

Adaptive Audio in INSIDE and 140

SpilBar 41
Jakob Schmid

slides available at <https://www.schmid.dk/talks/>

Who Am I?

Jakob Schmid

INSIDE, Playdead, audio programmer

140, Carlsen Games, music and sound design

Co-founded new game studio in 2017



IGF award 2013

Excellence in Audio

Honorable mention: Technical Excellence

Spilprisen 2014

Sound of the Year

Nordic Game Award 2014

Artistic Achievement



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

Adaptive Audio in INSIDE



INSIDE Audio Team



Martin Stig Andersen

audio director, sound designer, composer

Andreas Frostholtm

sound designer

Søs Gunver Ryberg

composer, sound designer

Jakob Schmid

audio programmer

Audio Engine: Audiokinetic Wwise

The screenshot displays the Audiokinetic Wwise software interface, titled "test - Wwise v2014.1.6 (64-bit)". The interface is divided into several panels:

- Project Explorer:** Shows a hierarchy of audio assets, including "Master-Mixer Hierarchy", "Actor-Mixer Hierarchy", "Default Work Unit", "SoundEffects", "MySound", "sound1", "sound2", and "Interactive Music Hierarchy".
- MySound - Random/Sequence Container Property Editor:** The main panel, showing a graph of "Voice Volume" (Y-axis, ranging from -200.0 to 200.0) versus "MyParameter" (X-axis, ranging from 0 to 100). A red curve shows the volume increasing from -200.0 at parameter 0 to approximately -10.0 at parameter 100. A dashed red horizontal line is at -10.0. A data point is labeled "MyParameter (50.000)".
- Coordinates:** A table mapping parameters to axes:

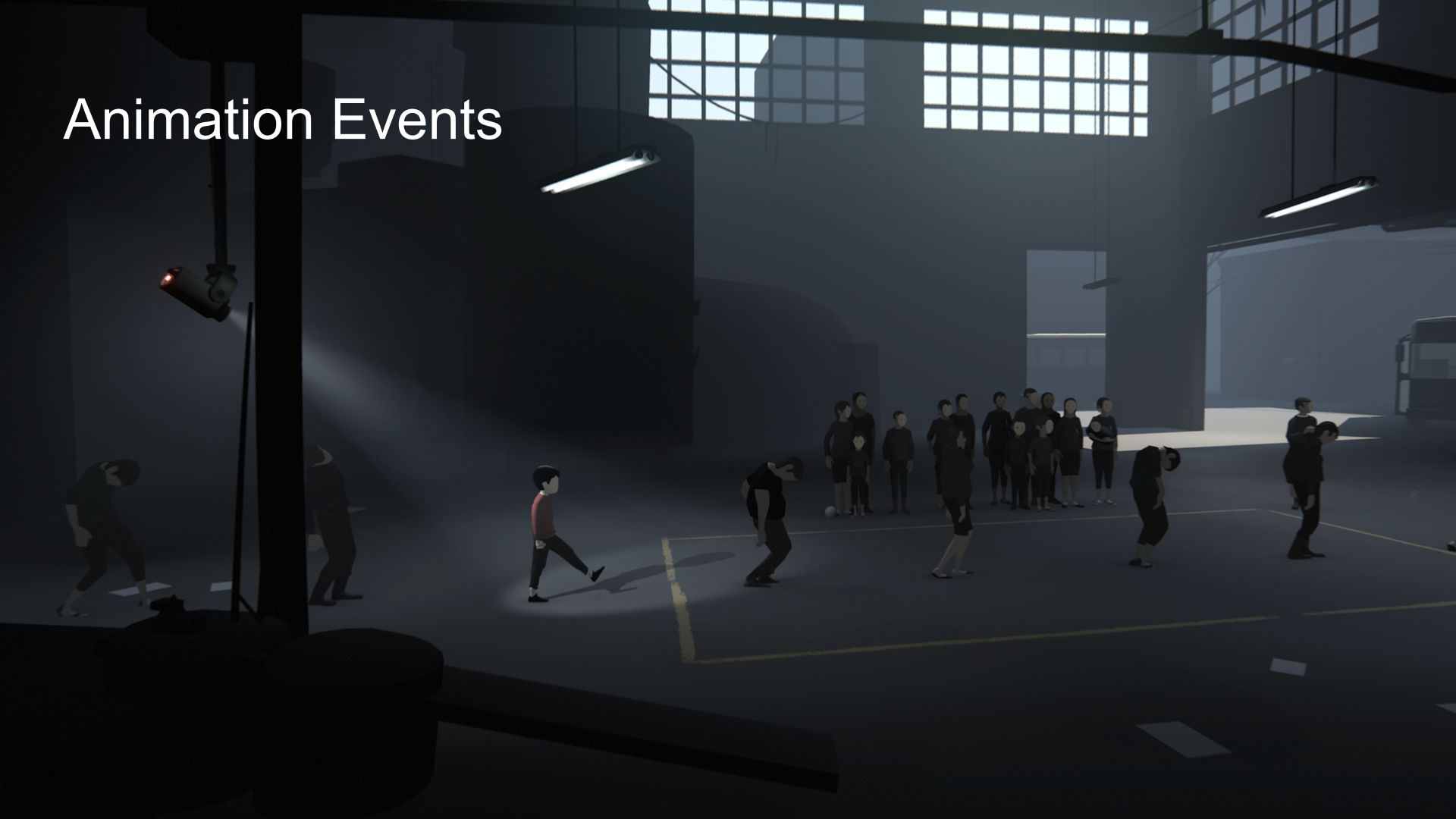
Coordinates	Y Axis	X Axis	Mode	Notes
X:	>> Voice Pitch	>> MyParameter		
Y:	>> Voice Volume	>> MyParameter		
	>>	>>		
- MySound - Contents Editor - 2 children:** A table listing sound assets:

Name	Weight	Voice Volume	Voice Pitch	Voice LFP	Notes	
sound1	<input checked="" type="checkbox"/> PF	50	0	0	0	/Notes
sound2	<input checked="" type="checkbox"/> PF	50	0	0	0	/Notes
- MySound - Transport Control:** Includes buttons for "Original", "PF Only", "Reset All", "States", "RTPCs", "Switches", and "Triggers".
- Event Viewer:** Shows "Filtered", "Current Selection", and "Orphans (0)".
- Master Audio Bus:** Shows a "Peak" meter on the right side of the interface.

INSIDE Video

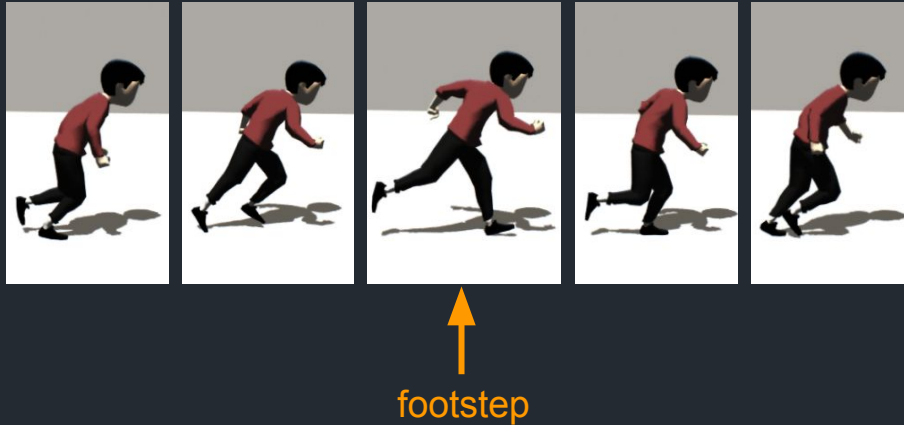


Animation Events



Animation Events

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



Context-Sensitive Events



idle



jog

previous action: idle

current action: 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					

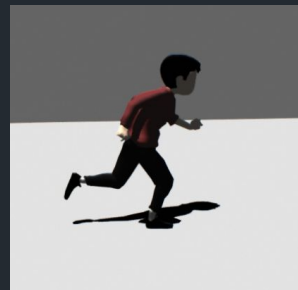
play sound 'takeoff_mf'

Context-Sensitive Events

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



sneak



jog

previous action: sneak

current action: 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					

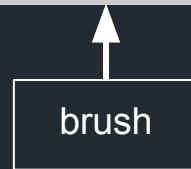
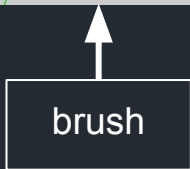
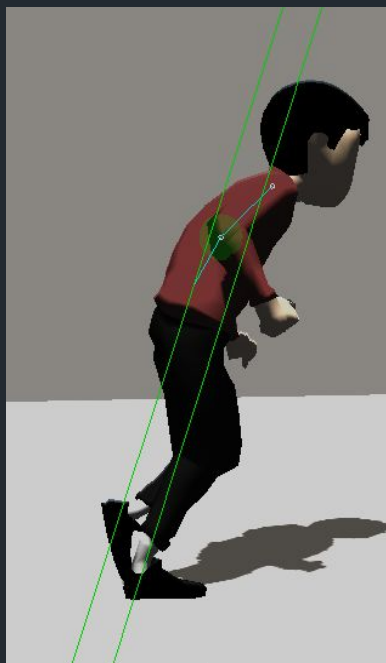
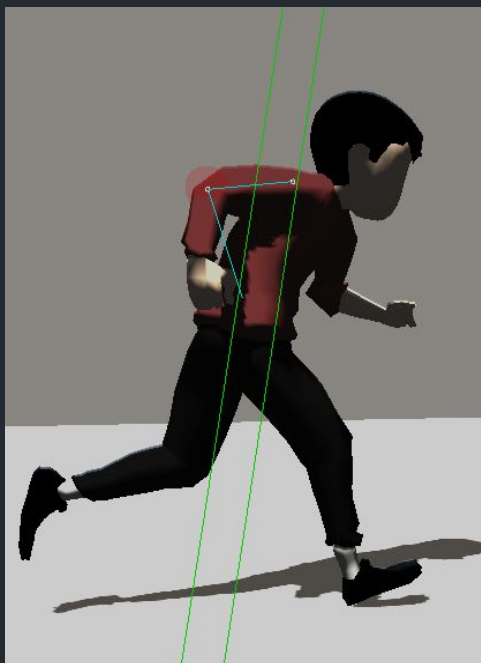
play sound 'walk'

Wet Animation Events

- Shoes can get wet
- Adds wet sound on top of footstep
- Wetness value is used to set volume
 - Is set high when in water or on a wet surface
 - Dries out over time



