Quartz: Audio / Visual Sync for Procedural & Interactive Systems.

I’m Max Hayes.

Audio Engine Programmer

At Epic Games focused on medium-to-low level audio systems for Unreal Engine. One of which is the Quartz Subsystem.

Musician

Initially studied Music Production & Engineering and Electronic Production & Design at Berklee College of Music (guitar principal) before transferring to…

DigiPen

Where I completed a BS in Computer Science & Digital Audio (graduated in 2019).

Agenda

What is this place?
Overview of Game & Audio Engines

Goal: I want to do stuff perfectly on beat
Ability to play sounds on strongly-timed boundaries in sync with gameplay / visuals

Problem: I seem to not be able to do stuff perfectly on the beat
Multithreading, arbitrary grids, latency.

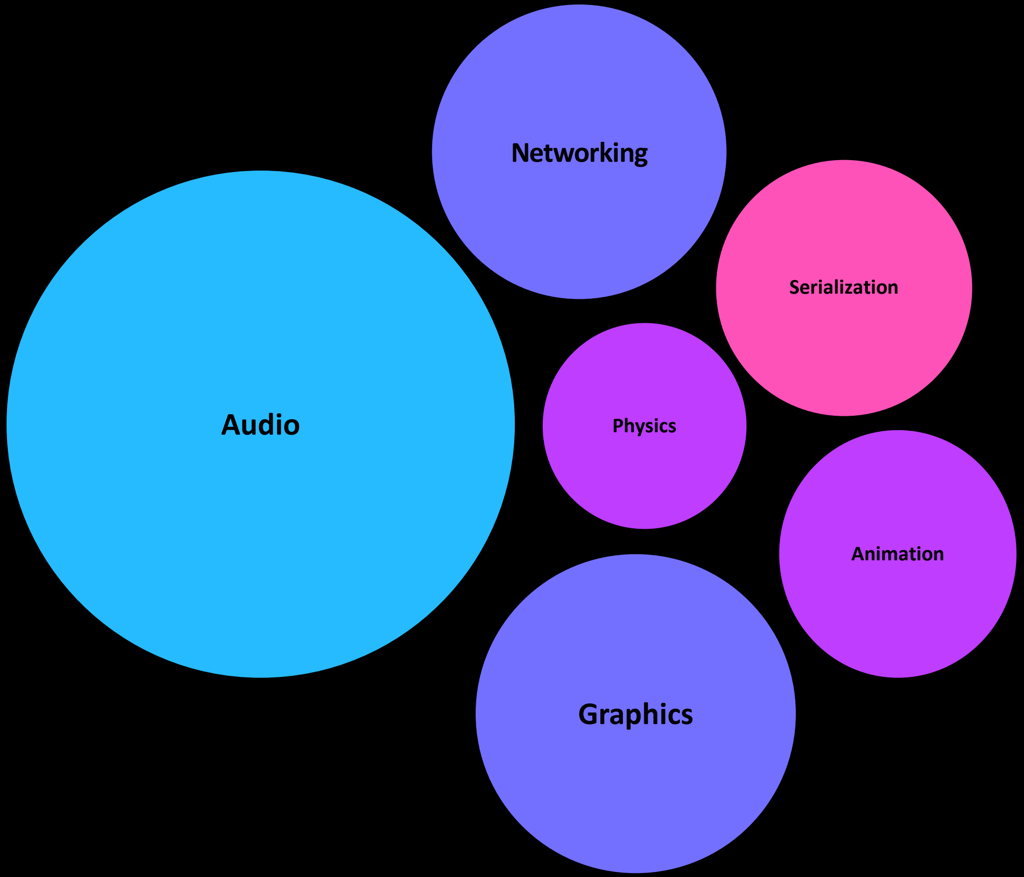
Solution: How do I do it on the beat
How Quartz approaches the problem space.
(Not a Quartz tutorial)

Outcomes:
What is unlocked once you have such a solution?

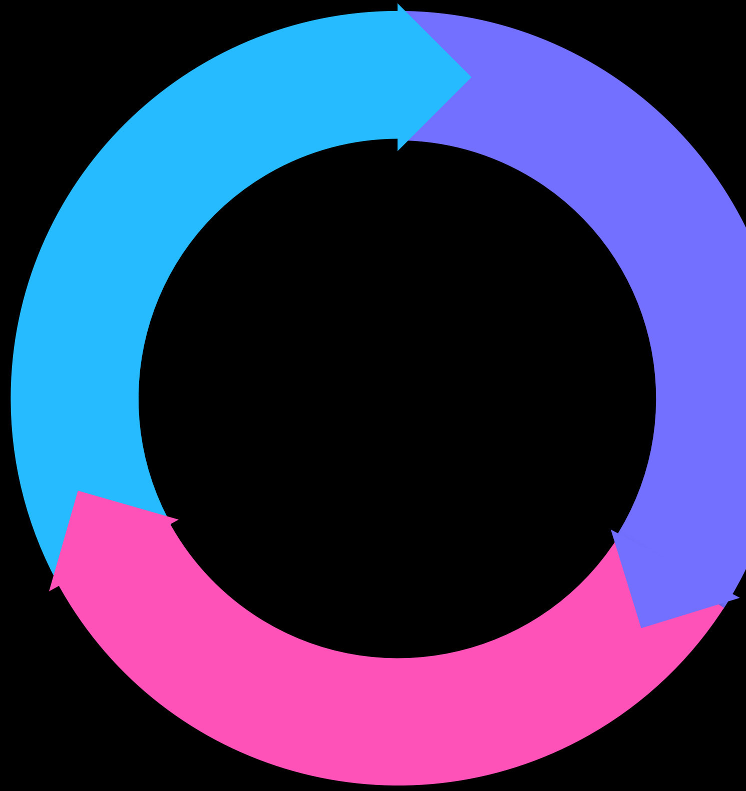
What is a Game Engine?

(Especially the sound part)

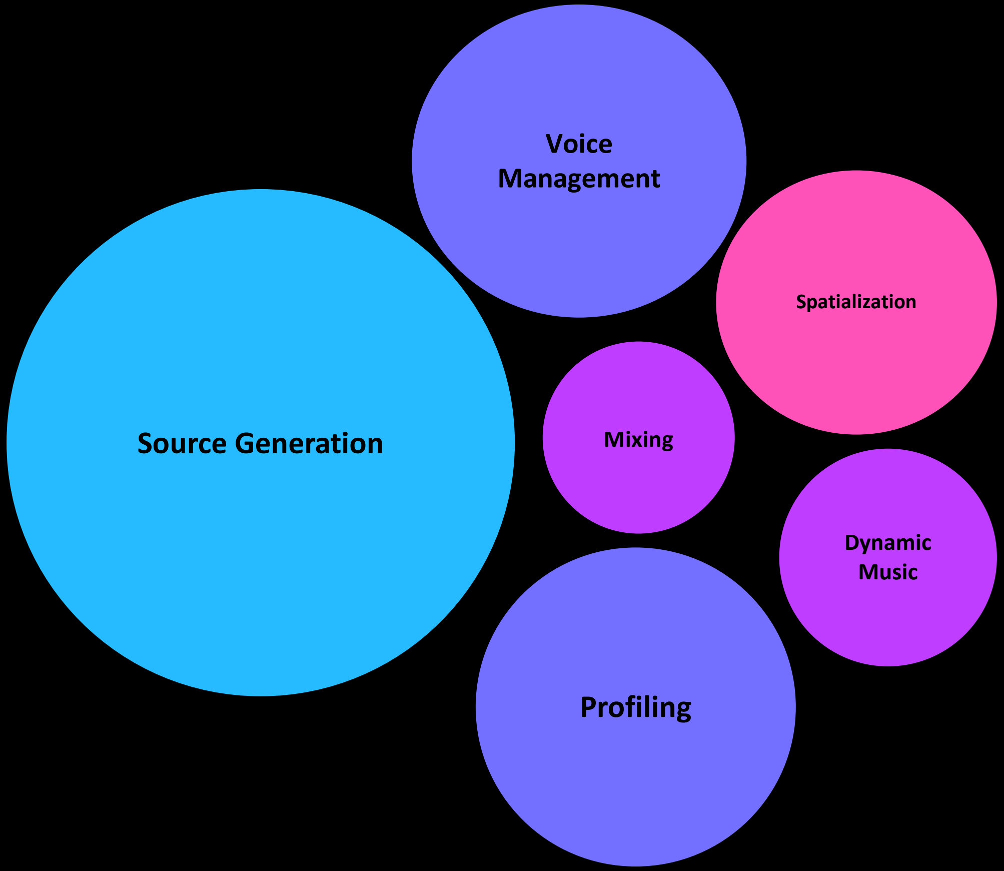
Game Engine:
A collection of real-time software systems working in concert to create a dynamic, user-driven experience.



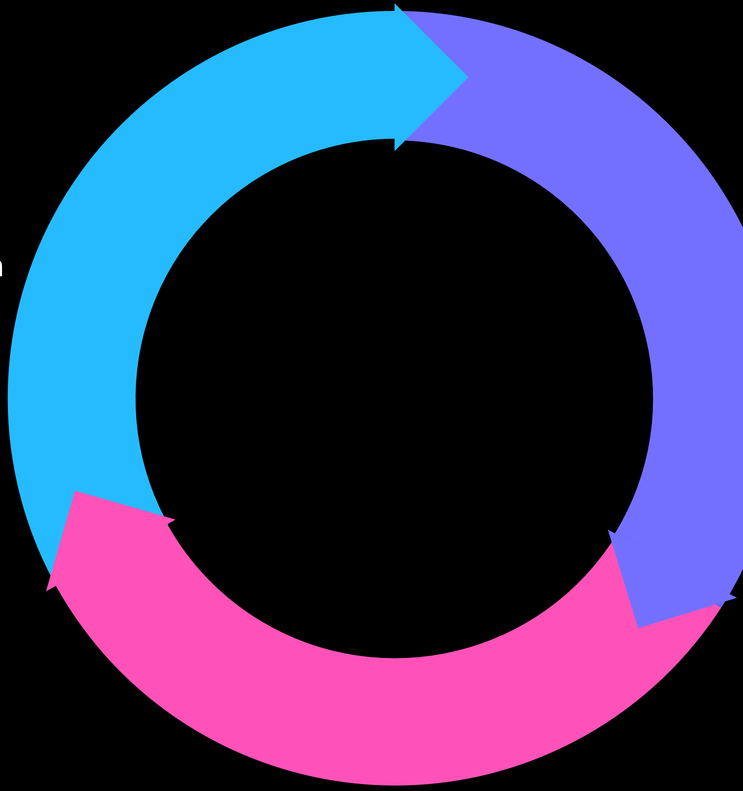
Game Loop


Check for Input / Player action
Jump? Pause? Quit?
Update Actors / Systems
Animation, Physics, Game Logic, send Audio Commands
Draw To Screen
Send vertex data to the graphics card

Audio Engines:
A constellation of features Sound Designers use to create the dynamic sonic experience for the player or audience.



Audio Engine Update



Generate Source Audio

Start/stop sounds, run decoders, synths, apply source-level effects

Mix and Submix

Submix graph, analyzers, submix effects, etc.

Mixdown to final buffer

Stage the data to be sent to the OS

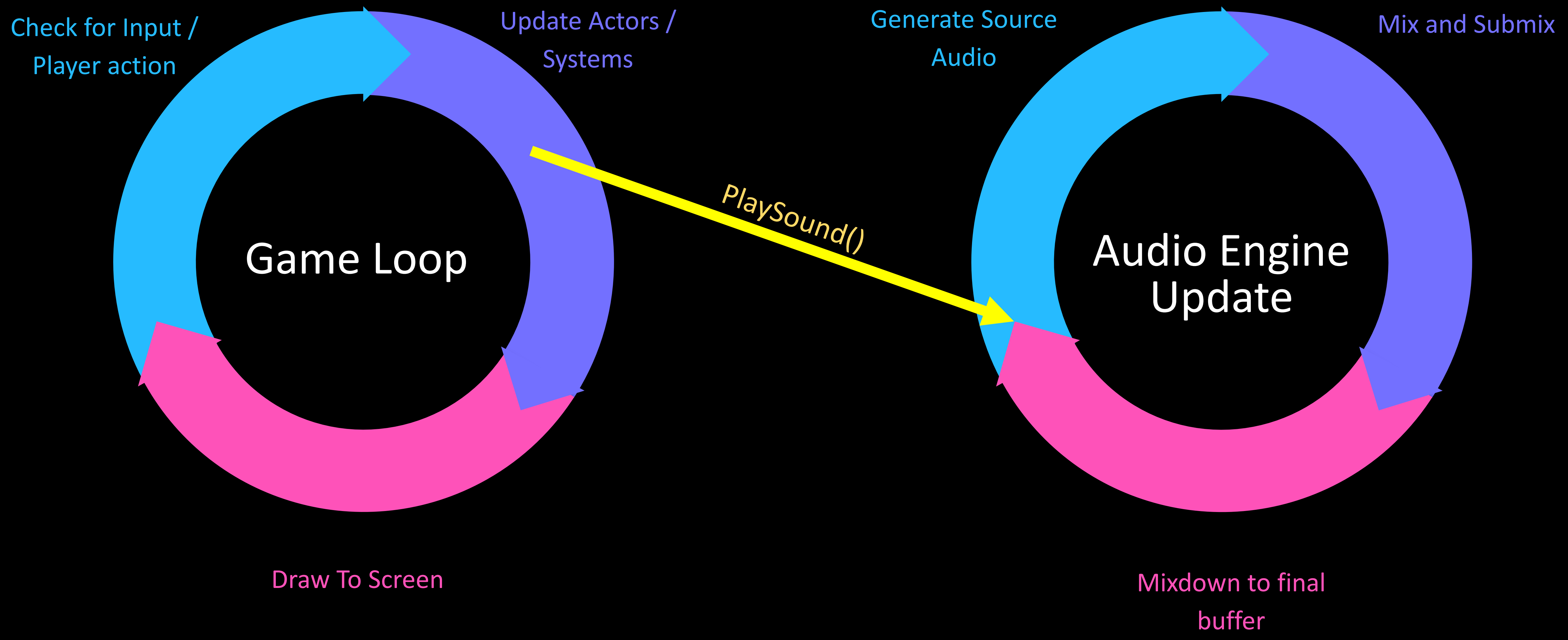
Games are multi-threaded:

Audio (basically) always has its own thread.
The operating system periodically asks our audio engine for the next chunk of audio samples.

Our logic runs on the game thread.
This is where things get updated, we decide to play a sound (or not), and the next frame gets drawn to the screen.

