Wwise Tutorials

If you're familiar enough with the "Wwise: Getting Started" chapter, it's time to learn ways this tool can be used. For each tutorial, the process will be covered for multiple game engines. If you see one missing, be the person to add a how-to, if ya know how!

- Posting Sounds on Collision
- Creating Audio Reactive Objects
- Using States

Posting Sounds on Collision

> This tutorial will show how to post sounds upon a collider experiencing collision, both for Unity and Unreal (5.1).

UNREAL (5.1)

Collision can happen in many different ways: The harder you kick a soccer ball, the louder the impact should sound, and vice versa. Using Unreal's Physics System and Wwise RTPCs, we can create a collision detection system that will only post a sound event on collision if the impact reaches a certain threshold.

Linked here is a web tutorial on Audio Kinetic's website for a reiterated breakdown on the process.

a.PNG

In Wwise, you need to create an RTPC in the Game Syncs tab of your Project Explorer window. I named mine *Collision_Velocity* and upon making it, went to the sound SFX object in my audio tab that I am using for our collision scenario: *Cube_Collision*.

By clicking this SFX object and navigating to RTPC tab in the Property Editor, I can add effects onto a Y Axis and on the X axis, define the RTPC these effects should be tied to. I used two effects, Lo-pass and Volume, and tied both of these to my *Collision_Velocity* RTPC.

> *If you right click the line curves, you can change the type of curve an effect undergoes!*

b.PNG

When your sound if configured how you want it and the RTPC is created and connected to the SFX object, you need to create a Wwise Event to drop your sound into, enabling us to post this event in Unreal upon regenerating our sound bank(s). I named my event *Impact_Cube*, dragging and dropping the *Cube_Collision* SFX object into the empty space near the top.

c.PNG

Next, we'll create a blueprint for posting the sound upon collision with our desired object. In the scenario above, it will be one of these blue cubes. If you want to quickly test out this Wwise configuration, I am using the 3rd Person Demo scene that Unreal provides its users for free as a starting template for project creation.

d.PNG

In the right view, I have the details panel open. By clicking the blueprint icon in the top right, the window in the middle opens up with options for creating a blueprint. We want to create a new subclass and for this blueprint to be tied to the StaticMeshActor, since the mesh is what the player can collide with.

For clarity's sake in a developing project, I prefer to start the names of blueprints or scripts that are related to Wwise with "Wwise", alowing me to easily distinguish which components are for audio purposes. I named my Blueprint *Wwise_CubeCollision.* The next thing to do is open the details panel and check for the following options to be selected:

- Physics > **Simulate Physics**

- Collision > **Simulation Generates Hit Events**

e.PNG

If you're in the view above and do not see the components tab, go to the top toolbar and Click **Window --> Components** to open the Components tab.

f.PNG

Right click the Static Mesh Component in the Components View and click **Add Event --> On Component Hit**. This will open up the Blueprint Editor, showing you the added component.

image.png

By right-clicking empty spaces in this view, adding and connecting components to one another, you need to configure the setup above. If you're having trouble finding certain components, here are some tips:

- The grey box, a greater than or equal to if statement, is found by typing ">=" in your search for a new component to add.

- If you cannot find your RTPC in the Wwise drop down, try opening the Picker Tab window, then dragging and dropping your RTPC from the Game Parameters Folder. If you do not see the RTPC in your Game Parameters folder in Unity, save your project in Wwise, make sure the event was dragged into the sound bank upon generating it in the sound banks view, then save your project again and allow Unreal to reparse before checking to see if the RTPC is now showing up.

- You can use the drag and drop method from the Wwise picker tab into any component asking for a reference to an object in Wwise that is needed. This goes for the Post Event Component!

To Summarize what this Blueprint is doing, I will be echoing what the linked website at the beginning tells us:

1.) It detects when the object collides with another.

2.) It sets the value of the RTPC based on the Vector Length of the Component Velocity, and ensures that the value exceeds a minimum threshold of 40, which ensures that minor contacts do not cause impact sounds.