Elbow Brush Sounds

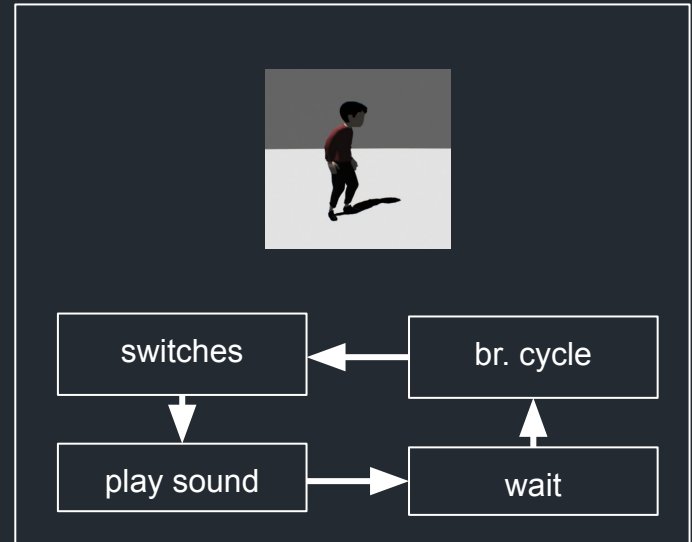
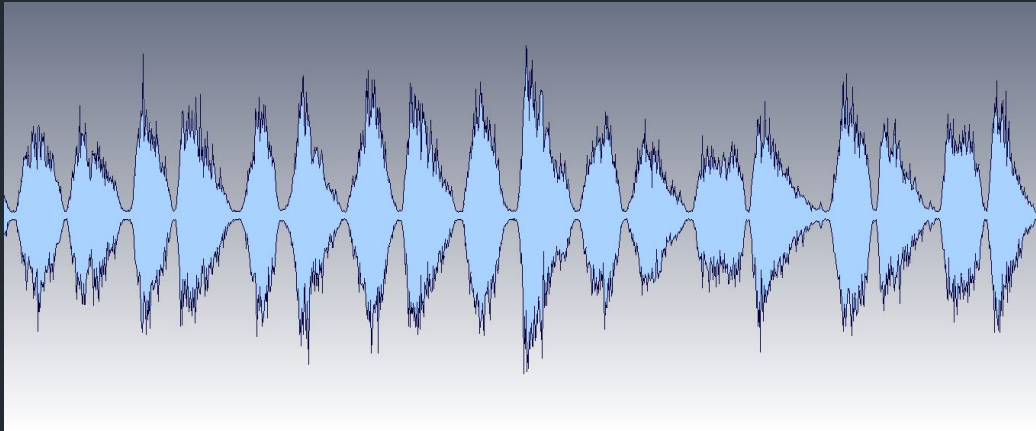


Voice Sequencer



Continuous Voice Sequencing

- Recorded breath sounds have varying durations
- 'Stitching' recorded sounds together results in natural, uneven breathing pattern

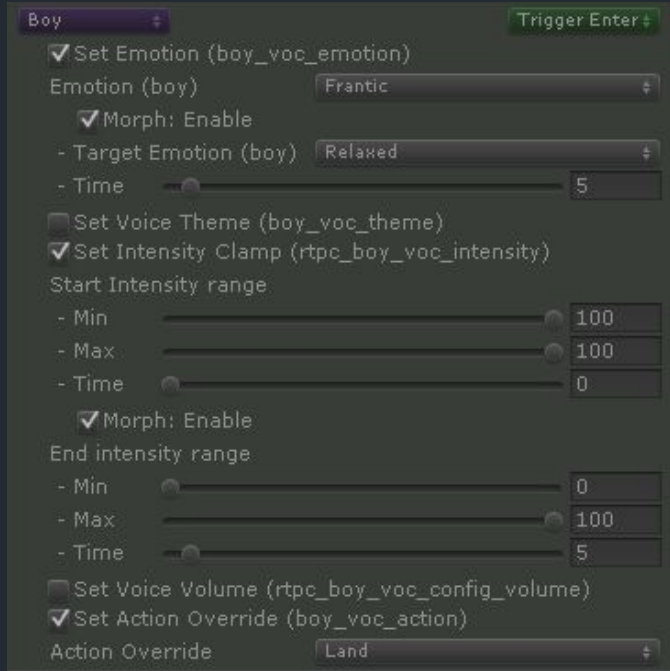


Voice Direction

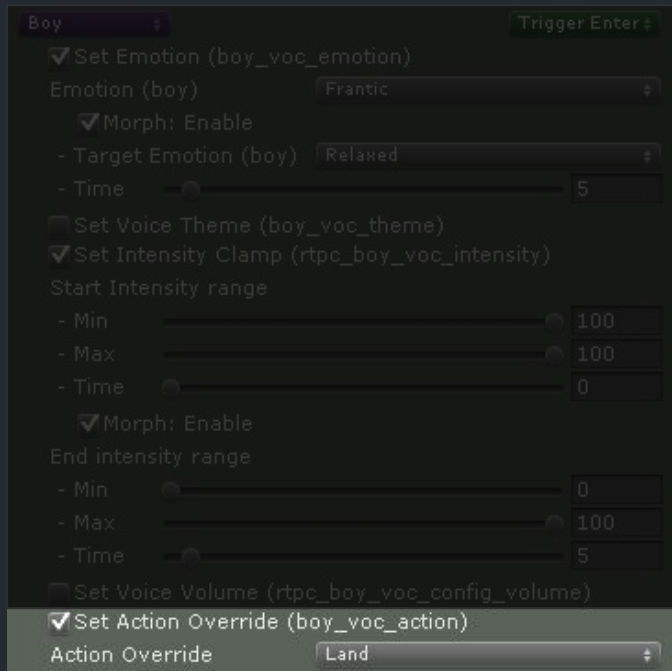
- Voice direction is done using voice configuration system
- The director (Martin) instructs the actor (the voice sequencer) how to emote:
 - based on location or
 - based on reacting to events



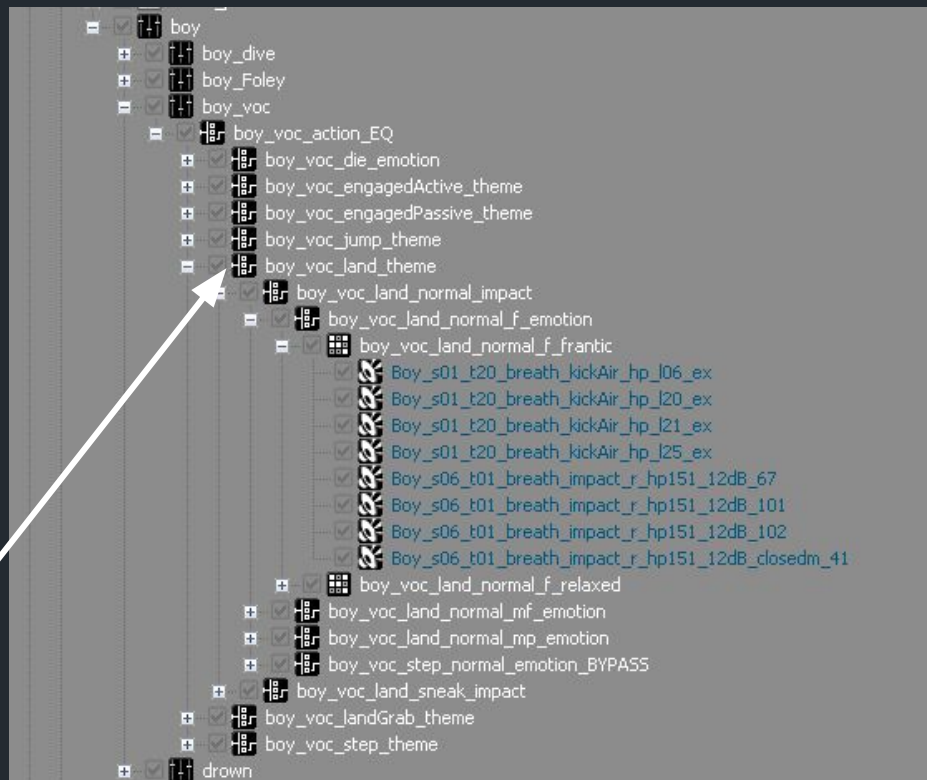
Voice Configuration



Action

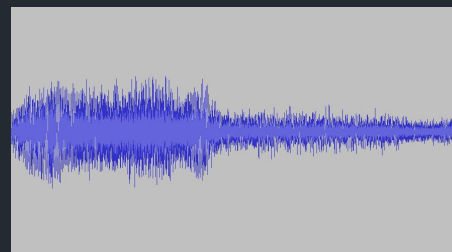
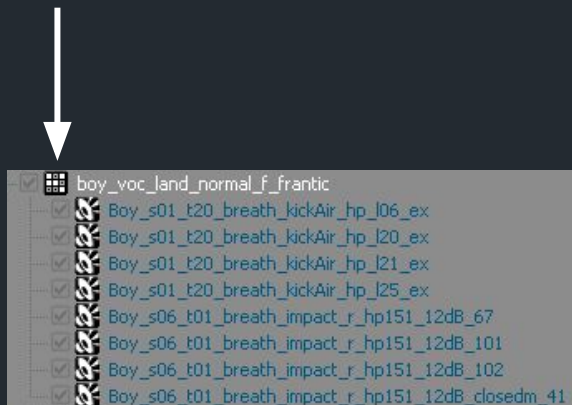


