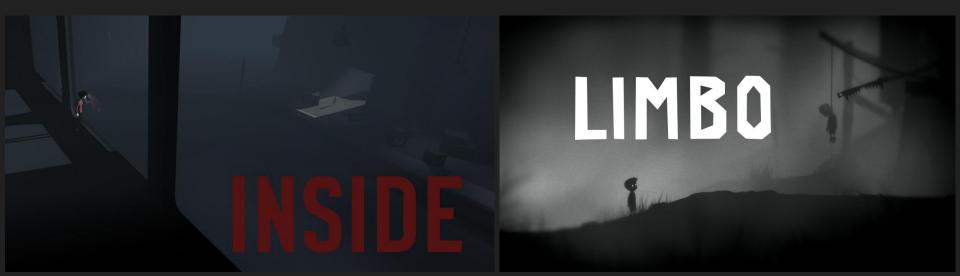
Unbreaking Immersion Audio Implementation for INSIDE

Wwise Tour 2016 Martin Stig Andersen and Jakob Schmid PLAYDEAD

Martin Stig Andersen

Audio director, composer and sound designer



Jakob Schmid

Audio programmer at Playdead

Composer and sound designer on 140



INSIDE

PLAYDEAD

June 29 on Xbox One, July 7 on Steam

Commenced 2010





Playdead Audio Team

Martin Stig Andersencomposer, sound designerSØS Gunver Rybergcomposer, sound designerAndreas Frostholmsound designerJakob Schmidaudio programmer

Unbreaking Immersion

- Introduction
- Voice
- Scene Change
- Performance

Slides available online. Link on last slide!



Voice Concept

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



INSIDE Technology

Unity

Audiokinetic Wwise

Modified Wwise-Unity plugin

PlayMaker

Voice Sequencer

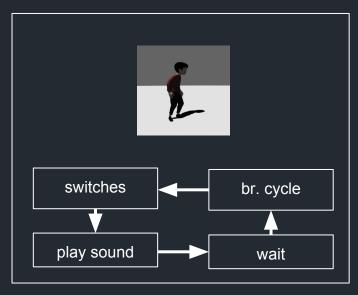
- Sequencer implemented in C# using Wwise callbacks
- Sequences voice sound events, alternating between inhale and exhale