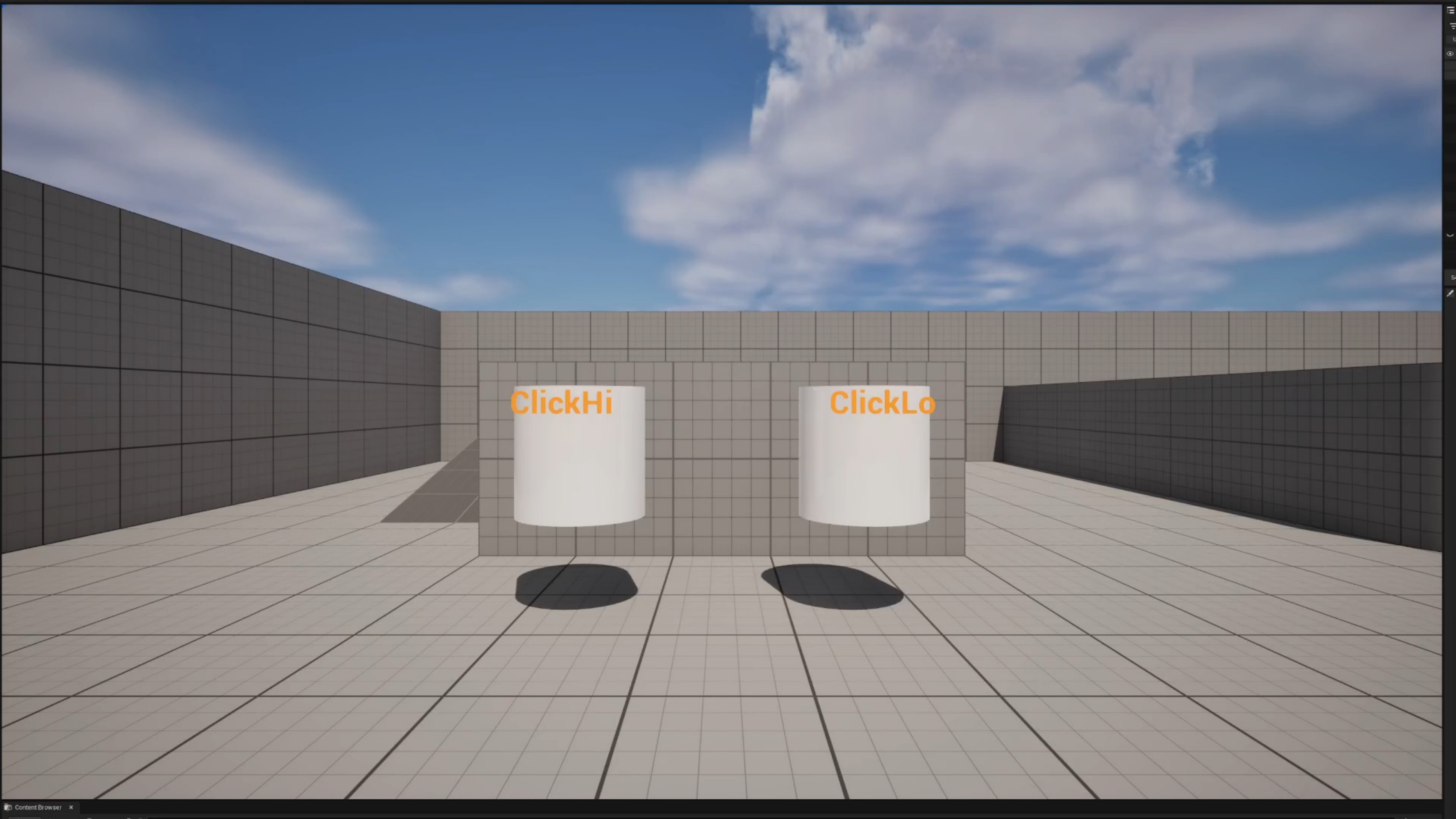
This is a slight over-simplification.
Unreal also has an Audio Thread and an Audio Render Thread (in addition to the Game Thread and OS audio call-back).

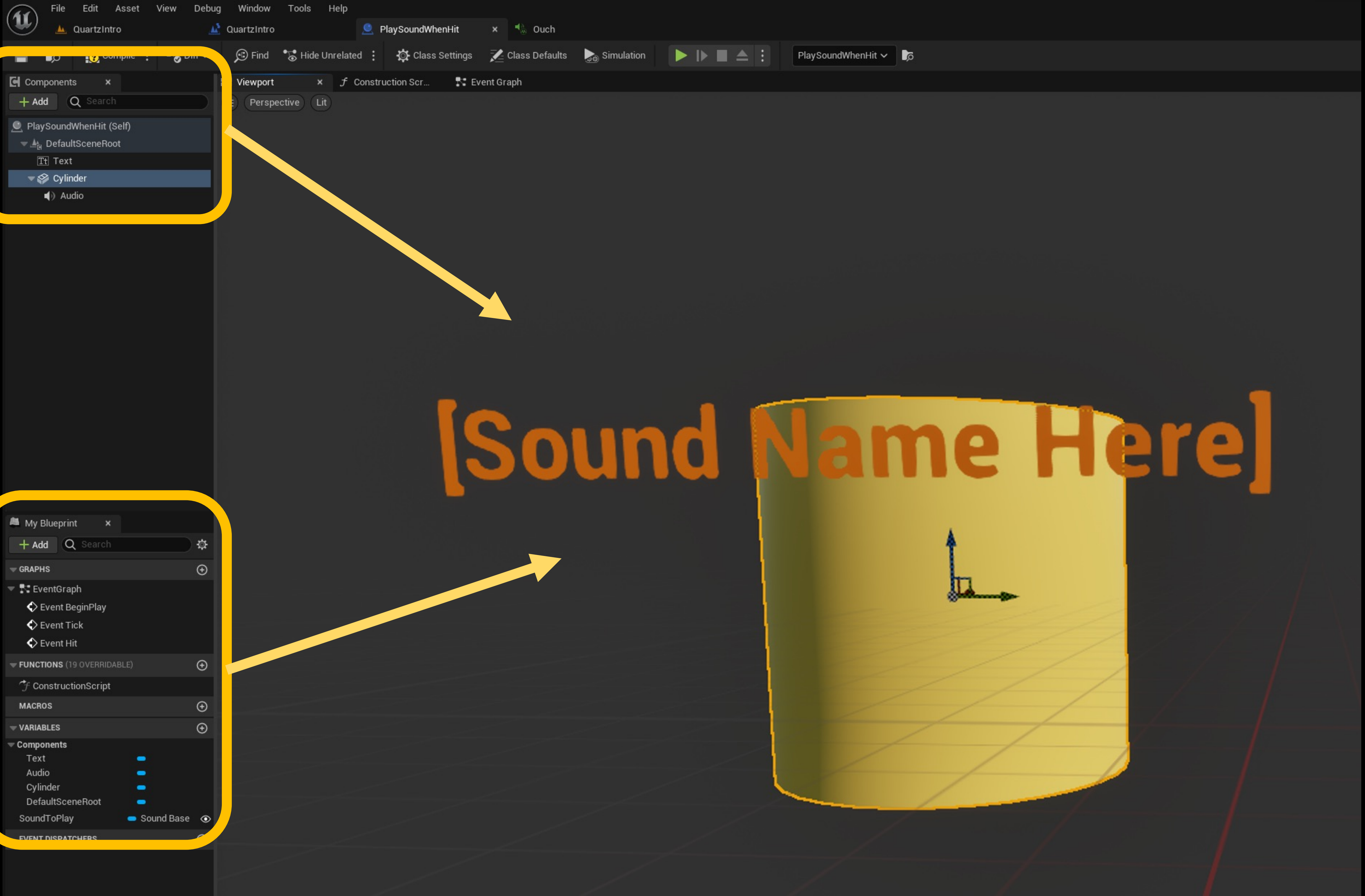
Games are multi-threaded

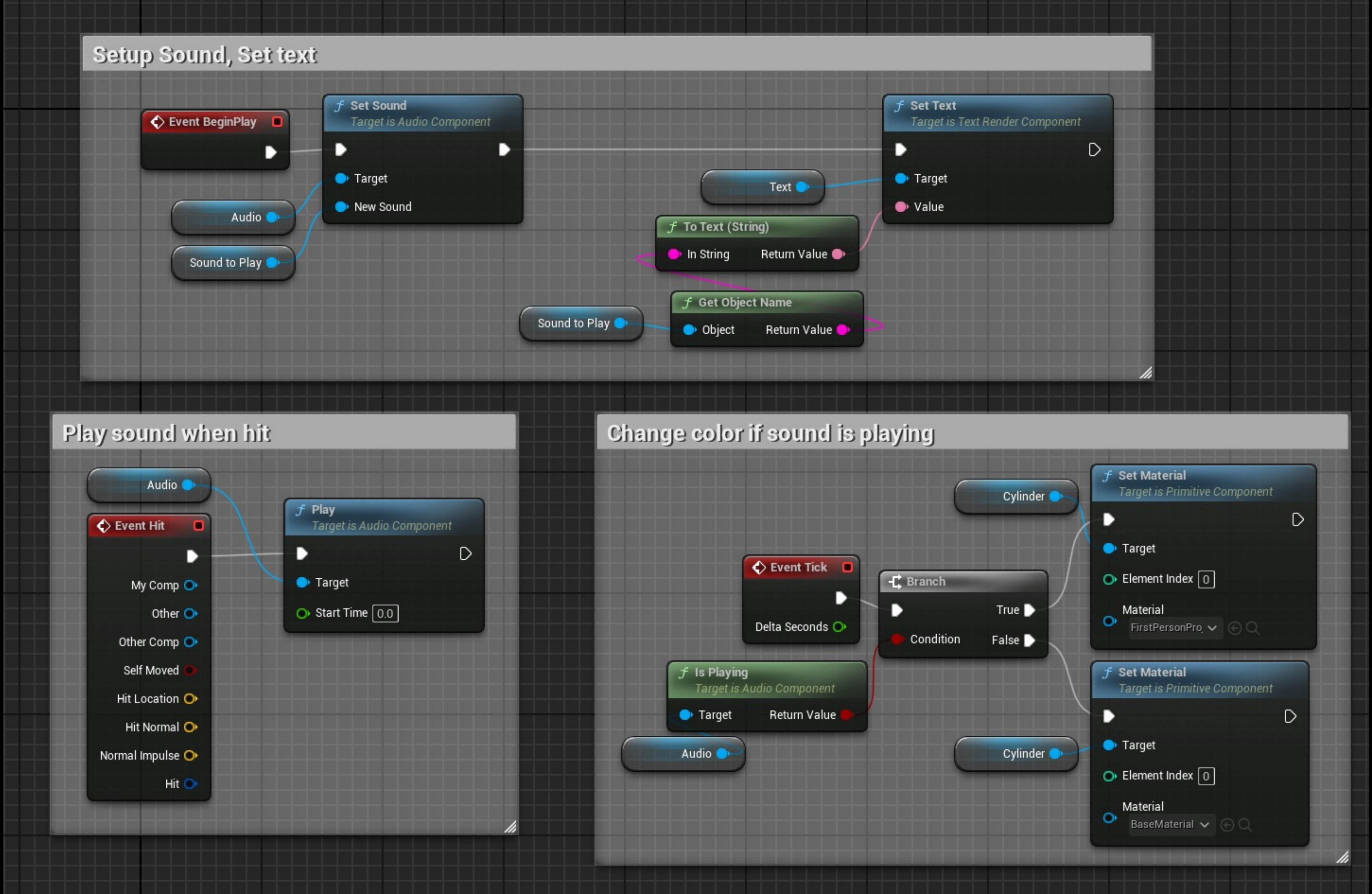


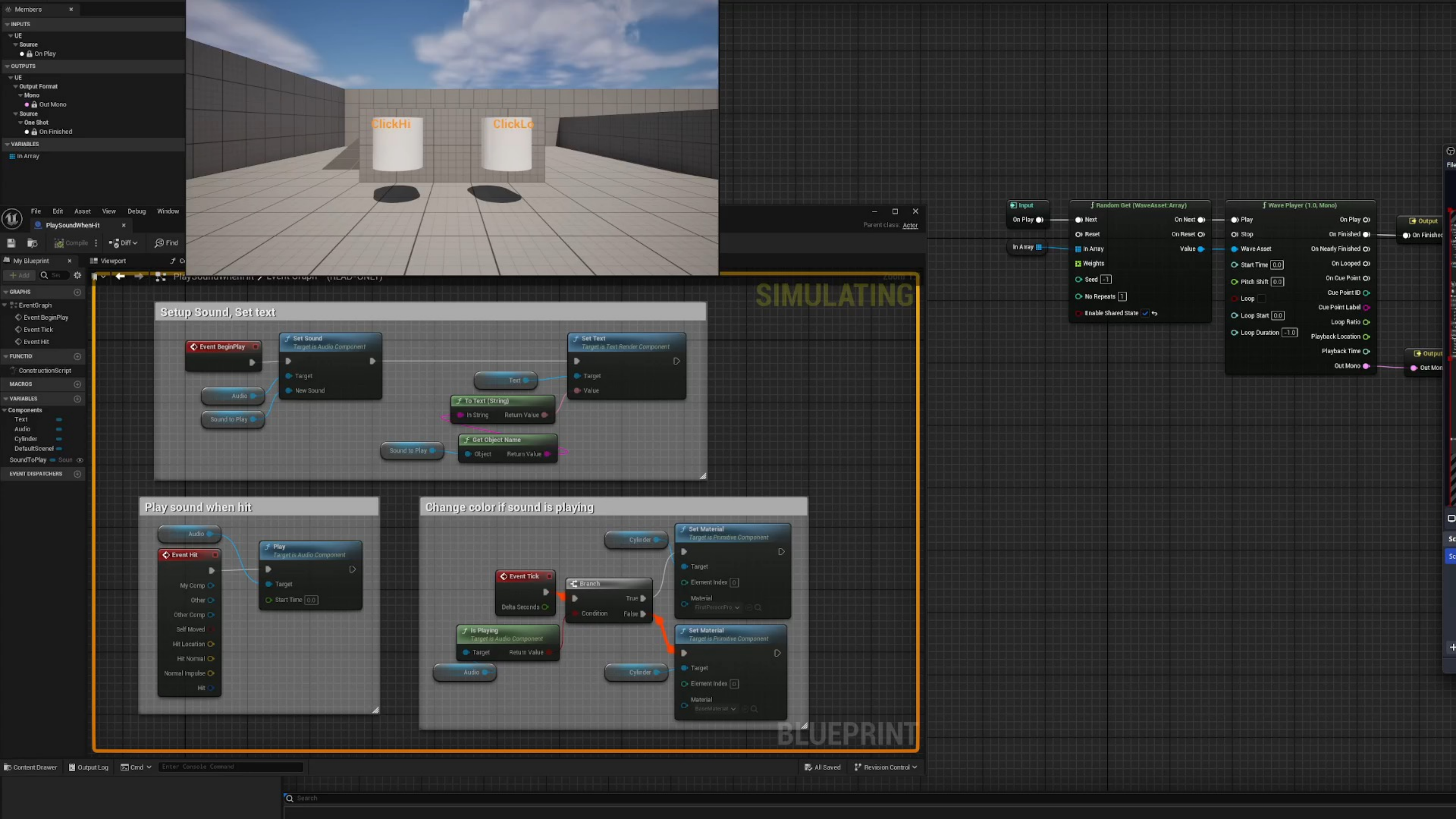
So lets do some work in a Game Engine.

