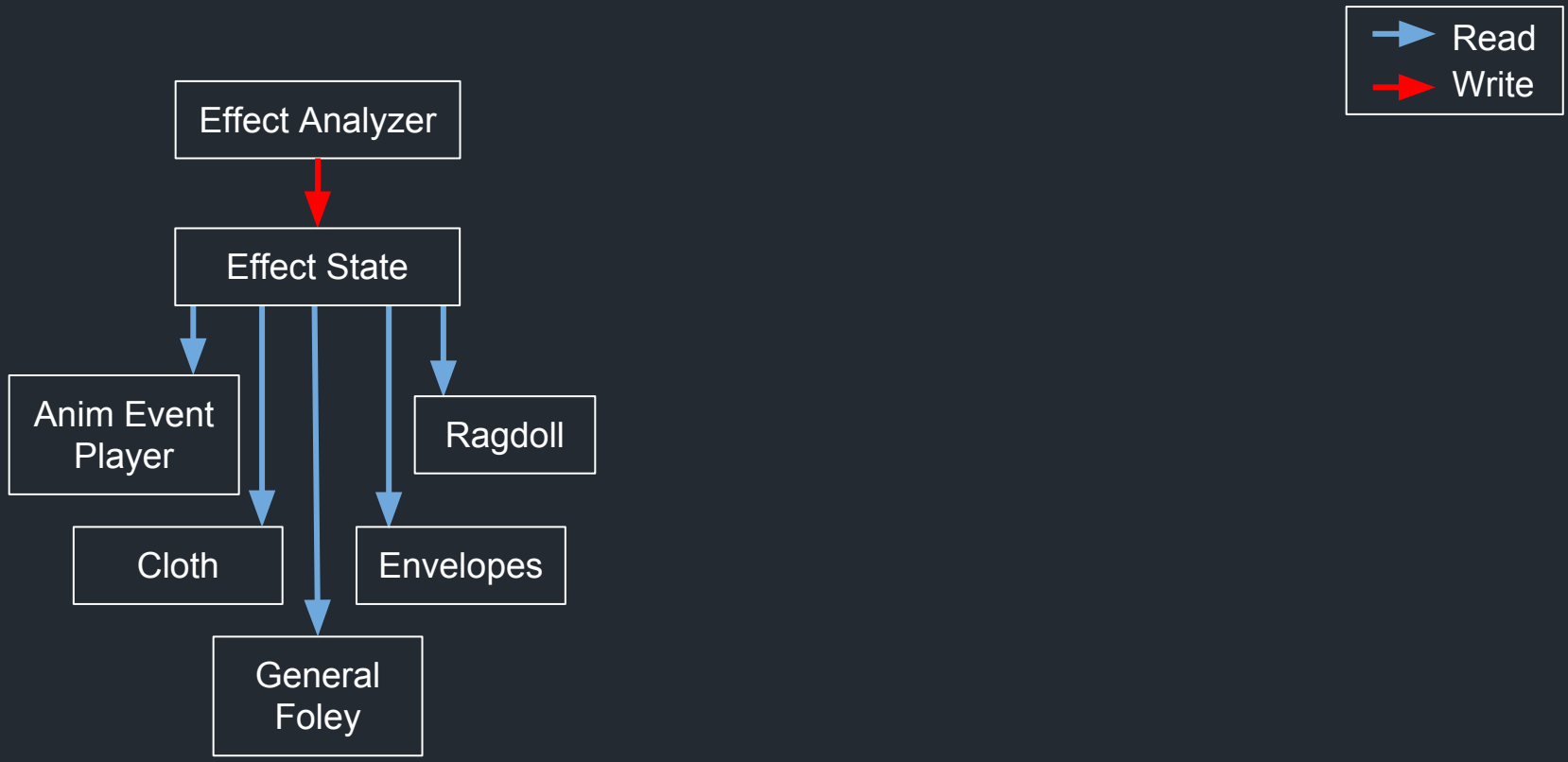
...

Update Effect State

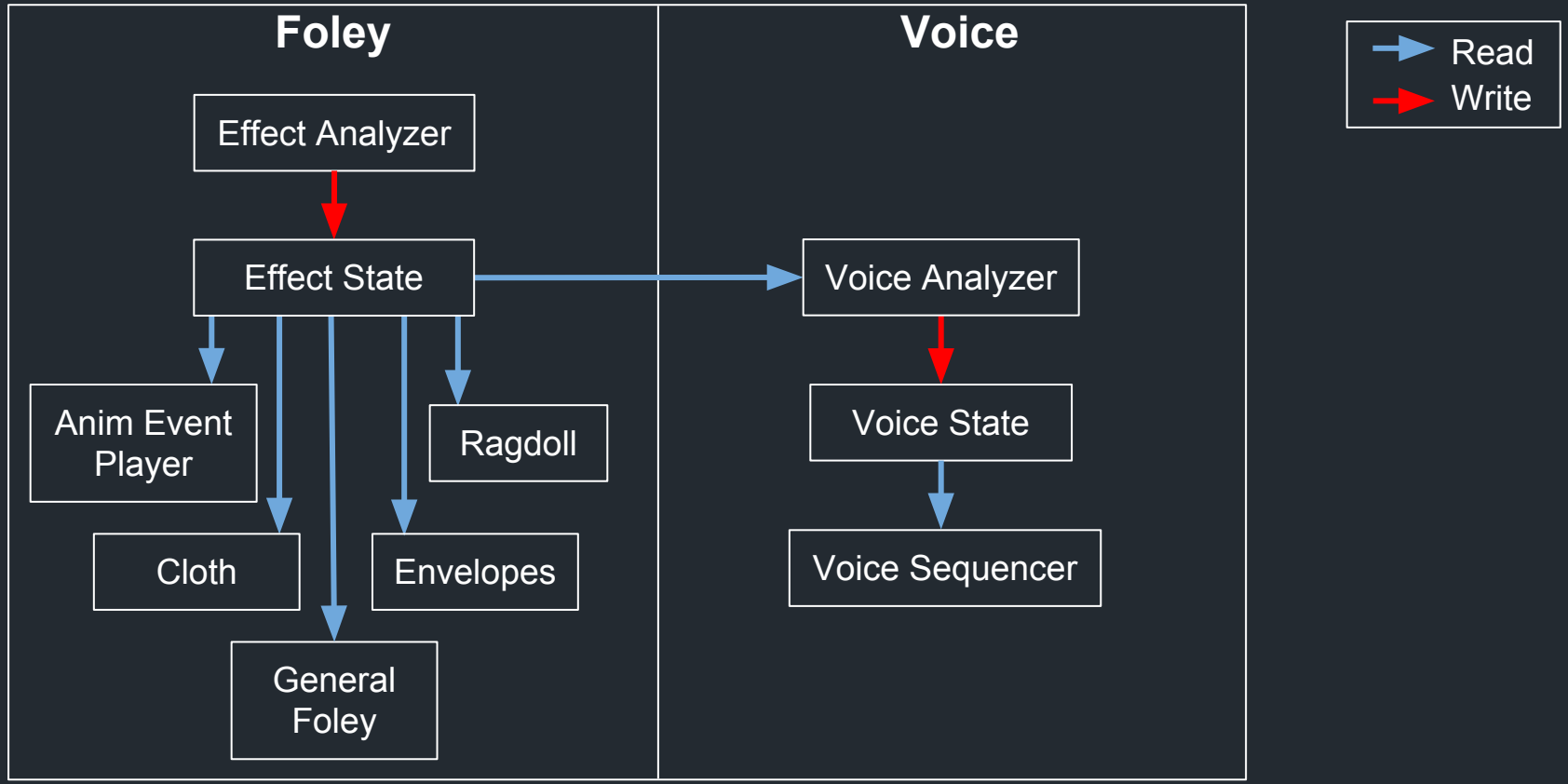
Boy Audio Architecture



Boy Audio Architecture

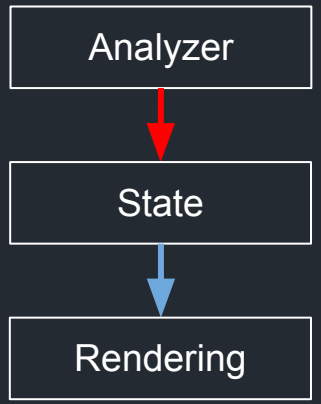


Full Boy Audio Architecture



State, Analysis, Rendering Summary

- Analyzer determines state from game
- State is double-buffered to detect changes and access previous frame data
- Rendering is performed based exclusively on State
- Debugging is easy and errors are trivially categorized