Hierarchy of sounds in Wwise project



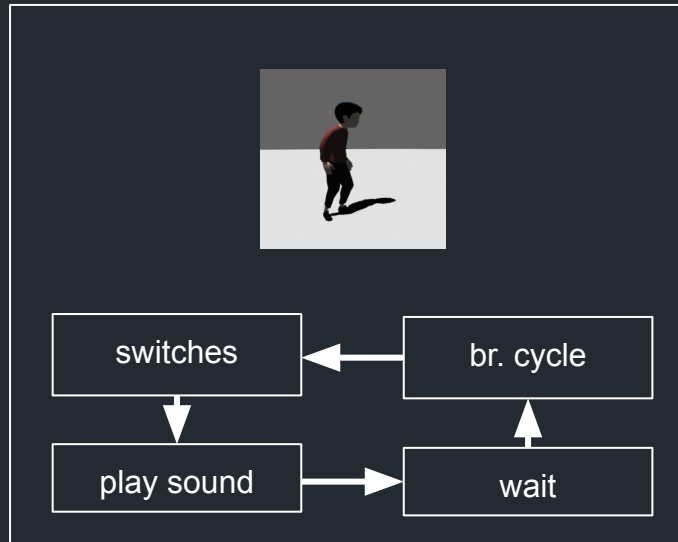
Random Selection

Randomly select and play one sound in a group

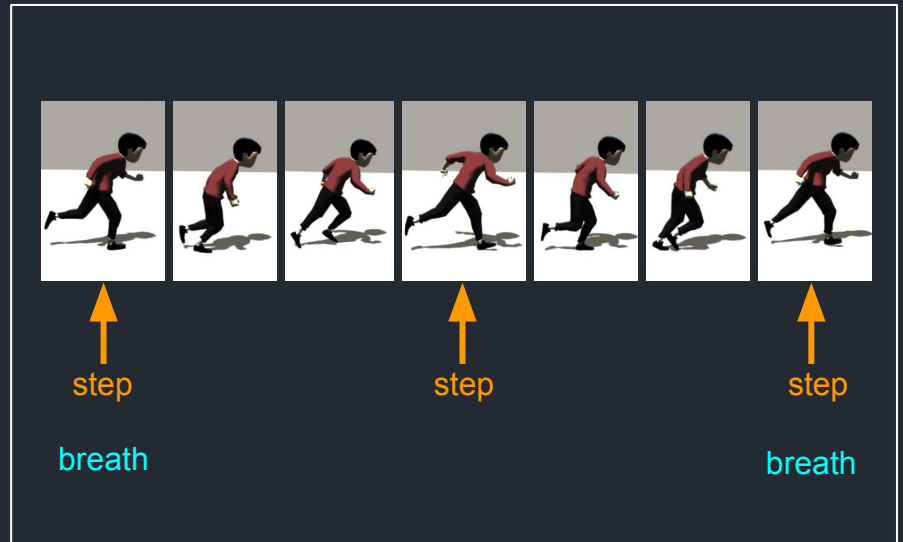


Voice Sequencer Modes

Continuous Mode



Run: Rhythmic Breathing



Adaptive Audio in INSIDE

- Context-sensitive animation events for footsteps
- Voice Direction selects sounds based on action, intensity, and emotion
- Voice sequencer continuous mode sounds natural
- Rhythmic breathing when running



Adaptive Music in 140

140

140

Jeppe Carlsen (design, programming)
Niels Fyrst, Andreas Peitersen (visual design)
Jakob Schmid (audio)

Developed as hobby project over 3 years

PSN

STEAM

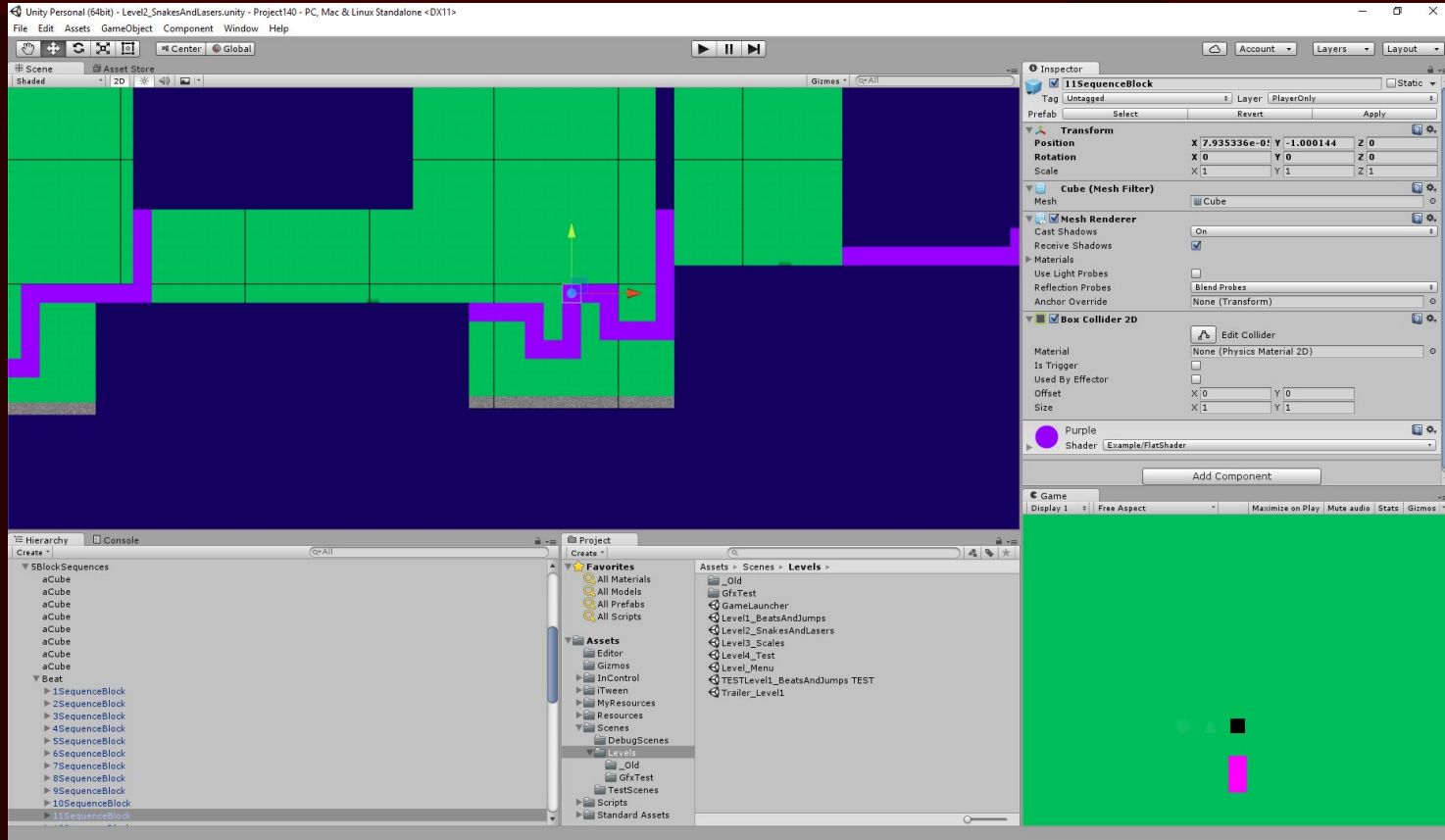
WiiU

Humble Bundle

XBOX ONE

gag
com

Developed in Unity 3



Ableton Live

The screenshot displays the Ableton Live 9 Suite interface, showing a multi-track session view. The top menu bar includes File, Edit, Create, View, Options, and Help. The session view is organized into tracks, each with a name, a solo button, and a volume knob. The tracks are:

- 1 Drum
- 2 Drum
- clap ve
- kywhl snrl
- laserba
- cuta drums
- real drums
- sub
- 9 level0
- theme
- l2-bass
- l2-swell
- l2-JK-noise
- l2-JK-jump
- l2-JK-build
- DONTUS
- bass
- acid
- acid
- Master

The tracks are arranged in a grid, with each track having a MIDI or Audio input section and a volume knob. The MIDI input section includes buttons for M. From, All Ins, All Chn, and Monitor. The Audio input section includes buttons for A. To, Master, and Monitor. The volume knob is located at the bottom of each track.

The bottom of the interface shows the Drum Rack, Instrument Rack, Operator, and Saturator sections. The Drum Rack shows a grid of drums with names like C2, C#2, D2, D#2, G#1, and others. The Instrument Rack shows a selected instrument, in this case, a keyboard instrument with a name like "nd and bass". The Operator section shows various parameters like Coarse, Fine, Fixed, and Level. The Saturator section shows a waveform and various parameters like Amount, Rate, Phase, and Shape.

Audacity

The screenshot displays the Audacity application window. The top menu bar includes File, Edit, View, Transport, Tracks, Generate, Effect, Analyze, and Help. Below the menu is a toolbar with various icons for playback, editing, and analysis. The main workspace shows a spectrogram of an audio file, with the frequency spectrum on the vertical axis (0.0k to 22.1k Hz) and time on the horizontal axis (29.0 to 48.0 seconds). The spectrogram shows a complex waveform with many vertical lines, indicating a high-frequency signal. The bottom status bar shows the Project Rate (48000 Hz), Snap To (Off), Selection Start (002,245,446 samples), End (000,000,000 samples), and Audio Position (000,000,000 samples). The status bar also indicates the audio is Stopped and provides instructions for selecting and scrubbing audio.

140.4

File Edit View Transport Tracks Generate Effect Analyze Help

Click to Start Monitoring MME Stereo Mix (Realtek Hi 2 (Stereo) Ret Speakers (Realtek High

29.0 30.0 31.0 32.0 33.0 34.0 35.0 36.0 37.0 38.0 39.0 40.0 41.0 42.0 43.0 44.0 45.0 46.0 47.0 48.0

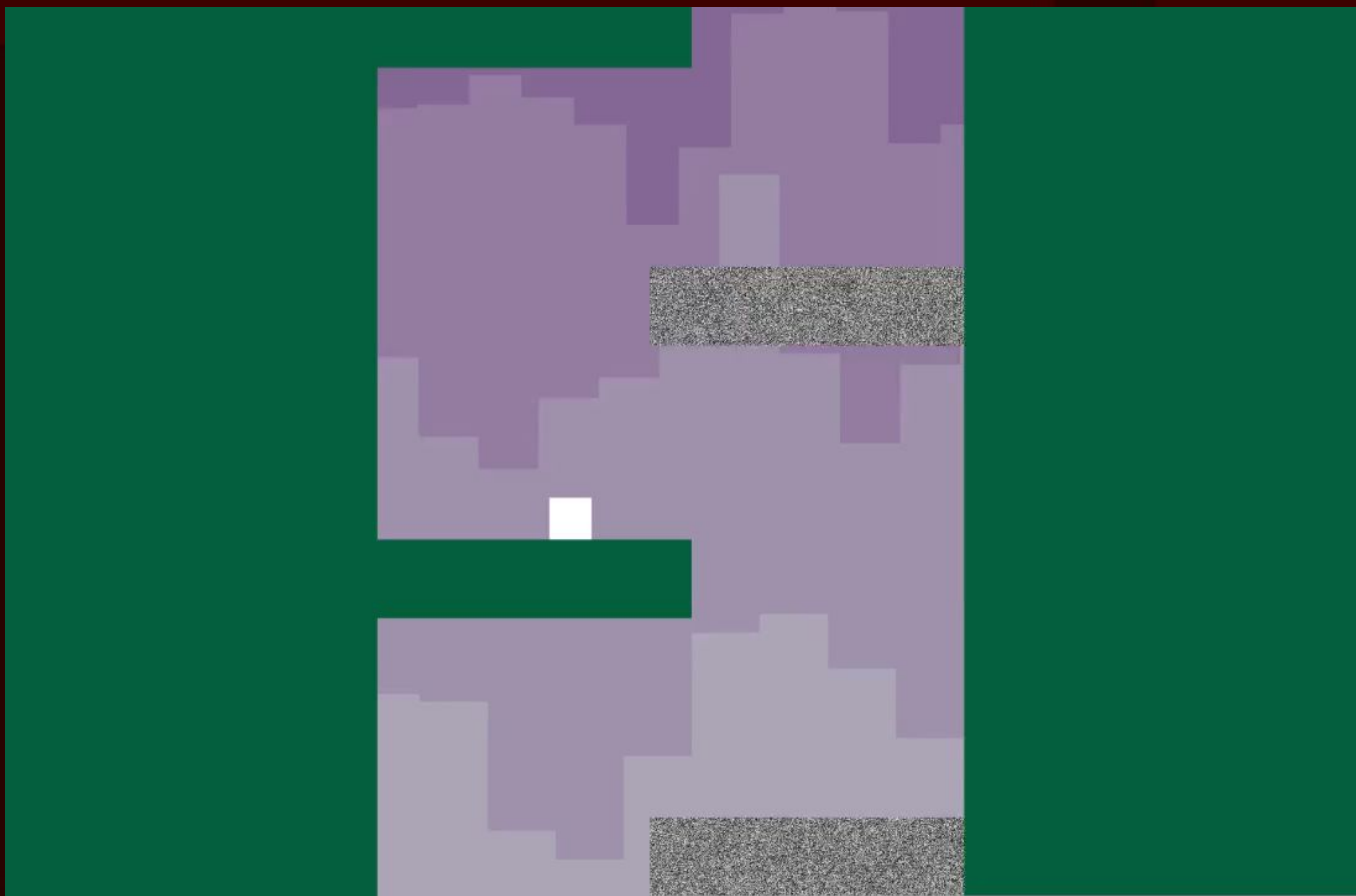
X 140.4 22.1k
Stereo, 44100Hz
32-bit float
Mute Solo