Lets make an actor make sound.


ClickHi
ClickLo
Content Browser


File
Edit
Asset
View
Debug
Window
Tools
Help
QuartzIntro
QuartzIntro
PlaySoundWhenHit
Ouch
Find
Hide Unrelated
Class Settings
Class Defaults
Simulation
PlaySoundWhenHit
Components
Add
Search
PlaySoundWhenHit (Self)
DefaultSceneRoot
Text
Cylinder
Audio
Viewport
Construction Scr...
Event Graph
Perspective
Lit
[Sound Name Here]
My Blueprint
Add
Search
GRAPHS
EventGraph
Event BeginPlay
Event Tick
Event Hit
FUNCTIONS (19 OVERRIDABLE)
ConstructionScript
MACROS
VARIABLES
Components
Text
Audio
Cylinder
DefaultSceneRoot
SoundToPlay
Sound Base
EVENT DISPATCHERS


Setup Sound, Set text
Event BeginPlay
Set Sound
Target is Audio Component
Target
New Sound
Audio
Sound to Play
Set Text
Target is Text Render Component
Target
Value
Text
To Text (String)
In String
Return Value
Get Object Name
Object
Return Value
Sound to Play
Play sound when hit
Audio
Event Hit
My Comp
Other
Other Comp
Self Moved
Hit Location
Hit Normal
Normal Impulse
Hit
Play
Target is Audio Component
Target
Start Time 0.0
Change color if sound is playing
Cylinder
Set Material
Target is Primitive Component
Target
Element Index 0
Material
FirstPersonPro
Event Tick
Delta Seconds
Branch
True
Condition
False
Is Playing
Target is Audio Component
Target
Return Value
Audio
Cylinder
Set Material
Target is Primitive Component
Target
Element Index 0
Material
BaseMaterial


Members
INPUTS
Source
On Play
OUTPUTS
Output Format
Mono
Out Mono
One Shot
On Finished
VARIABLES
In Array
ClickHi
ClickLo
File
Edit
Asset
View
Debug
Window
PlaySoundWhenHit
Compile
Diff
Find
My Blueprint
Viewport
Parent class: Actor
GRAPHS
EventGraph
Event BeginPlay
Event Tick
Event Hit
ConstructionScript
MACROS
Components
Text
Audio
Cylinder
DefaultSceneRoot
SoundToPlay
EVENT DISPATCHERS
SIMULATING
Setup Sound, Set text
Event BeginPlay
Set Sound
Target is Audio Component
Target
New Sound
Audio
Sound to Play
Set Text
Target is Text Render Component
Value
To Text (String)
In String
Return Value
Get Object Name
Object
Play sound when hit
Event Hit
My Comp
Other
Other Comp
Self Moved
Hit Location
Hit Normal
Normal Impulse
Hit
Play
Start Time
Change color if sound is playing
Set Material
Target is Primitive Component
Element Index
Material
Event Tick
Delta Seconds
Branch
Condition
True
False
Is Playing
BLUEPRINT
Input
Random Get (WaveAsset:Array)
Next
Reset
Weights
Seed
No Repeats
Enable Shared State
On Next
On Reset
Wave Player (1.0, Mono)
Stop
Wave Asset
Pitch Shift
Loop
Loop Start
Loop Duration
On Nearly Finished
On Looped
On Cue Point
Cue Point ID
Cue Point Label
Loop Ratio
Playback Location
Playback Time
Output
Content Drawer
Output Log
Cmd
Enter Console Command
All Saved
Revision Control
Search

As a sound designer, I want to be able to...

As a sound designer, I want to be able to...

Play sounds on strongly-timed boundaries.

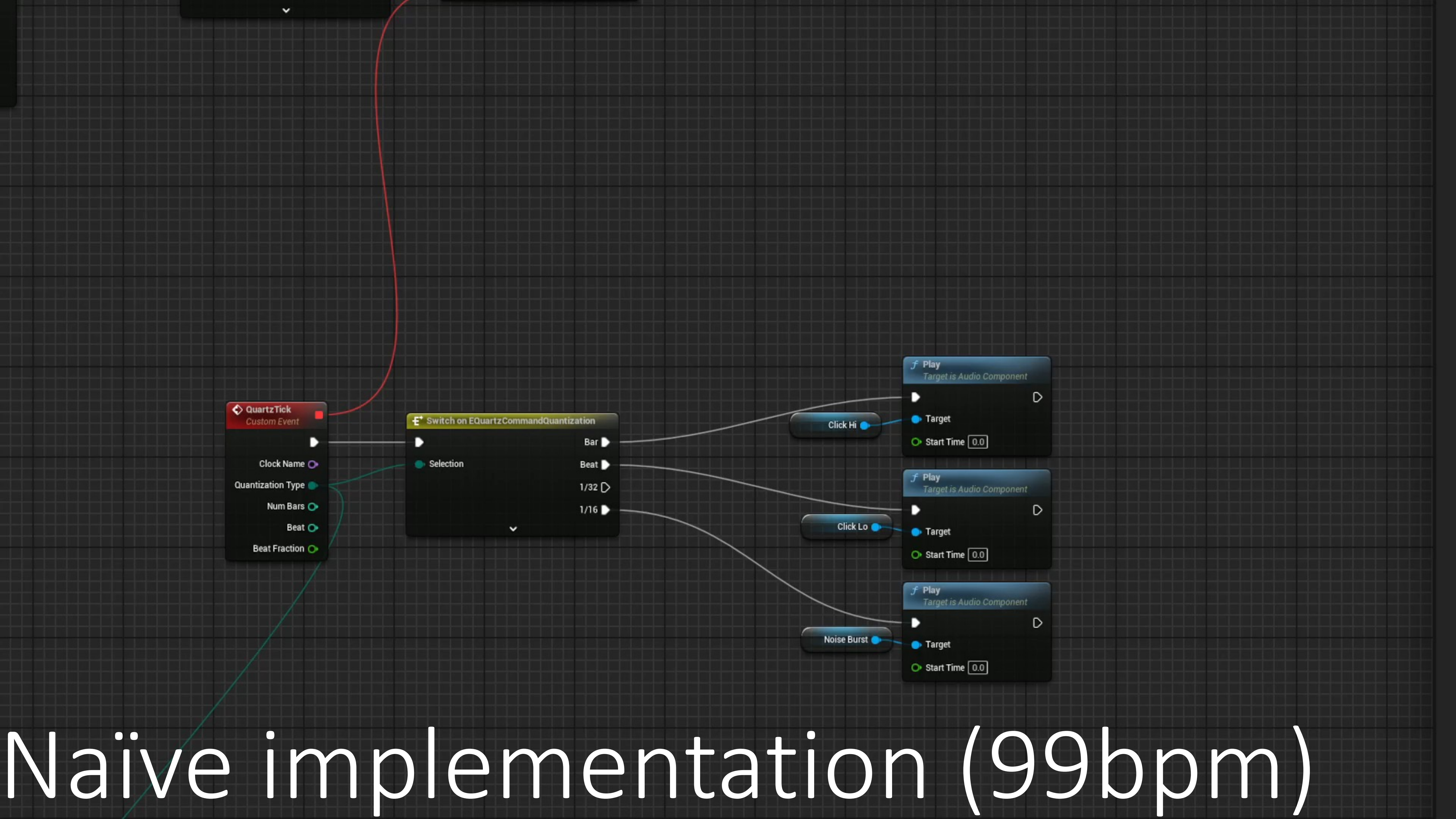
Could be dynamic music, machine guns, breathing/heartbeat systems, etc.

Trigger gameplay logic and VFX in sync with audio.

Let other disciplines tap into my audio system (audio-driven gameplay).

Why wouldn’t this work by default?

I can do some math, keep track of delta times, decide when to play my sounds...

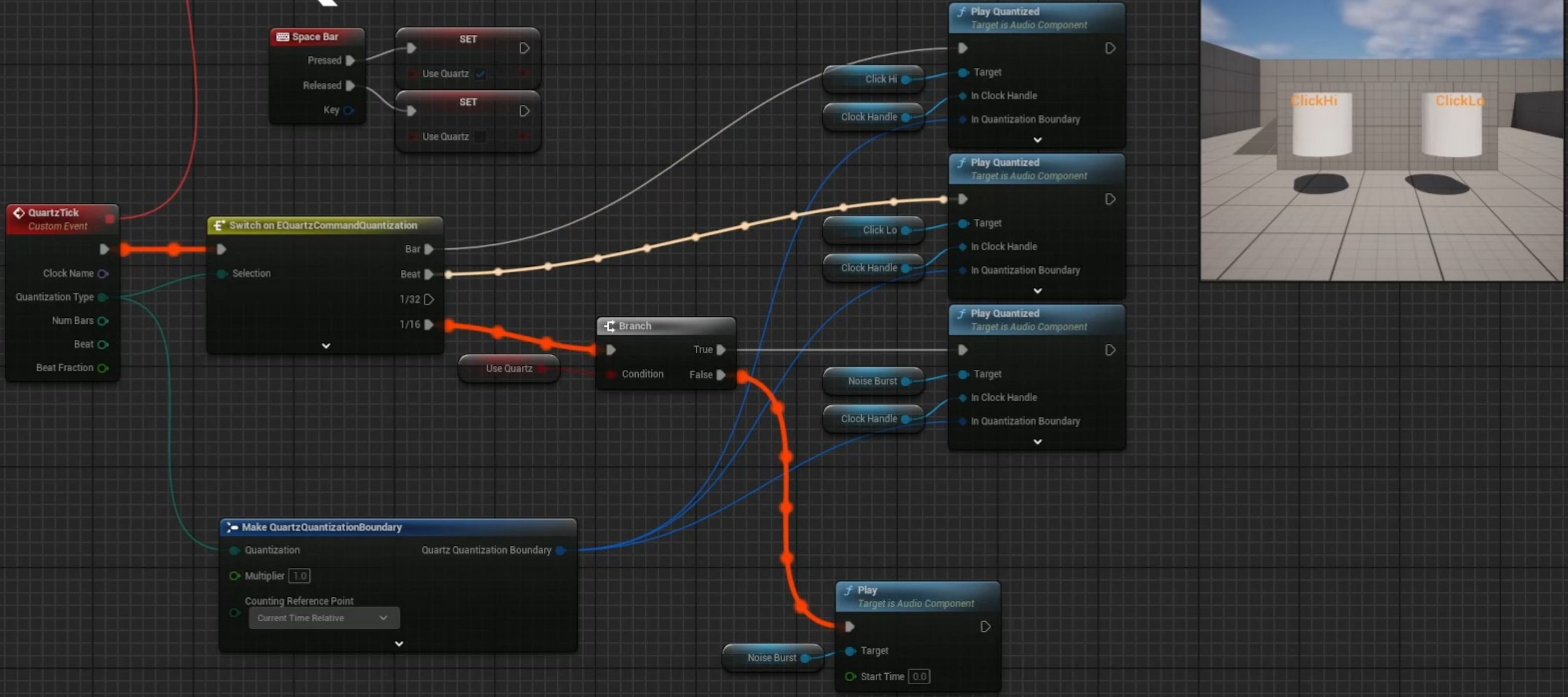


Naïve implementation (99bpm)

Why wouldn’t this work by default?

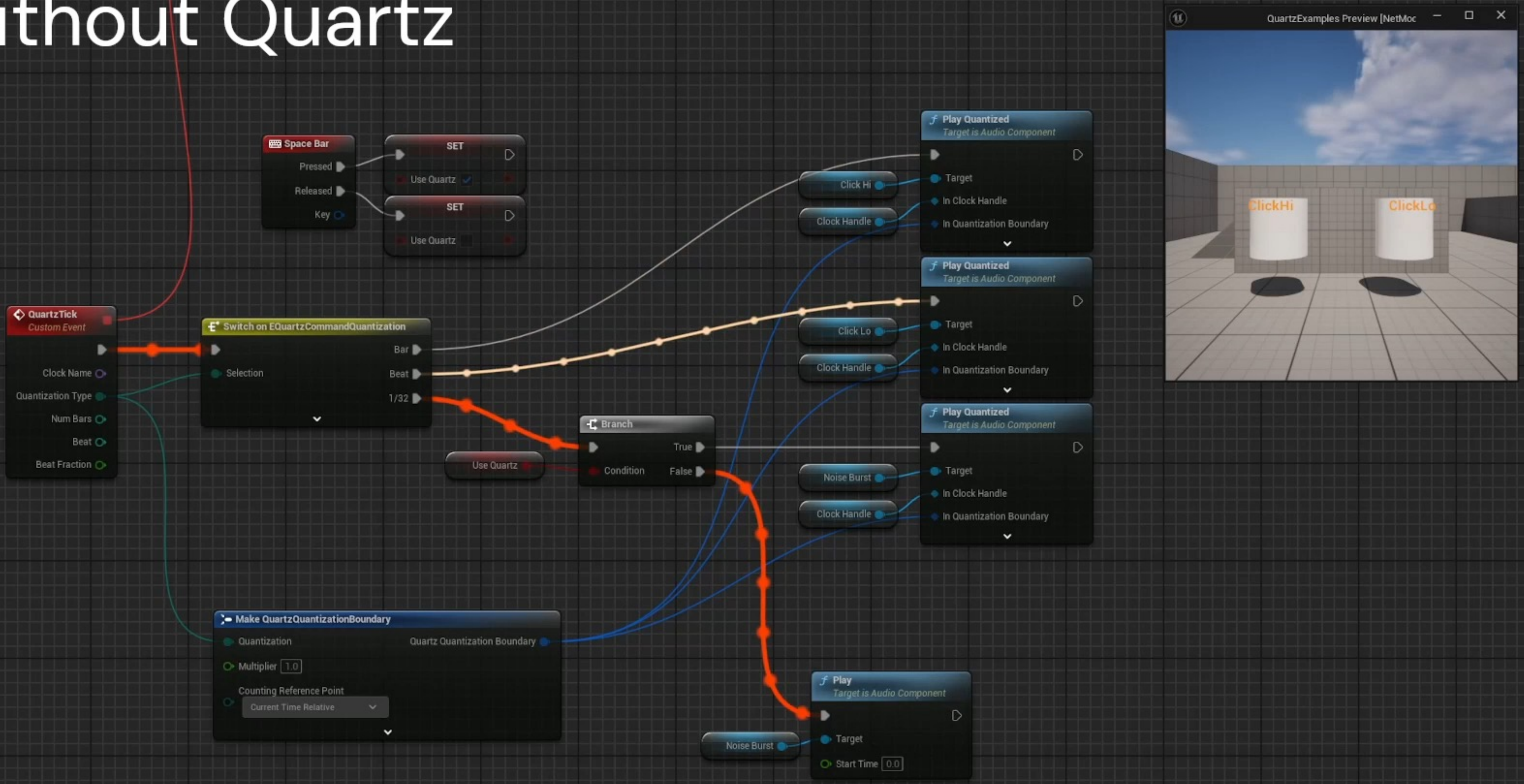
…Not quite my tempo

Without Quartz



A/B with 16th notes (99bpm)