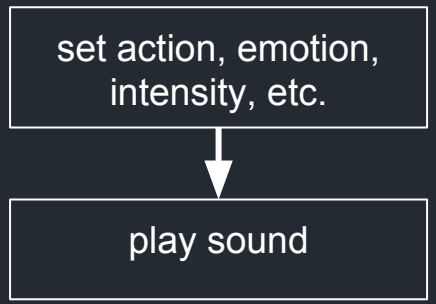


Voice Sequencer

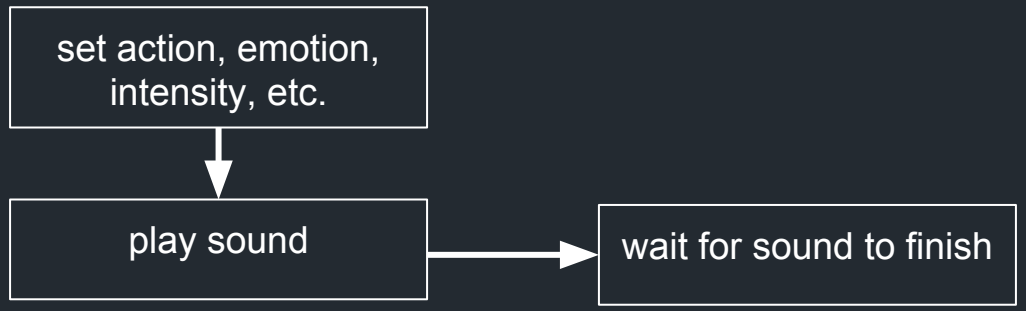
Voice Sound Events

- Played by voice sequencer
- Two modes:
  - Continuous
  - Rhythmic breathing
- Which sound to play is defined by parameters:
  - Action
  - Emotion
  - Intensity
  - etc.
- Intensity is a numeric value:
  - increases with physical exertion
  - decreases when idle

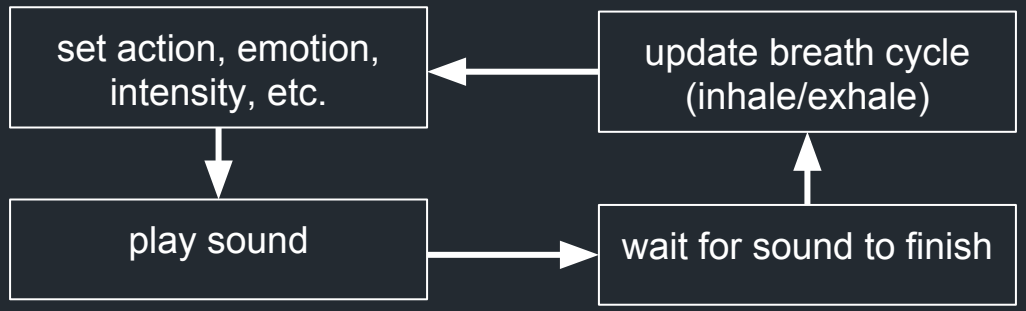
Voice Sequencer: Continuous Mode



Voice Sequencer: Continuous Mode



Voice Sequencer: Continuous Mode



Animation Feedback

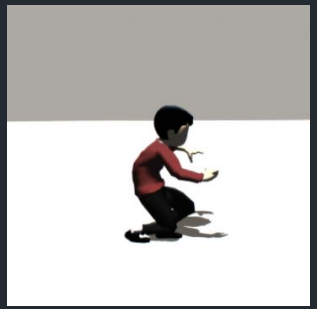
- Breath sounds have varying durations
- Continuous sequencing results in natural, uneven breathing pattern
- Every breath results in a callback to the game
- Callback controls additive breathing animation

Holding Breath

On jump:

if currently inhaling, stop afterwards

if currently exhaling, do a quick inhale, then stop



On land:

restart breathing with exhale

Engagement Actions

Special actions indicate performing work, uses different set of sounds

not engaged

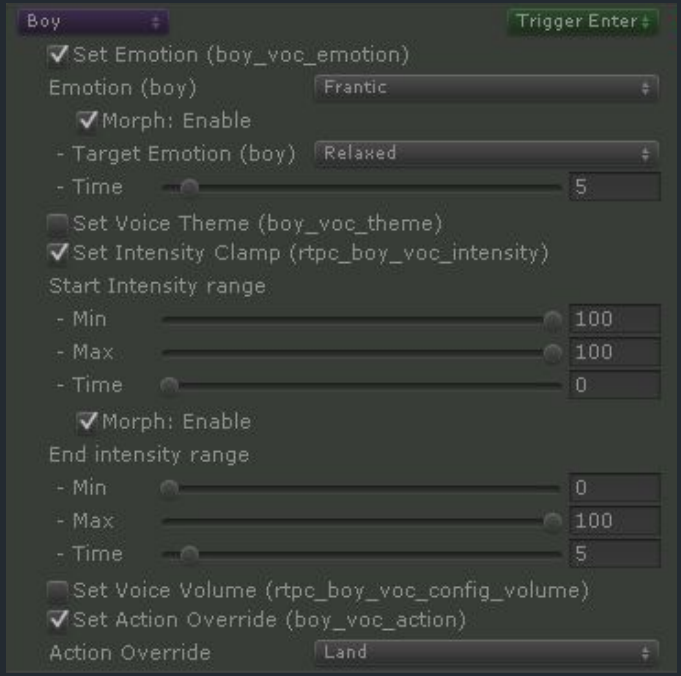
engaged passive

engaged active

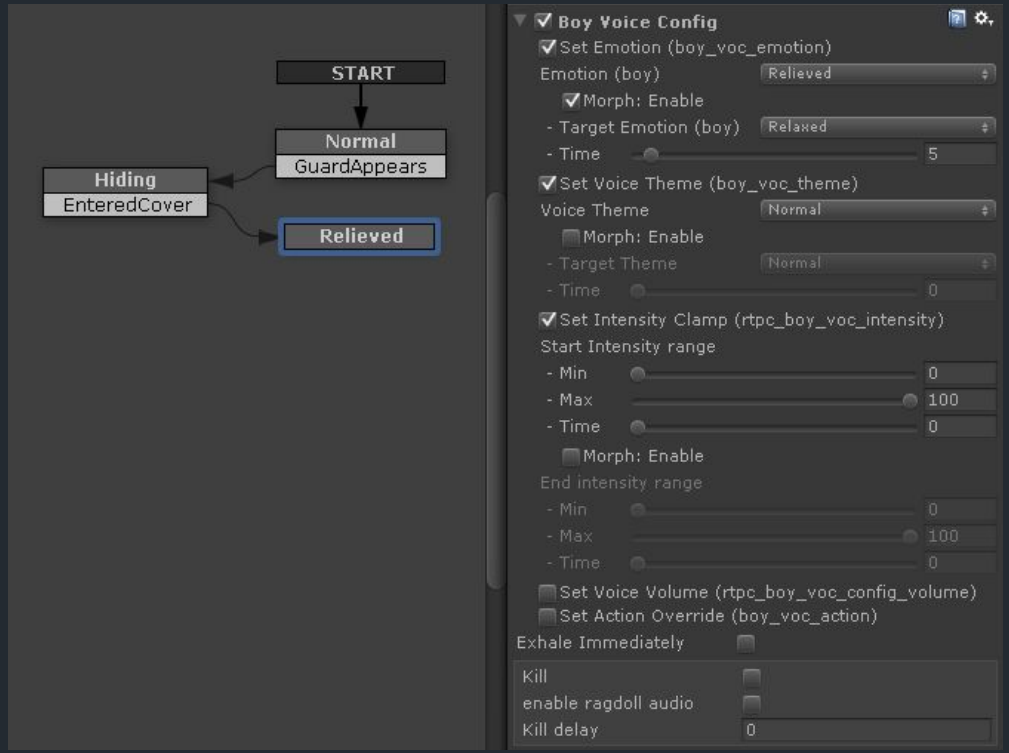
Voice Sequencer Configuration

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

Voice Sequencer Configuration: Trigger box



Voice Sequencer Configuration: State Machine

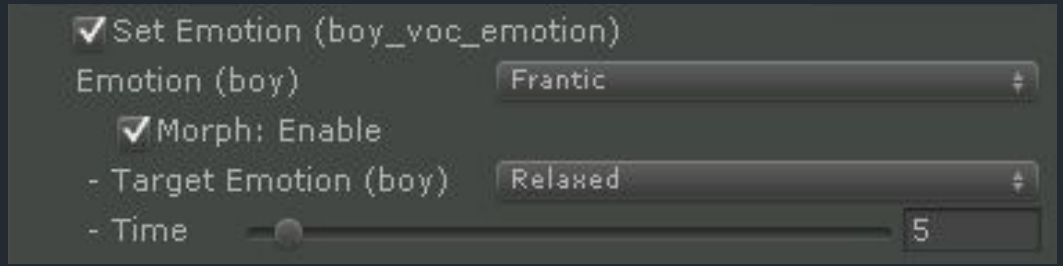


Voice Action Override

- Action is normally determined automatically from animation
- Can be overriden
- Enables defining voice reactions in custom situations
- Includes engaged active/passive

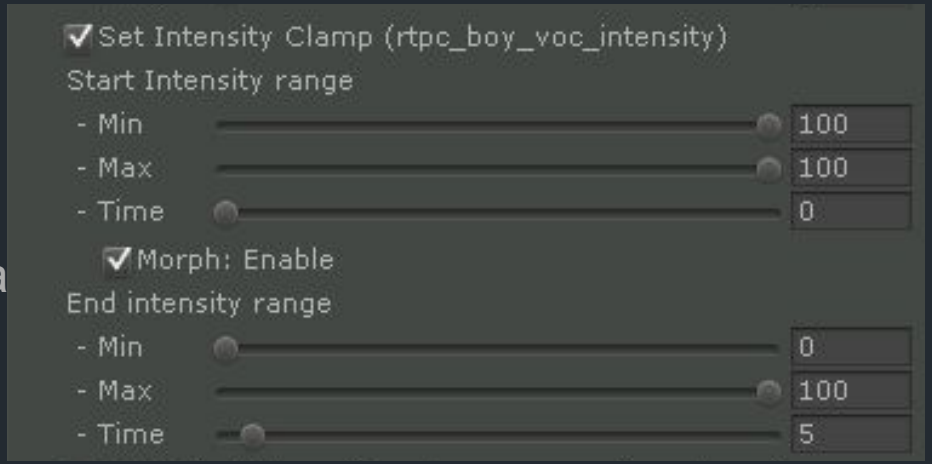
Voice Emotion

- Emotion selects a specific set of voice sounds
- Relaxed, frantic, relieved, etc.
- Morphing allows automatically changing emotion after a specified time



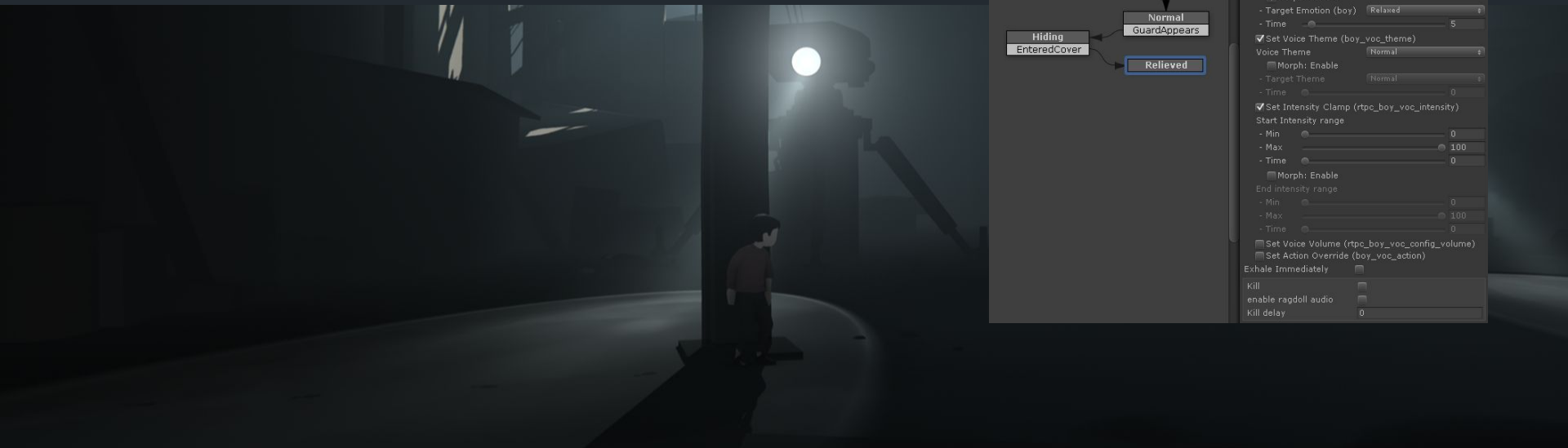
Voice Intensity Clamping

- Voice Intensity selects depth and force of breathing
- Clamping limits intensity value min and max
- Depending on the emotion parameter, intensity defines:
  - Physical exertion level
  - Intensity of character emotion
- Morphing allows clamping to cha gradually



Voice Direction

- Voice configuration is our way of doing voice direction.
- The director (Martin) instructs the actor (voice sequencer) how to emote:
  - based on location on the set (trigger boxes), or
  - based on reacting to events (state machines or scripts)



DEMO: Voice sequencer

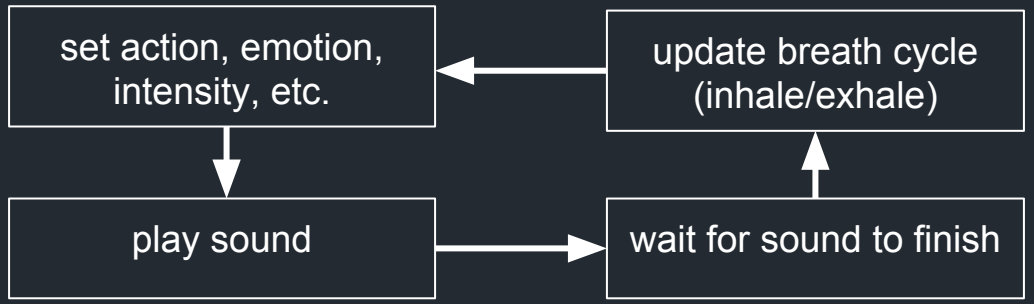
scene: #forestReveal

emotions, rhythmic breathing

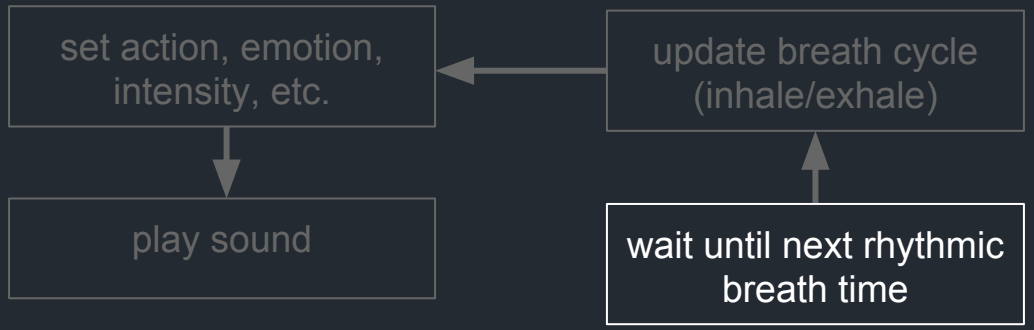
Rythmic Breathing

- Goal: breathing aligns with footsteps when running
- 1 breath for every 2 steps
- Aligns gradually to sound natural