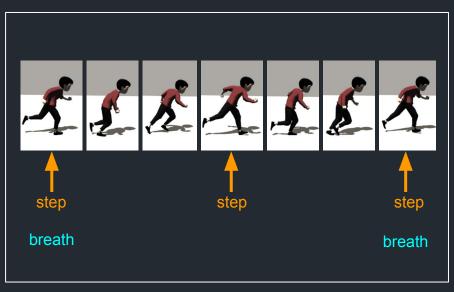
Voice Sound Events

- 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

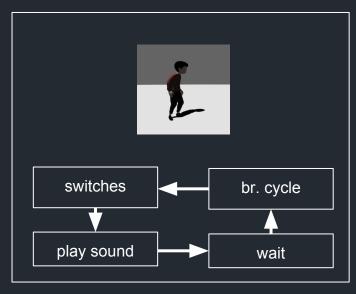
Continuous Mode

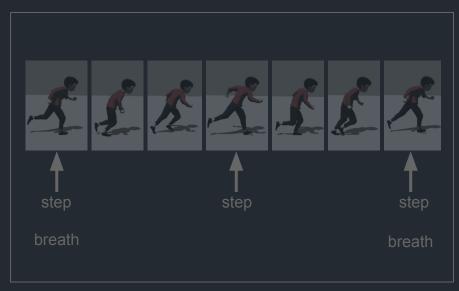


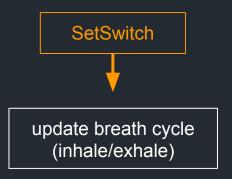


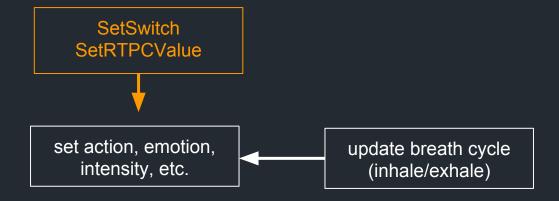
Voice Sequencer Modes

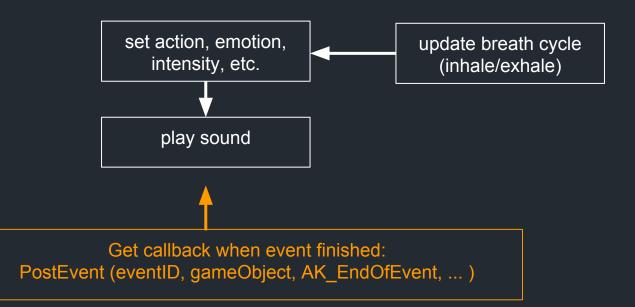
Continuous Mode

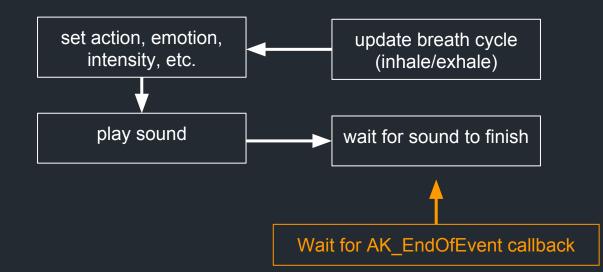


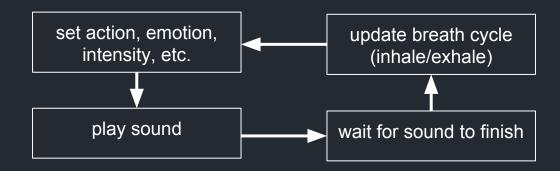






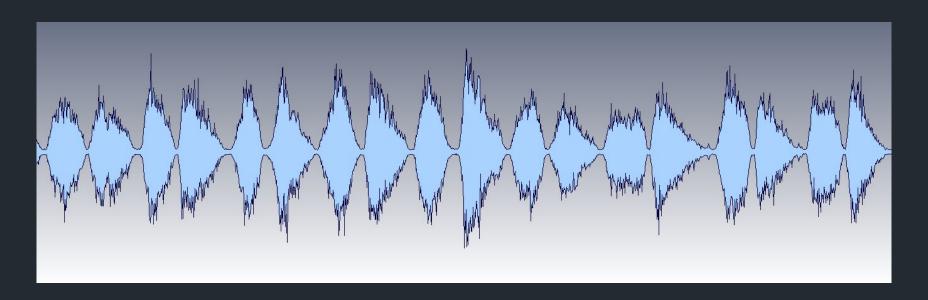






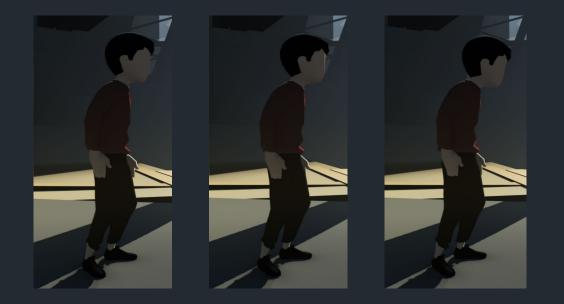
Natural Breathing

- 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:

On land:

if currently inhaling, stop afterwards

if currently exhaling, do a quick inhale, then stop



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

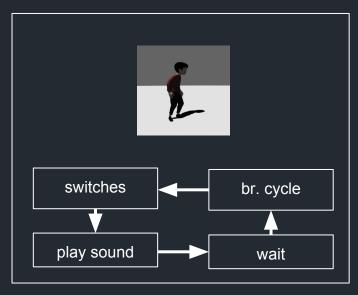
engaged active

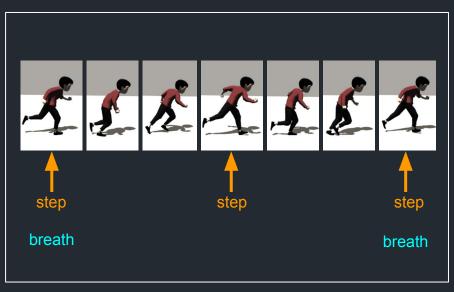
Voice Wwise Setup

-

Voice Sequencer Modes

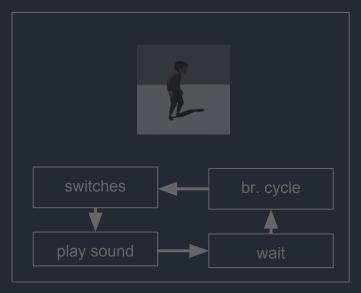
Continuous Mode





Voice Sequencer Modes

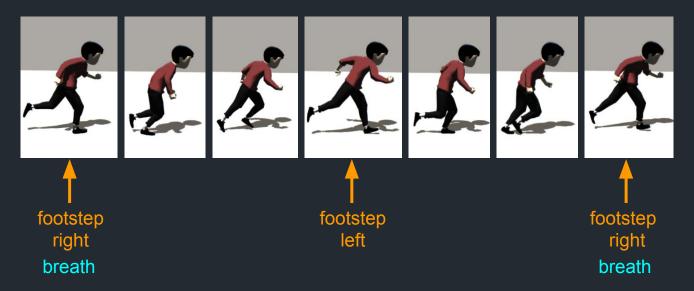
Continuous Mode





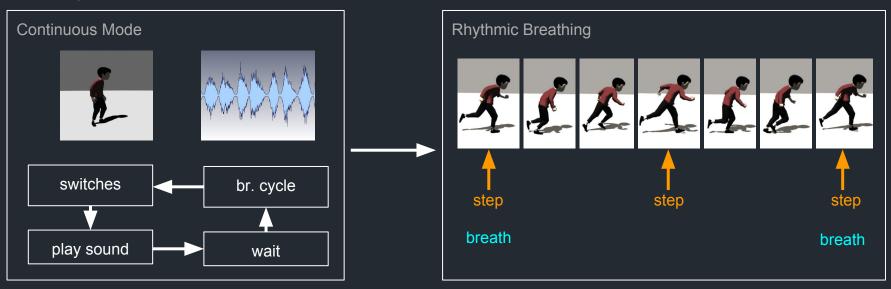
- Goal: breath should align with footsteps when running
- Non-continuous sequencing

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

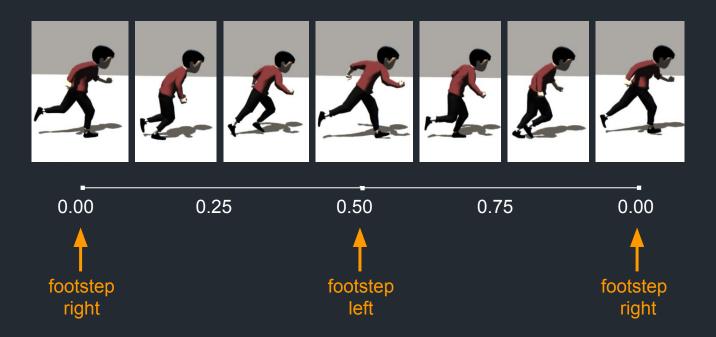


Rhythmic Breathing Transition

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

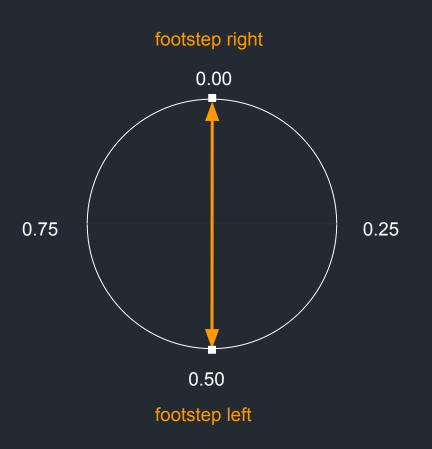


Run Cycle Phase



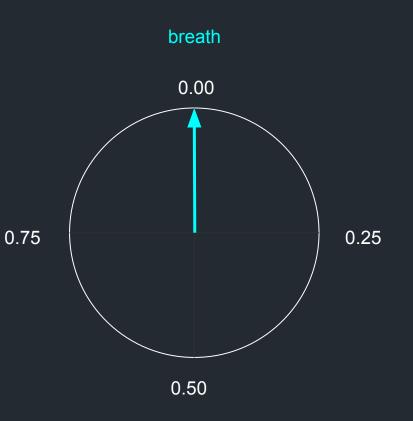
Run Cycle Phase

- 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

• Who knows about aligning two **frequency**, phase pairs?



Solution: Beat Matching

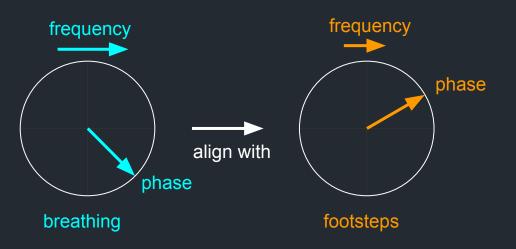
- Who knows about aligning two frequency, phase pairs?
- DJs do.