20.0k
19.5k
19.0k
18.5k
18.0k
17.5k
17.0k
16.5k
16.0k
15.5k
15.0k
14.5k
14.0k
13.5k
13.0k
12.5k
12.0k
11.5k
11.0k
10.5k
10.0k
9.5k
9.0k
8.5k
8.0k
7.5k
7.0k
6.5k
6.0k
5.5k
5.0k
4.5k
4.0k
3.5k
3.0k
2.5k
2.0k
1.5k
1.0k
0.0k

Project Rate (Hz): 48000 Snap To: Off Selection Start: 002,245,446 samples End: 000,000,000 samples Audio Position: 000,000,000 samples

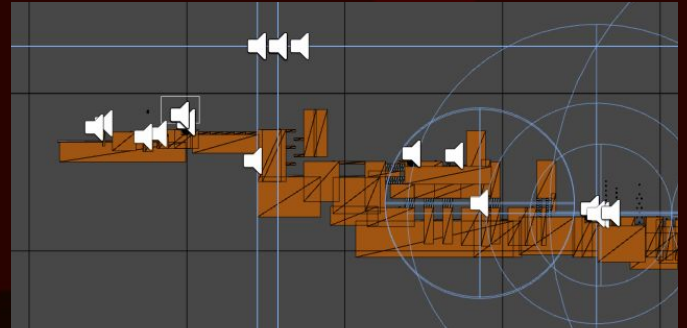
Stopped. Click and drag to select audio, Ctrl-Click to scrub, Ctrl-Double-Click to scroll-scrub, Ctrl-drag to seek

Level 4

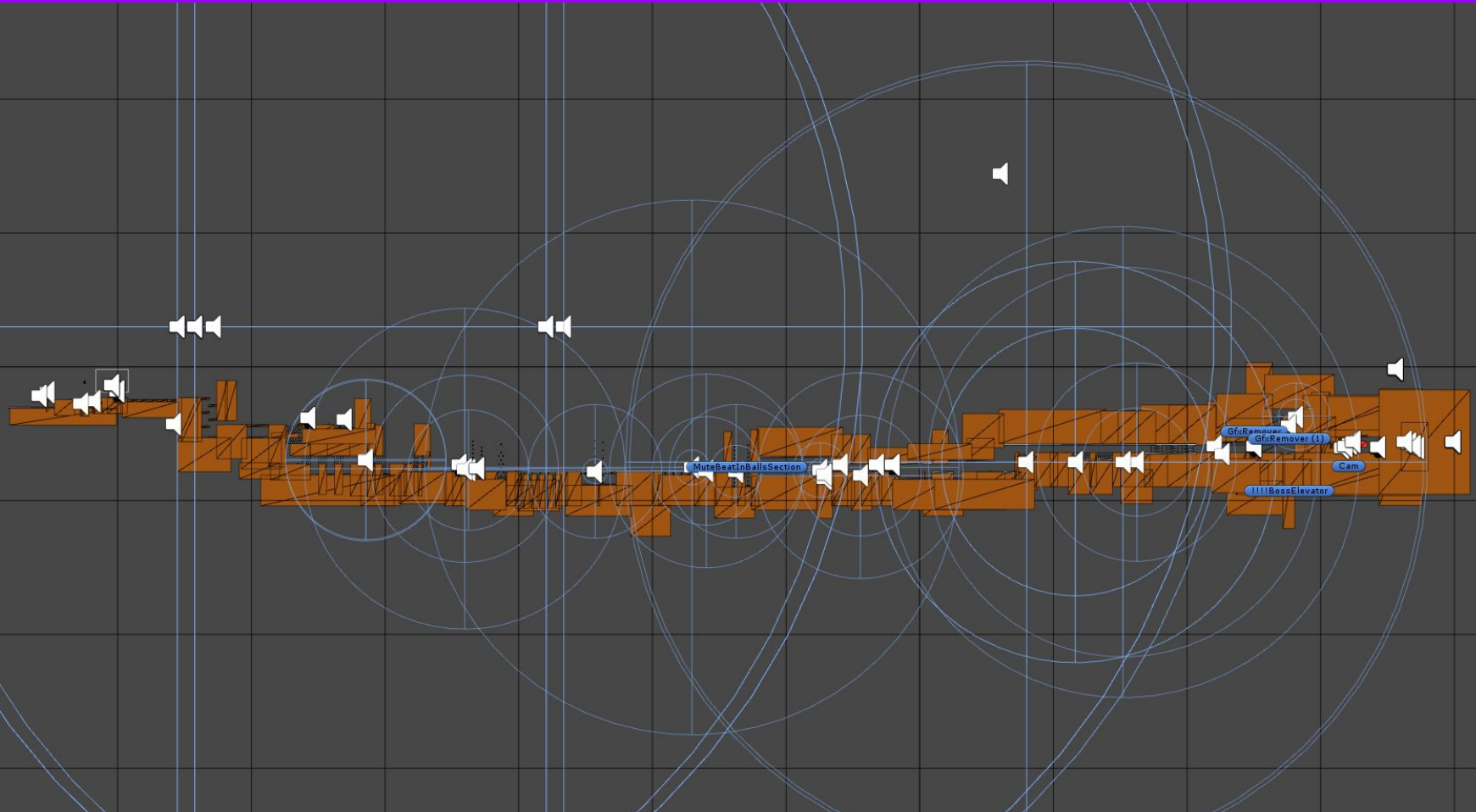


Position-Adaptive Music

- Music loop audio sources are placed directly in level geometry
- Adaptive mix occurs as player moves around
- Areas gain unique atmosphere based on music

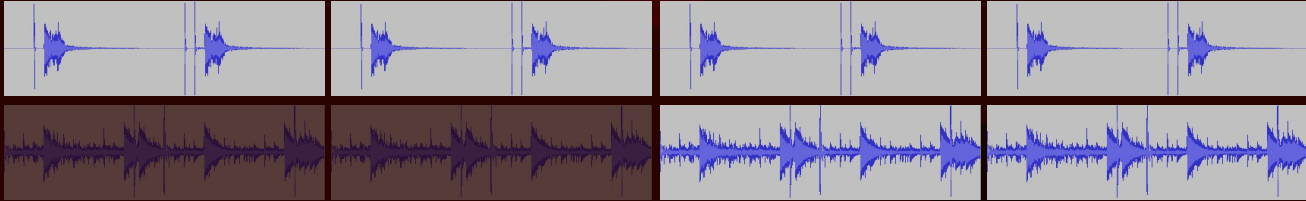


Music Loop Positions - Level 4



Synchronized Loops

- All loops must be exactly same length, or integer multiples
- All loops must be started in the same frame, possibly muted
- New loops cannot be started
- Never change pitch



AudioSync Component

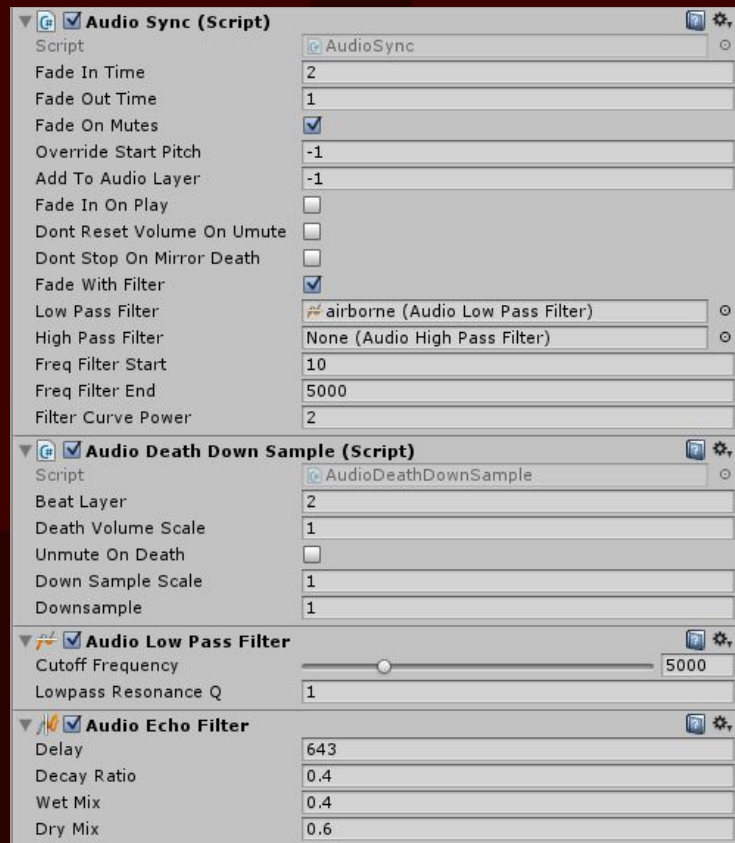
AudioSync component handles all music loops