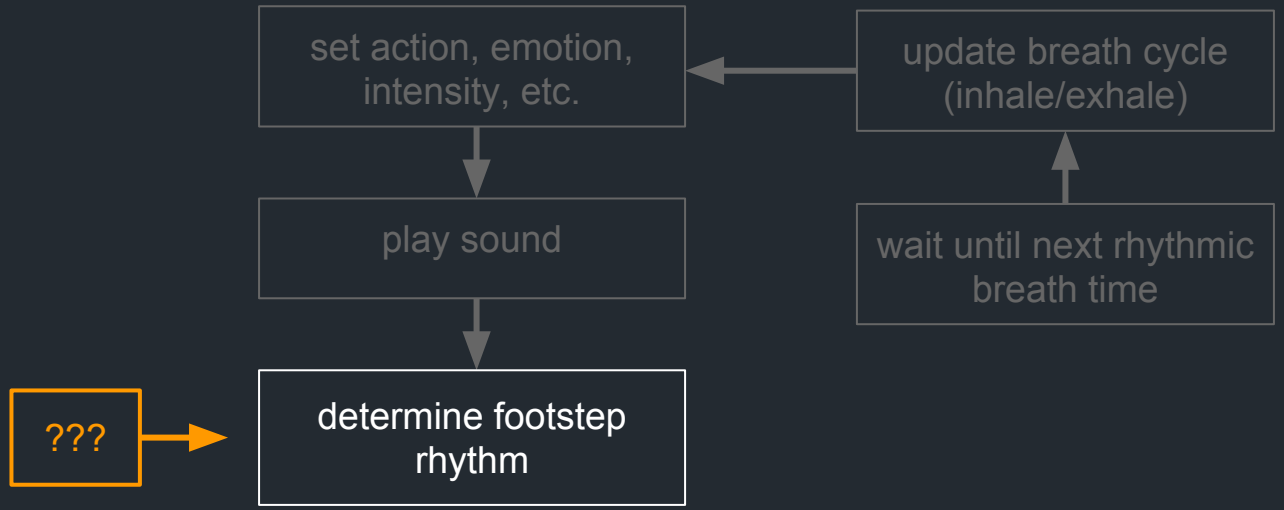
Voice Sequencer: Continuous Mode



Voice Sequencer: Rhythmic Breathing



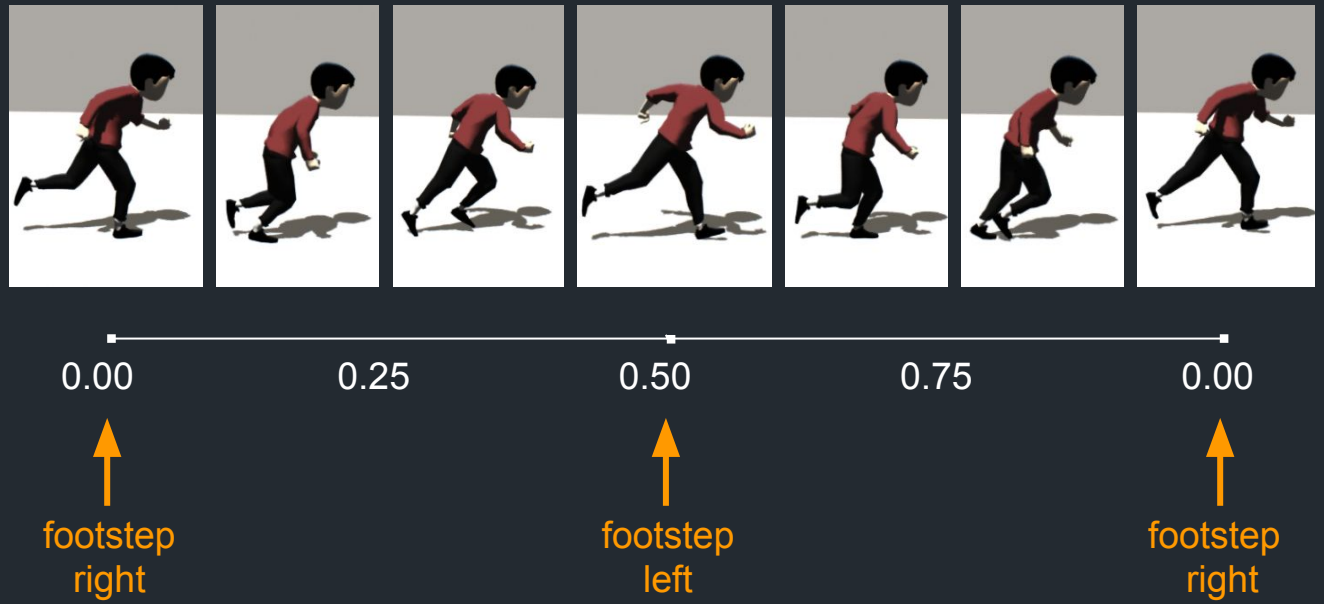
Voice Sequencer: Rhythmic Breathing



Defining Rhythm

- We want to define a rhythmic phenomenon at a point in time
- Frequency
- Phase

Footstep Phase



Footstep Phase

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

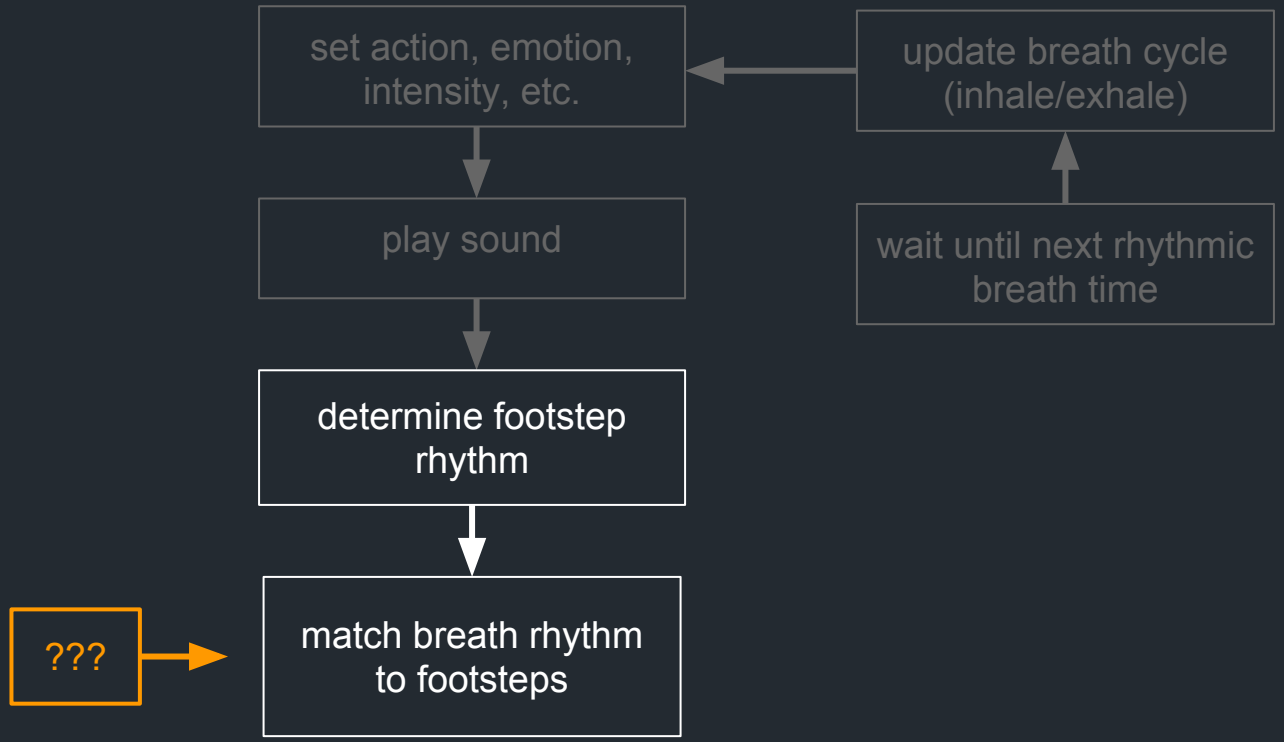
Footstep Frequency

On right footstep animation event:

```
            phase = 0
        frequency = 1 / (currentTime - lastRightStepTime)
lastRightStepTime = currentTime
```