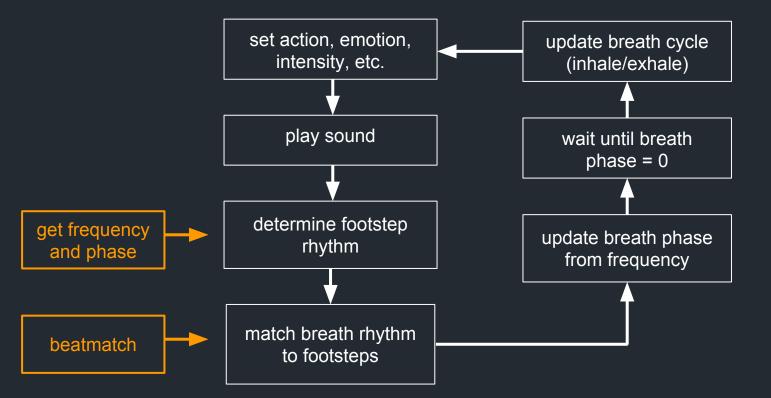
Solution: Beat Matching

- 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 Direction

- <u>Voice direction</u> 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
 - \circ action
 - \circ emotion
 - intensity

Voice Configuration: Trigger box



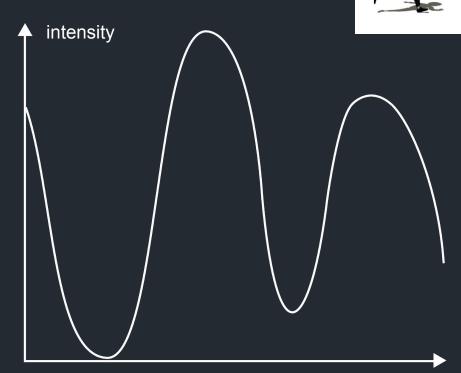
Boy 🗧		rigger Enter 🗧
▼Set Emotion (boy_voo	_emotion)	
Emotion (boy)		
✔Morph: Enable		
- Target Emotion (boy)	Relaxed	
- Time 🔲 🖳		
Set Voice Theme (boy	y voc theme)	
▼Set Intensity Clamp (y)
Start Intensity range		
- Min		100
- Max		100
- Time 🔍		— 0
▼Morph: Enable		
End intensity range		
- Min 🔿		- 0
- Max		- 100 100
- Time		
Set Voice Volume (rtp	oc_boy_voc_config_vo	lume)
▼Set Action Override (I		
Action Override		

Voice Configuration: State Machine

	Boy Voice Config Set Emotion (boy_voc	_emotion)	₪ ≎,
START	Emotion (boy)		
	🗸 Morph: Enable		
Normal		Relaxed	
GuardAnnears	- Time 🔄 🧑		
Hiding EnteredCover	Set Voice Theme (boy		
	Voice Theme		
Relieved	Morph: Enable		
	- Time 💿		
	Set Intensity Clamp (r	tpc_boy_voc_inten	isity)
	Start Intensity range		
	- Min 💿		0
	- Max		0 100
	- Time		
	Morph: Enable		
Į	- Time 💿		
	Set Voice Volume (rtp		volume)
	Set Action Override (b Exhale Immediately	oy_voc_action)	
	Kill delay		

Voice Intensity

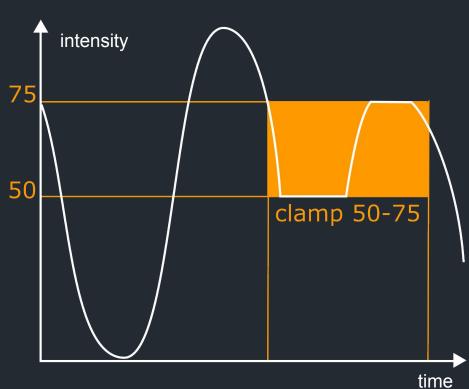
- Boy movement generates exhaustion
- Voice intensity = lowpass filtered exhaustion
- Voice Intensity selects depth and force of breathing
- Depending on the emotion parameter, intensity defines:
 - Physical exertion level
 - \circ Intensity of character emotion



Voice Intensity Clamping

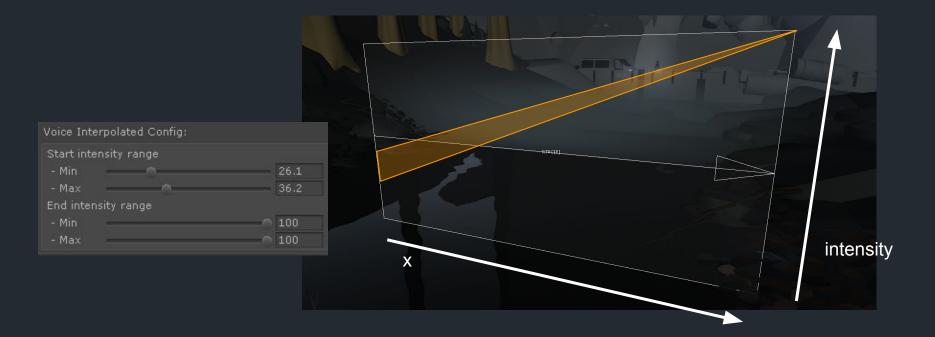
• Clamping constrains intensity to a given range