3.) It posts the Event to Wwise with the RTPC value, so that Wwise plays the impact sound at the appropriate volume.

g.PNG

The blueprint is complete and you can now click Compile in the top left.

image.png

A more fun way to test the collision effect is by giving you the ability to spawn cubes during playmode! Here are some tips for finding these components during your search.

You can reference a quick 1 minute video tutorial, linked here.

- The grey x, our multiplication component, can be found by typing Multiply and selecting *Multiply* under the Operators section. You can find the addition component by typing '+'.

- The dot in the grey box is a converter. It will convert a value from 'this' to 'that' and there are many types of conversions that can be made. To know what type of conversion you're doing, the pins are colored and represent a type of value. In this case, the conversion is from a Vector to a Transform value. You can make this box appear by dragging the orange pin next to 'Spawn Transform' to the upper right yellow pin in the + node, or you can look up 'Vector to Transform' and should find the component.

- F is a *keyboard event*, so when you type F in your search, look for the keyboard event section and select the one that says 'F'.

- You may have noticed the pin in the bottom left of the x component is green, whereas yours starts off yellow. You can convert this pin by right clicking it and selecting **Convert --> Float (Double-Precision)**.

- "SpawnActor Wwise Object" is not what you should type, when searching for this component. You only need SpawnActor because what comes next is the name of the gameObject that will be spawned.

Creating Audio Reactive Objects

> It is assumed you know how to obtain and integrate Wwise in this tutorial. If you do not, refer to the chapters in the "Wwise: Getting Started" chapter.

Multiple ways exist to influence an object's behavior from audio events posted by Wwise. We are going to explore some ways of configuring audio-reactive objects in Unity. Remember, no matter the game engine, this setup relies on tools in Wwise and Wwise-type references in scripting, meaning these concepts should be able to translate into other game engines pretty seamlessly.

Color Changing Objects

Size Changing Objects

Topic Related Blog LINK

Objects Moving Rhythmically

Wwise 301 | Callbacks Lesson

>>Picture of AkEvent Component, highlighting "Use Callback" checkbox<<

Rather than post sound events in an AkEvent component, Wwise has a checkbox in this component called "Use Callback". This feature relies on your use of Callback Flags in a Wwise project, which can be assigned to Music Segments. These flags serve as notifications for when to call a function in your game engine.

What we are going to do is assign callback flags to a music segment, then use those in an AkEvent component that calls forth a custom script for affecting an object" 's scale. By Doing so, we can make an object look as if it's breathing in rhythm with a song.

Doppler Effect

Related Article LINK

A car approaches you, rising in pitch as it nears. When it passes, it begins to fade out and its mechanical whirr lowers in pitch. We want to recreate this effect, known as the Doppler effect. To do so, we'll need to configure RTPCs in Wwise, and make a custom script referencing them

Using States

> Use cases for States in both Wwise and Unreal will be shown on this page.

States are a type of game sync in Wwise that are useful for applying immediate or gradual transitions in your soundscape. This tutorial will show the process of setting up States in Wwise, applying them to your sound, and implementing the effect in your game engine.

Unreal

Setting Up In Wwise

VR_Slider.PNG

Here is the scenario: In the French75 scene, you have a bunch of ambient sounds that complement the void-like setting and are echoes of the past. When the player has loaded the shell, pulls the cable, and fires, you want all ambient sounds to immediately duck out so that the firing sound is fully embraced. As the firing sound fades off, the ambient sounds can slowly creep back in. To achieve this effect, we are going to use States, so let's start in Wwise and see how this is set up.

Ambiances.PNG

In my Wwise project, I have my Ambiances as children objects under the INTRO_LEVEL Folder, for organizational purposes. In the first scene of the French75 experience, this is where the sounds first come into play, which are later reused in the French75 scene. I decided to separate sounds specific to the French75 scene in a different folder.

Ambient Sounds.PNG

The only sounds I am worried about are my ambient sounds, which have been color-coded yellow. These are what will duck out when the cannon is fired, so we're going to set up states for them in the Game Syncs tab.