Without Quartz



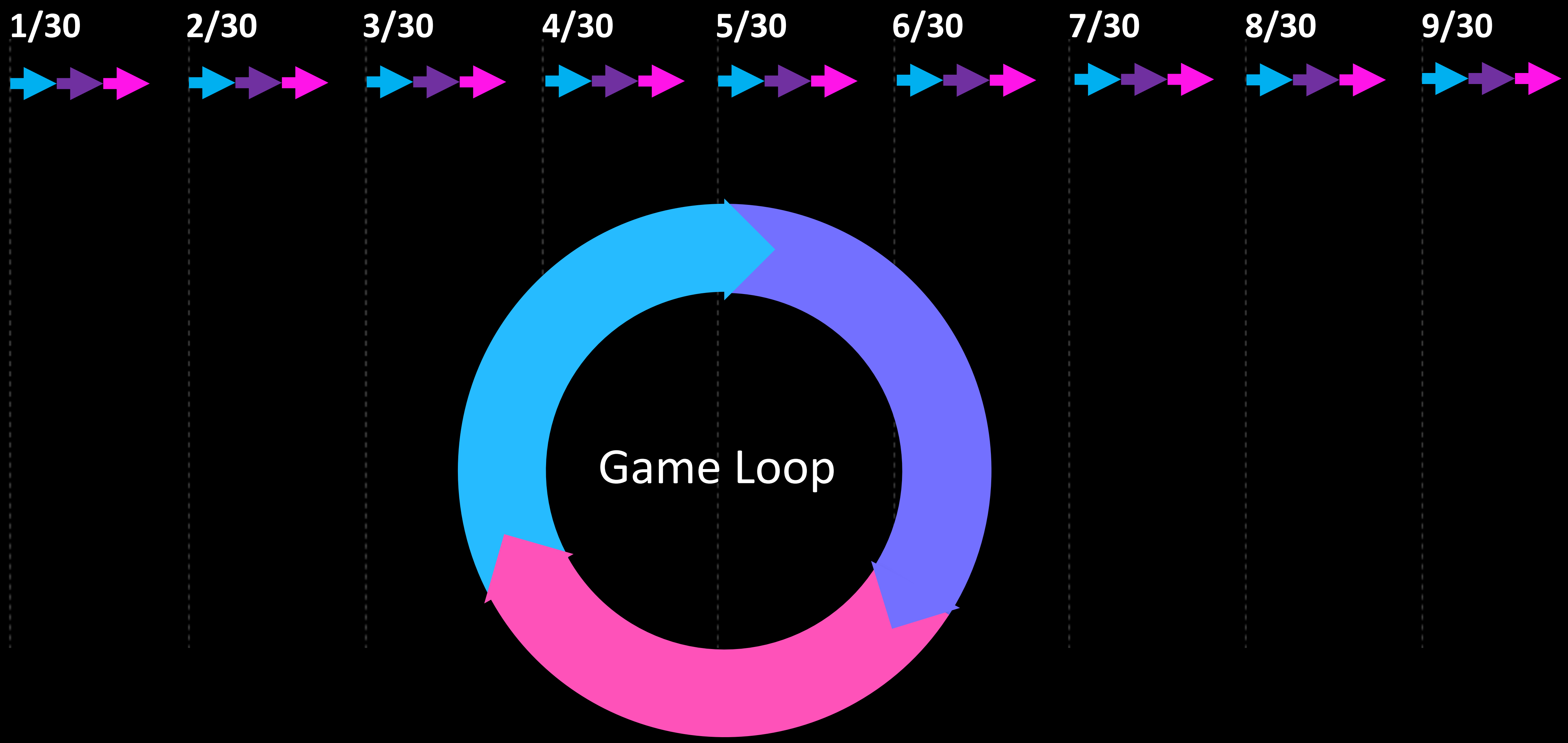
A/B with 32nd notes (99bpm)

Obstacle Number 1:

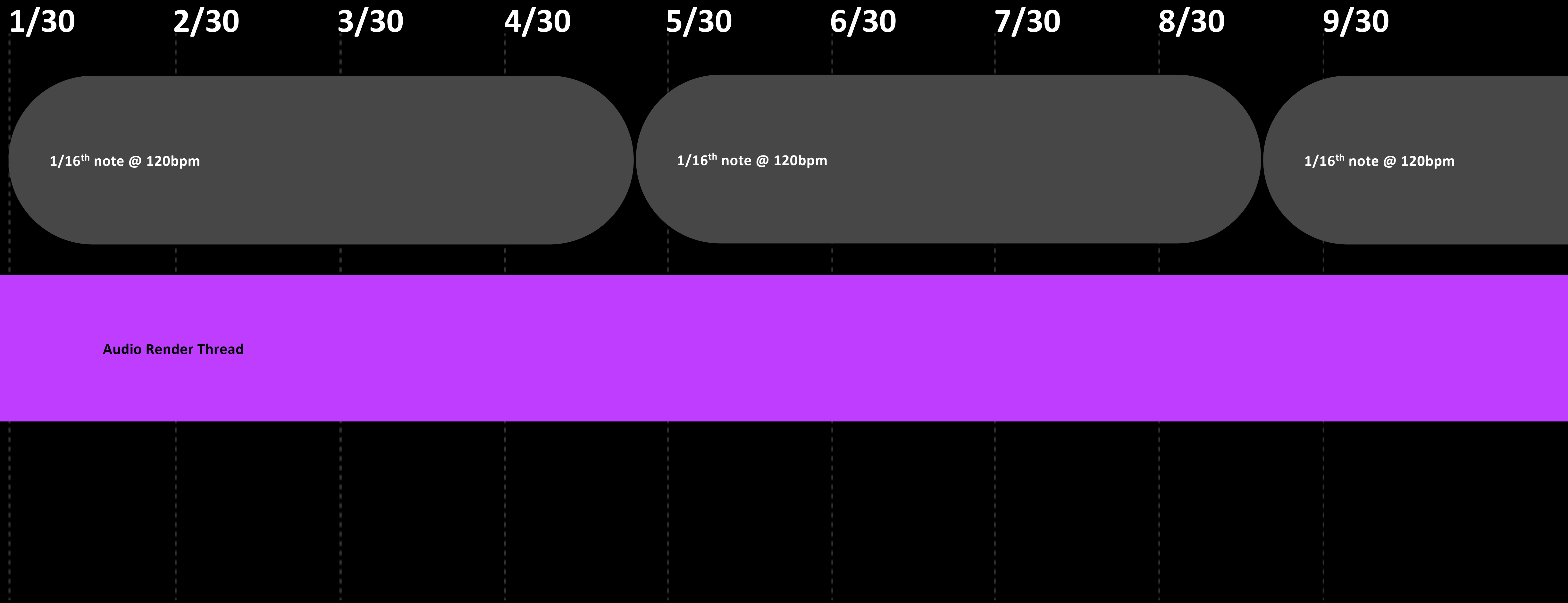
The game thread only ticks at a limited rate.

(30fps, 60fps, 100fps, etc.)

At 30 fps we only get to update every 33.3 ms



At 30 fps we only get to update every 33.3 ms



As a sound designer, I want to be able to...

Play sounds on strongly-timed boundaries.

Could be dynamic music, machine guns, breathing/heartbeat systems, etc.

Trigger gameplay logic and VFX in sync with audio.

Let other disciplines tap into my audio system (audio-driven gameplay).

Not be bound to the game's frame rate.

I need to be able to trigger sounds between game frames, and be unaffected by fps dips/drops.

But once we solve that it should be easy…

If we could sort that out it should just work right?

Audio is very high-res right??

Basically 48,000fps… right???

…Audio is not rendered in samples

Audio is rendered in blocks of samples we call buffers.

Obstacle Number 2:

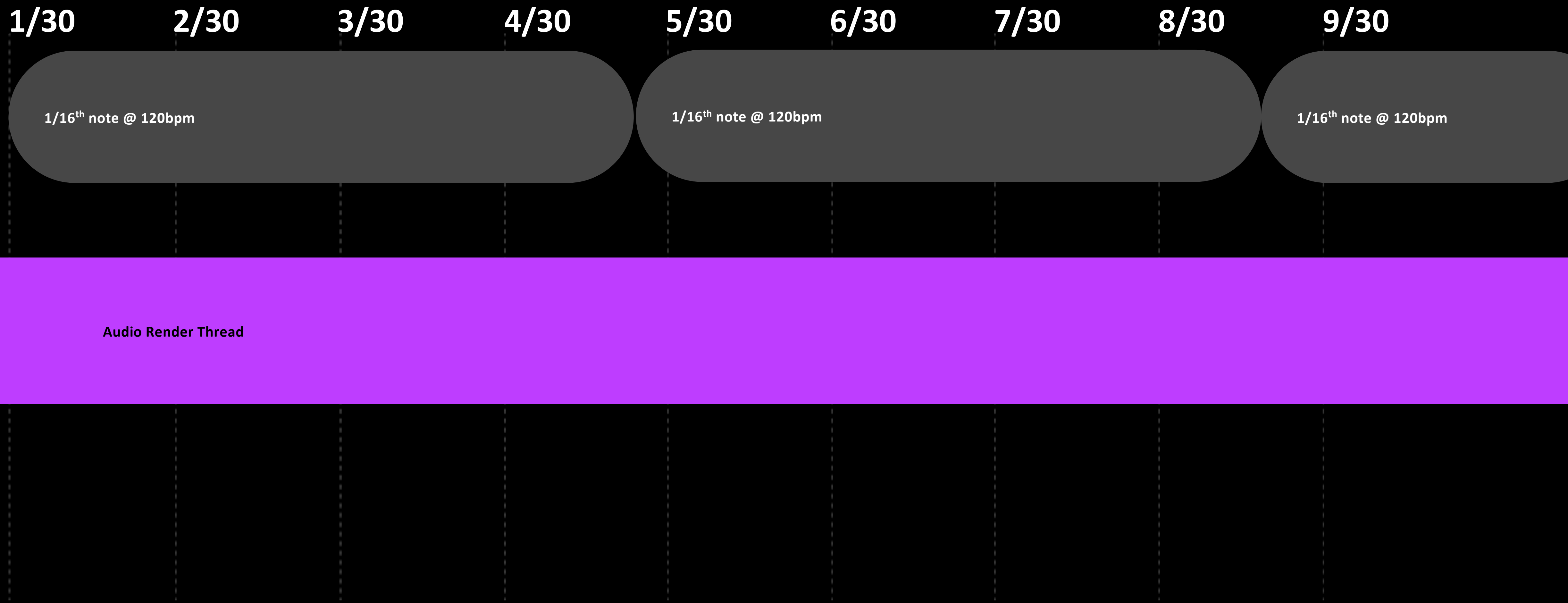
Audio is rendered in buffers.

The audio engine only processes pending requests right before each buffer is rendered.

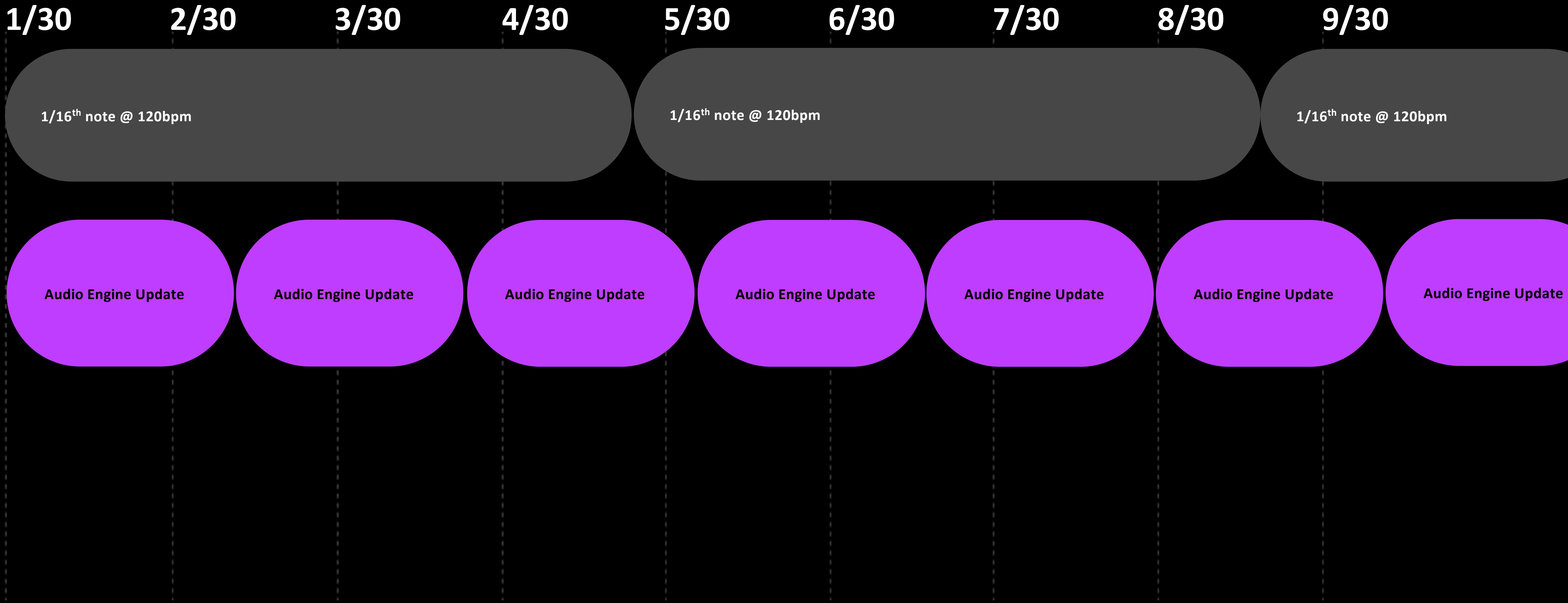
New sounds will only play at the beginning of the next buffer.