- Handles fading in or out
- Controls filters



Death filter

Low-pass filter and echo masks transitions and loops



The screenshot displays the Unity Inspector for three audio components:

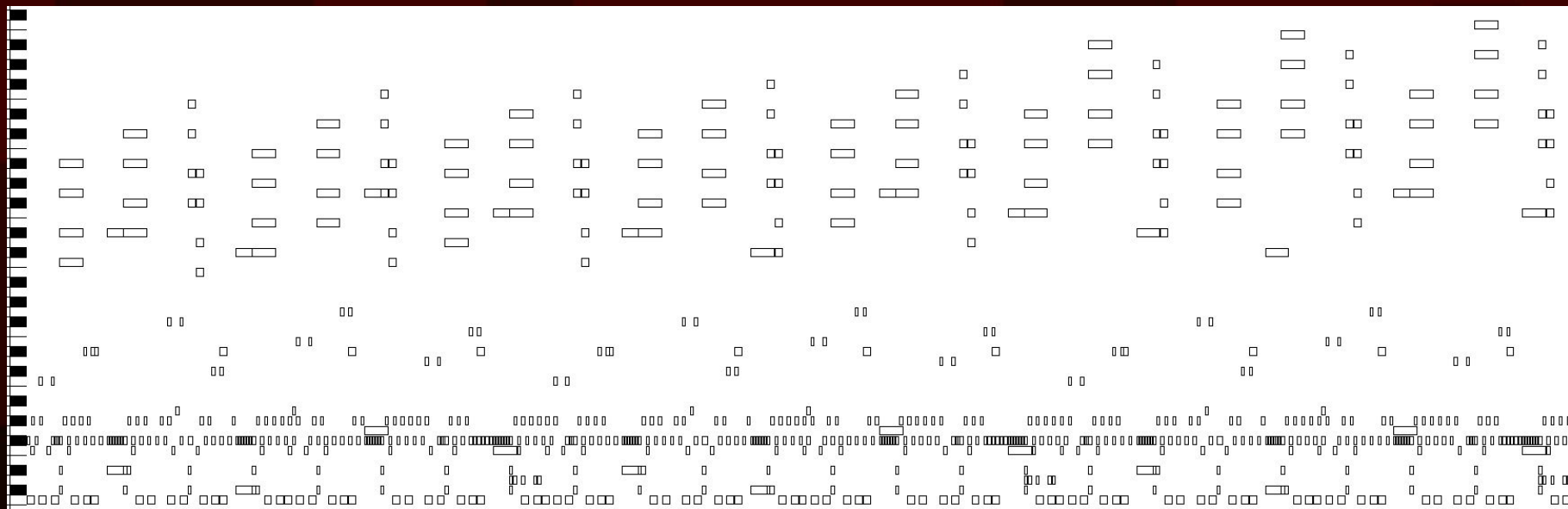
- Audio Sync (Script)**
 - Script: AudioSync
 - Fade In Time: 2
 - Fade Out Time: 1
 - Fade On Mutes:
 - Override Start Pitch: -1
 - Add To Audio Layer: -1
 - Fade In On Play:
 - Dont Reset Volume On Umute:
 - Dont Stop On Mirror Death:
 - Fade With Filter:
 - Low Pass Filter: airborne (Audio Low Pass Filter)
 - High Pass Filter: None (Audio High Pass Filter)
 - Freq Filter Start: 10
 - Freq Filter End: 5000
 - Filter Curve Power: 2
- Audio Death Down Sample (Script)**
 - Script: AudioDeathDownSample
 - Beat Layer: 2
 - Death Volume Scale: 1
 - Unmute On Death:
 - Down Sample Scale: 1
 - Downsample: 1
- Audio Low Pass Filter**
 - Cutoff Frequency: 5000
 - Lowpass Resonance Q: 1
- Audio Echo Filter**
 - Delay: 643
 - Decay Ratio: 0.4
 - Wet Mix: 0.4
 - Dry Mix: 0.6

140 Music Production

The image displays a detailed view of a digital audio workstation (DAW) interface, likely Ableton Live, used for music production. The top section features a transport bar with playback controls and a piano roll for MIDI editing. Below this is a multi-track arrangement view with 32 tracks, each containing various audio and MIDI clips. The tracks are organized into several groups: 'MUSIC' (tracks 1-32), 'SOUND EFFECTS' (tracks 33-36), and 'KEY PICKUP' (tracks 37-38). The bottom section shows the mixer and effect racks. The mixer includes a filter section with a frequency response curve, a volume knob, and a solo button. The effect racks contain several audio effects, including a 'Waveshaper' (set to 'wvshaped' with a gain of 0.27 dB), a 'Saturator' (set to 'Drive' with a gain of 6.66 dB), a 'Dynamic Tube' (set to '100%' output), and an 'EQ Eight' (set to '80.0 Hz' frequency). The interface is highly detailed, showing various parameters and controls for each track and effect.

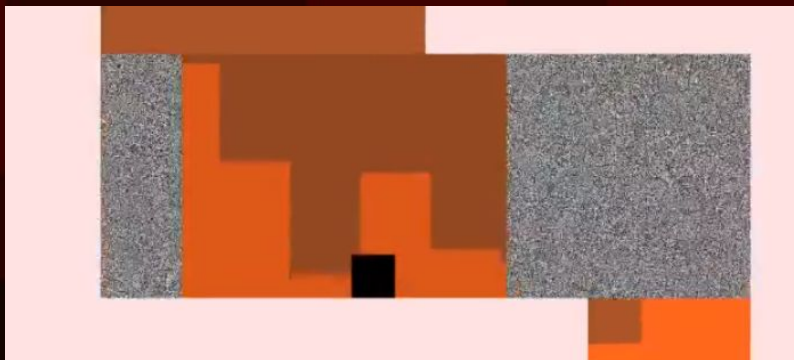
The Puzzle of Music

Making music for a *music game* can be like solving a puzzle

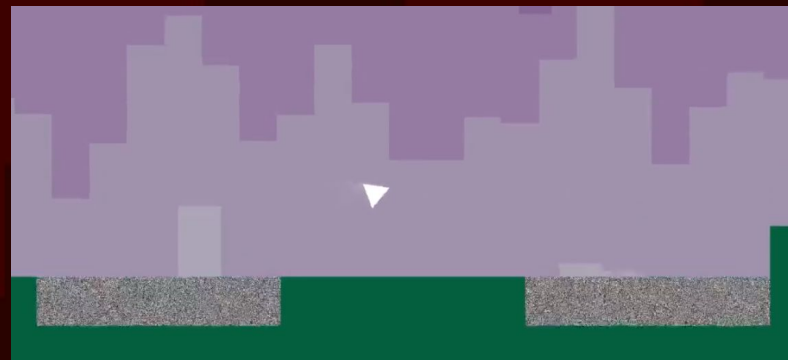


KillBlocks and Toggler

Two example game mechanics

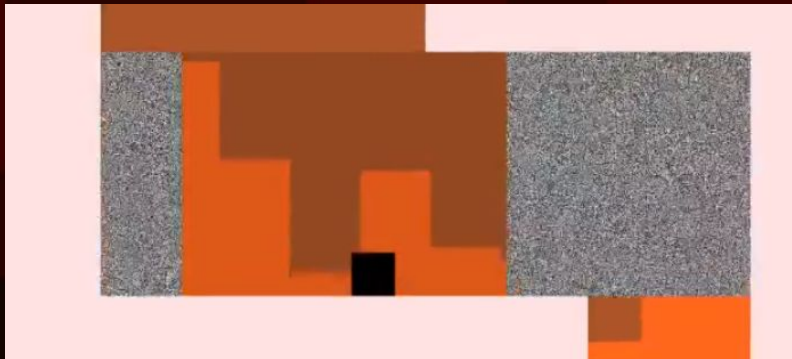


KillBlock



Toggler

KillBlocks

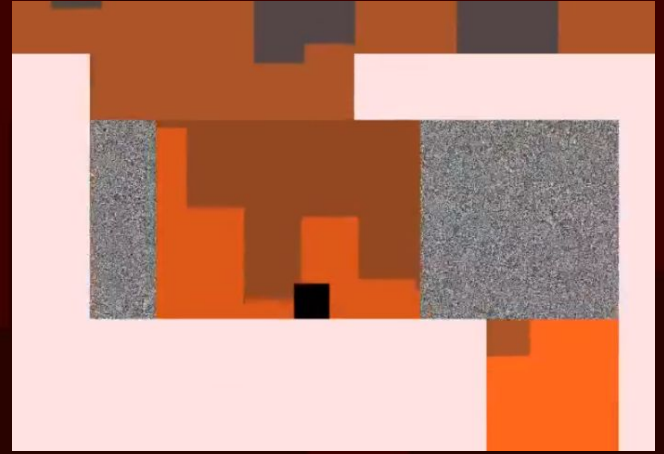


KillBlock



KillBlock Rhythm Pattern

- Communicates to player exactly when certain game areas are either safe or lethal
- Must correspond exactly to game logic timing



time 

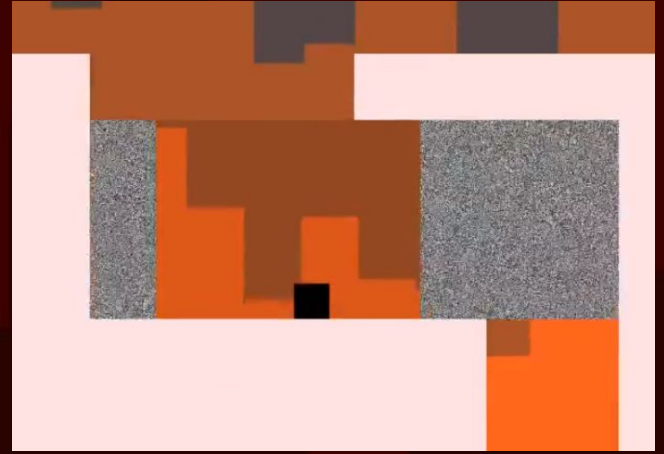
> > > > x

> MOVE blocks right

x TOGGLE all blocks

KillBlock Rhythm Pattern

- Music runs at 140 beats/minute
- A 'bar' is 4 beats ~ 1.7 seconds
- Our KillBlock rhythm is exactly 2 bars



time (bars) \longrightarrow

1 . . . : . . . 2 . . . : . . .