VSet Int	ensity Clai	mp (rtpc_bo	oy_voc_	intensity)
Start Inte	nsity rang	le		
- Min				50
- Max	0 <u></u>		<u></u>	75
- Time	6			0
Mor	oh: Enable			



Voice Intensity Interpolation in Space

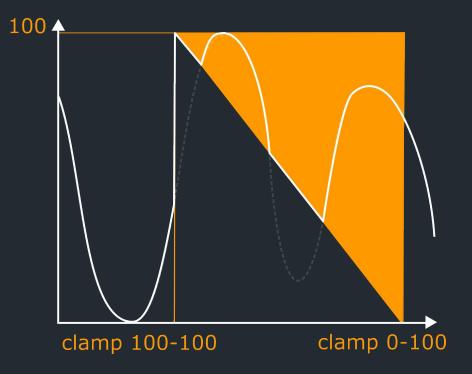
• Useful for indicating proximity to danger



Voice Intensity Interpolation over Time

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

▼Set Int	ensity Clamp (rtpc_boy_v	oc_intensity)	
Start Inte	ensity range		
- Min			100
- Max		0	100
- Time			0
🗸 Mor	ph: Enable		
End inter	isity range		
- Min			0
- Max	-		100
- Time	-0-		5



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

Shockwave Demo

Scene Change

- When main character dies, scenes are reloaded
- Audio should retain state and continue during load
- When reload is complete, audio should switch to new state instantly
- We call this a <u>scene change</u>



Image credit: The New York Times: 'Times Lapse Video: Behind the Scenes at the Metropolitan Opera

Scene Change Events

Boy death Fade out start Fade out complete Unload scenes Load scenes Fade in start Fade in complete

Scene Change Implementation

Boy death Fade out start Fade out complete

Unload scenes Load scenes Fade in start

Fade in complete

- death event
- prepare_spawn_[savepoint]
- pause Wwise updates (RenderAudio)

- scene stop events

- scene and global start events
- post spawn_[savepoint]

- resume Wwise updates 🤜

Scene Change Implementation

Wwise updates are paused during scene change:

- Retains audio state during scene change
- Wwise commands accumulated during load
- All commands are executed at once when scene change is complete



Image credit: www.artsjournal.com

Creates the illusion of no time passing during scene change

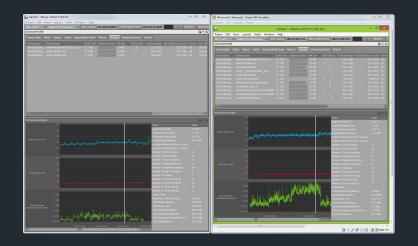
Performance

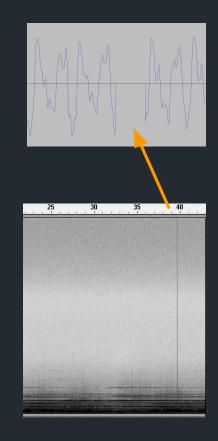
ĩ

1

Debugging INSIDE Audio

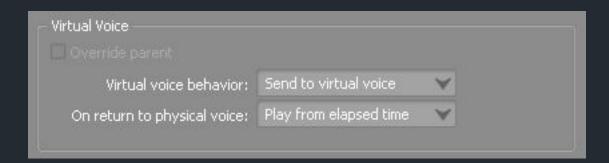
- 2D gameplay: predictable, testable performance
- Profile entire playthrough and analyze the data
- Record audio digitally and inspect for glitches





CPU Performance

- Virtual Voices are your friends
- Inaudible sounds are still updated, but not mixed



Fixing glitches

- Scene Change requires large command queue (2 MB)
- Caused glitches with standard 512 sample audio buffer
- Audio buffer size adjusted to 1024 samples

initSettings.uCommandQueueSize = 2048 * 1024; initSettings.uNumSamplesPerFrame = 1024;

Wwise-Unity Plugin Modifications

- Wwise API wrapped in C#
- General Unity performance concerns:
 - Unity API calls are slow
 - Runtime allocations cause CPU spikes on garbage collection

C# API Optimization

Unity API calls are **slow**.

