Lesson 1: Quick Start–From Silence to Sound

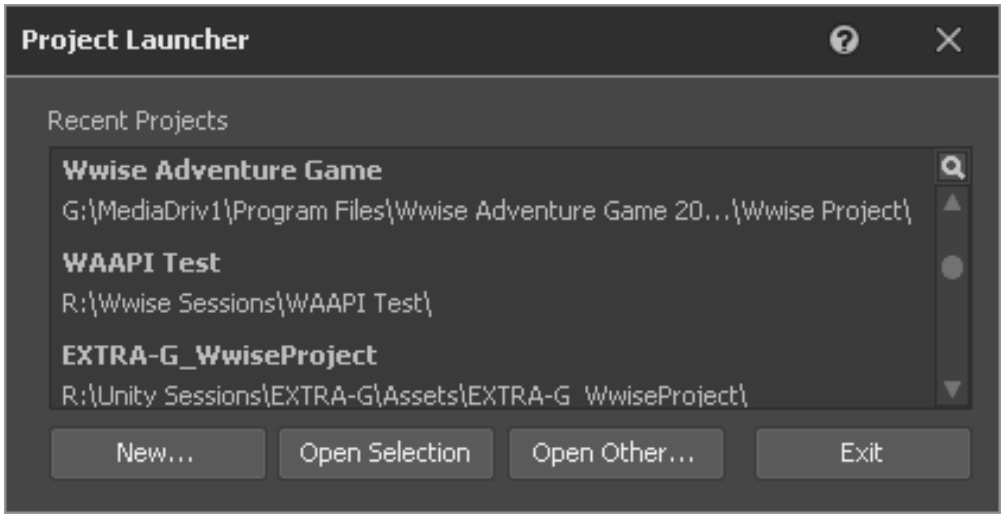
In the Installation of Wwise you installed and briefly played Cube, the game you'll use throughout this tutorial. At this point, Cube is a silent game and that's not very exciting. You'll soon fix this! By the end of this lesson you'll bring the game to a point where one of the games most noticeable visual animations, the firing of a shotgun, is supported by a shotgun blast sound.

Launch Wwise

You'll begin by creating a new Wwise project. Wwise projects are not a single file, but a folder with a number of sub folders that collectively contain the various resource files needed to carry out the instructions of how you've integrated sound into a game.

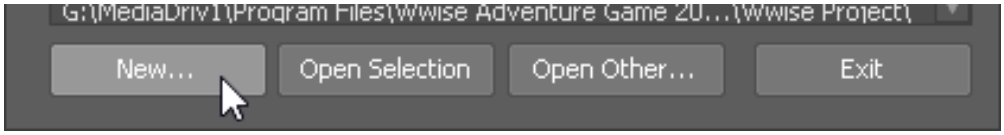
1. From the Wwise Launcher application, go to the Wwise tab and launch the Wwise application.

   The Wwise Project Launcher opens and all available Wwise projects are displayed.



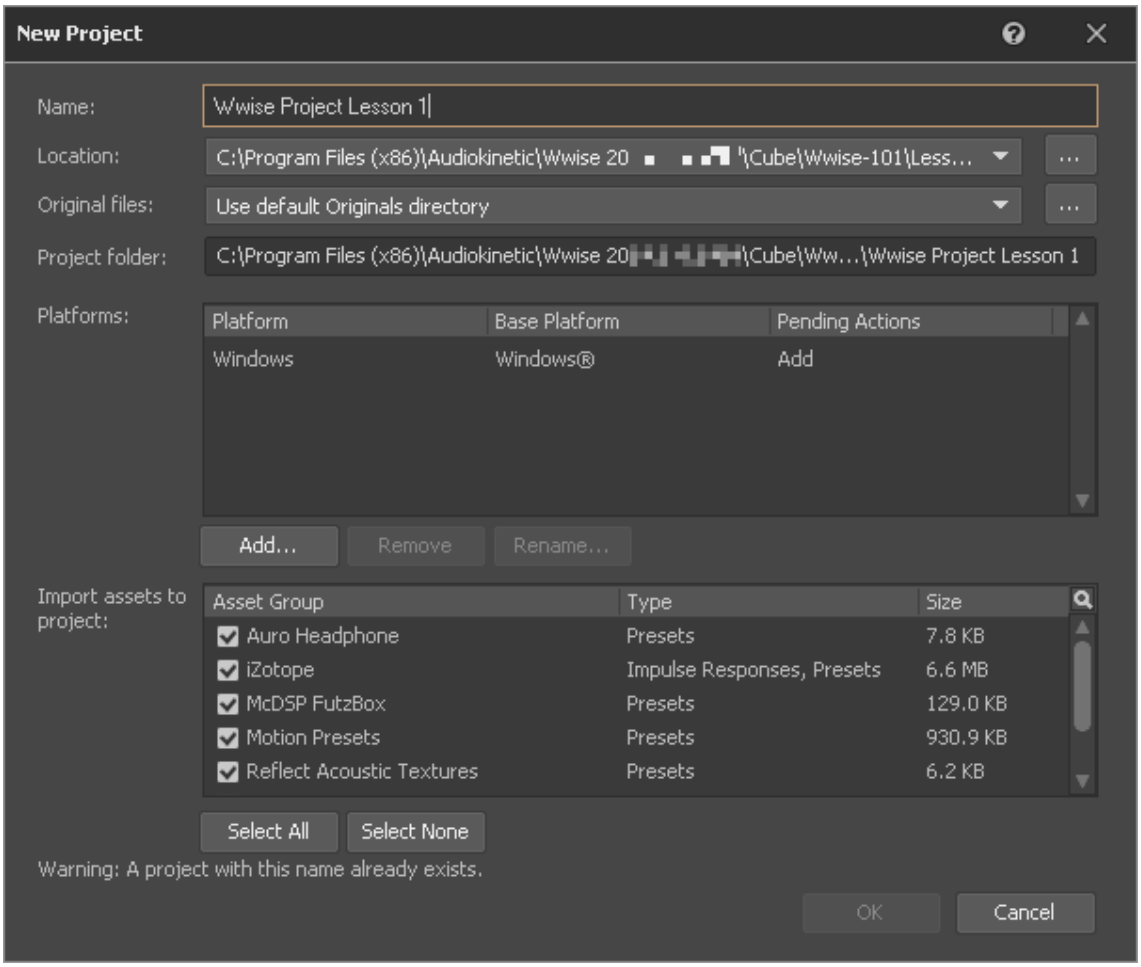
   The number of projects you see here will vary depending on what other projects you may have opened on your computer. Most likely you see nothing in this area at this point. In this case it makes no difference because you are going to be creating a completely new project for this lesson.

2. In the lower left corner of the Project Launcher, click **New**.



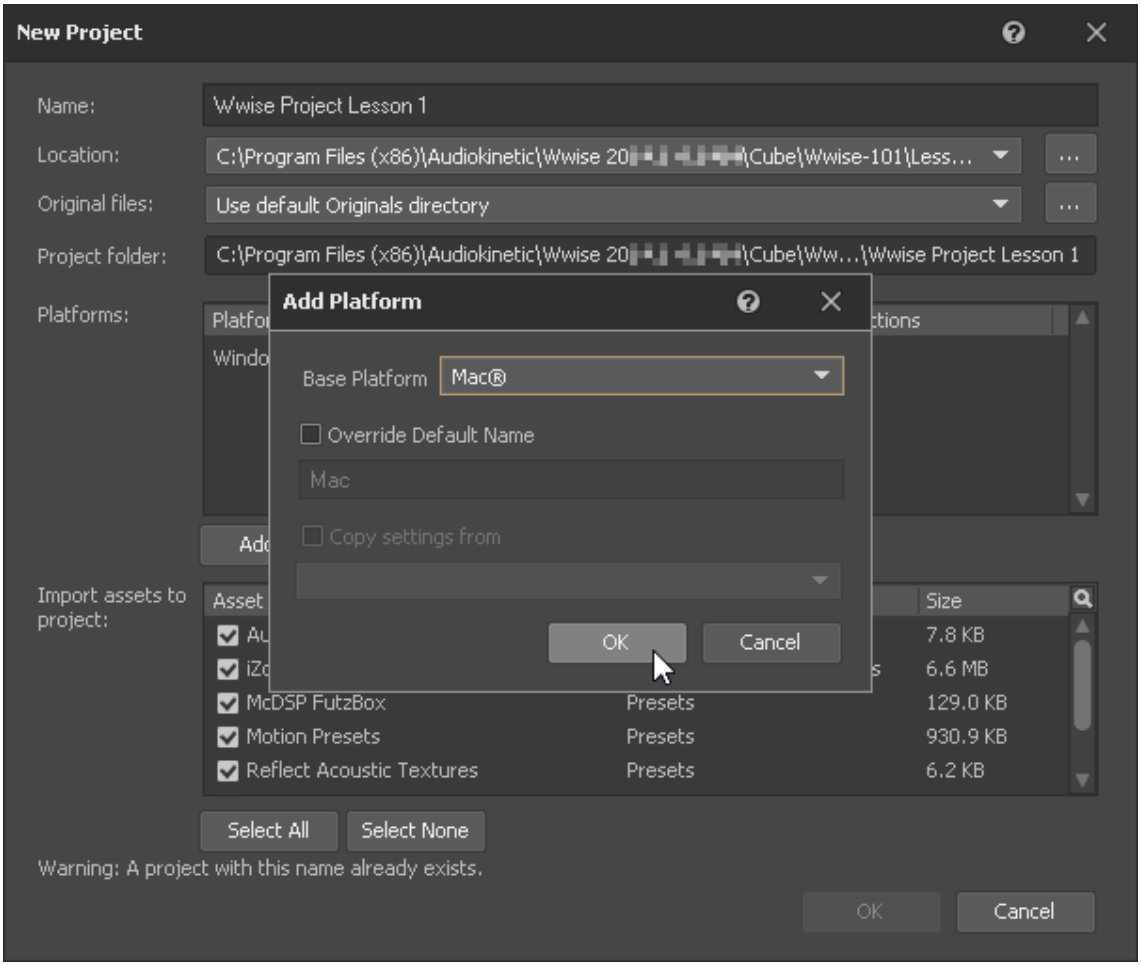
   A dialog window opens asking what you want to call your new project and the line below indicates where the the Wwise project folder that's about to be built will be located.

3. In the Name field, type *Wwise Project Lesson 1*. Click the [...] button to the right of the location field, navigate to your Wwise-101 folder and select the Lesson 1 folder.



Mac users: Upon selection, the path in the Location field will be displayed as `Z:\Documents\WwiseProject\Wwise Lessons\Lesson 1\`. The Z: path designation occurs because Wwise for Mac runs within a PC emulator and it uses this letter as a drive designation.

4. In the Platforms panel, click *Add...*. The Add Platform dialog pops up. Select **Mac®** as a Base Platform.

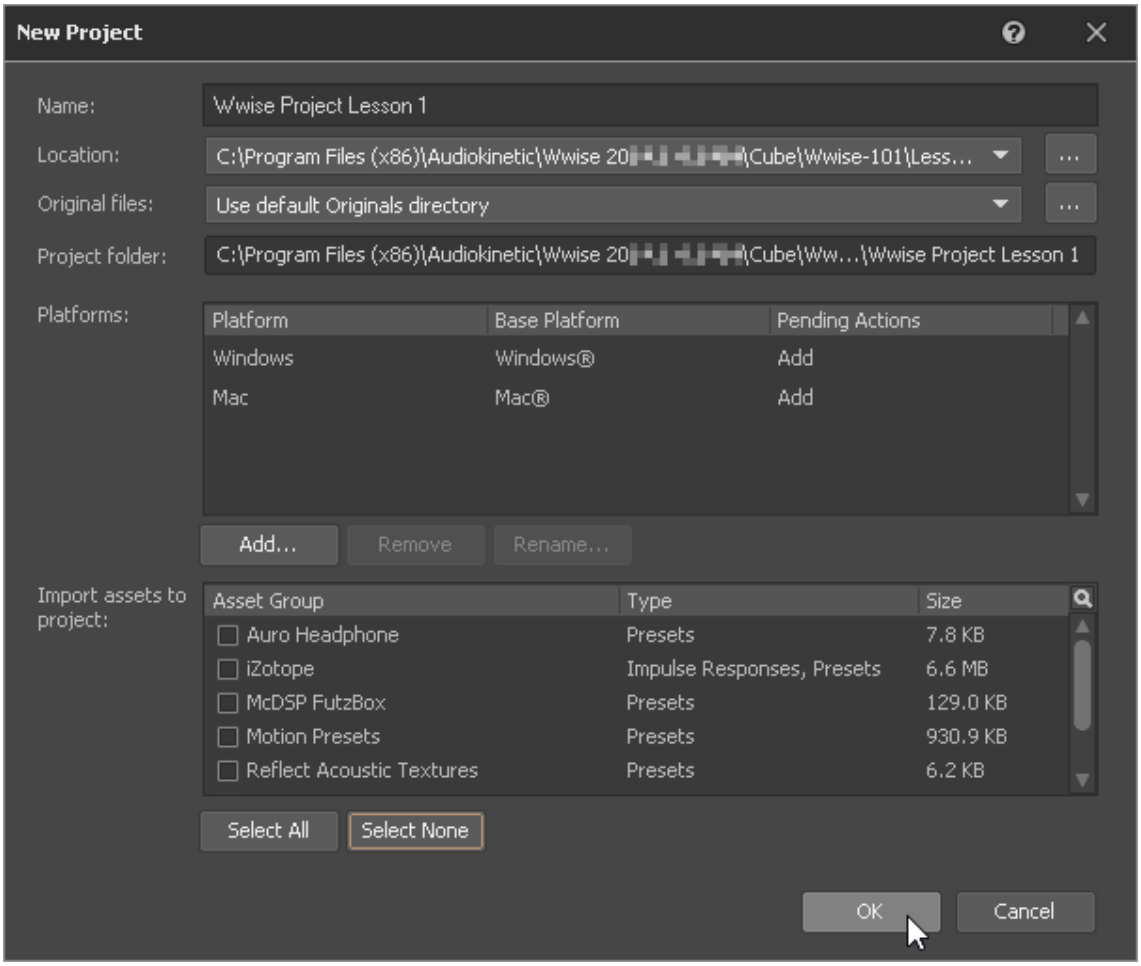


5. Click **OK** in the Add Platform dialog,

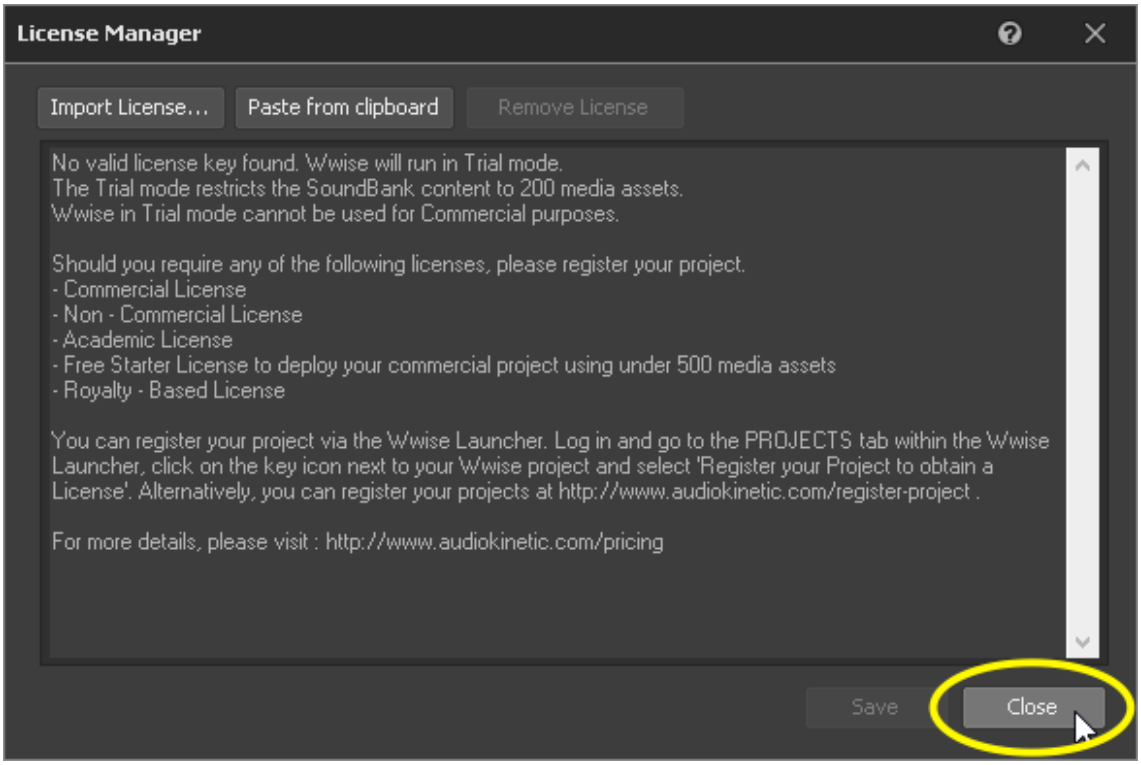
   You'll see that this project will now support building your sound design for both Mac and Windows versions of the Cube game.