(buffer size ‘N’ = 1024, 2048, 4096, etc.)

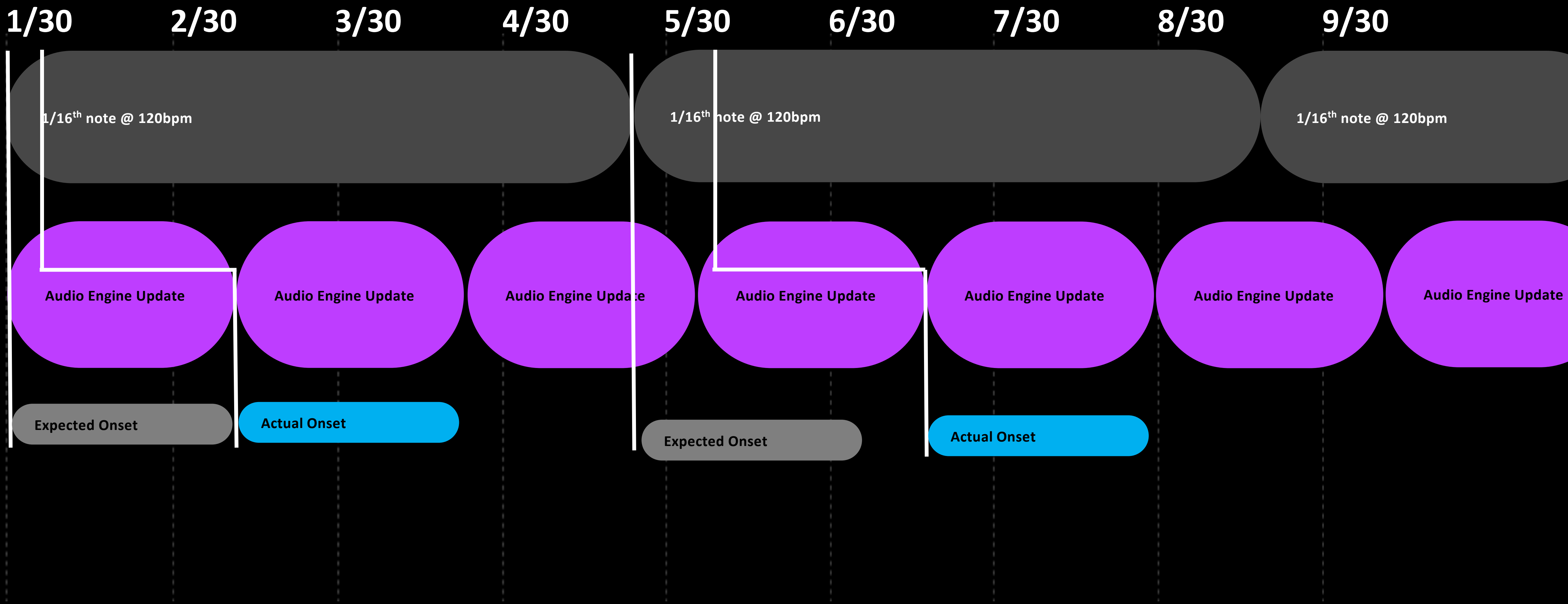
At N=2048 & SR=44.1kHz the audio engine “ticks” every 46.44ms



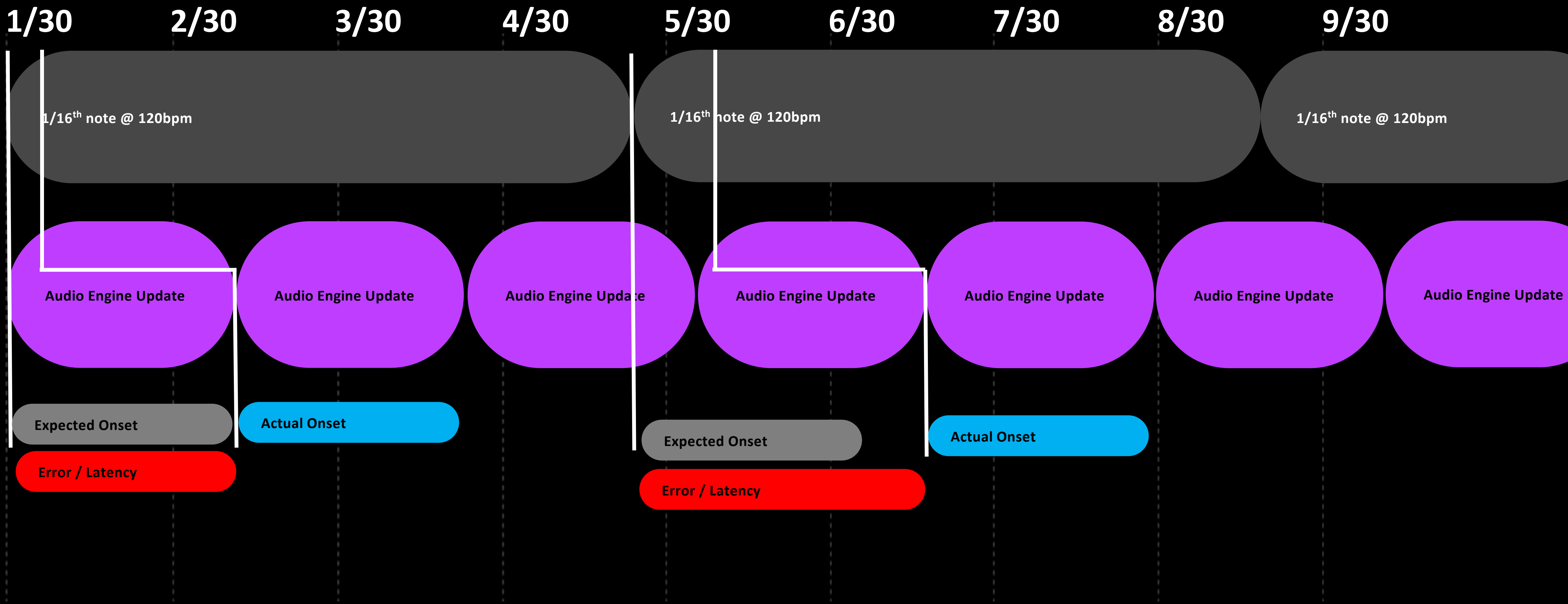
At N=2048 & SR=44.1kHz the audio engine “ticks” every 46.44ms



At N=2048 & SR=44.1kHz the audio engine “ticks” every 46.44ms

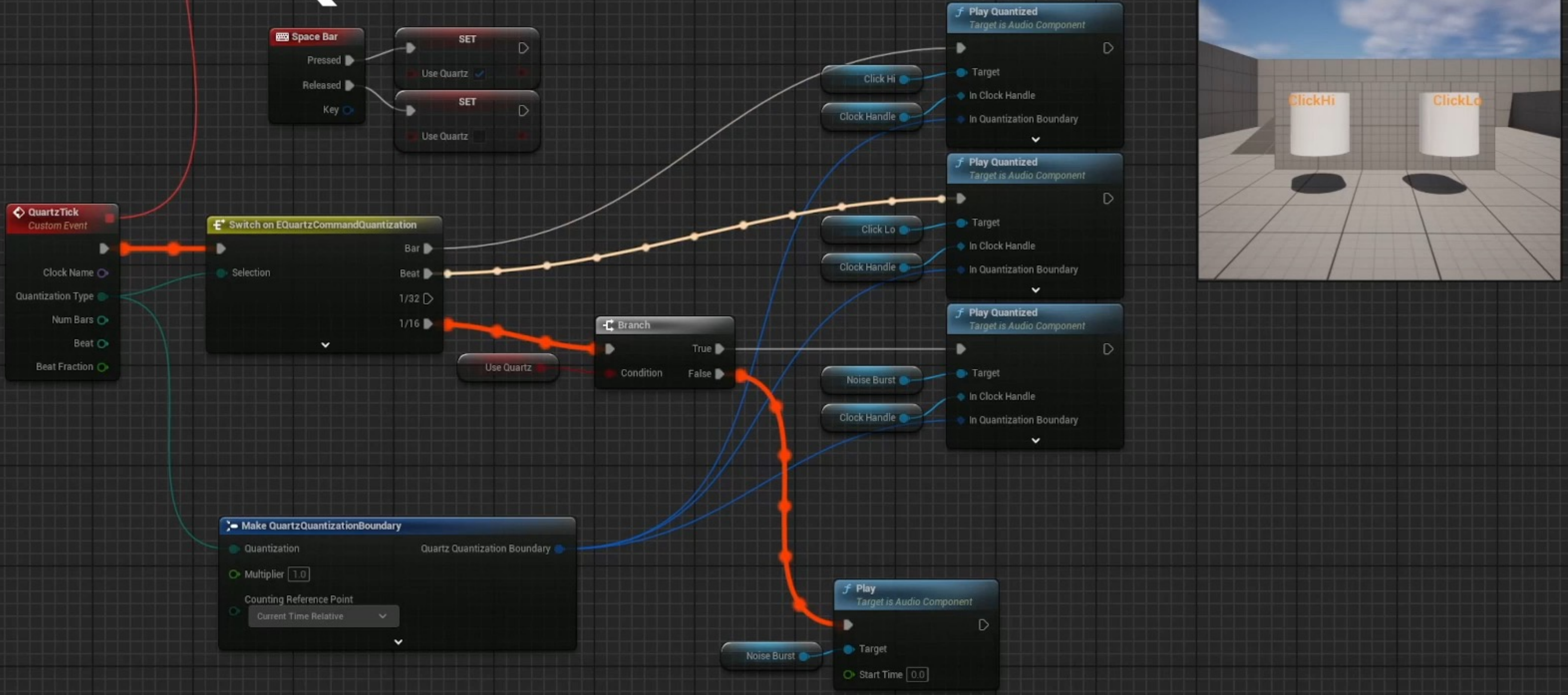


At N=2048 & SR=44.1kHz the audio engine “ticks” every 46.44ms

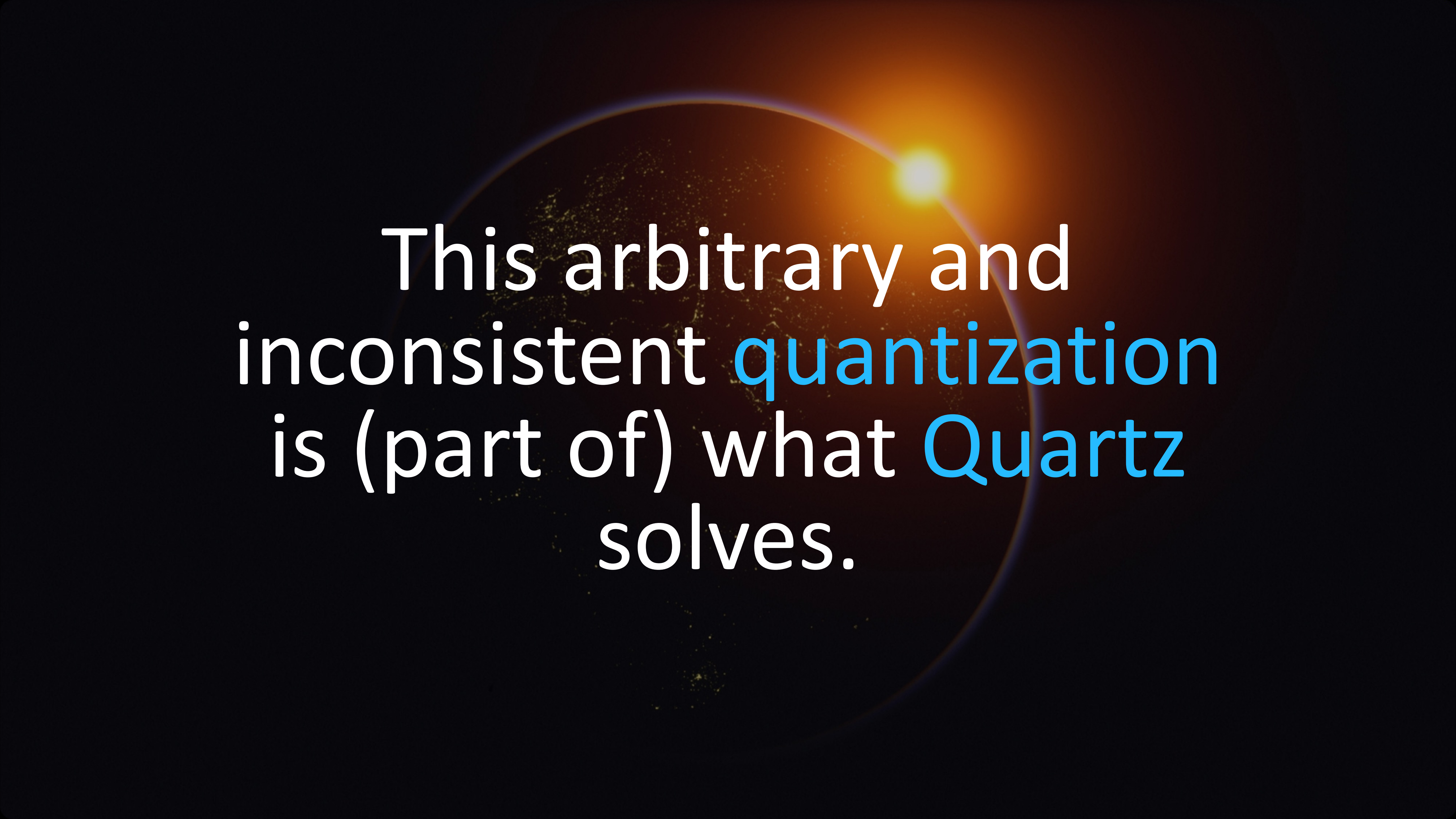


This error is VARIABLE

Without Quartz



A/B with 16th notes (99bpm)



This arbitrary and inconsistent quantization is (part of) what Quartz solves.

Quartz is a scheduler:

Create and control clocks.