> > > > x

KillBlock Sounds

- > MOVE sound is a 'Landlord stab'
- x TOGGLE sound is a 808 cowbell

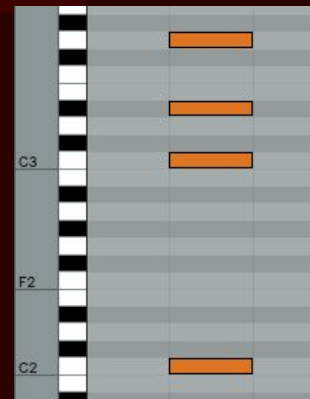


> MOVE Sound

- 'Landlord' stab
- Classic house sample
- Sampled minor chord played on piano-like FM synth
- Origin of sample seems to be 1984 Linndrum demo tape 
- Made famous by Landlord's 'I Like It (blow out dub)' (1989) 



→
sounds
like




Sampled Chords

- Sampled chord is played back at different sample rates
 - Resulting output is the same chord with new base notes
- (foundational for all sampler-based music)



KillBlock Harmony

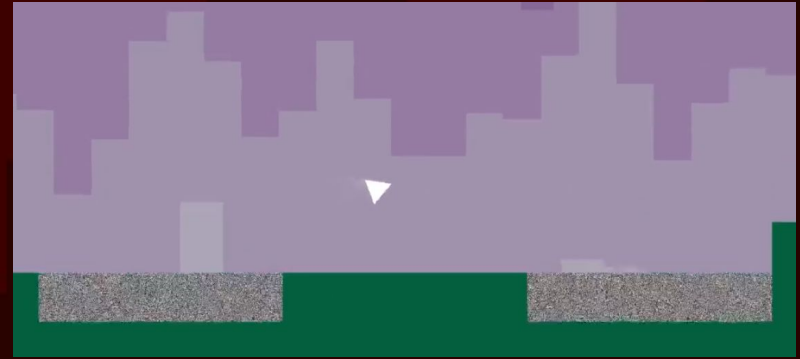
- The result can be heard in the **KillBlock loop** 
- 2-bar rhythmic loop
- 8-bar harmonic loop

bars

1 . : . 2 . : . 3 . : . 4 . : . 5 . : . 6 . : . 7 . : . 8 . : .
Cm D#m F#m C#m Em Gm Dm Fm



KillBlocks and Toggles



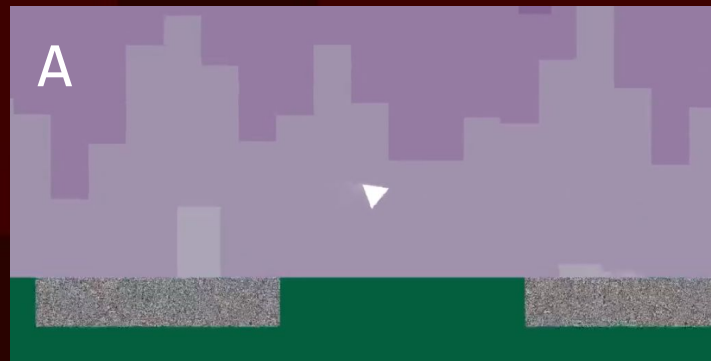
Toggler

Toggler Sound

The toggler loop alternates between two different Operator patches

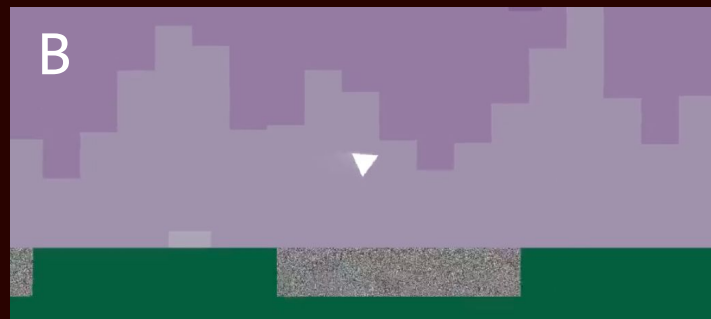
State A Sound

The screenshot shows the State A Operator patch interface. The left panel contains four sliders for Coarse, Fine, Fixed, and Level, with values: Coarse 3, Fine 0, Fixed -12 dB, Level -12 dB; Coarse 2, Fine 0, Fixed -14 dB, Level -14 dB; Coarse 1, Fine 0, Fixed -7.6 dB, Level -7.6 dB; Coarse 1, Fine 0, Fixed -12 dB, Level -12 dB. The center panel shows an Envelope graph with parameters: Attack 0.00 ms, Decay 1.77 s, Release 1.37 s, Time<Vel 0%, Wave Sin. The right panel shows LFO (Sine, Rate 111.88, Amount 32%), Filter (12, 24, Clean, Freq 18.5 kHz, Res 20%), Pitch Env (0.0%), Spread (0%), Transpose (0 st), Time (29%), Tone (70%), and Volume (-2.3 dB).



State B Sound

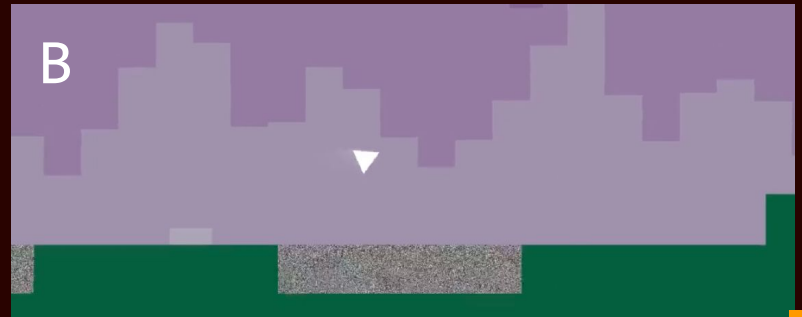
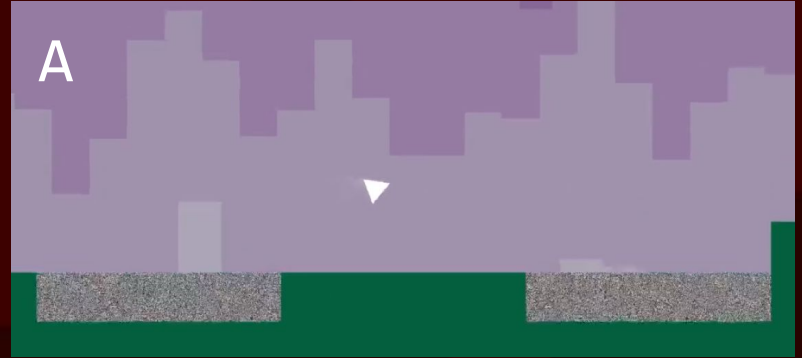
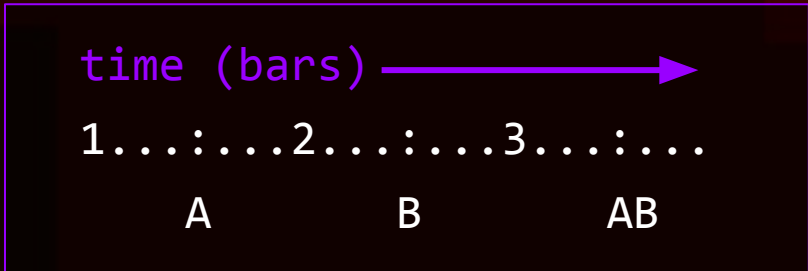
The screenshot shows the State B Operator patch interface. The left panel contains four sliders for Freq, Multi, Fixed, and Level, with values: Freq 186 Hz, Multi 0.01, Fixed -58 dB, Level -58 dB; Coarse 5, Fine 0, Fixed -7.0 dB, Level -7.0 dB; Coarse 1, Fine 0, Fixed -8.1 dB, Level -8.1 dB; Coarse 1, Fine 0, Fixed 0.0 dB, Level 0.0 dB. The center panel shows an Envelope graph with parameters: Attack 0.00 ms, Decay 600 ms, Release 6.42 s, Time<Vel 0%, Wave Sin. The right panel shows LFO (Sine, Rate 96.76, Amount 52%), Filter (12, 24, OSR, Freq 55.3 Hz, Res 37%), Pitch Env (0.0%), Spread (0%), Transpose (0 st), Time (0%), Tone (70%), and Volume (-13 dB).



Toggler Rhythm Pattern

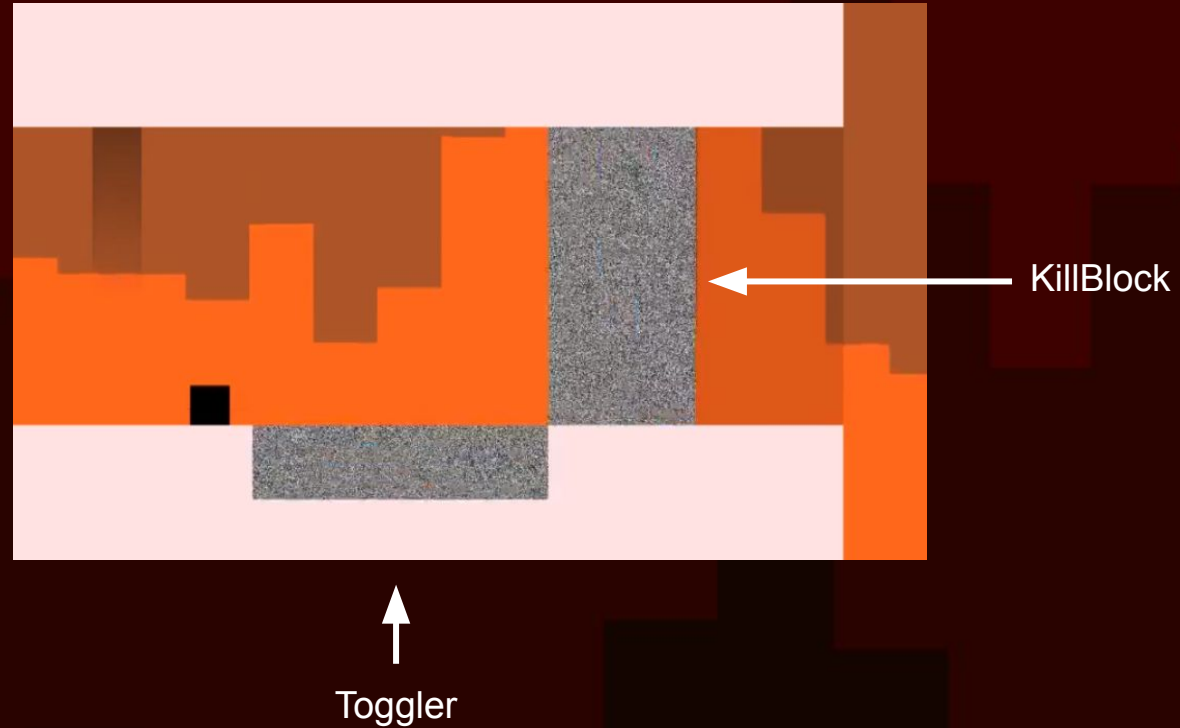
The toggler loop: 

- 3-bar rhythmic loop
- Game logic toggle floors between lethal and non-lethal
- Two states: A and B



Toggler and KillBlock

Both play at once in this jump puzzle!