   In the lower section of the window there are checkboxes where you can choose to import necessary assets for some of the optional plug-ins or features within Wwise. These are unnecessary for the Cube game, so you won't be importing those assets.

6. In the Import assets to project area, click the **Select None** button and then click **OK** to create the Wwise project.

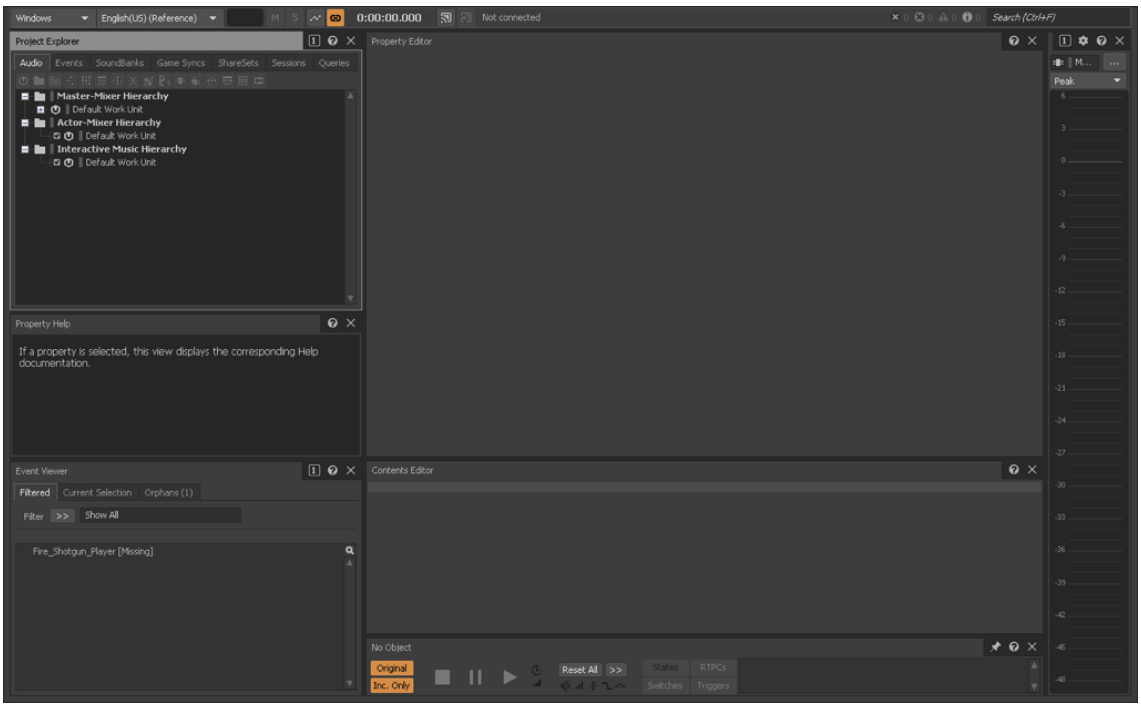


There's a brief pause while Wwise builds your new project folder. When complete, the Wwise License Manager window opens and explains that Wwise is running in its trial version. The trial version of Wwise is identical to the fully licensed version with the exception that there is a maximum of 200 media items across all SoundBanks. For more information on the fully licensed version, please visit the frequently asked questions page on licensing here: https://www.audiokinetic.com/licensing/faq/ . You'll learn more about SoundBanks later in this lesson, but be assured that the material provided with the trial version is more than sufficient for you to gain an understanding of Wwise.



7. In the License Manager window, click **Close**.

   You now see the Wwise software interface.



Depending on your screen size and resolution, it may look slightly different, but across the top you'll see a main menu bar, with toolbars below. These are static and will always be visible. Below you'll see multiple areas called views. Each view provides functionality that can be used to visualize and manipulate how sound is integrated into your game. There are around forty different views available within Wwise. This may seem a bit overwhelming, but it is not imperative that you know what all of these views do to make Wwise work for you. In fact, rarely will any one person use all of these views as Wwise is used by teams of people, many of which have specific jobs that only require them to use specific sections of Wwise.

Profiling the Game

To integrate sound into Cube, you need to first evaluate the kind of information that is being sent from the game to Wwise. To do this you'll need to run the special "profiler" version of Cube that you tested in the Installation chapter.

1. Launch the Cube game.

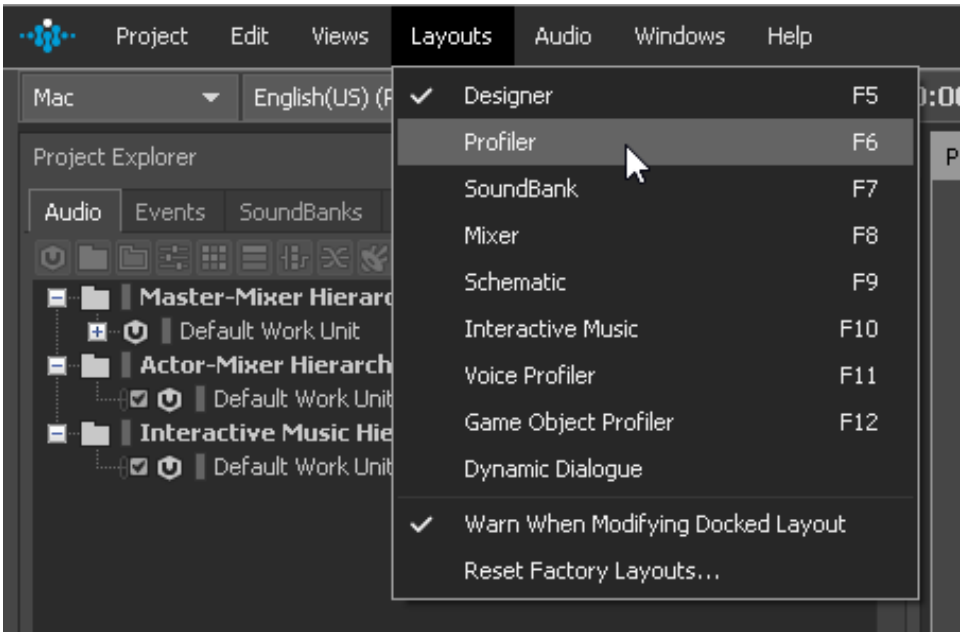
If you're not sure how to launch Cube, review the *Playing Cube* exercise at the end of the Installation chapter written for your computer's operating system. If you play Cube and there is already sound in the game, then refer to the Silencing Cube exercise also found in the installation chapter.

You'll need to now return to Wwise. In fact you'll be switching back and forth between the game and Wwise quite often throughout this tutorial, so you'll want to become familiar with using Alt+Tab on Windows, or Command+Tab on a Mac to switch between programs.

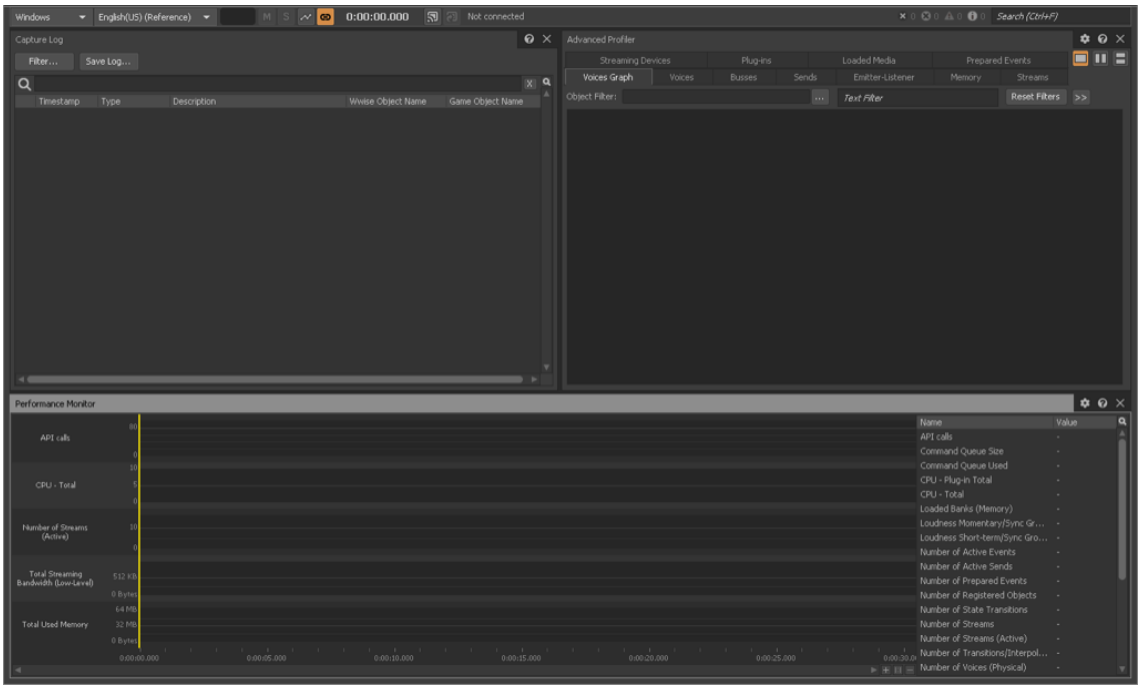
2. Press **Alt**+**Tab** on Windows or **Command**+**Tab** on Mac to return to Wwise.

   There are a number of views that work together to show you information related to reviewing information being received from the game, but rather than opening them one at a time you can quickly open them in one step using a layout. A layout is a predefined collection of views that are commonly used together.

3. In the main menu click **Layouts** and choose **Profiler** or press F6.



The Profiler layout is displayed. You can use the views in this layout to evaluate information about messages generated in the game and monitor details about sound engine performance.

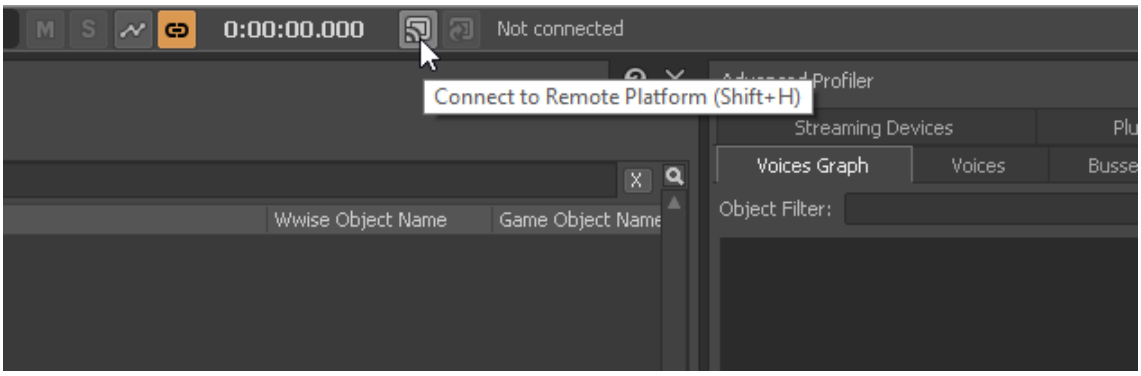


You're going to use the Capture Log view in the upper left to display information that is being generated in game. To access this information, you

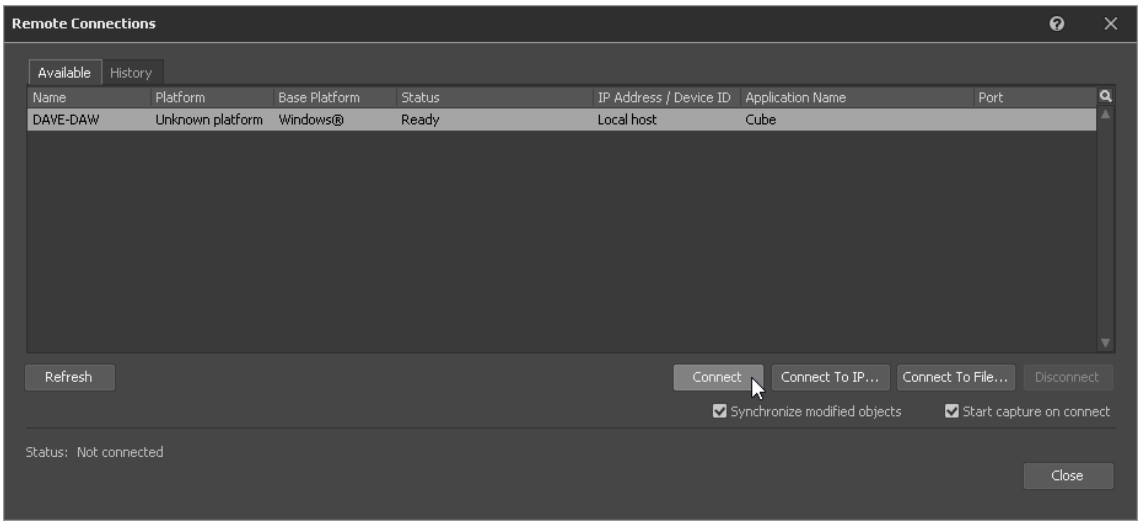
need to connect Wwise to the Cube game that you should still have running in the background.

Hovering over buttons displayed as icons will display a help tag indicating their function.

4. In the toolbar, click the **Connect to Remote Platform** button located to the right of the numeric Cursor Time display.