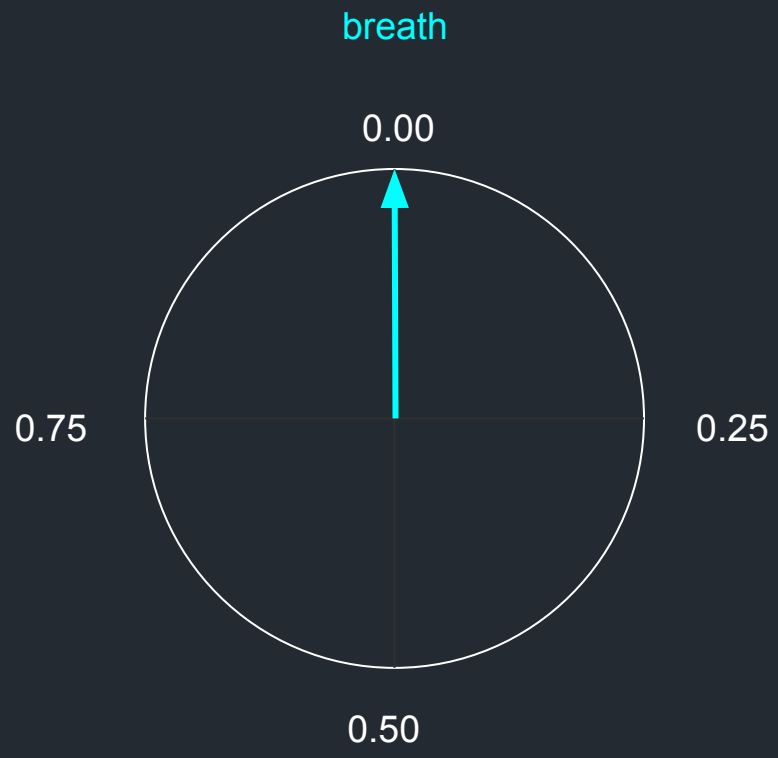
- Interpolation is used to smoothen measured frequency
- We actually use footsteps from both feet for more precision

Voice Sequencer: Rhythmic Breathing



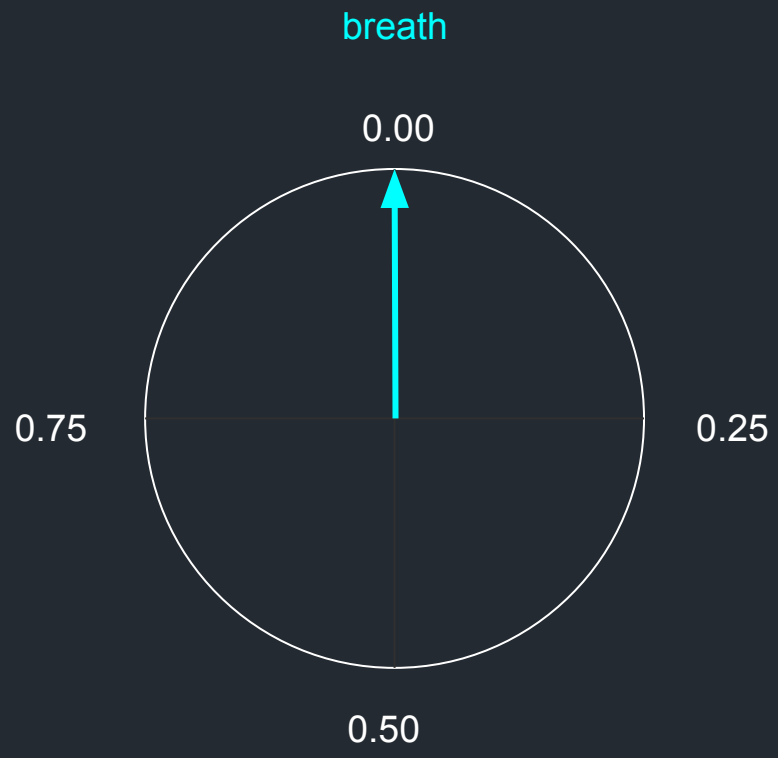
Breath Phase

- Breathe when phase is 0



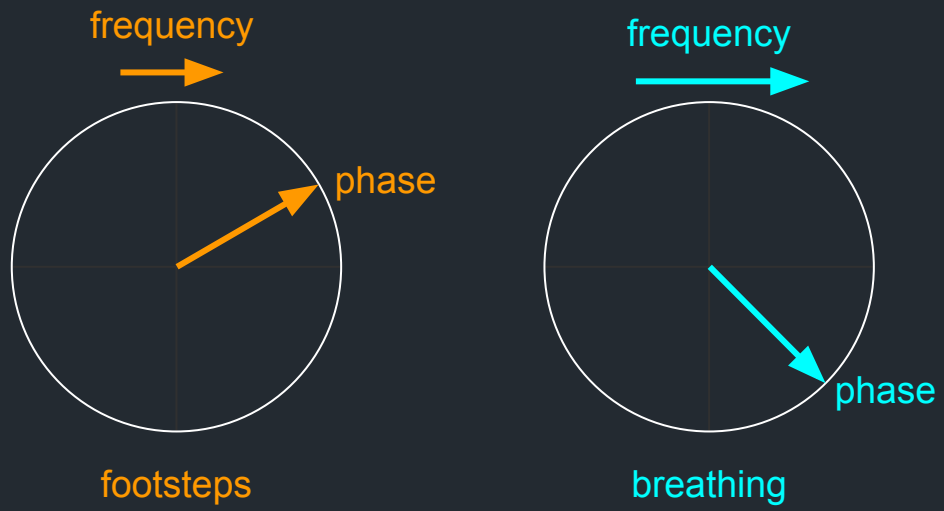
Breath Frequency

- 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



Breath Frequency

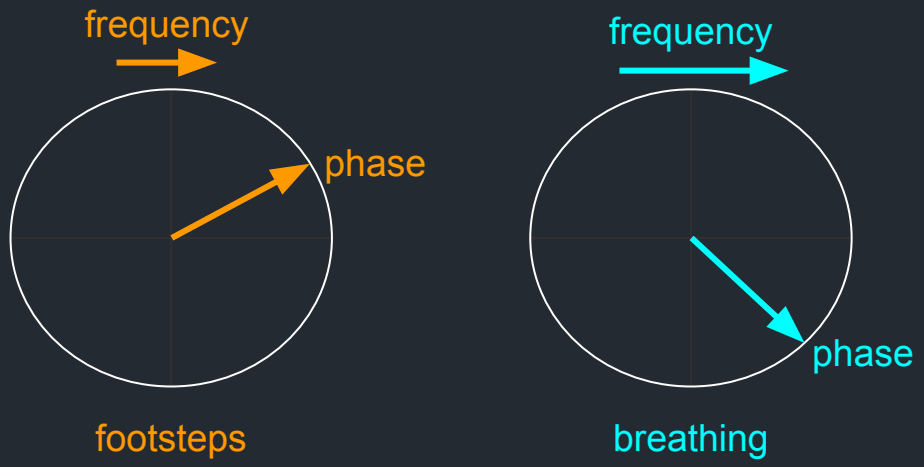
- Gradually align breath rhythm to footstep rhythm
- Align two **frequency, phase** pairs