Toggler and KillBlock Rhythms

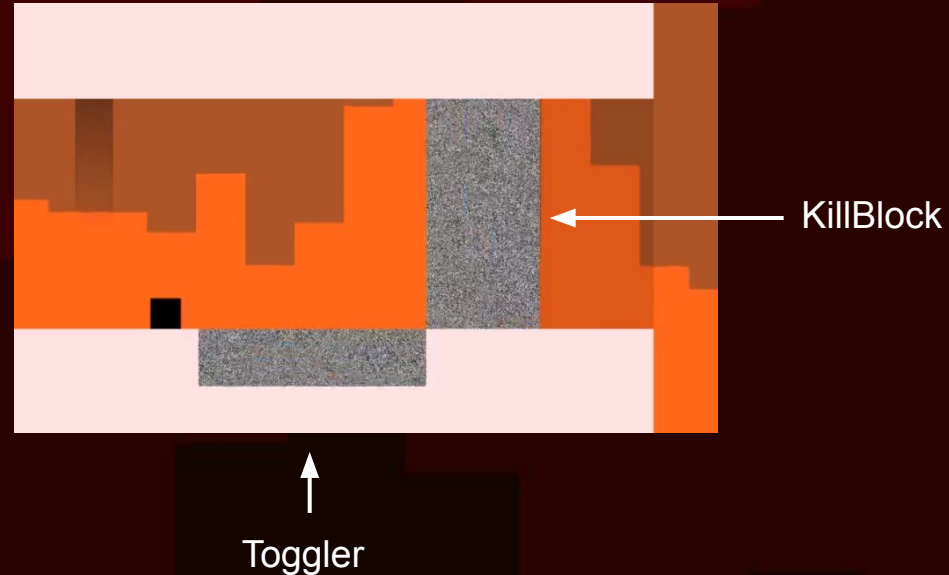
- KillBlock loop is 2 bars
- Toggler loop is 3 bars

KillBlock

```
1 . . . : . . . 2 . . . : . . .  
  > >      > > x
```

Toggler

```
1 . . . : . . . 2 . . . : . . . 3 . . . : . . .  
          A          B          A B
```



Toggler and KillBlock Looped

Loop simultaneously after $2 \times 3 = 6$ bars

KillBlock x 3

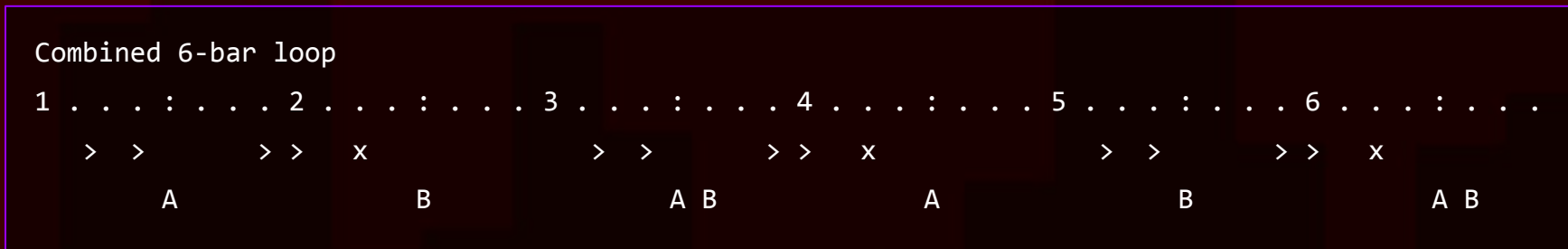
```
1 . . . : . . . 2 . . . : . . . 3 . . . : . . . 4 . . . : . . . 5 . . . : . . . 6 . . . : . . .  
  > >      > > x          | > >      > > x          | > >      > > x  
                    loop                    loop
```

Toggler x 2

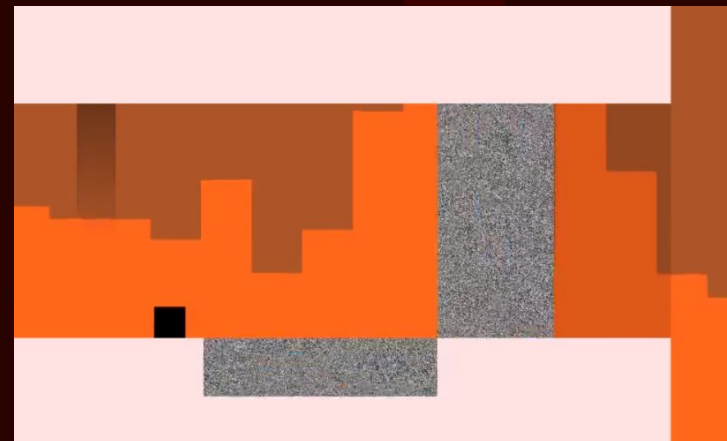
```
1 . . . : . . . 2 . . . : . . . 3 . . . : . . . 4 . . . : . . . 5 . . . : . . . 6 . . . : . . .  
          A          B          A B          |          A          B          A B  
                    loop
```

Toggler and KillBlock Combined

The combined 6-bar loop of **Toggler and KillBlock**: 



This pattern is what the player must grasp
to pass the jump puzzle



Toggler Harmony

- Toggler loop must be in harmony with KillBlock loop
- 8-bar harmonic loop

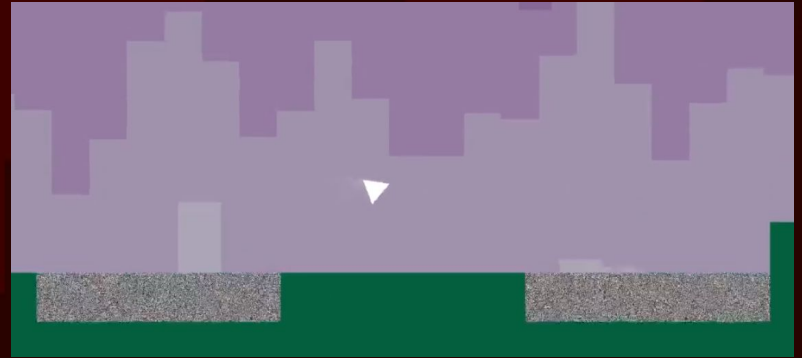
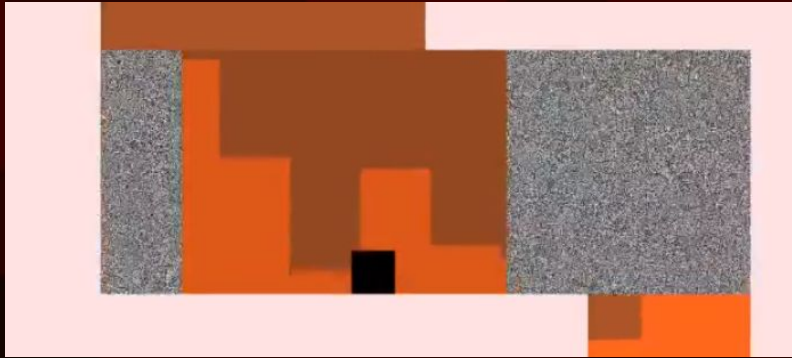
KillBlock:	Cm	D#m	F#m	C#m	Em	Gm	Dm	Fm
Toggler:	Cm7	D#m7	Bm11	C#m7	Em7	Cm11	Dm7	Fm7

Toggler Full Loop

- 3-bar rhythmic loop
- 8-bar harmonic loop
- Full loop: $3 \times 8 = 24$ bars

1	2	3	4	5	6	7	8	9	10	11	12			
A	B	AB		A	B	AB		A	B	AB		A	B	AB
Cm7	D#m7	Bm11		C#m7	Em7	Cm11		Dm7	Fm7		Cm11	D#m7	F#m7	C#m11
13	14	15	16	17	18	19	20	21	22	23	24			
A	B	AB		A	B	AB		A	B	AB		A	B	AB
Em7	Gm7	Dm11		Fm7	Cm7	D#m11		F#m7	C#m7	Em11		Gm7	Dm7	Fm11

Level 4 Video



Toggler Full Loop

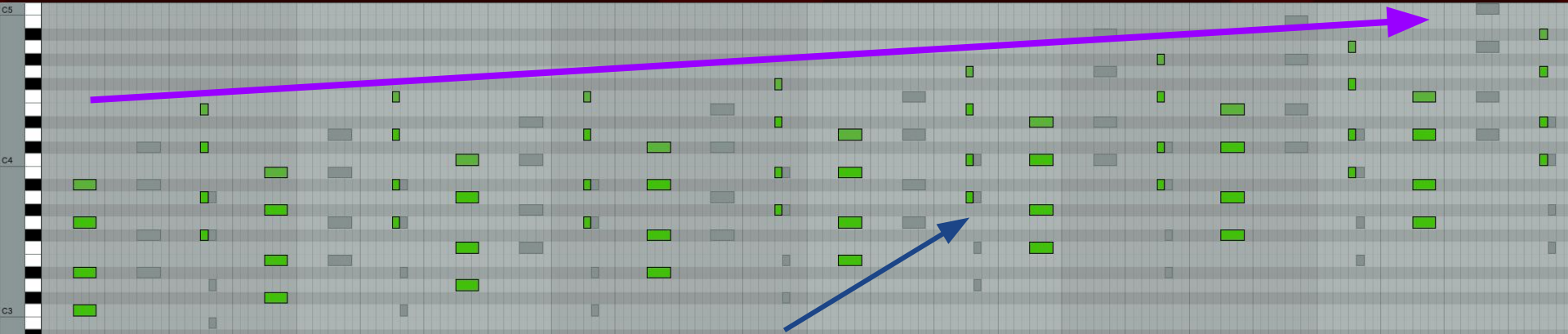
1	2	3	4	5	6	7	8	9	10	11	12	
A	B	AB	A	B	AB	A	B	AB	A	B	AB	
Cm7	D#m7	Bm11	C#m7	Em7	Cm11	Dm7	Fm7	Cm11	D#m7	F#m7	C#m11	
13	14	15	16	17	18	19	20	21	22	23	24	
A	B	AB	A	B	AB	A	B	AB	A	B	AB	
Em7	Gm7	Dm11	Fm7	Cm7	D#m11	F#m7	C#m7	Em11	Gm7	Dm7	Fm11	