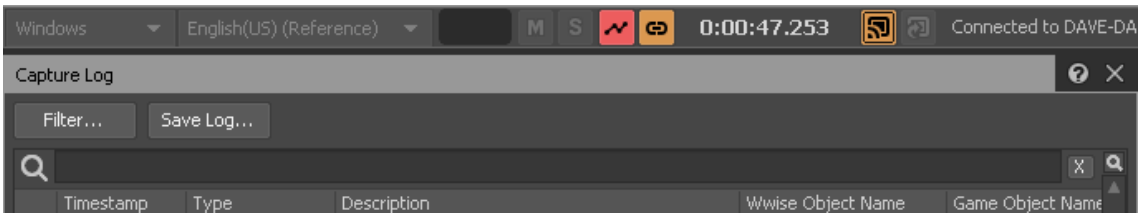
The Remote Connections window opens displaying computers on the network (including your own local computer) that are running a version of the game that uses the Wwise Sound Engine and are designed to communicate with Wwise.



5. Select your computer from the list and click **Connect.**

The window closes and because the **Start capture on connect** check box was selected, you'll see the capture button in the toolbar turn red, while a counter ticks next to it.

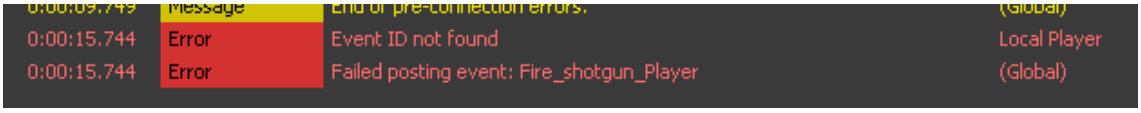
Capturing is the process of recording, in real-time, a log of any information coming from the Wwise Sound Engine in your game related to the game play, an invaluable asset as you develop the sound for your game.



In the Capture Log, you see messages in red and yellow that that Cube is unable to find expected files and information it needs in order produce sound. This is because you deleted those files as part of the installation process.

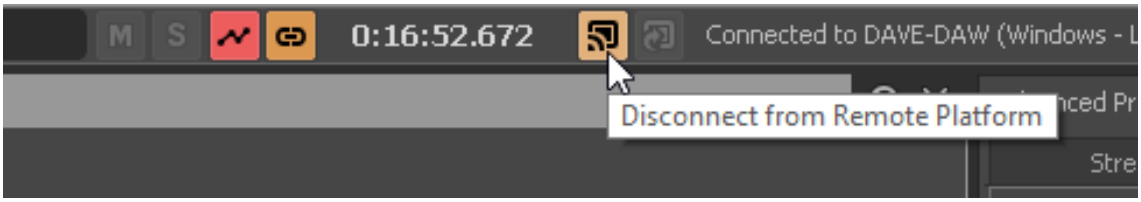
6. Return to the Cube game and fire the shotgun one time.
7. Go back to Wwise to view the Capture Log.

   In the Capture Log view you'll see a few lines of information, confirming that Cube transmitted information to Wwise when you fired the shotgun.



When connected to a game, some Wwise parameters cannot be adjusted. So, once you have the desired Capture Log information, you should disconnect from the game.

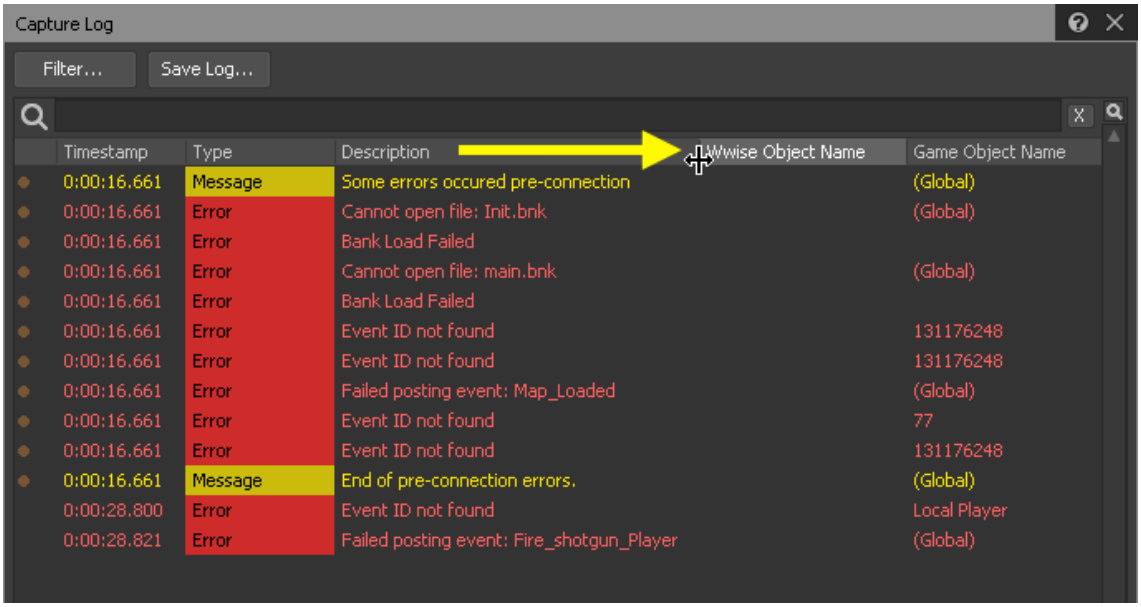
8. Click the **Remote Platform** button in the toolbar.



The capture process stops and Wwise disconnects from the game. You can now evaluate what just happened.

The information displayed represents errors that were generated because Wwise does not know what to do with the messages it has received. These message are generically referred to as game calls and there are a variety of different types of calls that may be triggered. You'll learn more about the different types of calls later, but for now you need to take a closer look at the error description.

9. If necessary, drag the right side of the Description column header to the right so you can view the entire error message.



The messages in the description area refer to something called Events. Events are a type of Wwise Game Call that the game engine sends to the Wwise audio engine, indicating that something has occurred in the game. Usually when an event is transmitted to Wwise, it's used to trigger a sound, to modify one of its properties, or even to stop the sound from playing. Events are given names to identify what the event is being used for. In this case the Fire_shotgun_Player event name that's being referenced makes it clear that it is associated with the act of the player firing their shotgun.

Also notice that there is a Game Object column that specifies Local Player. This identifies which entity in the game the message is associated with. For example, there may be multiple characters in the game that carry a shotgun, so the game engine needs to know which character fired the shot to appropriately play the sound. For example, if the monster that just fired a shotgun appears to be 50 meters away, then the sound probably shouldn't be played at the same volume as when our hero fires his shotgun. This will be explored in greater detail in *Lesson 4: Creating Space*.

For now, the problem is that Wwise is saying that it has no idea what to do with this event, but you're about to change that!

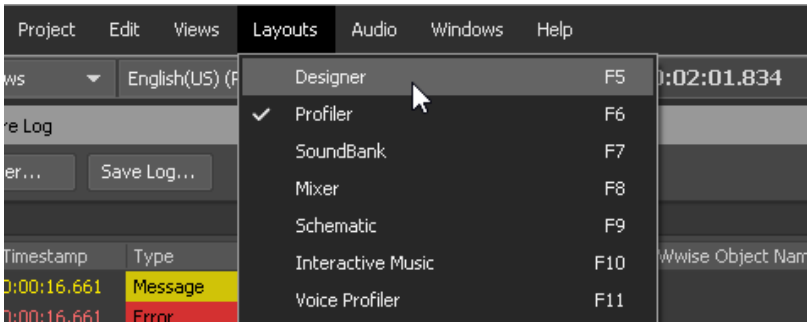
Creating an Event

For everything that happens in the game that may need an audio response, a programmer has added lines of code to transmit a message informing Wwise about what has occurred. This message is referred to as a game call. Game Calls are just simple messages that say things like "Hey, the shotgun was fired", but in

reality this message is sent as a string of text or numbers. When this message is received by the audio engine, any number of things can happen as defined by you, the game audio designer.

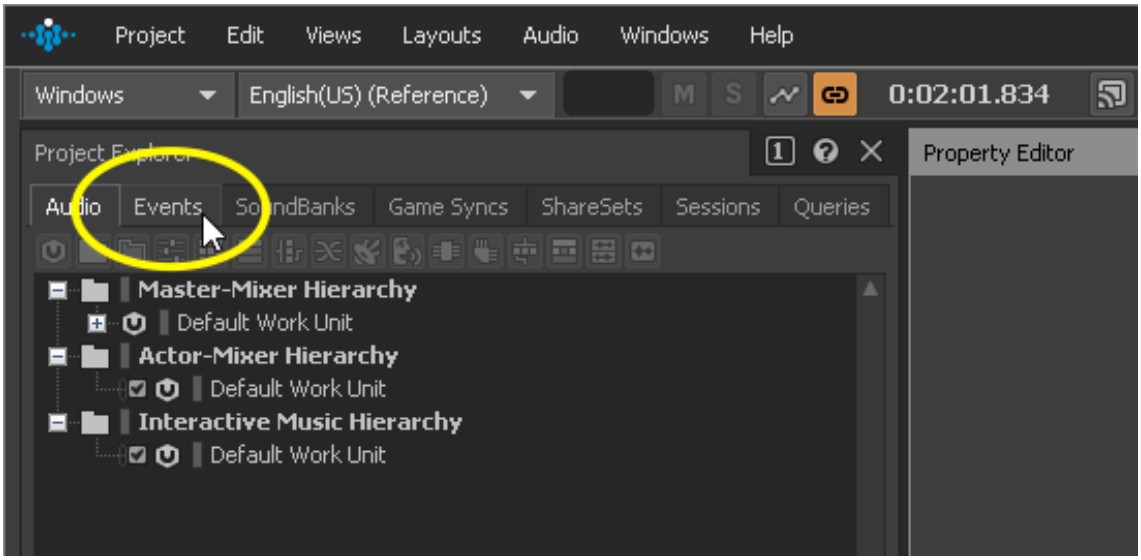
The first thing that you need to do is to create an Event, so that Wwise can catch an incoming Game Call. Think of it like a game of catch between the game engine and the audio engine. The ball being thrown from the game engine is the Game Call, while an Event is a type of object within Wwise designed specifically to catch Game Calls. The important thing to note is that each Game Call needs an identically named Event to be received by Wwise.

1. From the main menu select **Layout** > **Designer** or press **F5**.



2. In the Project Explorer view, click the Events tab.

   It's in the Events tab where you're going to allow Wwise to understand the incoming Event Game Calls that Wwise should expect to see transmitted from the game engine.

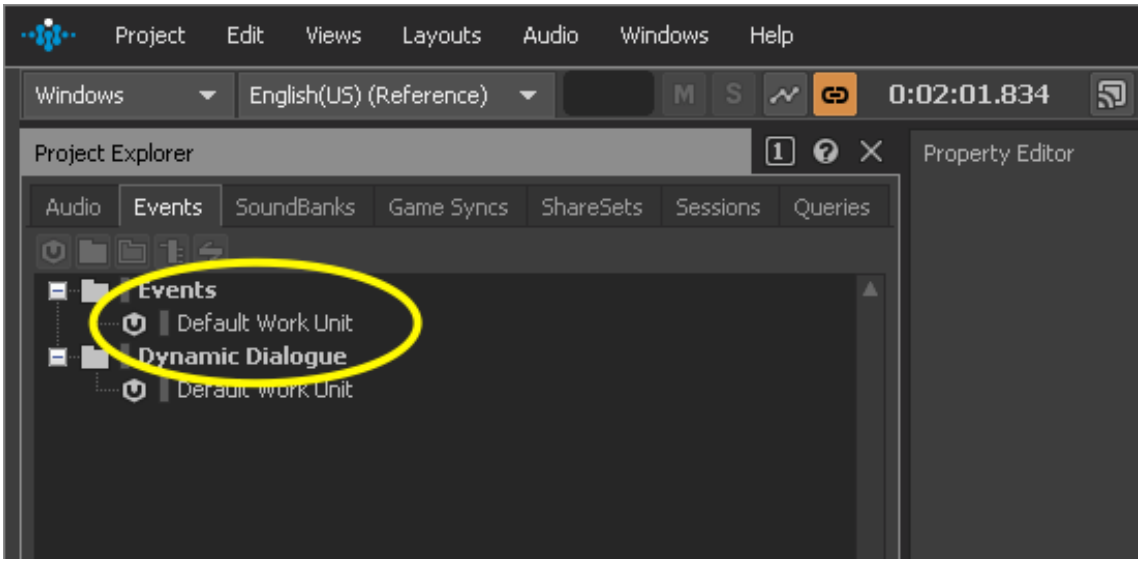


   Within the Project Explorer's Events tab you see a folder titled Events and within that folder there is an object called Default Work Unit. Work Units serve as the foundation for Wwise. Work Units contain information related to a

particular section or element within your project and help you organize your project.

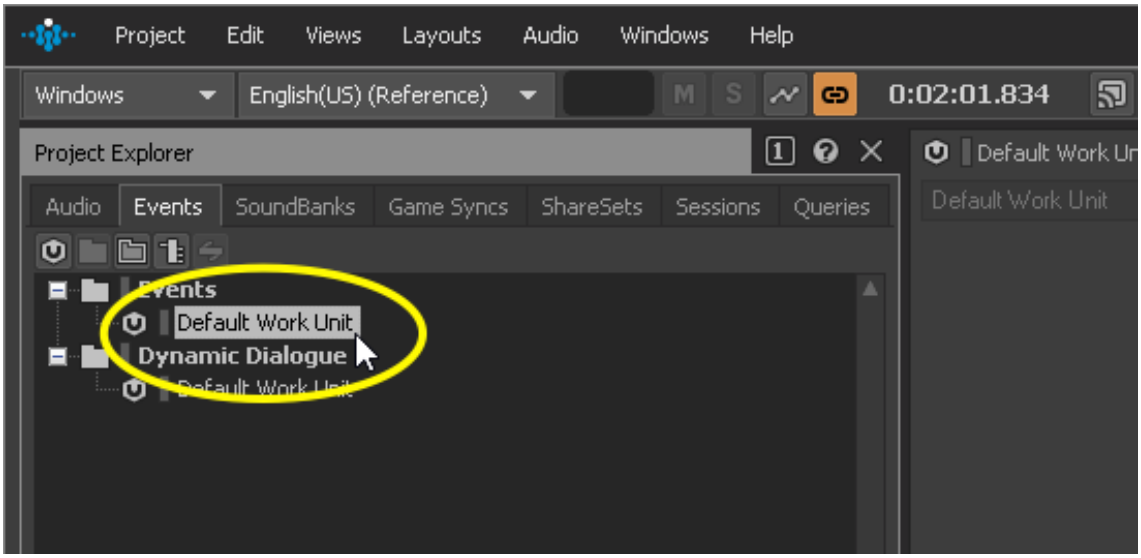
In many games, many people or even companies are concurrently working on different parts of a game. For example, one team may be working on all of the weapon sounds, while another focuses on ambient sounds. In this scenario, each team could have its own copy of Wwise and create their own unique Work Unit that houses their assigned Events. Later in the production, multiple Work Units can be brought into a single project in order to bring all of the elements in a game together.

Work Units are actually XML files created within the project structure of Wwise.