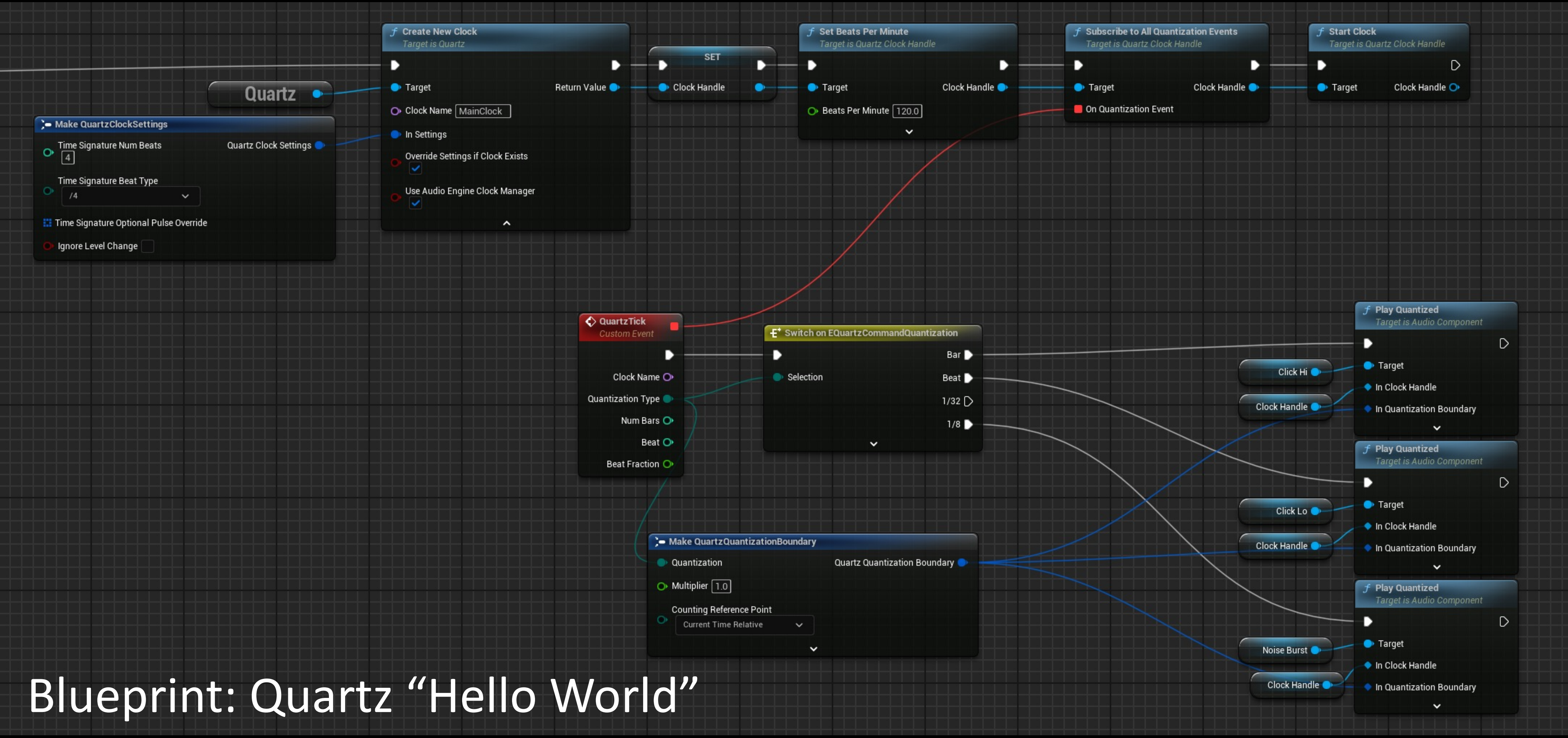
These clocks live on the Audio Engine (Audio Render Thread) and can be controlled from the game thread.

Schedule commands like "PlayQuantized()"

Quartz calculates how many audio frames until the command should execute.

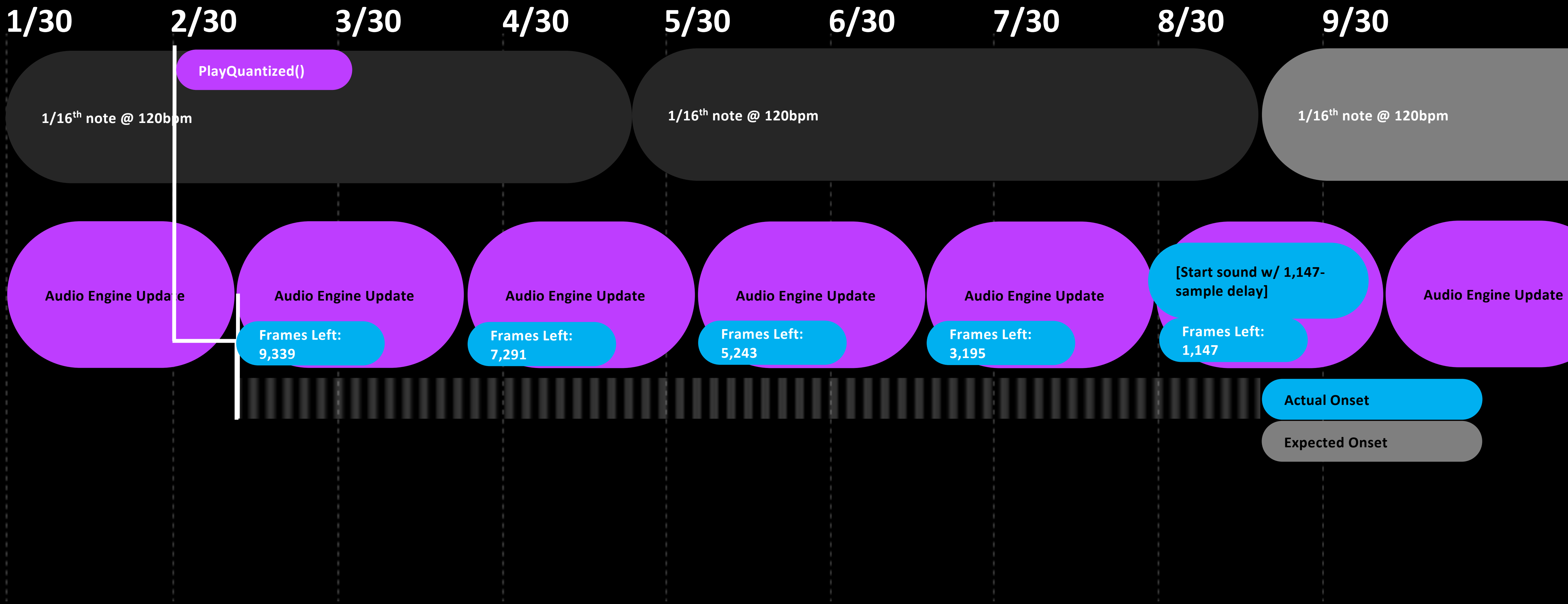
Game visuals in sync with audio.

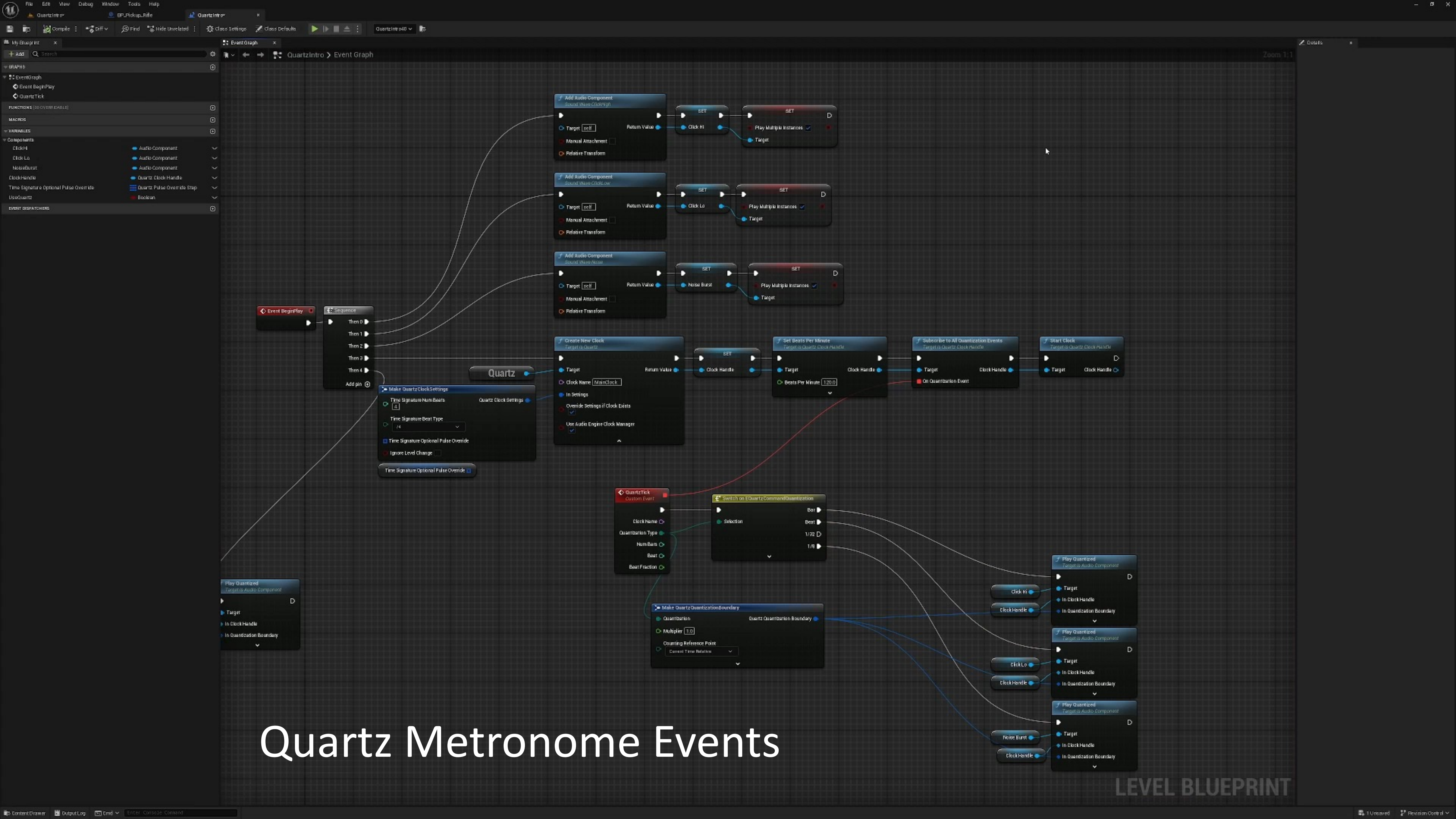
Game logic can piggyback on this scheduling worrying about things like BPM changes, voice limits, etc. to let audio trigger VFX & Gameplay.



Blueprint: Quartz “Hello World”

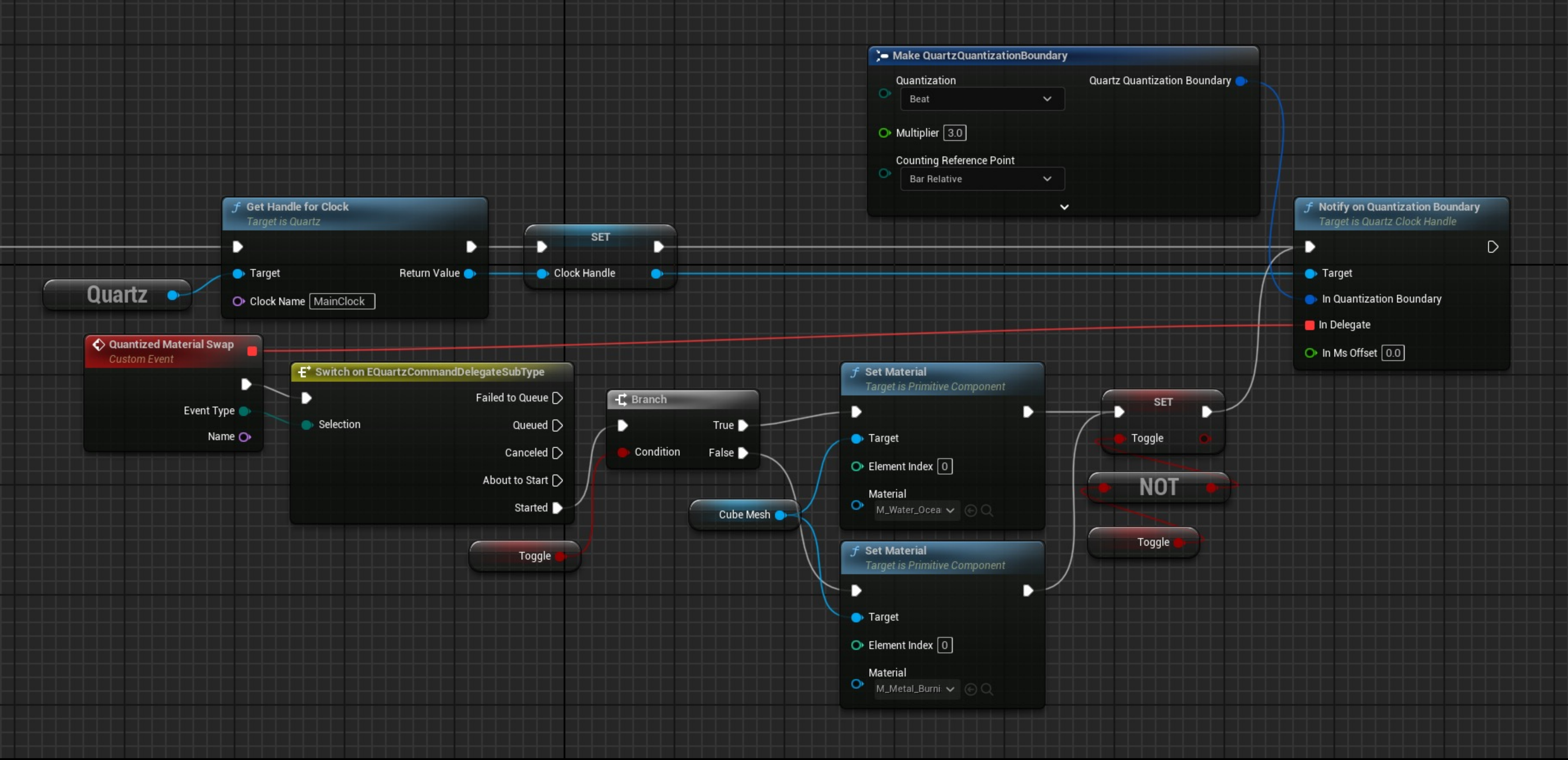
How PlayQuantized() works:
@ N=2048 & SR=44.1kHz