We removed AkGameObj check from all Wwise API calls (except in editor):

```
#if UNITY_EDITOR
if (in_gameObjectID.activeInHierarchy) {
    if (in_gameObjectID.GetComponent<AkGameObj>() == null) {
        in_gameObjectID.AddComponent<AkGameObj>();
        Debug.LogError("Missing AkGameObj", in_gameObjectID); // no AkGameObj = error
    }
}
#endif
```

Avoiding Callback String Allocation

User Cues and Markers are sent from Wwise as hashes instead of constantly allocating C# strings:

AkCallbackSerializer.cpp (Wwise-Unity plugin code): const char *s = pCueInfo->pszUserCueName; akCallbackInfo.cueHash = AK::SoundEngine::GetIDFromString(s);

And recognized based on hash in Unity:

```
Custom callback handler (Unity C# code):
    string cueNameToWaitFor = "my cue";
    // if(info.pszUserCueName == cueNameToWaitFor) { ... }
    uint hash = AkSoundEngine.GetIDFromString(cueNameToWaitFor);
    if(hash == akCallbackInfo.cueHash) { ... }
```

Performance Summary

- Virtual Voices
- Glitches caused by large command queues:
 - Audio buffer size 1024
- Wwise-Unity plugin optimized:
 - Removed slow Unity API calls
 - User cues and markers are hashed to avoid allocations

Questions?

playdead.com

Slides are here:

schmid.dk/talks/2016-06-16-wwise

Martin Stig Andersen

martinstigandersen.dk

E-mail: martin@martinstigandersen.dk

Jakob Schmid

Twitter: @jakobschmid

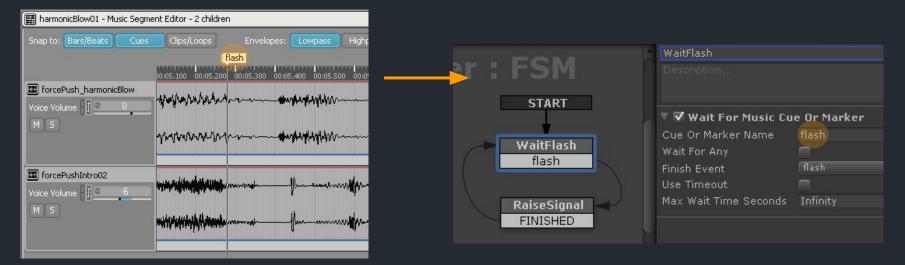
E-mail: jakob@schmid.dk

game140.com

BONUS SLIDES

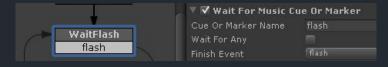
Audio-driven Gameplay: User Cues

- Named User Cues can be placed in music segments
- Received in Unity as callback when AkCallbackManager.PostCallbacks is called (normally, the next frame after cue occurred).



Receiving User Cues

• Receiving User Cues:



Getting Music Time

• The game can also get music time information directly from Wwise:

AkCallbackType flags = AkCallbackType.AK_EnableGetMusicPlayPosition; uint id = AkSoundEngine.PostEvent(eventID, gameObject, (uint)flags); AkSegmentInfo info = new AkSegmentInfo(); AkSoundEngine.GetPlayingSegmentInfo(id, info, doExtrapolation); float musicTime = info.iCurrentPosition * 0.001f;

Snap to: Bars/Beats Cues	Clips/Loops	Envelopes:	Lowpass Highp			WaitPreBlow	
forcePush_harmonicBlow	00:05.900	00:05.950	20:06.000	er : FS	5M		
/oice Volume 🔠 👛 🛛 🔤	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	mm	~~~~~		START	▼ ▼ Wait For Music Tin	1e
MS	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	mm	·		tPreBlow NISHED	Time (Finish Event	5.95 FINISHED
forcePushIntro02 foice Volume		-mmm			seSignal	Use Timeout Max Wait Time Seconds Use Loop Length Requirer	
MS	~~~~~	man have			IISHED	Minimum Loop Length Maximum Loop Length	0 0

Voice Action and Emotion Override

- Action is normally determined automatically from animation
- Action and emotion can be overriden in Voice Configuration
- Enables defining voice reactions in custom situations
- Morphing allows automatically changing emotion after a specified time

▼Set Action Override (b	oy_voc_action)	
Action Override	Land	

▼Set Emotion (boy_voc_	emotion)	
Emotion (boy)	Frantic	
🗸 Morph: Enable		
- Target Emotion (boy)	Relaxed	
- Time 🔜 💮	5	