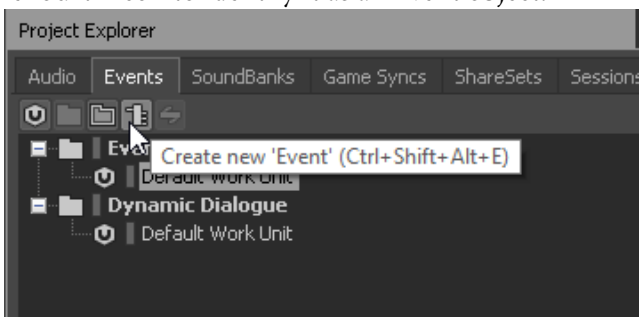
Anything you create in Wwise is referred to as an object. Be careful not to confuse this with the Game Object term discussed earlier in this lesson which relates to items within the game. Objects in the Wwise interface are represented by small square icons. There are over 20 types of objects, each providing unique functionality for how you can create and control sound within Wwise. Objects exist within a hierarchy and typically work units are at the top of this hierarchy. Like the bricks that make up a building, these objects are building blocks that can be used for purely practical applications, or they can be arranged in intricate and creative ways. For now, you're going to create a single Event object within the Default Work Unit that will serve as a glove that will be used to catch the shotgun Game Call that is being transmitted from the Cube game engine.

3. Click the Default Work Unit within the Events folder.



When an object is selected, an icon bar is displayed in the Project Explorer that shows which other types of objects can be housed within the selected object. When you hover over the icon, a tooltip indicates the type of object represented by the icon.

4. Hover over the fourth icon to identify it as an Event object.



5. Click the Event icon to create a new Event Object within the Default Work Unit.

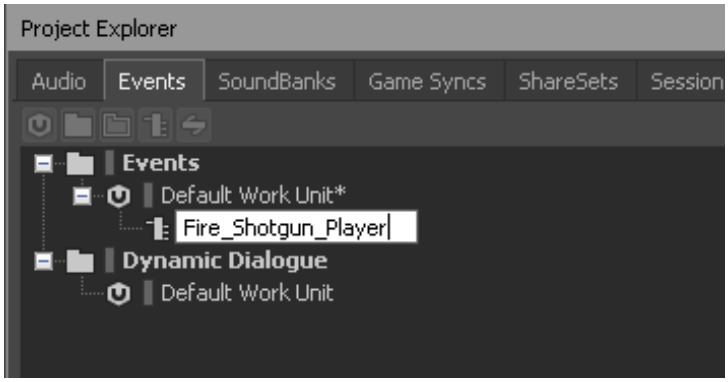
   A new event object is created and you are prompted to type its name.

   How you name an Event object is critical. It is imperative that the name of the Event object perfectly matches that name of the game call being transmitted from the game engine. Just like dialing a phone number, if you're off at all, it won't connect.

   To ensure that this works well, typically the sound designer decides what kinds of things the sound engine needs to be aware of, and creates the Events for each of these things in Wwise. The sound designer then informs the programmer of the Event names so that they can program the game engine to transmit game calls with the same name. In our scenario, the game engine for Cube

has already been programmed, and the names for the calls have already been defined, and Fire_Shotgun_Player is the name for the call used each time a shotgun is fired.

6. Type Fire_Shotgun_Player and press **Enter**.



Wwise will later create software code based on your project. In that code, all object name references are handled in lower case, therefore object names are not case sensitive.

Selecting an object reveals its associated properties in the Event Editor to the right which you can confirm by the Fire_Shotgun_Player - Event Editor in the view's title bar. Different objects have editors of varying complexity. Most have a Notes field that is incredibly important because you can add extra information to avoid any confusion about the Event name. With a name like Fire_Shotgun_Player, the Event likely relates to the shotgun being fired, but not all Events are intuitively named. Notes provide a way to provide more information to help you and others who might be reviewing your work better understand what you've created.

Importing a Sound

Now that you have an Event setup to catch the incoming shotgun Game Call, you need to bring in the sound that you want to hear when that event is received. Wwise can create sounds in a variety of ways, including synthesizing sounds from scratch; however, the most common way to generate a sound is to use a recorded audio file. There's no need to go out and record a shotgun as an audio file of a shotgun blast is already provided. You simply need to bring it into your project.

Sounds are usually managed within the Audio tab of the Project Explorer which is found in the Designer layout.

1. In the Project Explorer view, select the Audio tab.

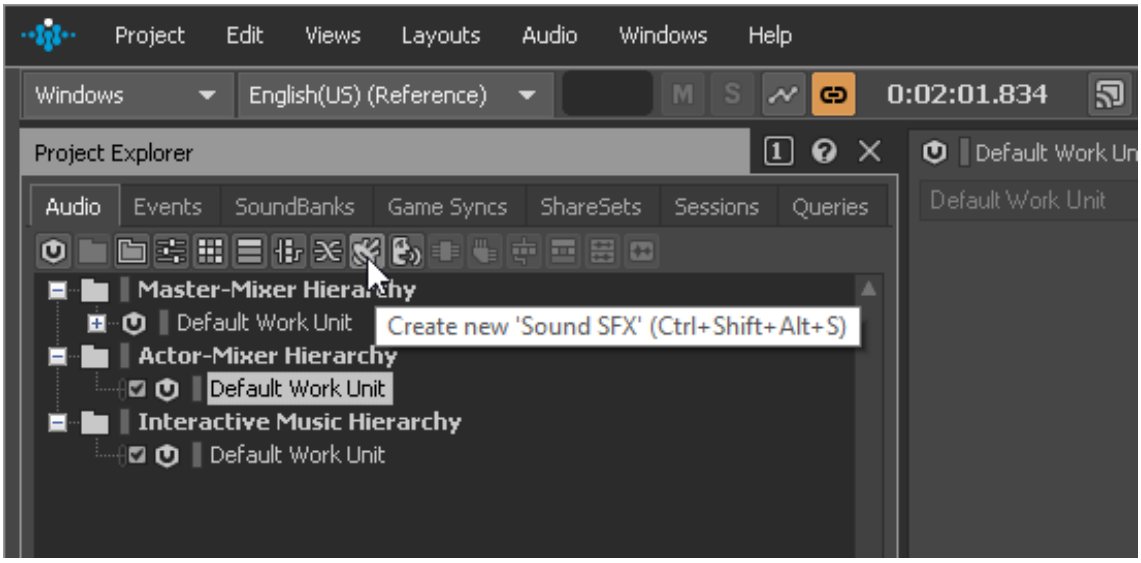
   The Audio tab of the Designer layout is where you'll spend most of your time when designing your game's soundscape. The audio tab has three different hierarchies that each provide unique objects to accomplish various tasks related to triggering or manipulating how audio works in Wwise.

With the exception of music, you'll keep your sounds within a Work Unit within the Actor-Mixer Hierarchy.

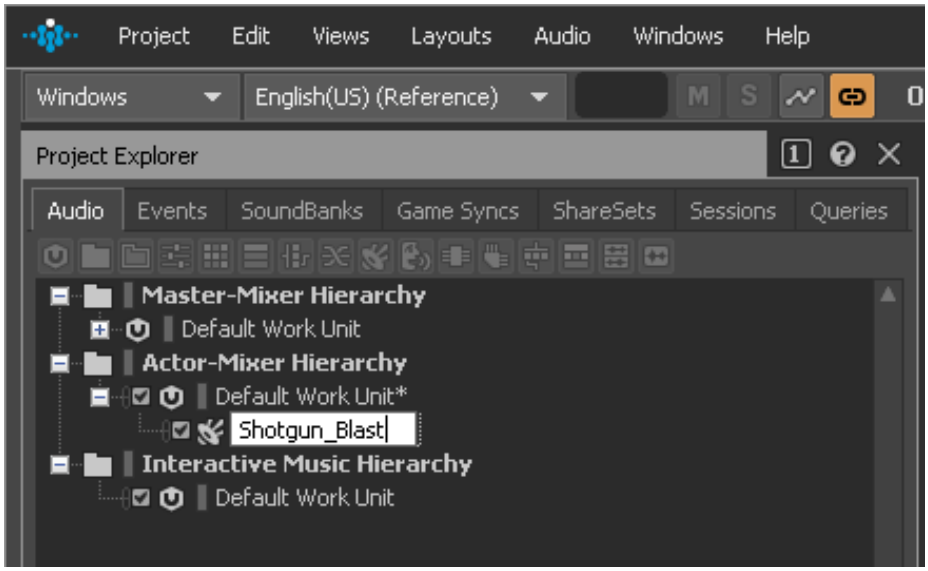
There are many different objects that can be created within the Actor-Mixer Hierarchy that you'll explore over the next several lessons; however, if you simply want to play a given audio file, this is accomplished via the Sound SFX (Sound Effects) object.

It is also possible to play sounds through Sound Voice objects; however, they are generally used for spoken dialogue as they have specific localization features used when releasing a game in multiple languages.

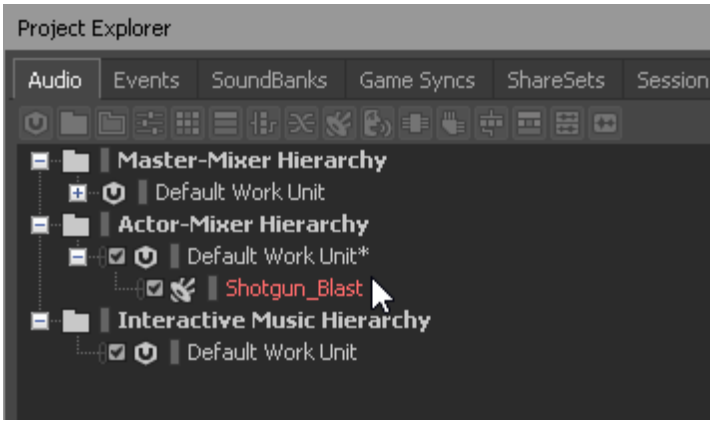
2. In the available object row, click the Sound SFX icon.



3. Name the Sound SFX Object Shotgun_Blast.



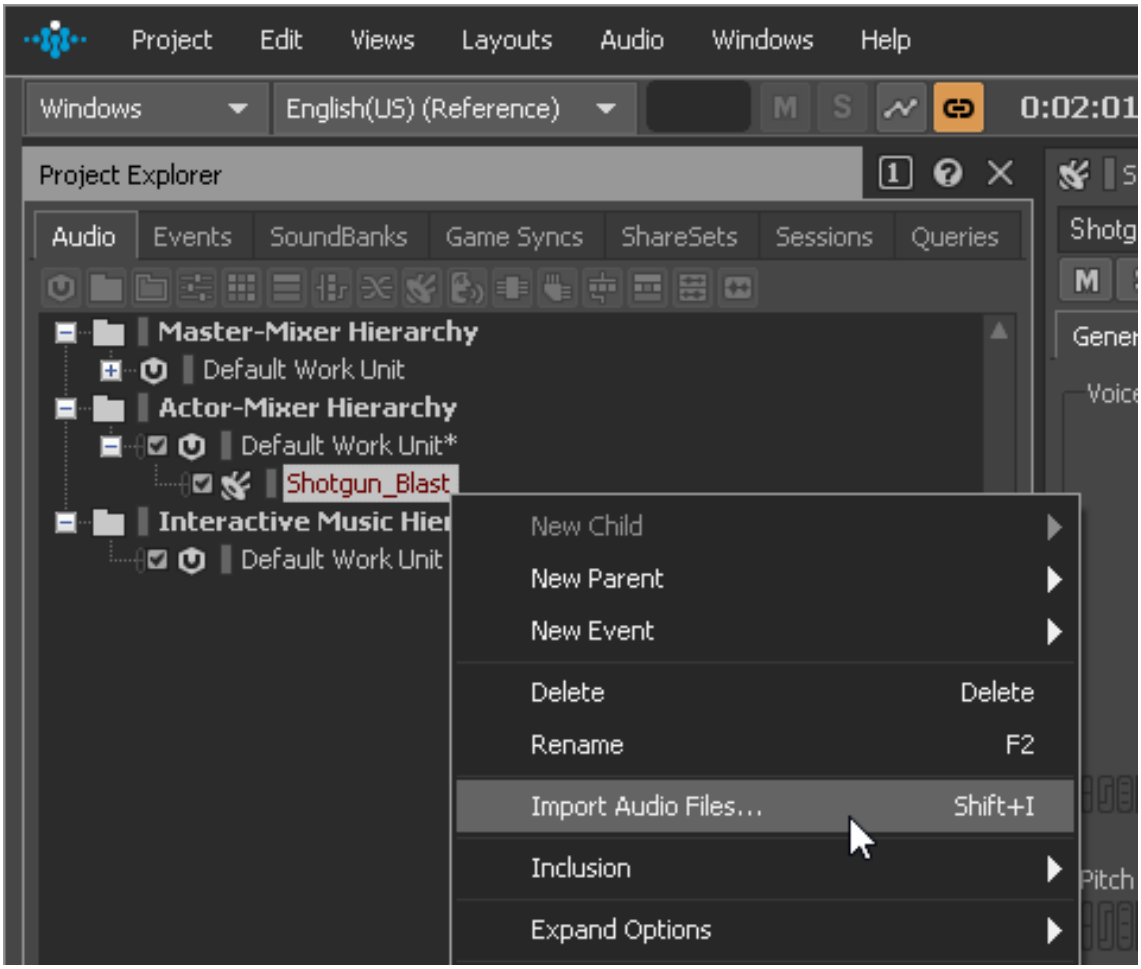
The object name appears in red.



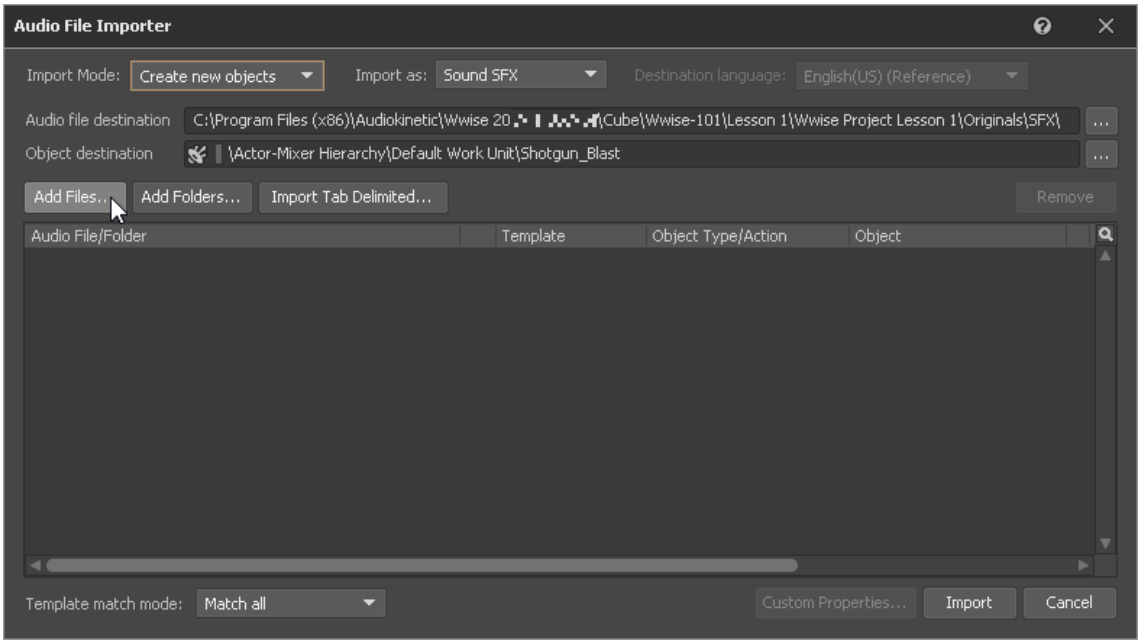
The red lettering represents that there isn't an audio file associated with this SFX object. You'll need to add the audio file in just a moment, but before you go any further it's important to understand that an SFX object does not directly represent an audio file. It instead represents the channel that the audio file will play through. You can equate it to the idea of a channel on a digital audio workstation. The channel has various controls that manipulate the actual audio files that are stored on the audio track that feeds through the channel. Once you understand this, you're ready to add the audio file to the Sound SFX object.