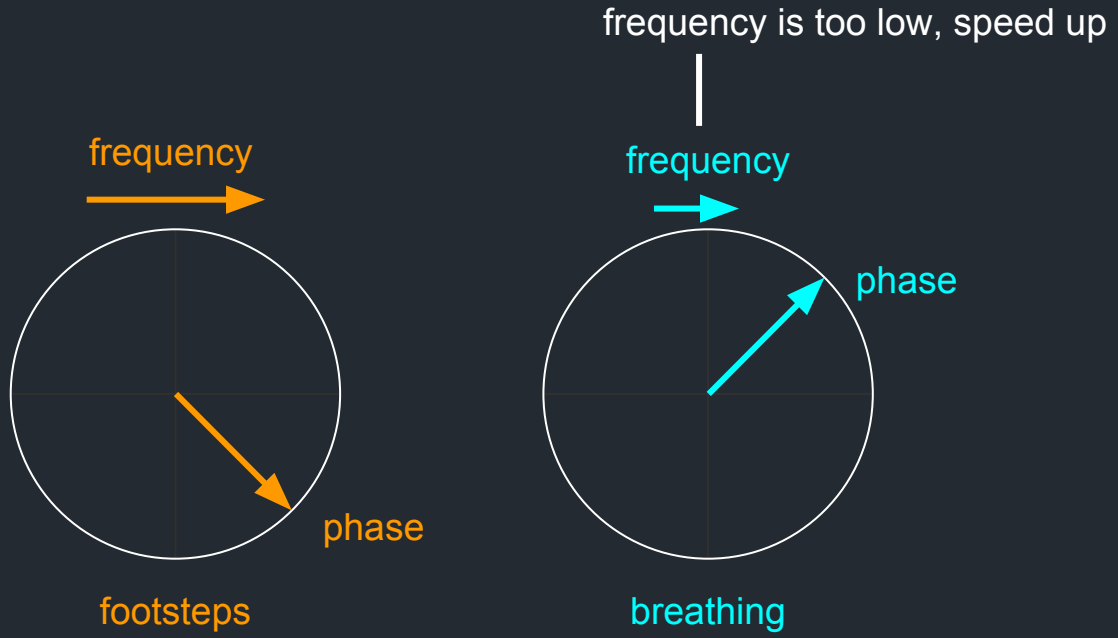
Solution: Beat Matching

- Gradually align breath frequency to footstep frequency
- Compensate breathing frequency for phase offset

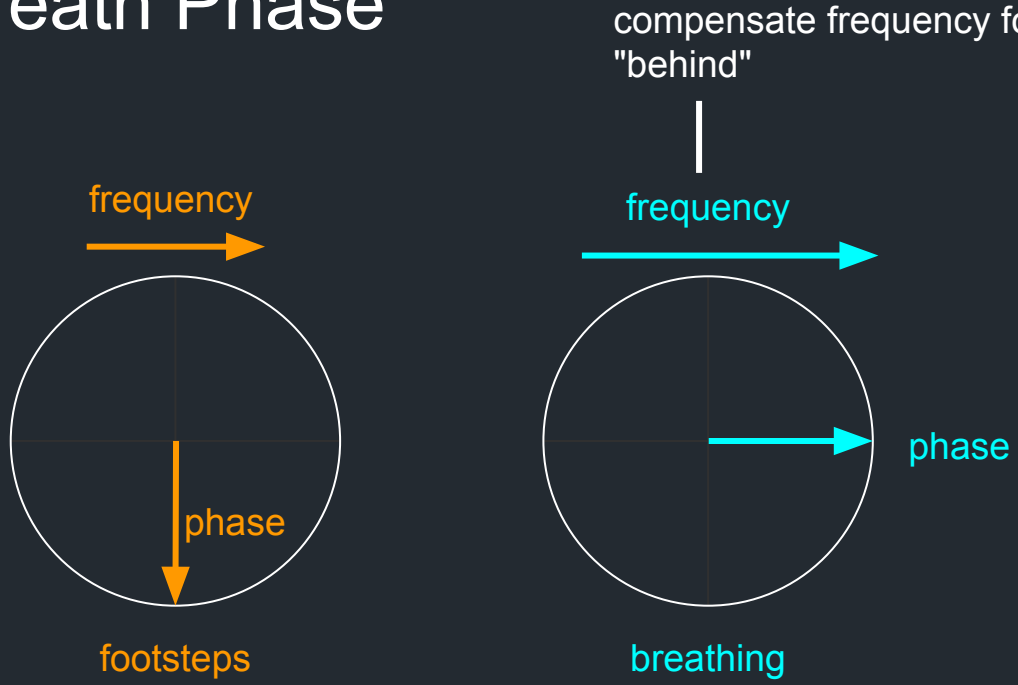
- Like a DJ that uses pitch adjust without nudging the record