Profiling Tips: Recording

- Record playthrough and use Connect To File
- Record large profiler sessions (~ 2 hours) by setting Capture Log Max Memory Usage to 3999 MB

R	emote Connect	ions			?)
	Available History	a			
	Name	Туре	Status	IP Address / Device ID	
1	Refresh		Connect Connect Te	o IP Connect To File Discon	
				Start capture on co	nnect
2					
	Status: Not connect	ed			
				Clos	e

Profiling Tips: Comparing

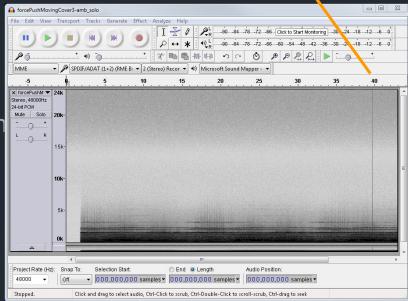
Wwise is single-instance. Compare two profile sessions by running another instance in a virtual machine (e.g. VirtualBox)

Game2* - Wwise v2014.1.6 (6	i4-bit)			L.		Windows8.1 [K	ørende] - Oracle VM VirtualBox				
Project Edit Views Layouts						Maskine Vis	Enheder Hjælp				
Xbox One''' 🛛 👻 English (US) (Reference) 🛛 👻 Start Capture 🗌	01:06:04.885 Follow Captu	re Time 00:43:31.669	M S Remo	ite Reconnect	E.		Game2 - Wwi	se v2014.1.6 (64-bit)		– 🗆 🗙
Advanced Profiler					88 ? X	and the second s	iews Layouts Audio Window				
Voices Graph Voices Busses	Sends Game-defined Sends	Memory Streams Streaming	Devices Plug-ins				English (US) (Reference) 🛛 💙 🕄		03.754 Follow Capture Ti	me 00:21:09.514	M S Remote
Device Name Stream Name	Buffer Size	Buffering Status File Size Fil	le Position Total Bandwidth	Bandwidth (Low-Level)	Priority Ref. Me	Advanced Profiler					88 ? >
Win32 Blocking sisterVoc6dBma		1.4 MB	223.3 K8/:			Water Graph La	oices Busses Sends Game-def	and Sonds Marrow	Streams Streaming Devi	sec Diveloc	
Win32 Blocking windVocBlade10						and the second s					
Win32 Blocking windVocBlade12					50 96.0 KB	Device Name	Stream Name sisterVoc6dBmarkHP1210_02_rev	Buffer Size Buffer 71.3 KB	ing Status File Size File 1.4 MB	Position Total Bandwidth Bandw 152.4 KB/s	dth (Low-Level) Priori 152.4 KB/s 50
							windVoc06ablurDel				
						Win32 Blocking					
						Win32 Blocking					
						Win32 Blocking					
						Win32 Blocking					
						Win32 Blocking Win32 Blocking					
						Win32 Blocking					
						Win32 Blocking					
						Win32 Blocking					
						CI.					Ø
01					6	Performance Monit					88 ?)
104							100			Name	
Performance Monitor					88 ? X					Audio Thread CPU	39.47 %
100			Name		Value					Command Queue Size	
80					37.28 %	Audio Thread CPL				Command Queue Used	
60					1.3 KB 0.13 %		4) Mr. Maringal	-	Rangel and Links of Strange	Loaded Banks (Memory) Loudness Momentary/Sync G	
Audio Thread CPU		A 1			487.9 MB		20 / 10 1			Loudness Short-term/Sync G	
4) A	and a spit simple approximation of the second	water want of the second states and	Loudnes 1 Prophy 14 Loudnes							Number of Active Events	
20							100			Number of Active Sends	
					10					Number of Fade Transitions	
100					13					Number of Prepared Events	
					0					Number of Registered Object	
~					144	Total Plug-in CPU	4)			Number of State Transitions Number of Streams	
50 Total Plus-in CPU					0					Number of Streams (Active)	
Total Plug-in CPU 40					3					Number of Voices (Physical)	
					3		0			Number of Voices (Total)	
° r		and the second			24		3 MB	1 14	THE READ AND	Number of Voices (Virtual)	
0					28 4			and 1		Output Peak	
3 MB			Output F						wa wyw yr I	Prepared Events (Memory) Total Media (Memory)	
					- O Bytes	Total Streaming Bandwidth (Low-Lev	2 MB			Total Plug-in CPU	
2 M8					487.9 MB		I MB	and district		Total Reserved Memory	
Total Streaming 2 MB Bandwidth (Low-Level)					12.95 %		512 KB			Total Streaming Bandwidth	
1 M8					550.4 MB		0 Bytes			Total Streaming Bandwidth (L	
512 KB	. Luistiut	ب إسامي الله الم المرامي (المبادر) الله]]			594.0 KB/s					Total Used Memory	
0.0-m /	alter de san de sal	A La Auct of the			594.0 KB/s 513.2 MB	G			0ei		
0 Dites 1		43/00.000 00/43/30.000	09:44:00.000		515.2 Mb						
			0034460.000							😝 💿 🖉 🛃 🖬	