Right-clicking an object provides many different options related to what can be done with that object, including importing an audio file to a Sound SFX object.

4. Right-click the Shotgun_Blast SFX Object and choose **Import Audio Files**.



5. Click **Add Files…**

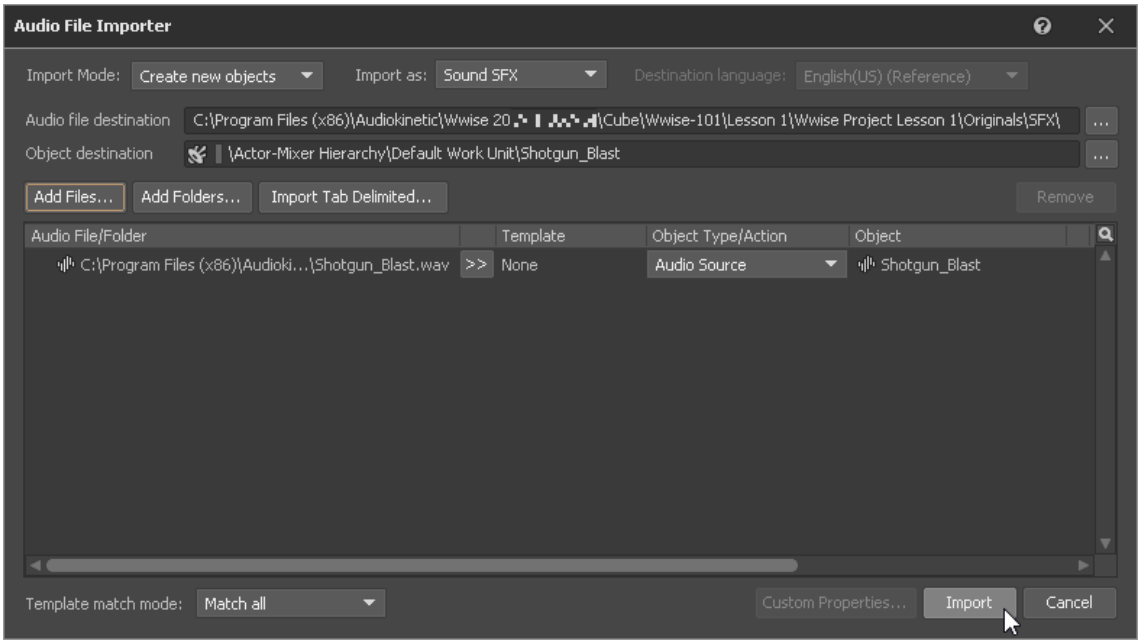


A dialog box opens prompting you to select the file you want to import.

The file you're going to import is a WAV file. Wave files are relatively large and you may well think that you'd first want to convert the file to something like an MP3 file or even reduce its sample rate or bit depth to reduce the amount of information you are bringing in. This is not the case. Actually, you want to import your best quality original into Wwise as a Wave file and don't worry about the size at this time. The beauty of Wwise is that you can later decide how you want to optimize the file's size before integrating into the game. In fact, Wwise has extensive features related to this which you'll explore in Lesson 7. Think of it like a photographer wanting to always keep his 25 megapixel original and only worry about how to crop or compress the image based upon the need at the time he needs to send someone the image.

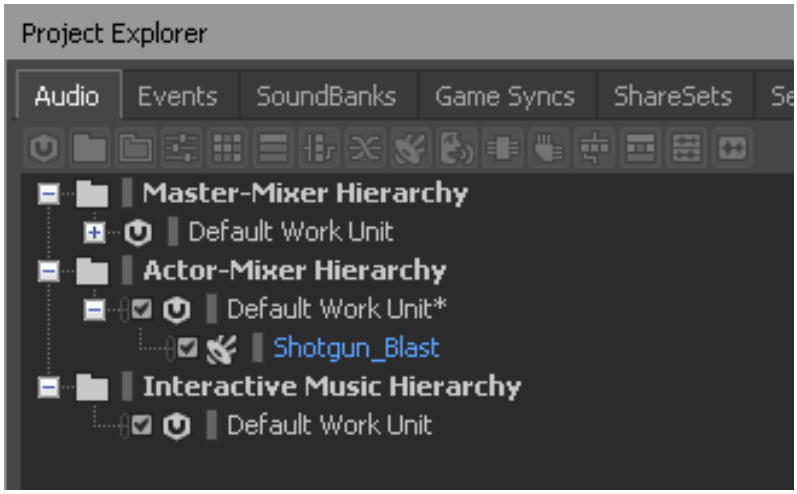
6. Navigate to `Wwise-101/Lesson 1/Audio files for Lesson 1/`, choose the Shotgun_Blast file and click **Open**.

   The Audio File Importer opens confirming which audio file you want to import to the SFX object.



7. Click **Import**.

The Audio File Importer window closes and you now see that Shotgun Blast SFX object has turned blue.



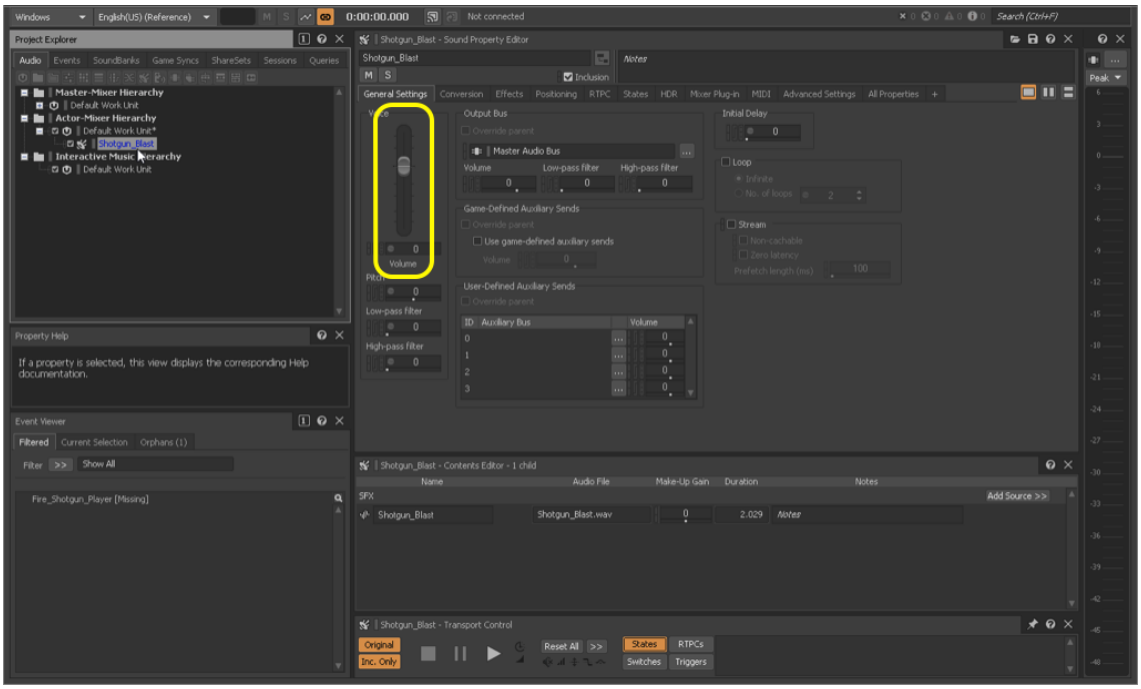
The blue color indicates that an audio source, in this case a WAV file, is properly associated with the SFX object and that the audio file is currently being referenced in its original imported format.

As you work, you may notice some Sound SFX objects names are blue while others are white. The color communicates if the associated file has been optimized through a conversion process which usually happens when you generate a SoundBank which you'll do later in this lesson. White object names indicate that the conversion has already taken place, where blue objects have yet to be converted.

At this point, don't worry if you see the color change from blue to white. You'll learn more about optimization and the conversion process in Lesson 7.

Now it's time to test your sound and make sure it's playing through your system.

8. Click the Shotgun_Blast Sound SFX object to make sure it's selected.

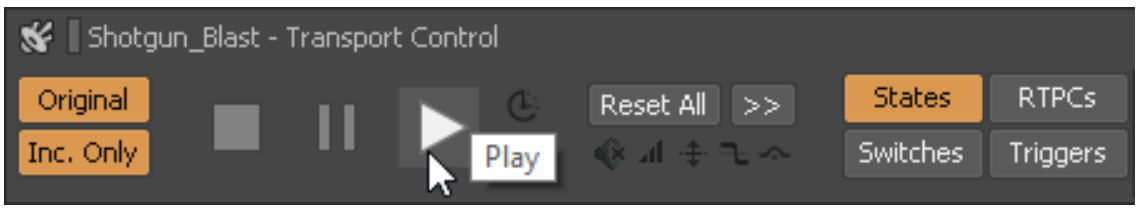


Notice that the Property Editor view displays audio controls for the Shotgun_Blast such as a volume fader.

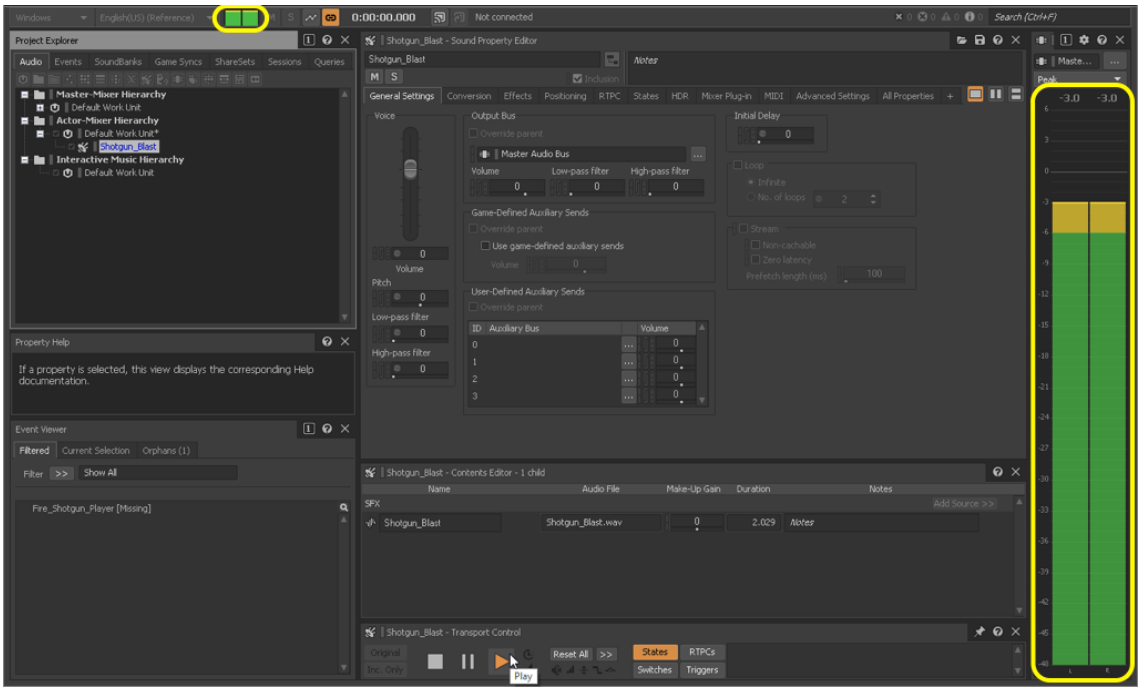
Also, look in the Transport Control view and you'll see the name Shotgun_Blast. This indicates that when you press the play button, you'll hear the sound as it's going to be played in the game.

The spacebar is the keyboard shortcut for the play button.

9. In the Transport Control view, click the play icon or press the spacebar.



You should hear the shotgun blast sound. You'll also notice that there's a meter in the toolbar as well as a meter view on the right side of the layout showing you the level of playback.

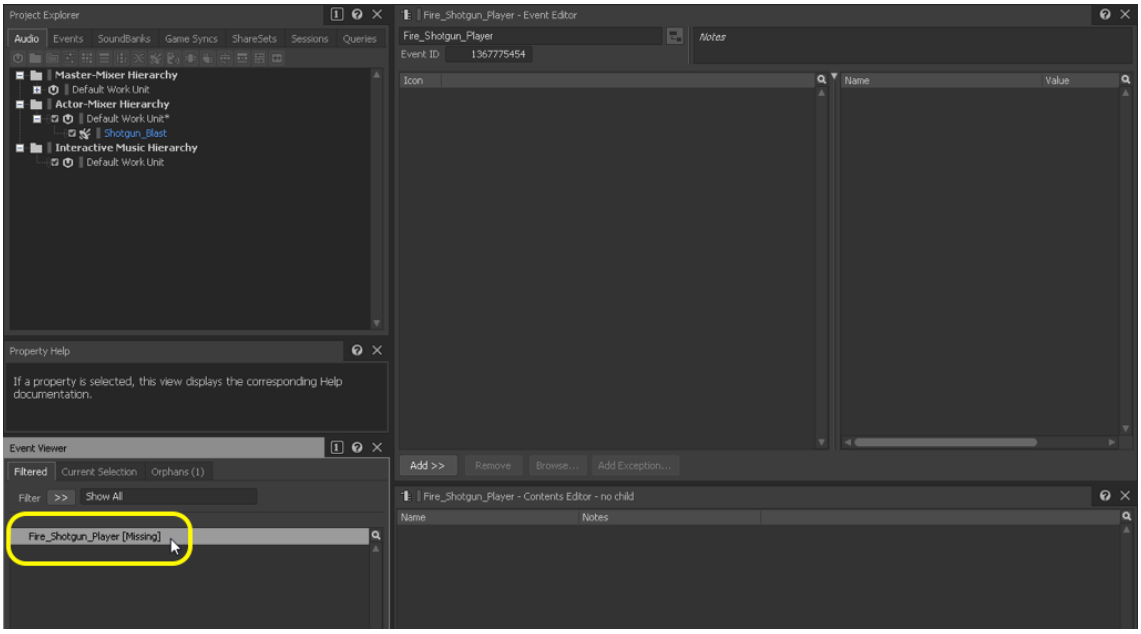


Applying an Action

Earlier you created the Event object that catches the game call transmitted from the game engine, and you've just created the Sound SFX object that contains the audio file you want to hear when that event is received. Now you are ready to connect these two objects together. This is accomplished via Actions that are created in a selected Event's Event Editor.