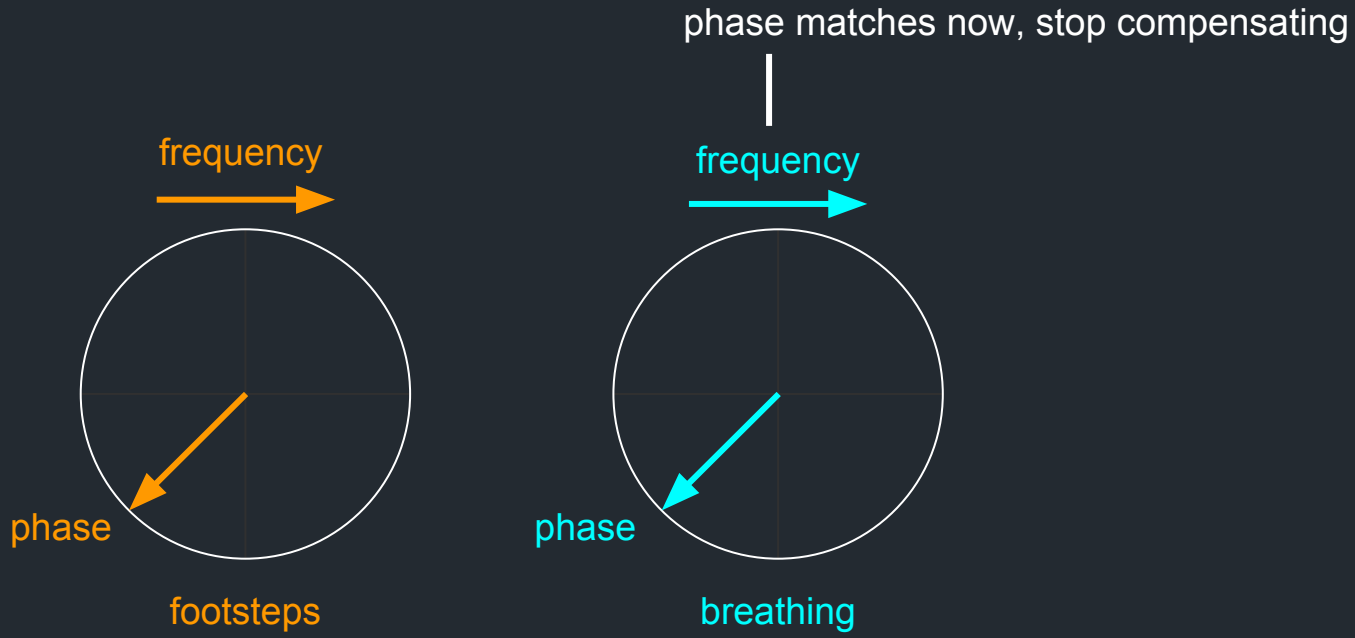
Adjust Breath Frequency



Adjust Breath Phase

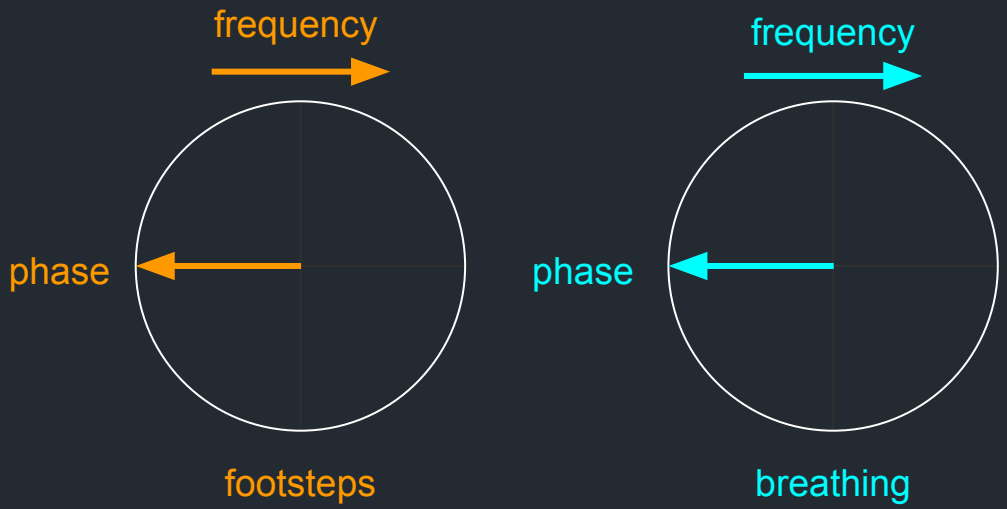


Breath Phase Adjusted

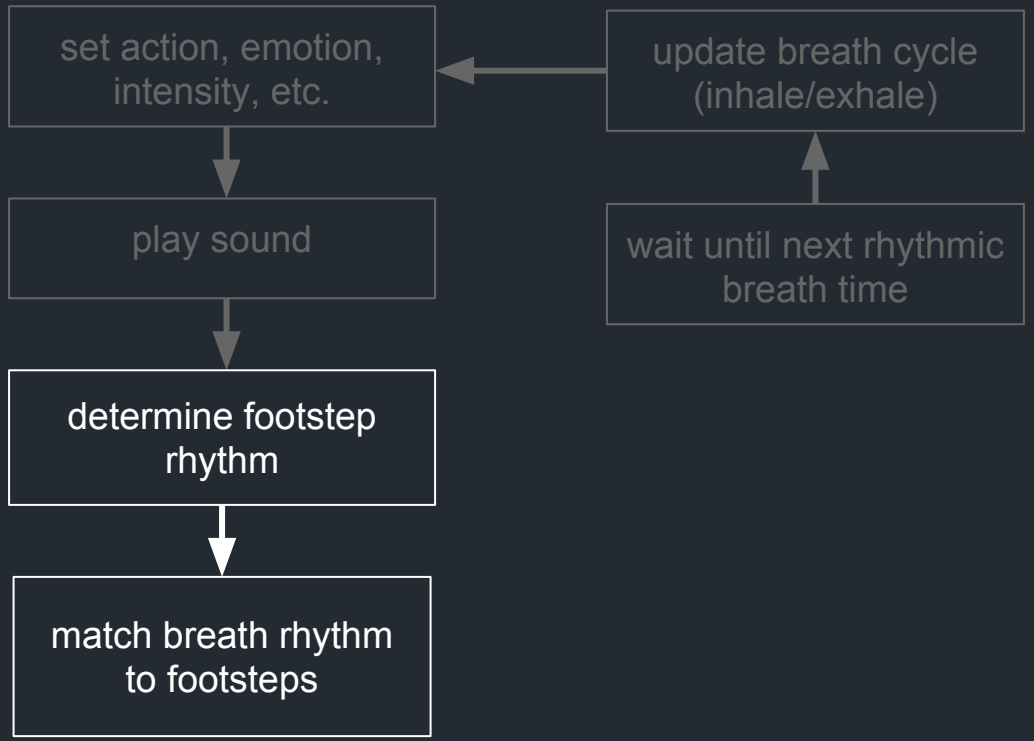


Synchronized

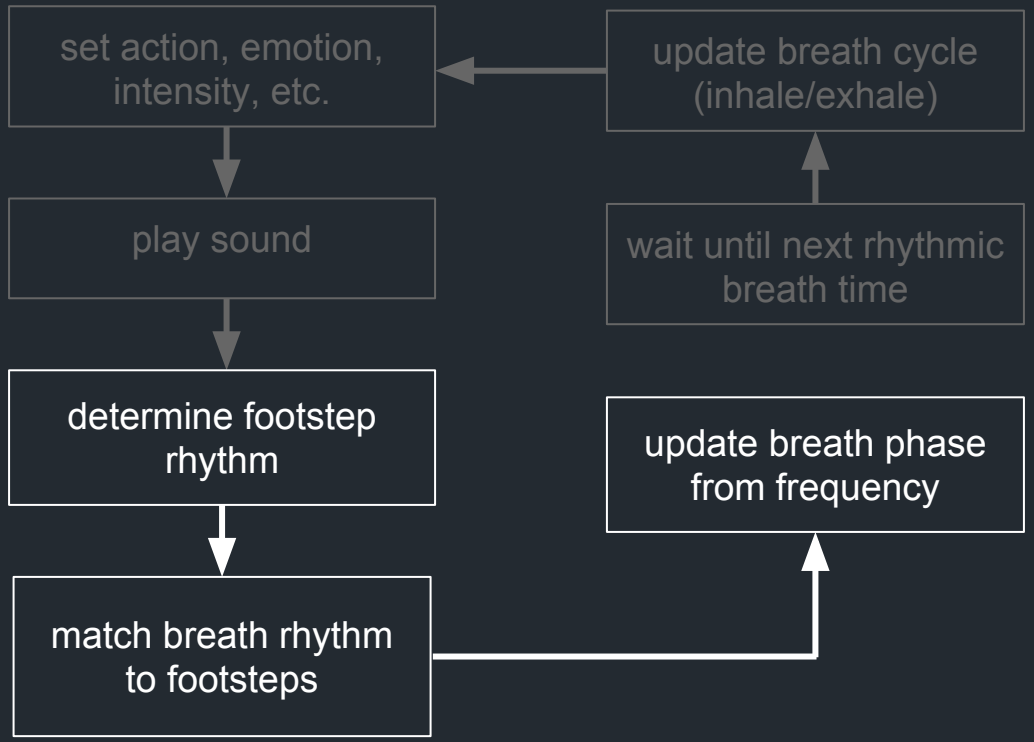
perfect sync!