Debugging Tips: Recording Audio

- Tiny audio glitches
- 256 samples of zeros when command queue was large
- No errors in Wwise profiler, only detectable using audio recordings
- Record console output using S/PDIF
- Barely audible glitches are easy to spot in spectrogram





CPU Performance

• Wwise runs on CPU core 5, Unity worker threads run on core 2-4:

platformSettings.threadLEngine.dwAffinityMask = (1 << 5)</pre>

Avoiding Callback Allocations

Single instance of callback data structure is reused for every callback.

I/O Performance

- Look for streams that are used a lot throughout the game, and convert them to non-streams. Searching and opening in List View is useful for this.
- We used PCM for music, and all other sounds vorbis (quality 10)

List View					
List View					8 ? X
Search windvoc					
Name	Voice Volume		e Low-pass Filter Voice High-pass	s Filter Is Streaming B	inabled Zero latency 🔯
🗢 🛄 windVoc	-24				
🖶 🕵 windVoc06ablurDel					
E 🕅 windVoc06bblurDel					
🖻 🔛 windVoc	-21				
🖶 💦 windVoc06ablurDel				🗹	∃ 💌
🗈 🔀 windVoc06bblurDel			e 0 🗍 🚺 e 0		∄ 💌
🖻 🏭 windVoc	-27.3	00000			
Similar Strategy S	0				
🖶 🕂 windVoc06bblurDel					
🖻 🏭 windVoc	● 0 ● -21.3		0 0		
Similar Strategy S	0	Ale 0 Ale	e 0 🗍 📴 0		
🗈 💦 windVoc06bblurDel	0				f 🗆 📗
🖻 🏭 windVoc	-24.3		e 0 🌔 e 0		
🖶 💦 windVoc06ablurDel	0	Alle O Alle	e 0 🗍 🤅 0		H 🗆
🗉 💦 windVoc06bblurDel	(e)				i 🗆
🖻 🏭 windVoc	-32) <mark>- 0</mark>	e 0 🗍 e 0		1. V
🗉 💦 windVoc06ablurDel					A 💌
🗈 💦 windVocO6bblurDel	0		e 0 👸 e 0		
🗢 🏢 windVoc	18.3	j <mark>e o</mark> je	e 0 j e 0		0

Teaser and Bios

Teaser

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, and the seemless handling of the death - respawn cycle.

The talk will cover both the Wwise setup and game engine tools used for audio features, and show how Wwise can be used as a compositional tool. Finally, performance results and considerations will be discussed in relation to the topics covered.

Bios

Martin Stig Andersen (b. 1973) has a background as a composer in the fields of acousmatic music, sound installations, electroacoustic performance, and video art, earning several international distinctions and awards. In 2009 he joined Playdead where he created the audio for the video game LIMBO which won Outstanding Achievement in Sound Design at the Interactive Achievement Awards, the IndieCade Sound Award 2010, and was nominated for best audio at the BAFTA Video Games Awards 2011. In the years following the release of LIMBO, Martin Stig Andersen has been working on Playdead's next title, INSIDE, which is to be released 2016.

Martin Stig Andersen graduated as a composer from The Royal Academy of Music in Aarhus, Denmark in 2003, after which he went on to study electroacoustic composition under Professor Denis Smalley at City University, London.

Jakob Schmid (b. 1976) graduated as a computer scientist specialized in game development and a minor degree in music science from the University of Aalborg, Denmark in 2007. Working in the danish video game industry since 2008, Schmid has specialized in developing novel dynamic audio systems for video games. He created the music and sound for Jeppe Carlsen's rhythm platformer '140', which went on to win the 'Excellence in Audio' award at IGF 2013, with honorable mention in Technical Excellence, as well as the 'Sound of the Year' award at SpilPrisen 2014.

Schmid joined Playdead in 2011 as audio programmer, mainly working on the studio's next title, 'INSIDE'.