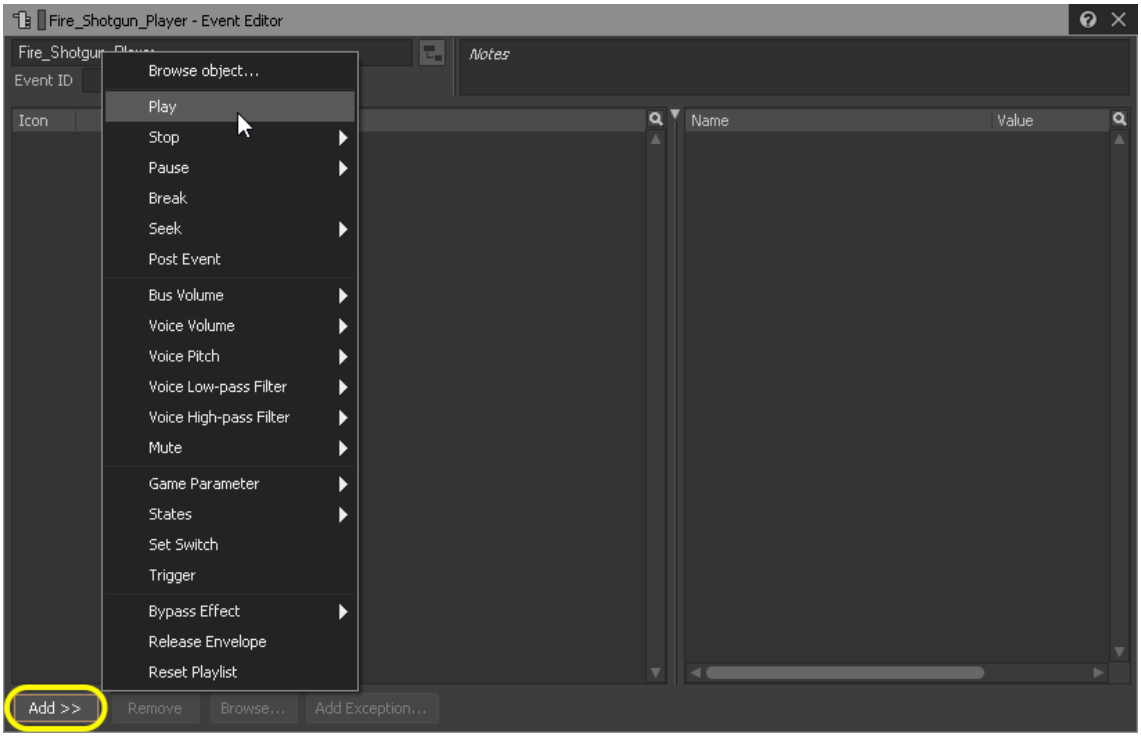
Earlier you worked with the Event Editor when you created the Fire_Shotgun_Player event within the Events tab of the Project Explorer. Although not currently in the Events tab, you may have noticed the Event Viewer view in the lower left corner is displaying the Fire_Shotgun_Player event you created earlier. There you see your Fire_Shotgun_Player event is indicating that it is missing associated actions and therefore isn't doing anyone any good.

1. Click the Fire_Shotgun_Player [Missing] Event in the Event Viewer.



The Fire_Shotgun_Player Event Editor is displayed showing two empty panes. This is where you indicate what Action you want to have happen when Wwise receives the Fire_Shotgun_Player event from the game engine.

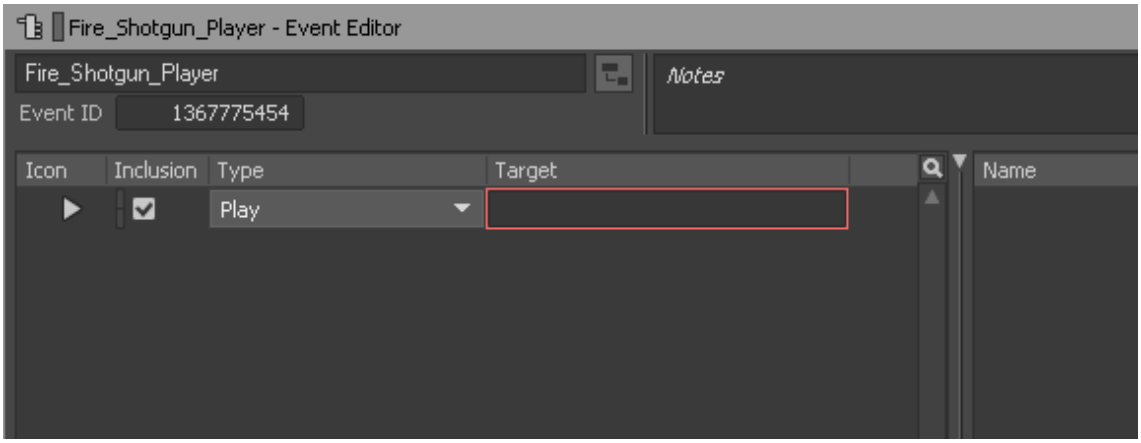
2. Click the **Add** >> button found in the lower left of the Event Editor.



A list of possible actions is displayed. As you can see, there are a lot of options, but all you need for now is the first option – Play.

3. Select **Play** from the Action list.

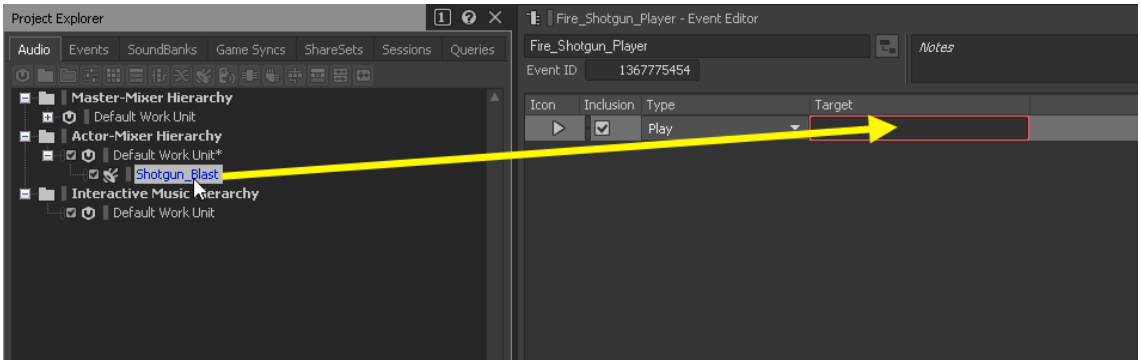
   The play action is added, but the action's Target column indicates that there is no object being referenced.



If you do not see the Target column in the left pane, then drag the border between the two panes in the Event Editor to the right until the target column is displayed.

Now you need to indicate that you want to play the Shotgun_Blast Sound SFX object that you created in the previous exercise.

4. Drag the Shotgun_Blast Sound SFX object from the Project Explorer to the Target column in the action list.

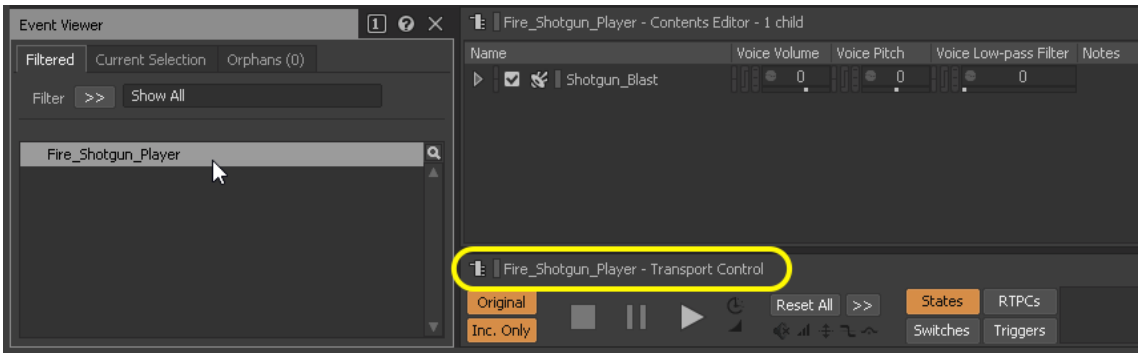


To save a step you can drag an object to an empty position in the action list and it will automatically add it with a Play action.

You've now connected the Fire_Shotgun_Player Event to the Shotgun_Blast SFX Object. Wwise lets you simulate what will happen in the game by allowing you to play not just Sound SFX objects, but also the Events themselves. Playing

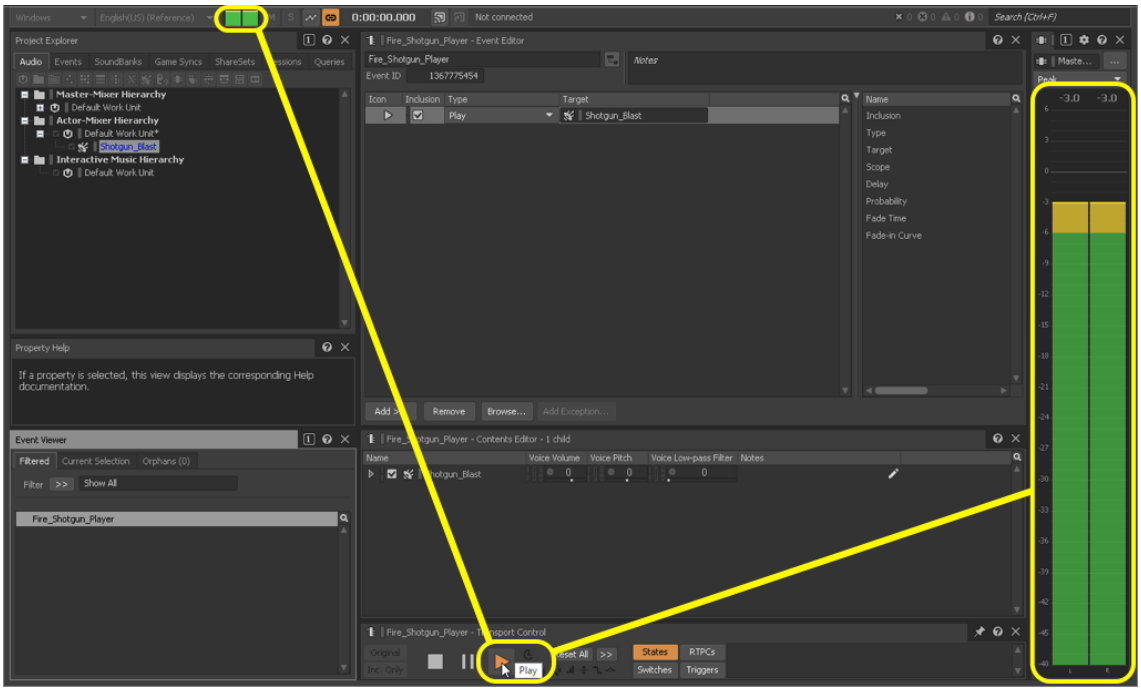
the Fire_Shotgun_Player event should trigger the shotgun blast sound. Before you can do this, you need to make sure that the Fire_Shotgun_Player Event is visible in the Transport Control view.

5. Look at the title bar of the Transport Control view and verify that it says Fire_Shotgun_Player. If it doesn't, click the Fire_Shotgun_Player event in the Event Viewer.



6. In the Transport Control view, click the play button or press the spacebar.

Bang! You should hear the shotgun blast, confirming that the Fire_Shotgun_Player Event is linked to the Shotgun_Blast sound you imported earlier.



Integrating Sound Into the Game

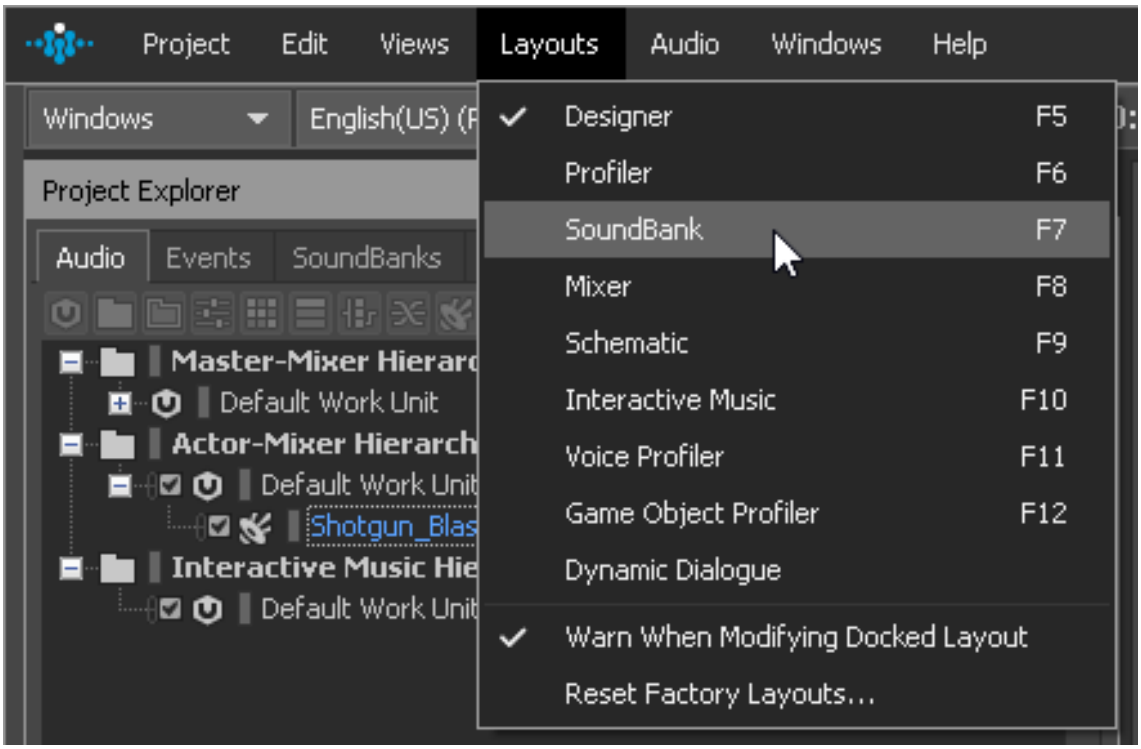
While playing the shotgun sound in Wwise is getting you closer to your goal, it's not complete until you can play the game and hear the result of your work. In

many respects, this last step is like the process of bouncing a playable mix of a music project, with the exception that there is a lot more going on than simply generating a single audio file. This is where you get to see the real benefits of the Wwise sound engine, as it essentially writes the code necessary to apply your sound design to the game for you.

Adding an Event to a SoundBank

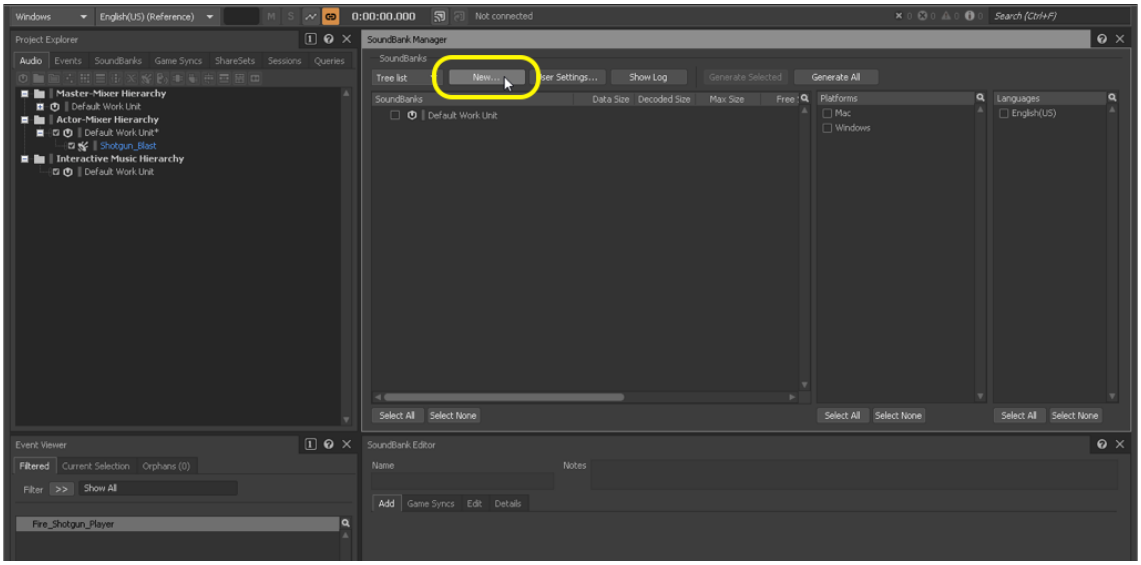
SoundBanks are the collection of both the code and the audio assets used at the time the game is played, also known as run-time. Every game created by Wwise has at least one SoundBank and, in larger more complex games, there may be many SoundBanks. In this case, you'll create a single SoundBank and place it into the directory where the Cube game engine expects to find it. In order to achieve this, you'll use a different layout.