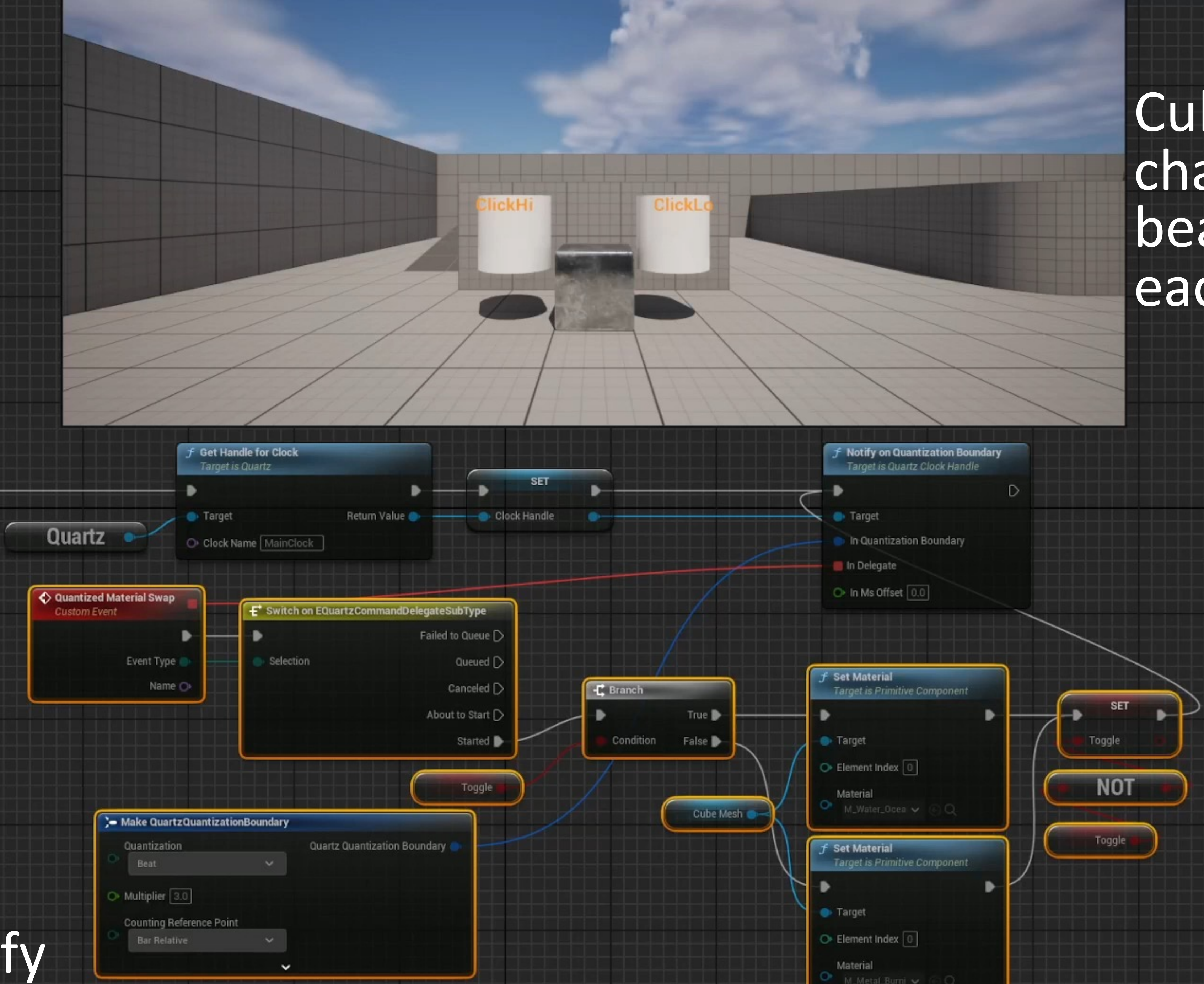


Quartz Metronome Events

LEVEL BLUEPRINT

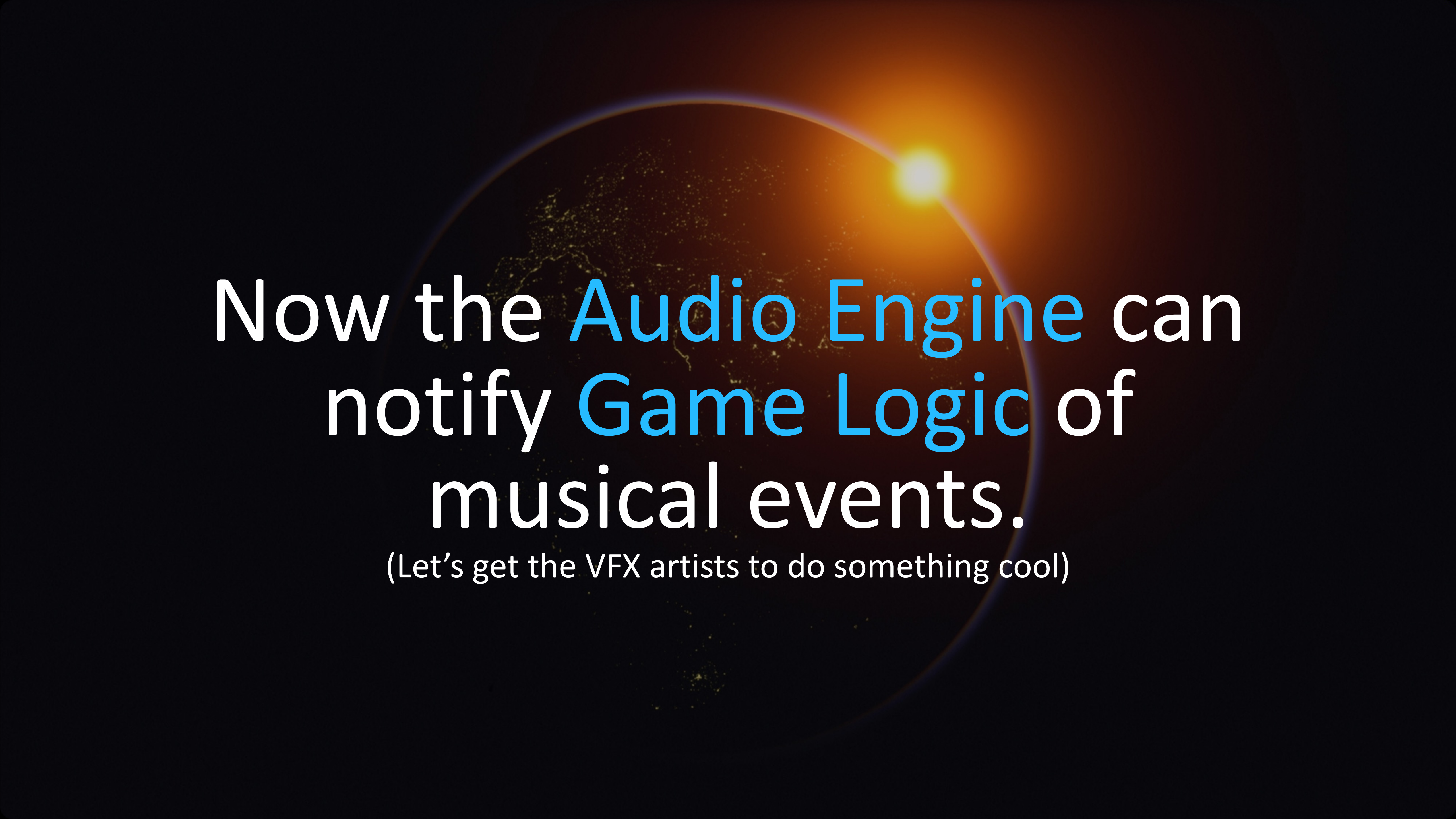


Quartz Notify



Cube changes on beat 3 of each bar

Quartz Notify

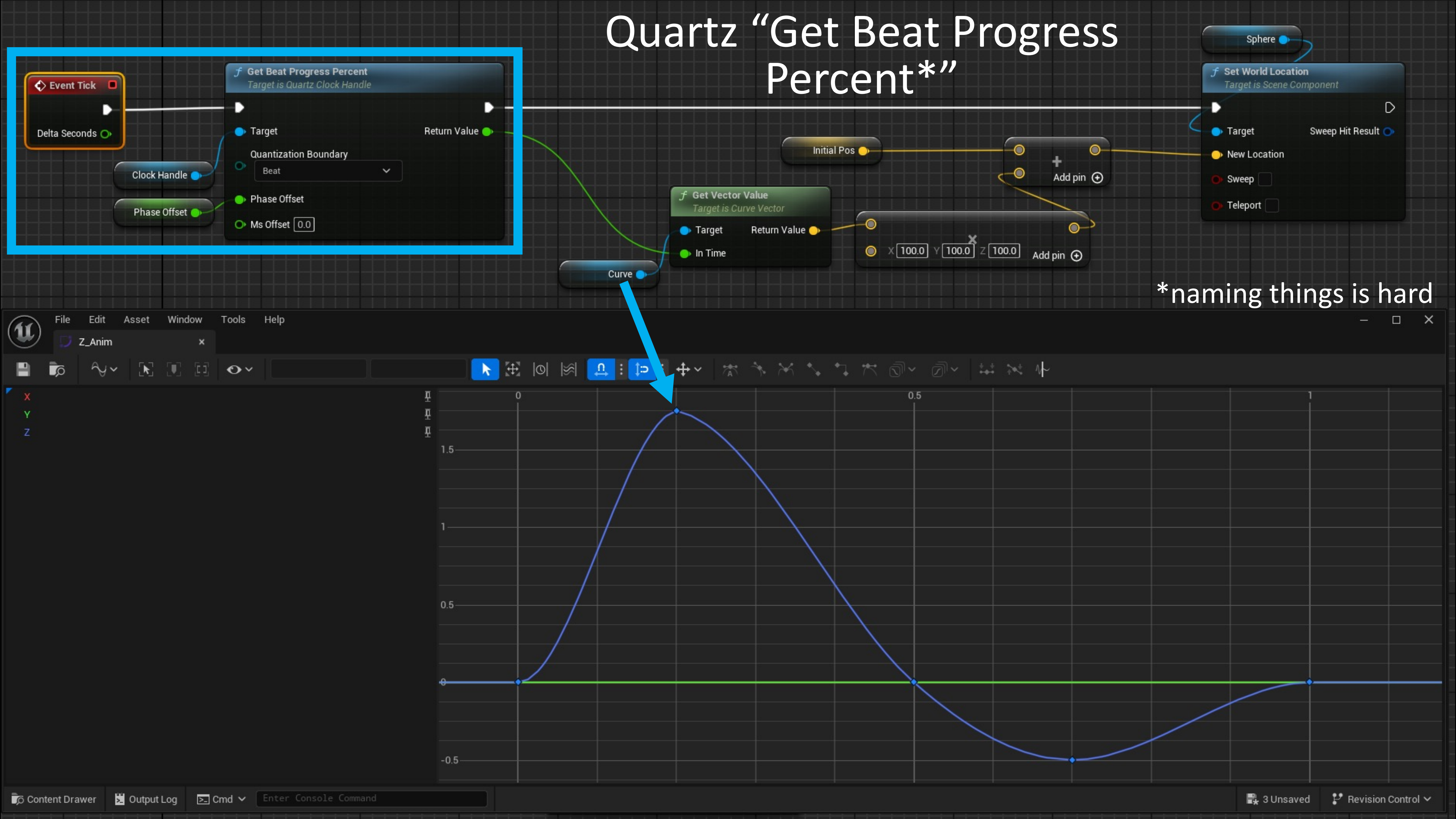


Now the Audio Engine can notify Game Logic of musical events.

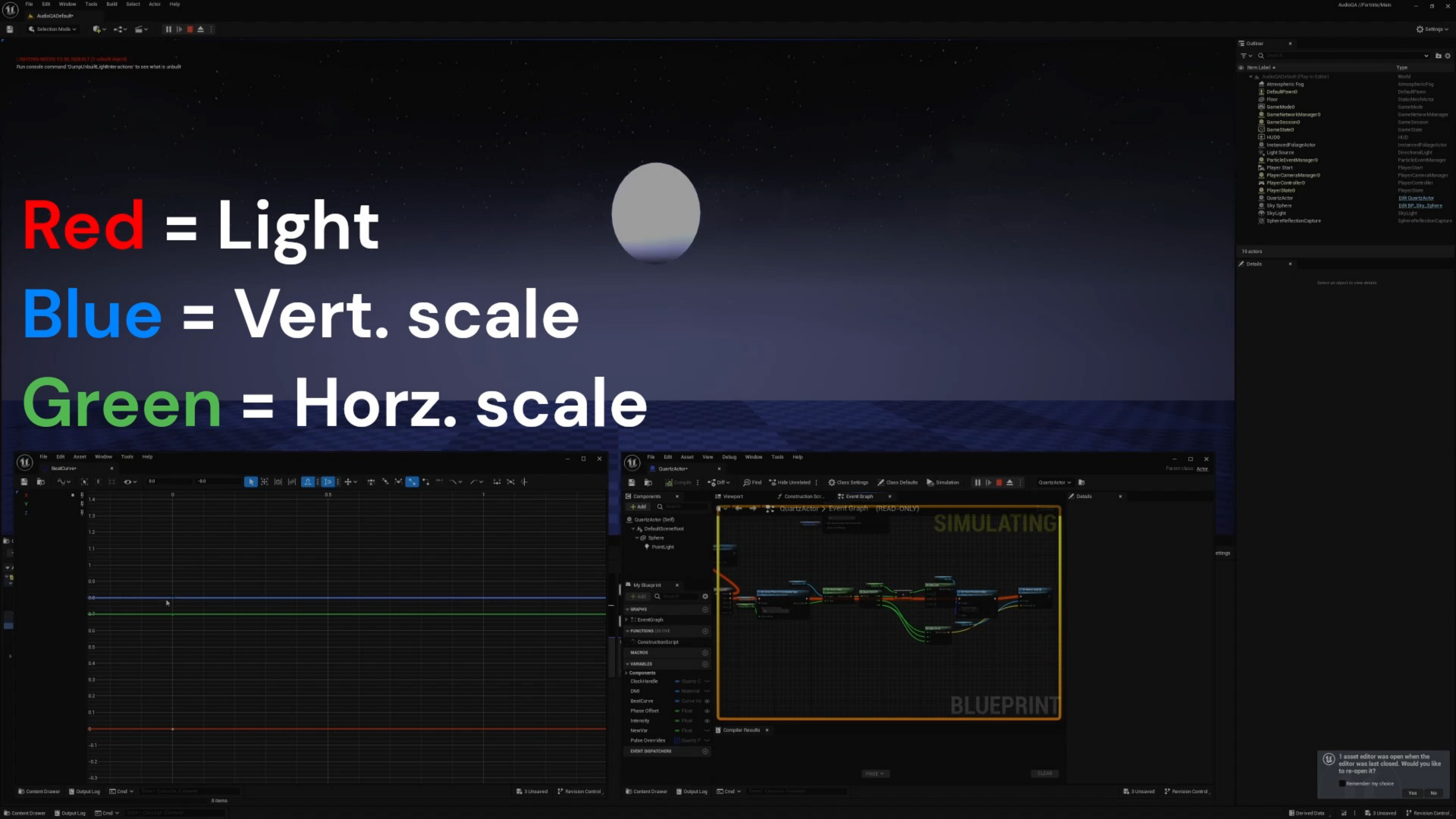
(Let’s get the VFX artists to do something cool)