States.PNG

Under the States hierarchy in the Game Syncs tab, I selected the default work Unit and created a State Group called "Cannon_FireAmbience". The state group object is what will hold our states, I have created two states: **Fired**, and **Not_Fired**.

property editor.PNG

The State Group object is what you will select in order to set up transition times from one state to the other. My vision for this state group's usage is that transitioning from Not_Fired to Fired should immediately fade out the audio, whereas, after firing, the Not_Fired state will be reapplied and provide a short fade-in of the ambient sounds.

I set up the aimed-for transitioning system by clicking insert at the beginning of the property editor and defining my states under the **From** and **To** properties.

The transitions can now happen but as of right now, have not been applied to any of the sound objects, nor have we said what should change during the transition: Volume? Lo Pass? Pitch?

Bus State.PNG

Rather than apply the state transition to however many Ambient parent objects would need it, I can filter my ambient sounds to an audio bus so that I can apply states to that Audio bus and it collectively affects all the ambient sounds I have routed to it.

You can create an Audio Bus in the Master-Mixer-Hierarchy in the property editor. Click the default work unit and then select the icon that identifies as Audio Bus. (Shortcut: Ctrl + Shift + Alt + B)

I named my Audio Bus "Ambience_Compressor" This is because I have a compressor effect being applied to all sounds that pass through this bus, and I wanted to also identify that any sounds filtered to this bus are solely ambient sounds.

Bus routing.PNG

To route your sounds to a different bus, click the object you want to route and in the Property Editor's General Settings tab, go to the Output Bus section and find your desired bus in the directory pulled up by the three dots button. I have my ambient sounds linked to an Actor Mixer object as the parent. Because of this, I am able to route all sounds at once to the Ambience_Compressor bus, simply by pulling up the General Settings for the Actor Mixer object and routing that to the Ambience_Compressor Bus. (I did this for 3D_Weaponry and Ambience)

Wthatever.PNG

Whatever you decide to do, click the object you need to apply the State transitions into. In the Property Editor, click the States tab. Click **Add State Group>>** and find the State Group you want and select it.

It will appear in the view with the states that are connected to it. My sounds are being influenced by effects, so you might notice that I have checked the **Bypass Effects** checkbox, in order for the transition to do what I want. As for what I want, I have specified in the Fire State's properties that the volume should be lower and that a Lo Pass filter should be applied.

Under the **Change Occurs at** Property, I have specified that it should be Immediate. THIS DOES NOT MEAN AN IMMEDIATE CHANGE IN DYNAMICS, rather it means the state transition should happen immediately though the time we specified in the Game Syncs time will still be however

long you set it. For example, in my transition from Fired to Not_Fired, I told the State Group to take 7 seconds to transition into Not_Fired. When the state is called in the game it will be immediate, but the change from Fired to Not_Fired will take seven seconds, dynamically speaking.

Test.PNG

You can test if your system works with the transport control view at the bottom of your project. Click the State Group Icon, which is the orange Icon in the image above, and the state group for the sound you selected and wish to test should appear. If it hasn't make sure you gave it a state group.

Now play your sound and in the dropdown menu that says none, select one of your states. The transition should begin immediately, and at the speed you gave it in your State Group's editor.

EVENTS.PNG

If you're happy with your configurations, go to the **Events** tab in the Property Editor. Create an event with whatever naming convention works for you. I have my ambiances as children objects under a separate work unit, for organization purposes.

Keep the view for your event open after selecting and navigating to the **Audio** tab to locate the sound you want applied to this event. Be sure not to click any audio objects during this process, or else the view you need to drag it into will disappear. Drag your audio object into the event view area and it should appear in the view with some information.

Alternatively, you can click the **Add>>** button at the bottom of the Event View, then find the sound object you want to add.

Event Viewer.PNG

Finally, go to **Layouts --> Soundbanks** , and find the soundbank you want to put your event into. Once selected, navigate to the **Event Viewer** in the bottom left and locate the Event(s) you want to drag and drop into your soundbank. Once you do that, it should appear in the **Hierarchy Inclusion** view.

Click **Generate All** in the top right, or you can do **Generate Checked** if you only want to update specific things. Save your project and you're now ready to move to Unreal!

Setting Up In Unreal