1. In the main menu, click **Layouts** > **SoundBank** or press **F7.**



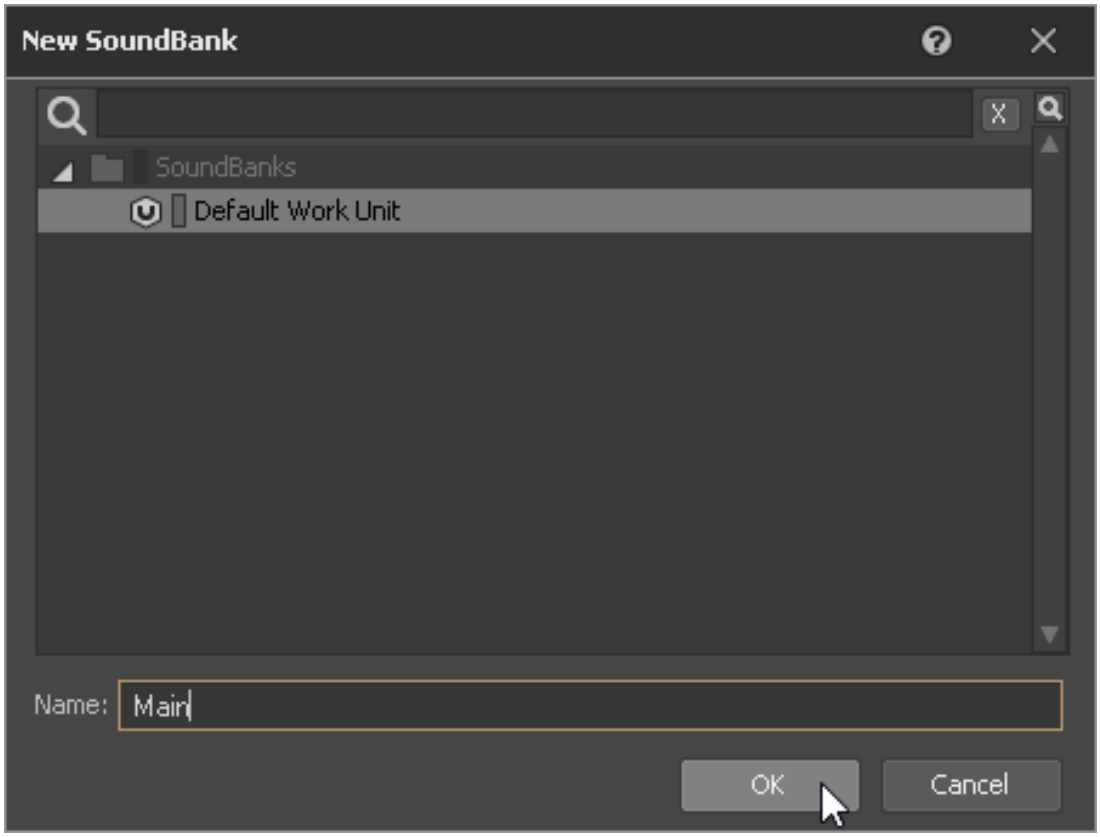
In the upper right area of the screen you can see the SoundBank Manager and, like other areas of Wwise, it contains a Default Work Unit. Cube has been coded to look for a SoundBank called Main, so you'll need to create that within this default work unit.

2. Click **New.**

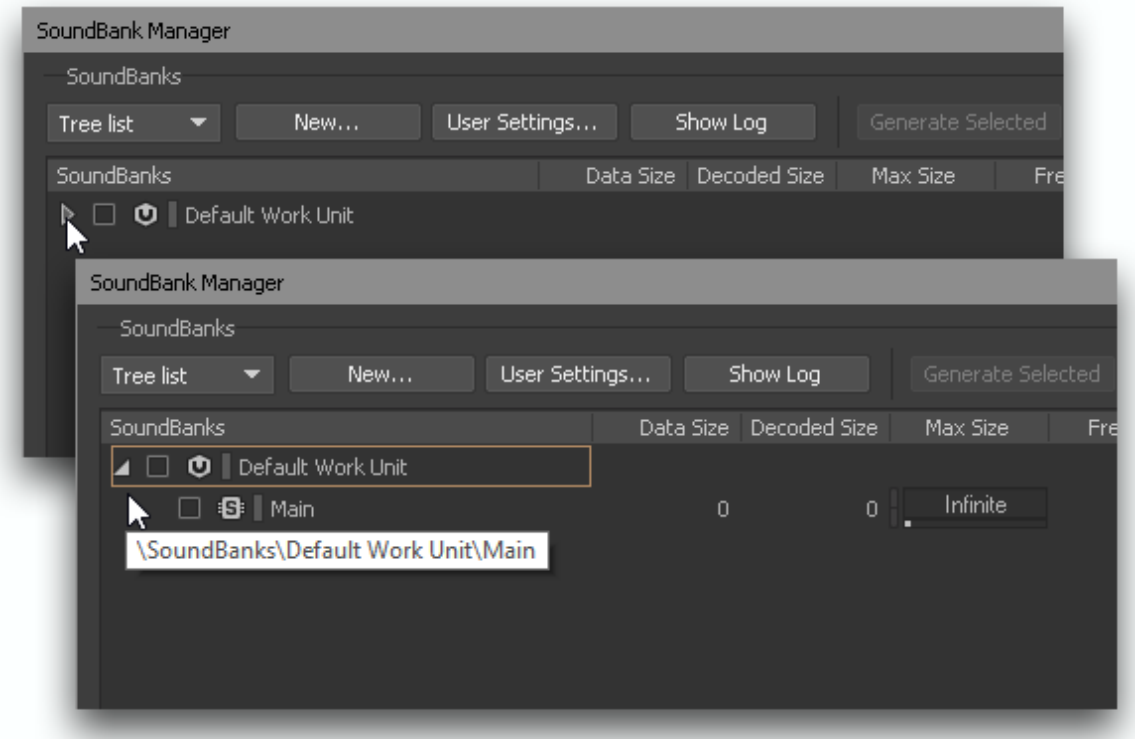


The New SoundBank dialog box opens.

3. In the Name field, type Main and click **OK**.

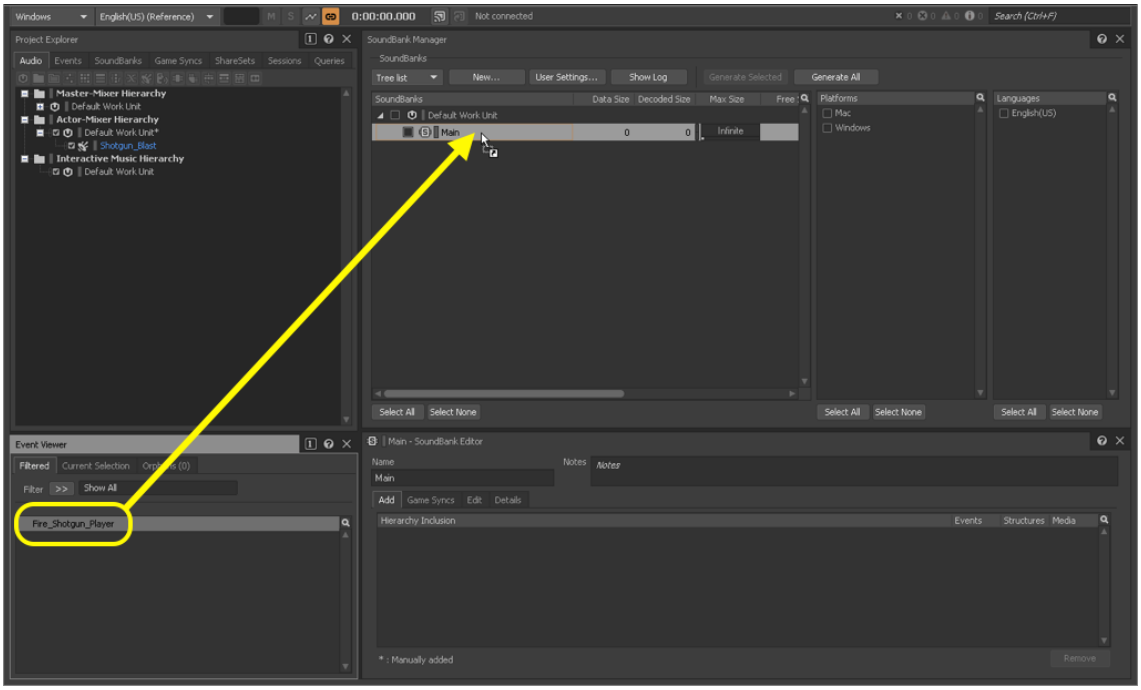


4. Click the disclosure triangle to the left of Default Work Unit.

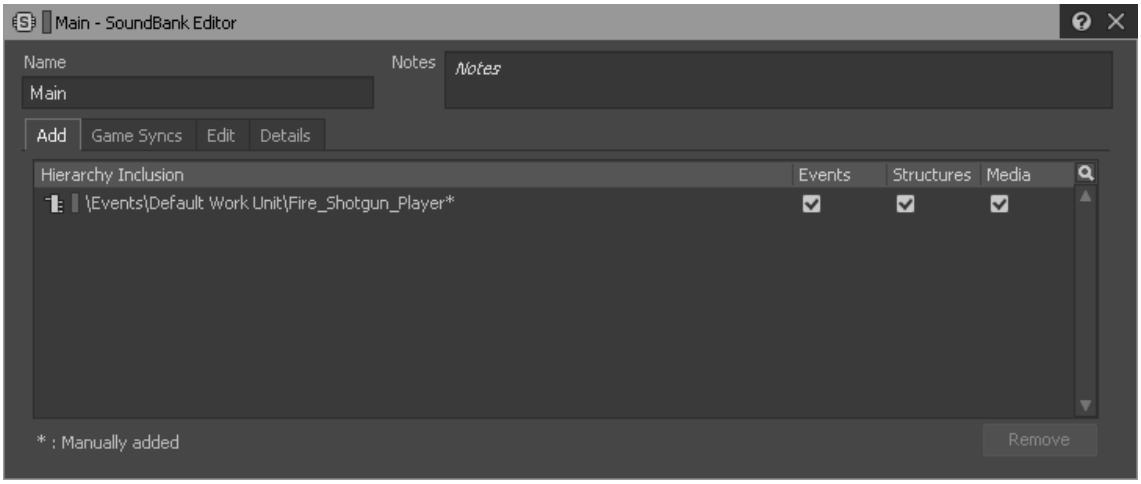


SoundBanks need to be populated with the Events that you intend to have as part of that SoundBank. Events can be assigned to a SoundBank by simply dragging them to the desired SoundBank.

5. Drag the Fire_Shotgun_Player event from the Event Viewer to the Main SoundBank in the SoundBank Manager.



In the lower right view, you'll see the Event you just added in the SoundBank Editor.



Generating a SoundBank (Mac).

Generating a SoundBank

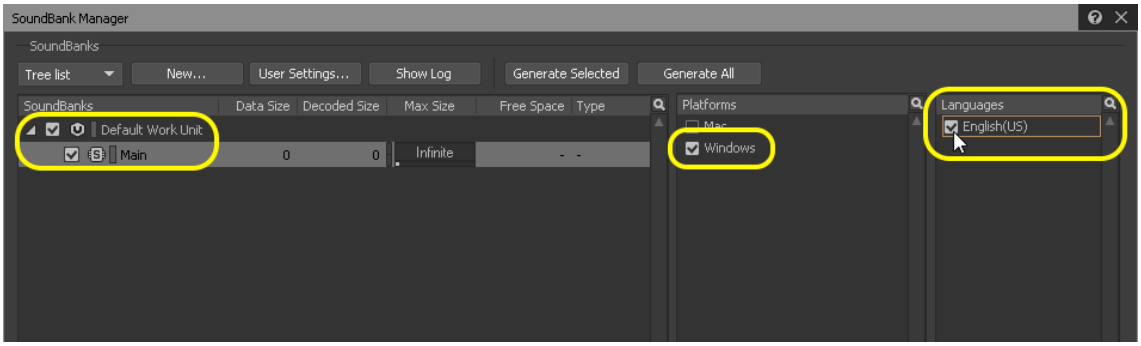
Wwise can simultaneously generate SoundBanks for multiple game platforms and even multiple languages, if you've provided dialogue content and used Wwise's localization features. This is a huge time saver considering that different game platforms have different ways in which audio code must be implemented. In essence, Wwise is like having multiple codewriters that know the specifics of

each game platform working for you. Using the **Generate All** button, Wwise can generate all of the possible SoundBanks in one swift step, or you can specify which SoundBanks you want to generate by choosing the necessary criteria. While this Wwise project can generate a SoundBank for both the Windows and Mac platforms, you'll specify that you only want to generate a SoundBank for the type of computer that you're currently using to play Cube. To do this you'll need to specify which game system and language you want to author SoundBanks for.

The specific steps you'll need to take in the following exercise will differ depending on if you are using a Windows or Mac computer. The images following each step reflect using the Windows platform. For Mac users, simply choose the available Mac option instead.

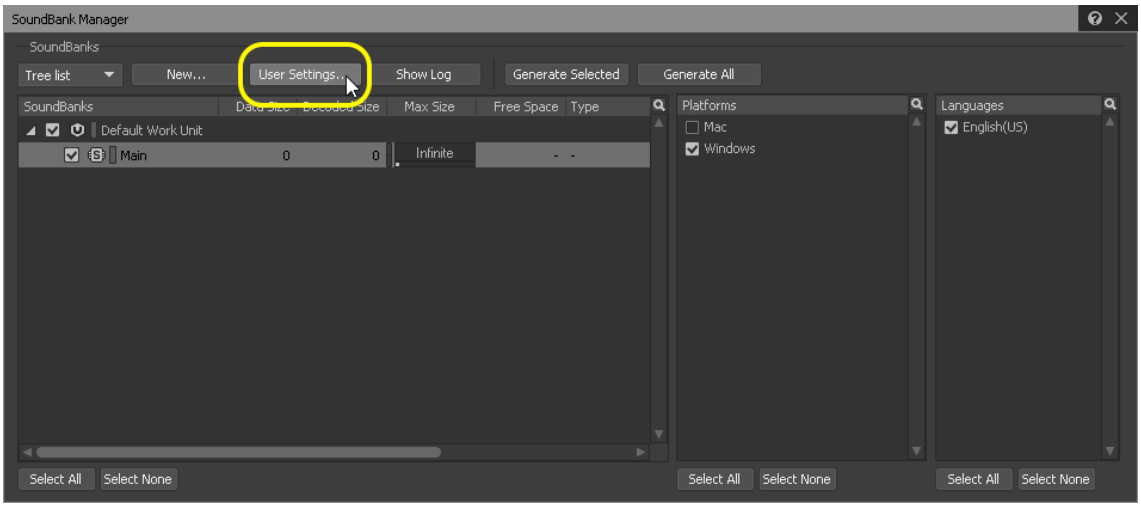
Because multiple SoundBanks can be generated for a single game, you must first select the SoundBanks that you want to be generated.