NORMALIZED FLOAT CURVES
VFX ARTISTS
"MUSIC WORDS"

Quartz "Get Beat Progress Percent*"



*naming things is hard



LIGHTING NEEDS TO BE REBUILT (1 unbuilt object)
Run console command 'DumpUnbuiltLightInteractions' to see what is unbuilt

Red = Light

Blue = Vert. scale

Green = Horz. scale

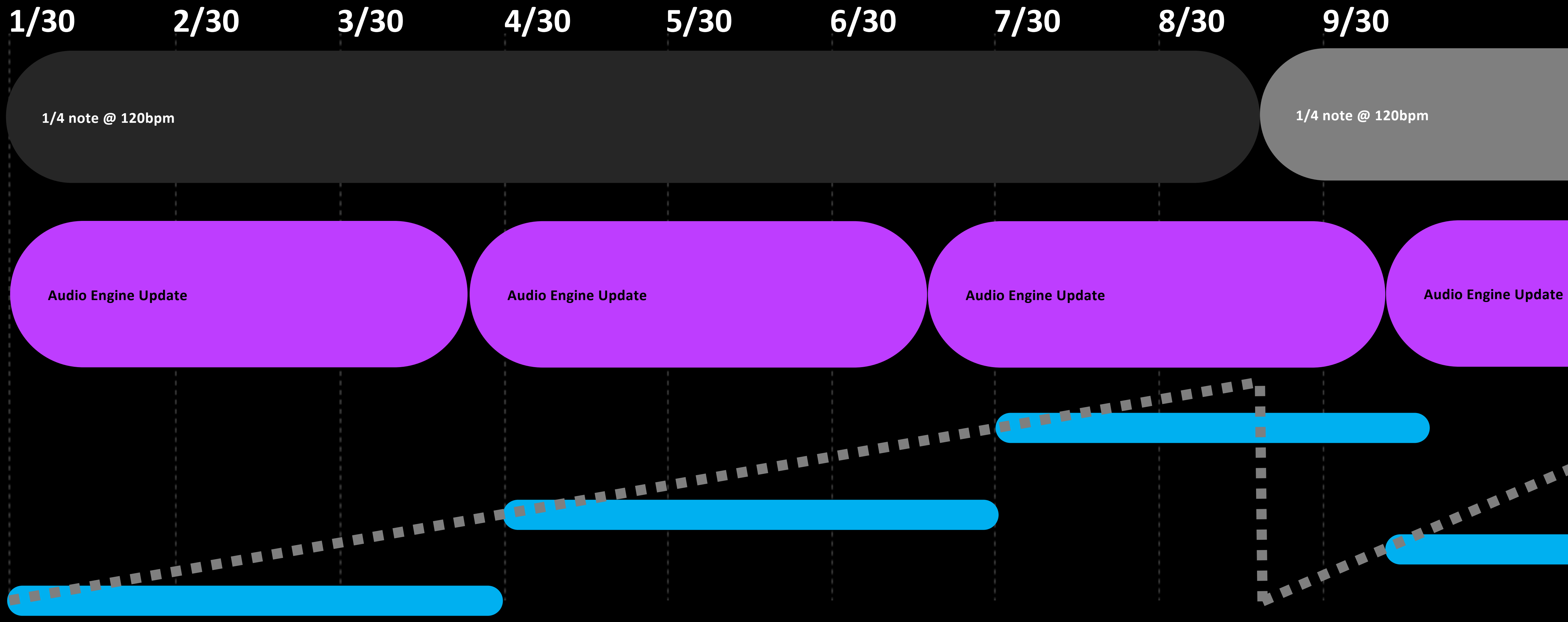
1 asset editor was open when the editor was last closed. Would you like to re-open it?

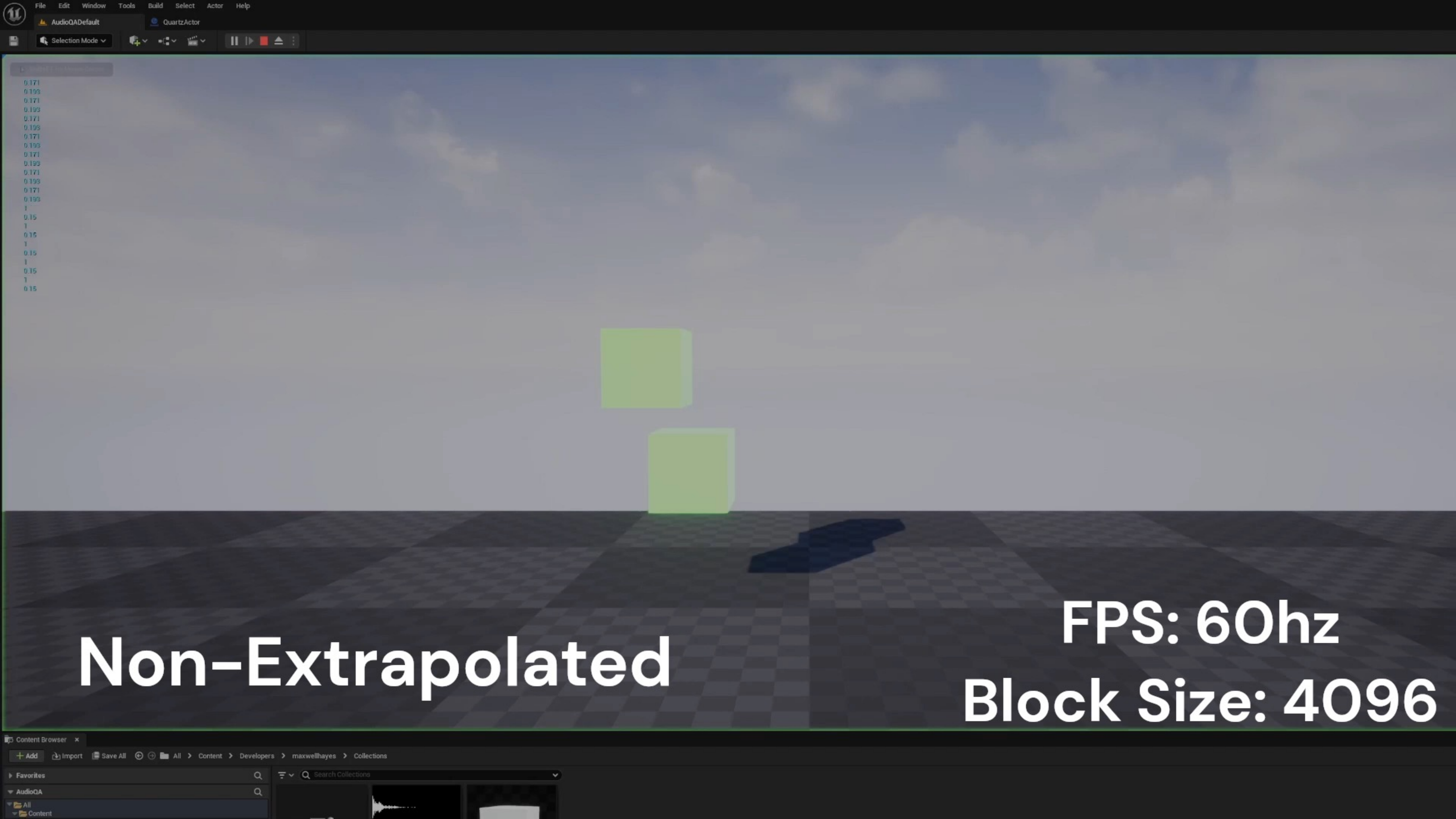
Obstacle Number 3:

At large buffer sizes, the audio engine's update rate becomes slower than the game's frame rate.

i.e. The game wants a smooth ramp but is getting the same value multiple frames in a row. (S&H)

At N=4096 & SR=44.1kHz the audio engine “ticks” every 92.88ms





```
0.171
0.193
0.171
0.193
0.171
0.193
0.171
0.193
0.171
0.193
0.171
0.193
0.171
0.193
1
0.15
1
0.15
1
0.15
1
0.15
1
0.15
```

Non-Extrapolated

FPS: 60hz

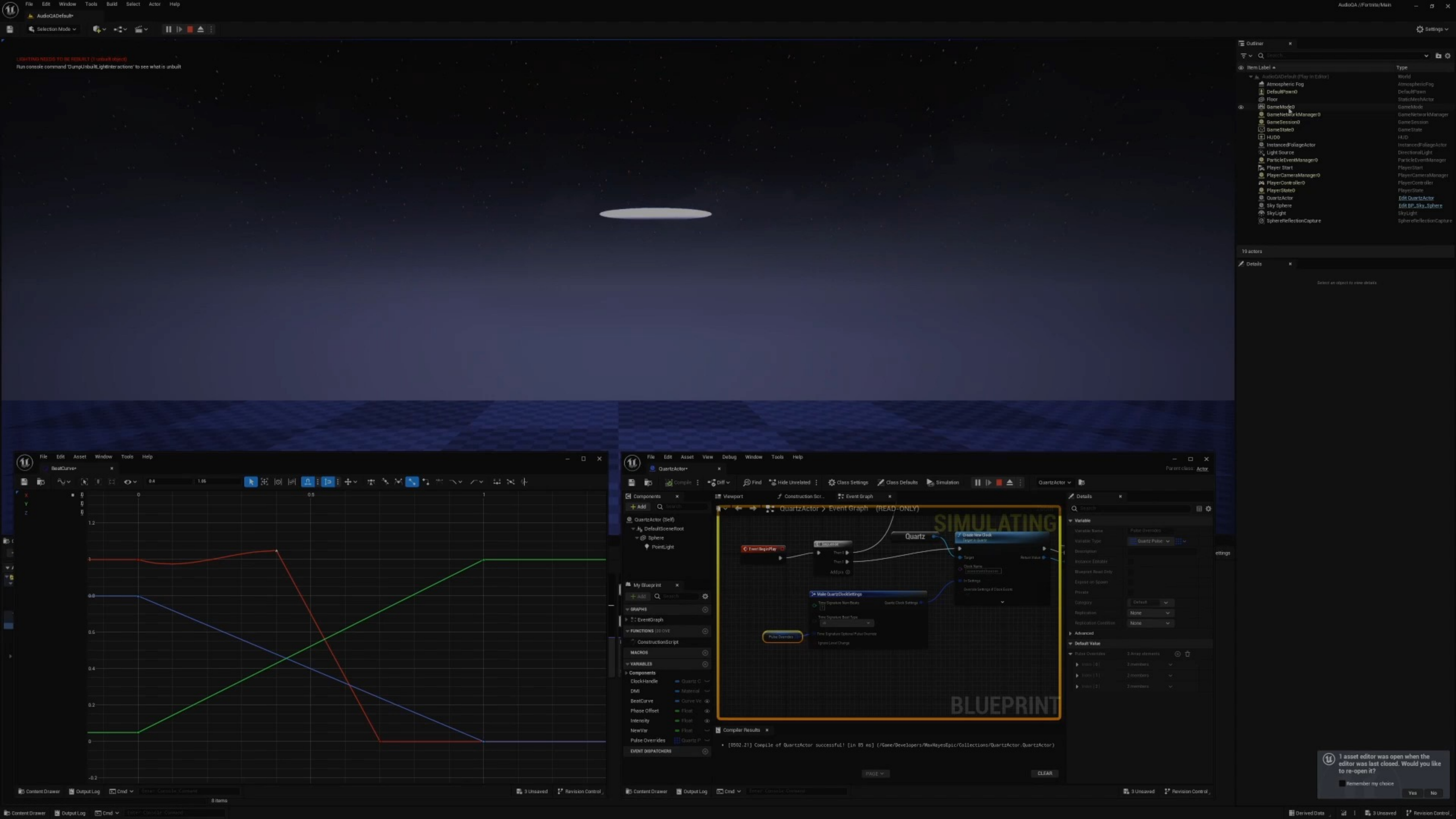
Block Size: 4096

Obstacle Number 4:

Light travels faster than sound.

Human error tolerance reflects this.

(Things we see cause things we hear)



LIGHTING NEEDS TO BE REBUILT (1 unbuilt object)
Run console command 'DumpUnbuiltLightInteractions' to see what is unbuilt

Item Label	Type
Atmospheric Fog	AtmosphericFog
DefaultPawn0	DefaultPawn
Floor	StaticMeshActor
GameMode0	GameMode
GameNetworkManager0	GameNetworkManager
GameSession0	GameSession
GameState0	GameState
HUD0	HUD
InstancedFoliageActor	InstancedFoliageActor
Light Source	DirectionalLight
ParticleEventManager0	ParticleEventManager
Player Start	PlayerStart
PlayerCameraManager0	PlayerCameraManager
PlayerController0	PlayerController
PlayerState0	PlayerState
QuartzActor	Edit QuartzActor
Sky Sphere	Edit BP_Sky_Sphere
SkyLight	SkyLight
SphereReflectionCapture	SphereReflectionCapture

```
• [0502.21] Compile of QuartzActor successful! [in 85 ms] (/Game/Developers/MaxHayesEpic/Collections/QuartzActor.QuartzActor)
```

1 asset editor was open when the editor was last closed. Would you like to re-open it?

Remember my choice

Yes No

In summary:

A bit about games, game engines, and game audio.
Talked about the mechanisms and undesired quantization between the Game Thread and Audio Render Thread.

How to schedule from game logic to audio renderer.
Stepped through how Quartz avoids these issues and allows to schedule ahead with single-sample accuracy.

How to let audio state drive gameplay and VFX.
Covered some pitfalls with audio engine state driving gameplay and VFX. And how Quartz approaches these issues as well.

Quartz in the wild

- Fortnite Infinite Downtime Music System
- Deadmau5 prototyping
- Mix Universe
- BlasterBeat

Fortnite: 10+ hours Procedural Downtime Music

Thank you!

Questions?

@MacksHazeAudio