Rising Pattern

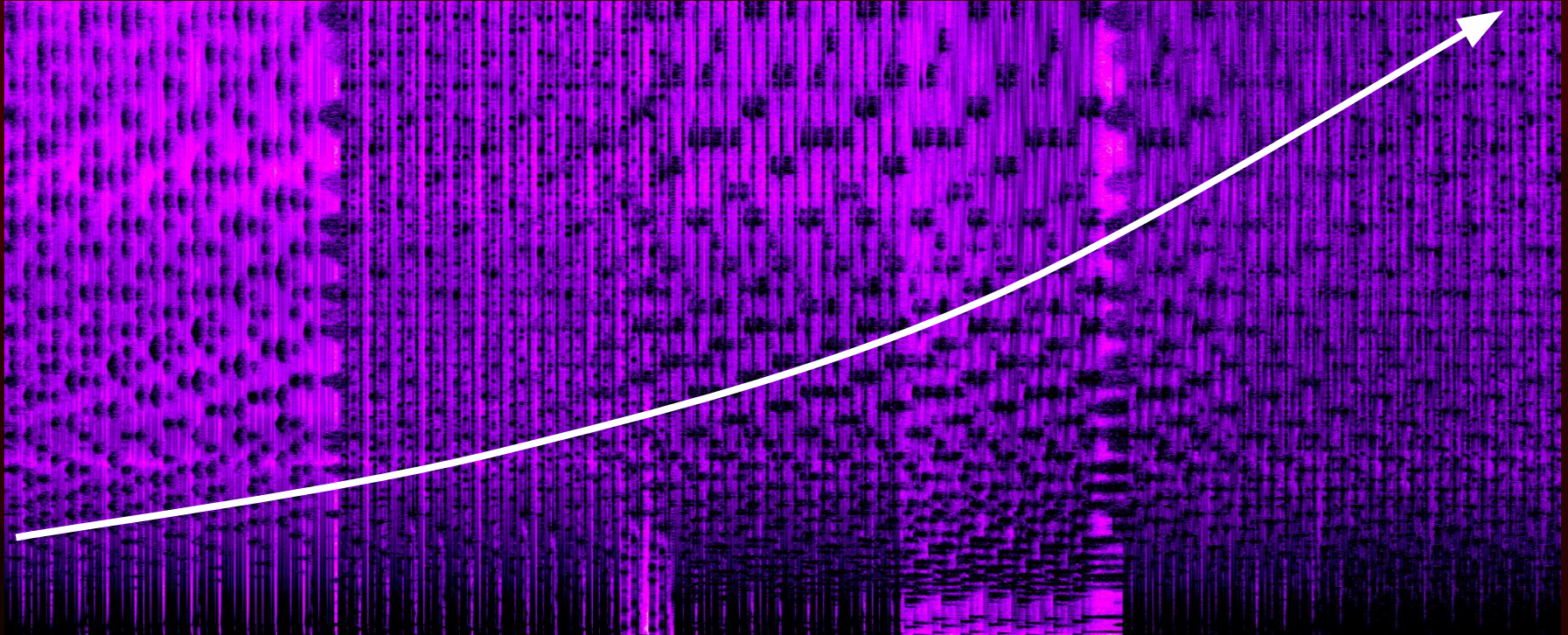
Level 4 is composed to emulate frequency continuously rising:

- Uses chord inversions to create 4-chord rising sequences
- Chord notes generally ascend over full 24-bar loop



Rising Pattern

Spectral analysis of soundtrack version shows rising frequency pattern



140 Soundtrack Available Now!

Vinyl

- iam8bit

Digital

- Steam
- GOG.com
- Spotify
- iTunes



The image shows the packaging for the 140 Vinyl Soundtrack. It features a large vinyl record with a colorful, concentric square pattern in shades of green, blue, and purple. The record is partially obscured by a black sleeve with the number '140' in white. The sleeve also has a vertical strip of static noise. Below the record, there is a small text box containing credits and a table of track durations.

Side A	Side B		
140 Title	2:50	140 Part 1	5:07
140 Part 1	5:00	140 Part 2	5:10
140 Part 2	5:00	140 Part 3	5:07
140 Part 3	4:54	Bonus	4:48
140 Part 4	4:50		
140 Menu	2:40		

Composed and produced by Jakob Schmid
www.schmid.dk / www.carlsongames.com
Created with Mikolaj Livo, Hodor, Korg M1 software synthesizer
Vinyl produced by South West Landfill.com 8817-8817
Mastered for vinyl by David Gardner, Infrazonic Mastering
The game 140 was created by Jeppe Carlsen, Jakob Schmid, Niels Fyrst, Andrea Arild Pedersen
Thanks to: Morten Stig Andersen, Peter Buchardt, Mikkel Sverdrup, Mikkel Gjøl, SØS Gunnar Nyberg, Christian Vogel, Arne Røse, Jan M. Skovum, Renee White, Christian Villum
140 © Carlsen Games. The 140 soundtrack by Jakob Schmid is licensed under a Creative Commons Attribution 4.0 International License. To view a copy of this license, visit www.creativecommons.org/licenses/by/4.0/.

140 Vinyl Soundtrack
Limited Edition of 1400

Includes:
Digital Soundtrack
Steam Code for Full Game

Music by:
Jakob Schmid

Carlsen Games
iam8bit
iam8bit.com

Adaptive Audio in 140

- Position-Adaptive Music: audio sources are placed in level geometry
- Music is a set of synchronized loops
- Music is created to fit game logic timing
- Fitting different harmonic and rhythmic patterns together is a puzzle in itself

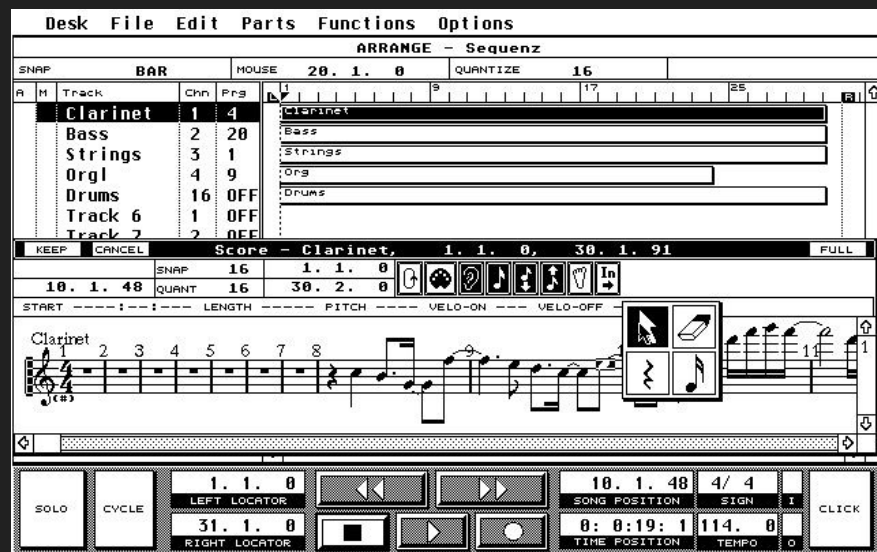


Realtime Synthesis



MIDI-like Sequencing

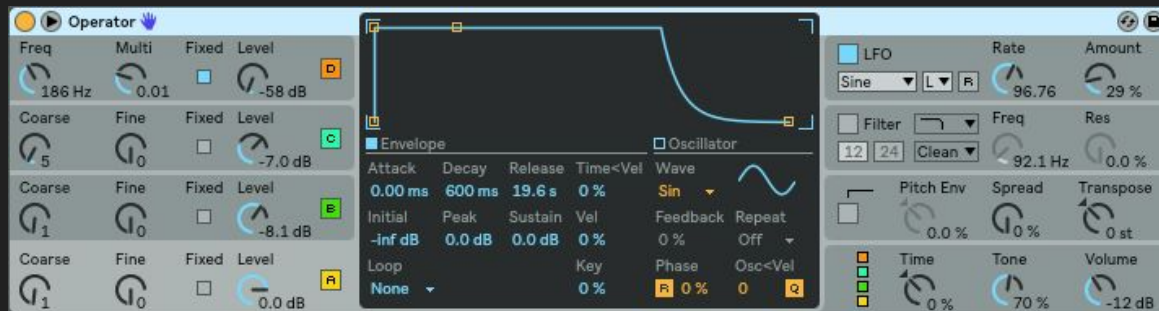
- Sequencing of samples or real-time synthesis
- Key changes
- Removing notes
- Procedural / generative music



Cubase (1989)

Real-time Synthesis

- Parameter changes controlled from game
- Subtle changes in timbre accompany game events
- Variations in timbre retain player interest even though sequence repeats



Ableton Live 10: Operator

Realtime Synthesis was the Norm

- 1970s to mid 1980s: hardware-based realtime synthesis
- Hardware synthesizer-based hardware platforms
 - Arcade machines (1970s and forward)
 - Atari 2600 (1979)
 - ZX Spectrum (1982)
 - Commodore 64 (1982)



Marble Madness



ZX Spectrum



Commodore 64



Atari 2600

Modern Realtime Synthesis

- Implemented as audio plugins in sound engines
- Normally rendered on CPU, not in dedicated hardware



FMOD Studio plugin



Questions?

Twitter: @jakobschmid

E-mail: jakob@schmid.dk

playdead.com

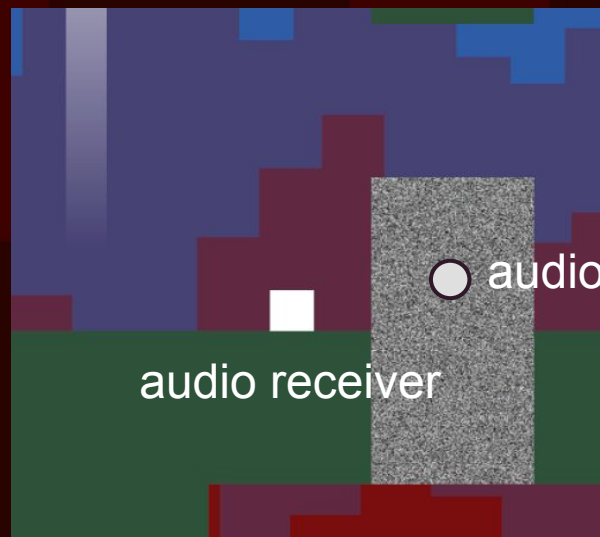
carlsengames.com

Slides are here: schmid.dk/talks/



Attenuation and Panning

Simple attenuation and panning for music loops using built-in audio system



audio source

audio receiver



audio source

audio receiver