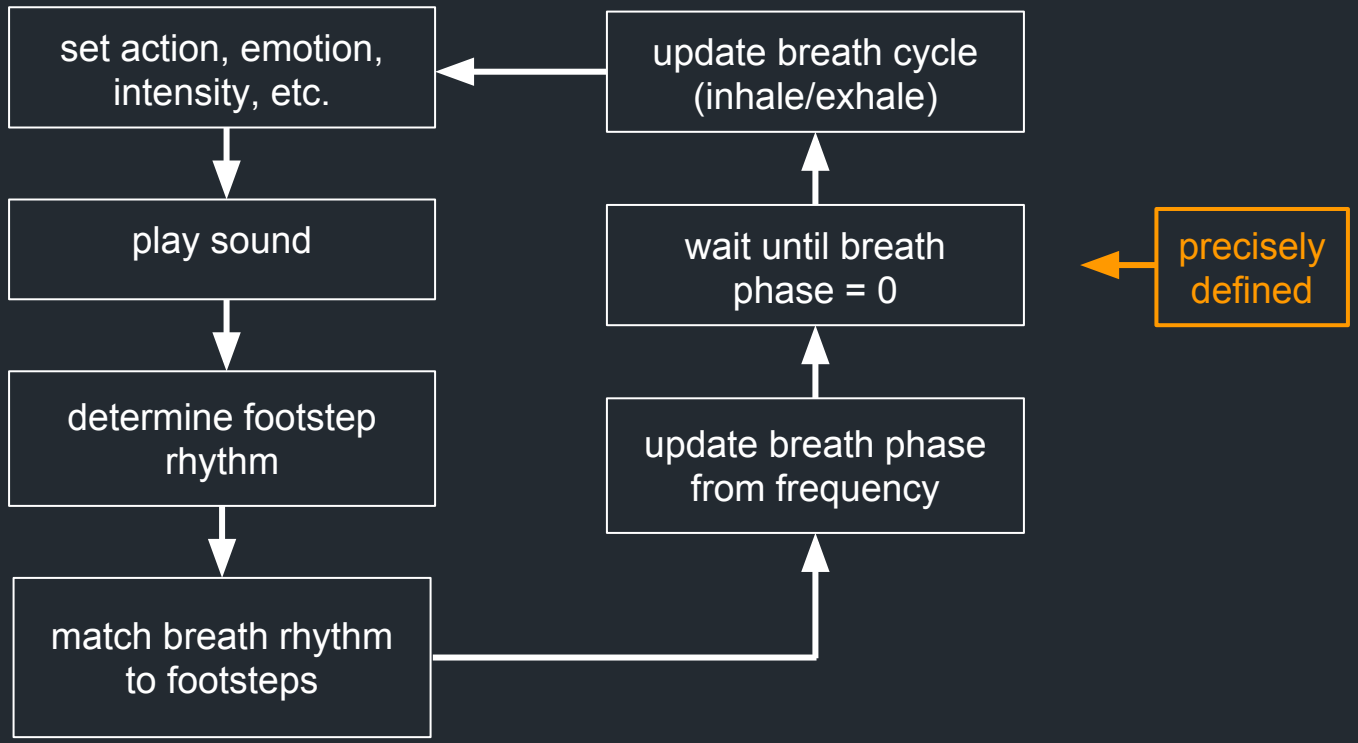
Voice Sequencer: Rhythmic Breathing



Voice Sequencer: Rhythmic Breathing

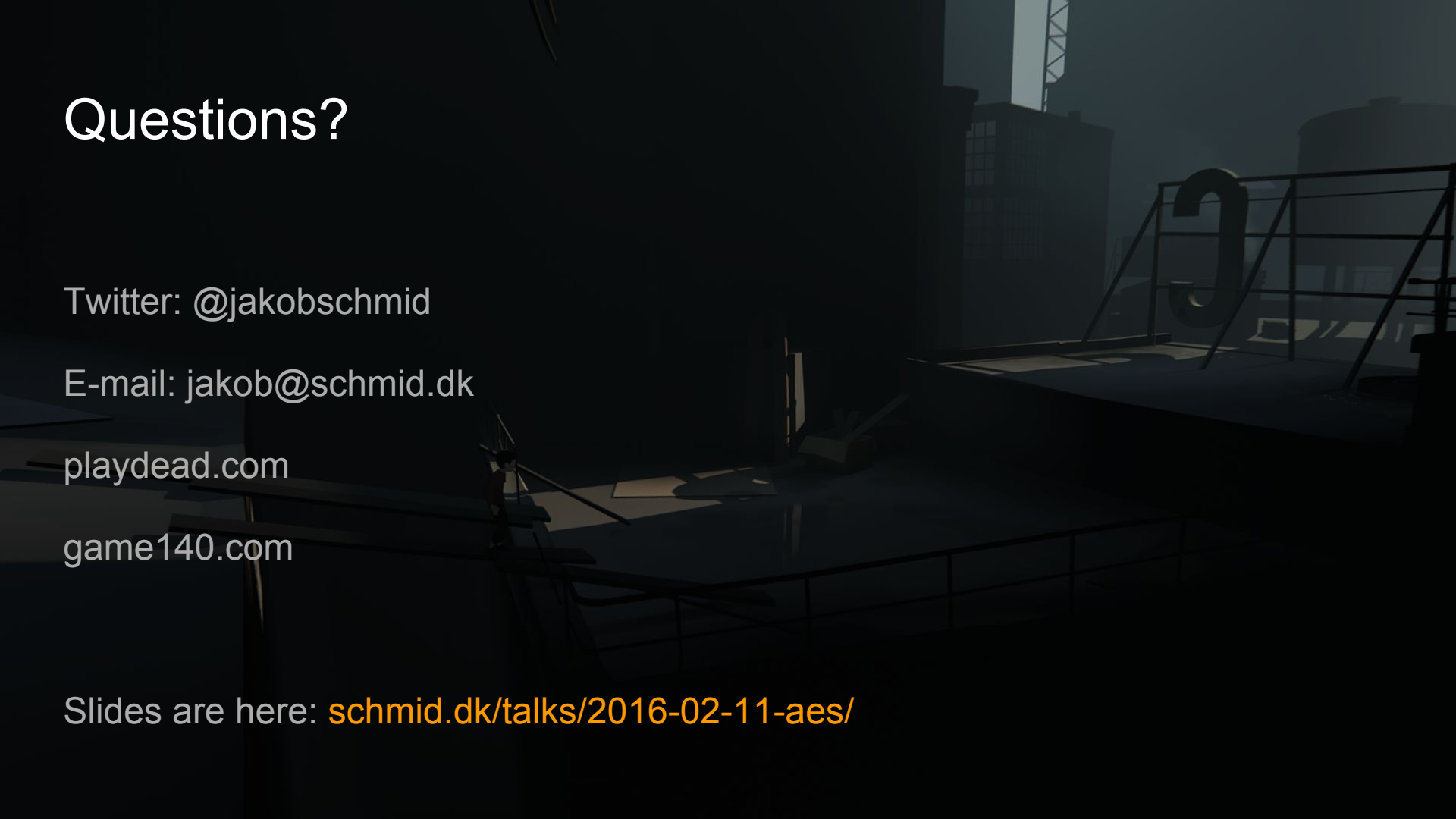


Voice Sequencer: Rhythmic Breathing



Voice Sequencer Summary

- Sound events are selected based on action, emotion, intensity, etc.
- Voice direction with trigger boxes and state machines
- Continuous sequencing of sound events
- Rhythmic breathing uses DJ-style beat matching to align breathing to footsteps



Questions?

Twitter: @jakobschmid

E-mail: jakob@schmid.dk

playdead.com

game140.com

Slides are here: schmid.dk/talks/2016-02-11-aes/

Beatmatching Code - steal it!

```
    // Gradually matches frequency and phase of a periodic function to another periodic function (~ beat
matching).
    // Code by Jakob Schmid, PLAYDEAD, 2016. Free to use for anything.
    void MatchFreqAndPhase(float t, float dt, float targetFreq, float targetPhase, ref float freq, ref float
phase)
    {
        float smallestPhaseOffset;

        // Get phase offset
        float deltaPhase = targetPhase - phase;

        // account for phase being a modular number (0.9 is equally close 0 and 0.8)
        // - see also http://en.wikipedia.org/wiki/Modular_arithmetic
        if (deltaPhase > 0.5f)
            smallestPhaseOffset = deltaPhase - 1f;
        else if (deltaPhase < -0.5f)
            smallestPhaseOffset = deltaPhase + 1f;
        else
            smallestPhaseOffset = deltaPhase;

        // 'Beat match' freq to targetFreq and adjust phase using frequency
        float adjustedFreq = targetFreq + smallestPhaseOffset * BREATH_PHASE_ACCEL;
        if (freq < adjustedFreq)
            freq = Mathf.Lerp(freq, adjustedFreq, BREATH_ACCEL_UP   * dt);
        else
            freq = Mathf.Lerp(freq, adjustedFreq, BREATH_ACCEL_DOWN * dt);
    }
```

AES 2016 - Abstract and CV

Title
The Boy from INSIDE: Uncompromising Character Audio Implementation

Abstract
A 5-year collaboration between sound designer Martin Stig Andersen and programmer Jakob Schmid on INSIDE, Playdead's follow-up to award-winning game LIMBO, has led to an uncompromising audio implementation, unique in its design choices and level of detail. This talk focuses on the design and implementation of foley and voice for the main character of INSIDE.

It will be explained how game state and character geometry is analyzed to provide data for audio systems. A method for context-dependent sound selection for footsteps is described, along with the implementation of a breath sequencer that reacts to player input and animation and matches rhythmic breathing to footsteps.

Finally, a selection of tools used to configure and debug audio will be described.

CV
Jakob Schmid is the audio programmer at Playdead in Copenhagen. He has a master's degree in computer science from Aalborg University. Since 2011, he's been working with Martin Stig Andersen on Playdead's upcoming game, 'INSIDE'.
In his spare time, he created the music and sound for colleague Jeppe Carlsen's game '140', which went on to win several awards including the 2013 Independent Games Festival award for 'Excellence in Audio'.