1. Select the Main SoundBank, appropriate Platform and English (US) language check boxes are selected.

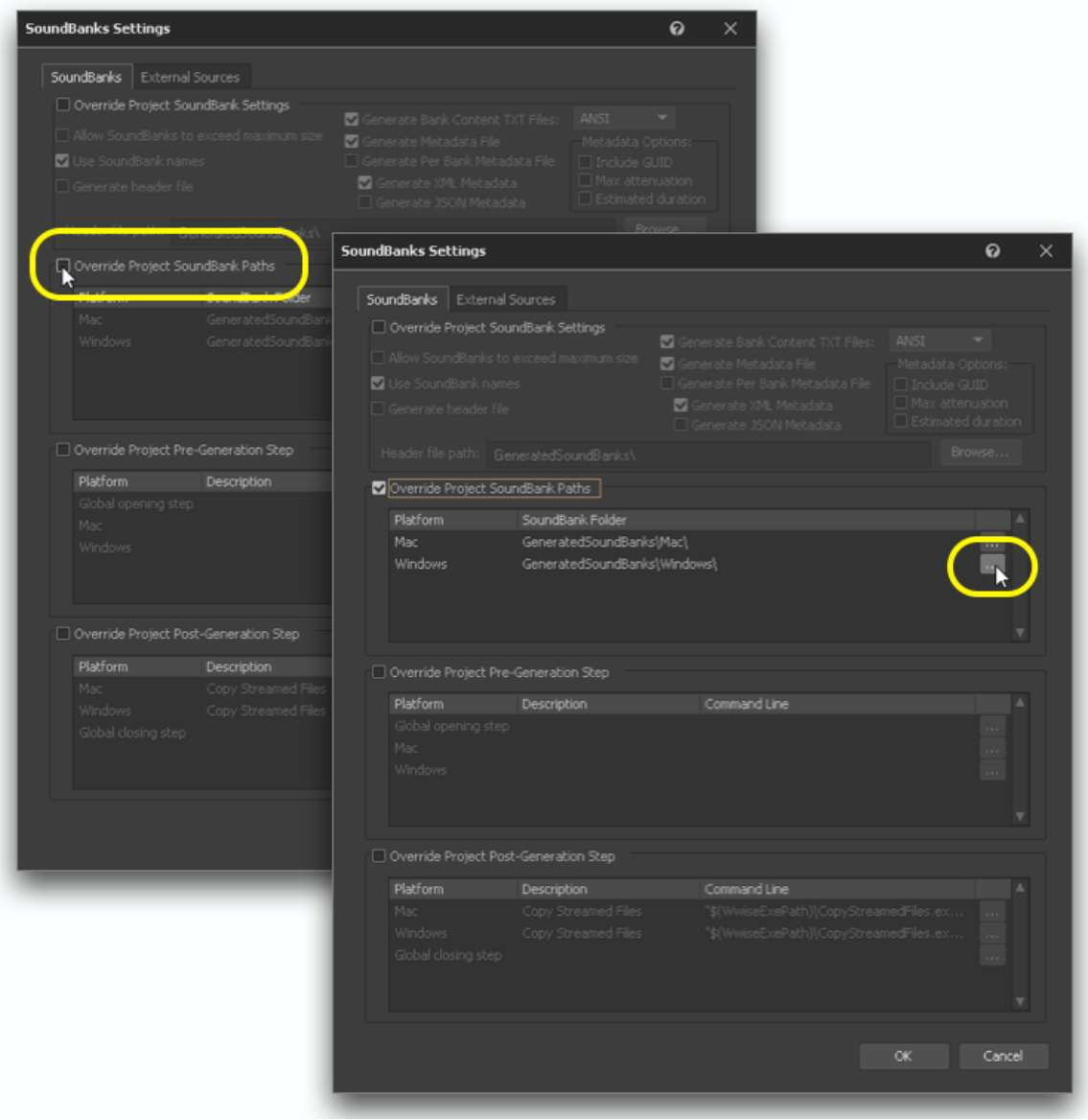


Next you must define the path indicating where the SoundBanks should be placed within the game's file structure. This information would typically be given to you by the game programmers.

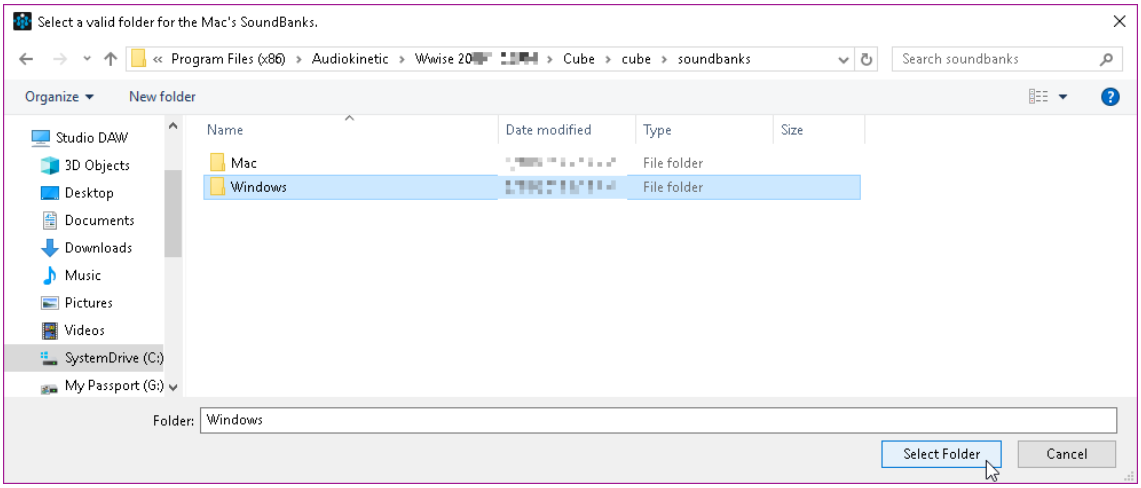
2. In the SoundBank Manager, click **User Settings.**



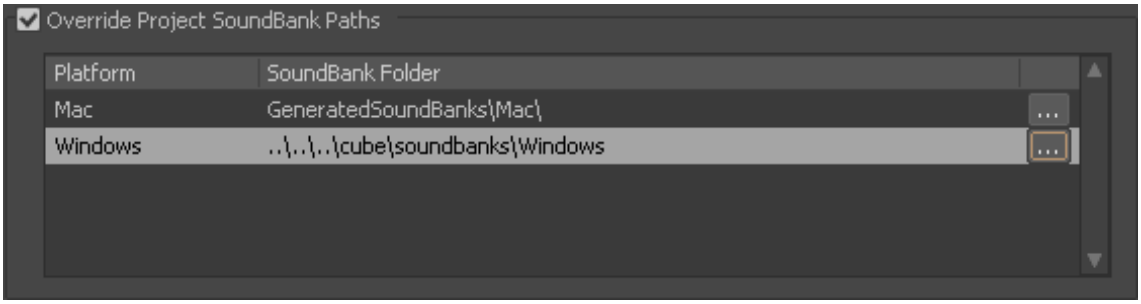
3. Select the Override Project SoundBank Paths option, and then click the path selector button for your platform.



4. Navigate to your Wwise Lessons folder, and then proceed to `Cube\cube\soundbanks`, choose the folder for your platform and click **Select Folder**.

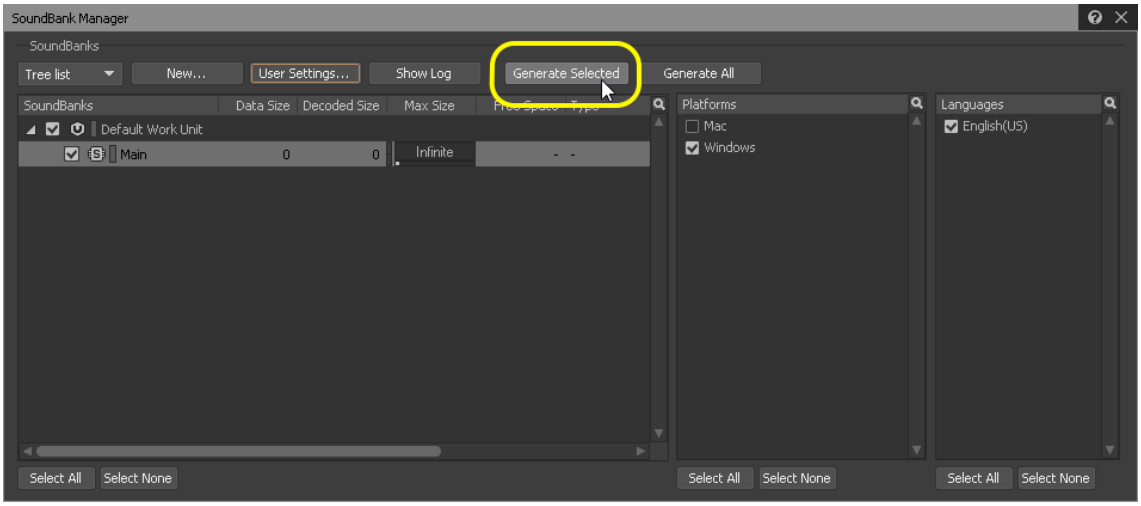


The updated SoundBank path appears for the platform you selected.

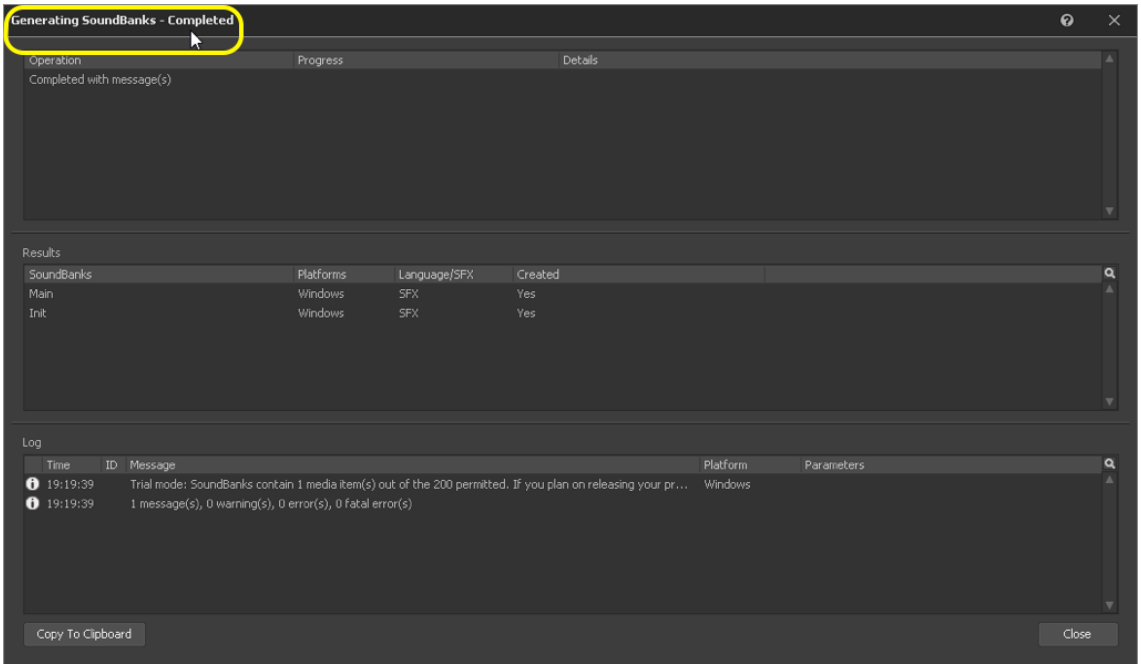


You now need to generate your SoundBank. Generating a SoundBank is the payoff of all of your work. It's much like bouncing a file in a conventional digital audio workstation. The result is what gives you and everyone else the ability to benefit from your work!

5. Click **OK** to close the SoundBanks Settings window then click **Generate** in the SoundBank Manager.



The SoundBank Generation view is displayed. In this moment, the code and all necessary files to implement your shotgun sound have been generated.



6. Click **Close**.

You'll be generating a SoundBank at the end of Lessons 2 through 8. For those lessons, the path to SoundBank folder will already be set, so you'll only need to make sure that you check the appropriate SoundBank, platform and language check boxes.

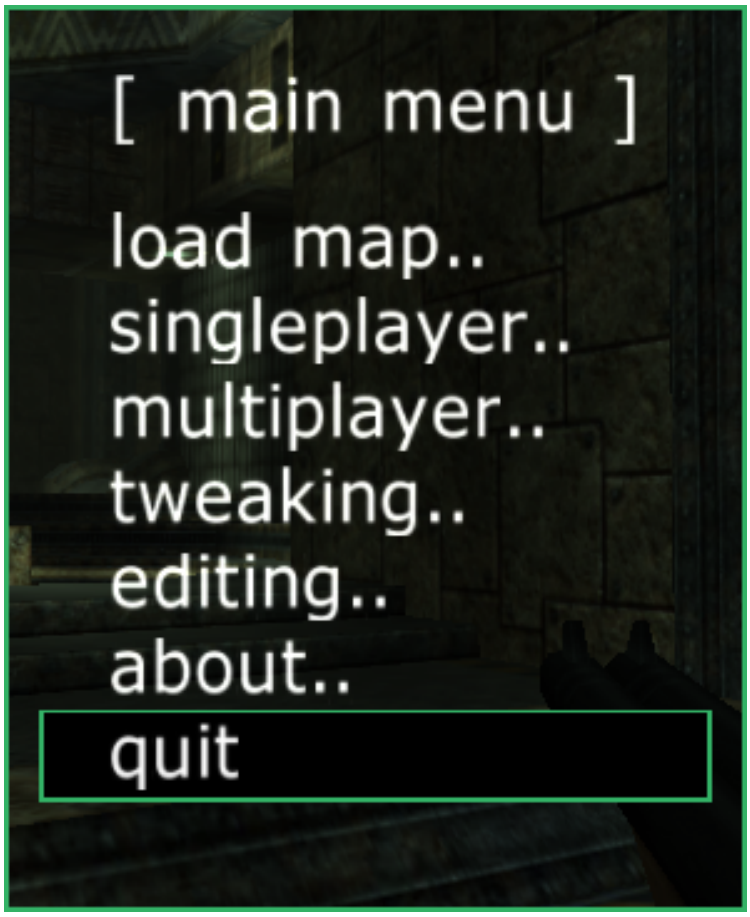
Related Video

Wwise-101-01 - Integrating a Sound

Play the Game!

Now it's time to see the result of your work in the actual game. To hear the changes you've made, you'll need to re-launch Cube, so you must first quit the currently running game.

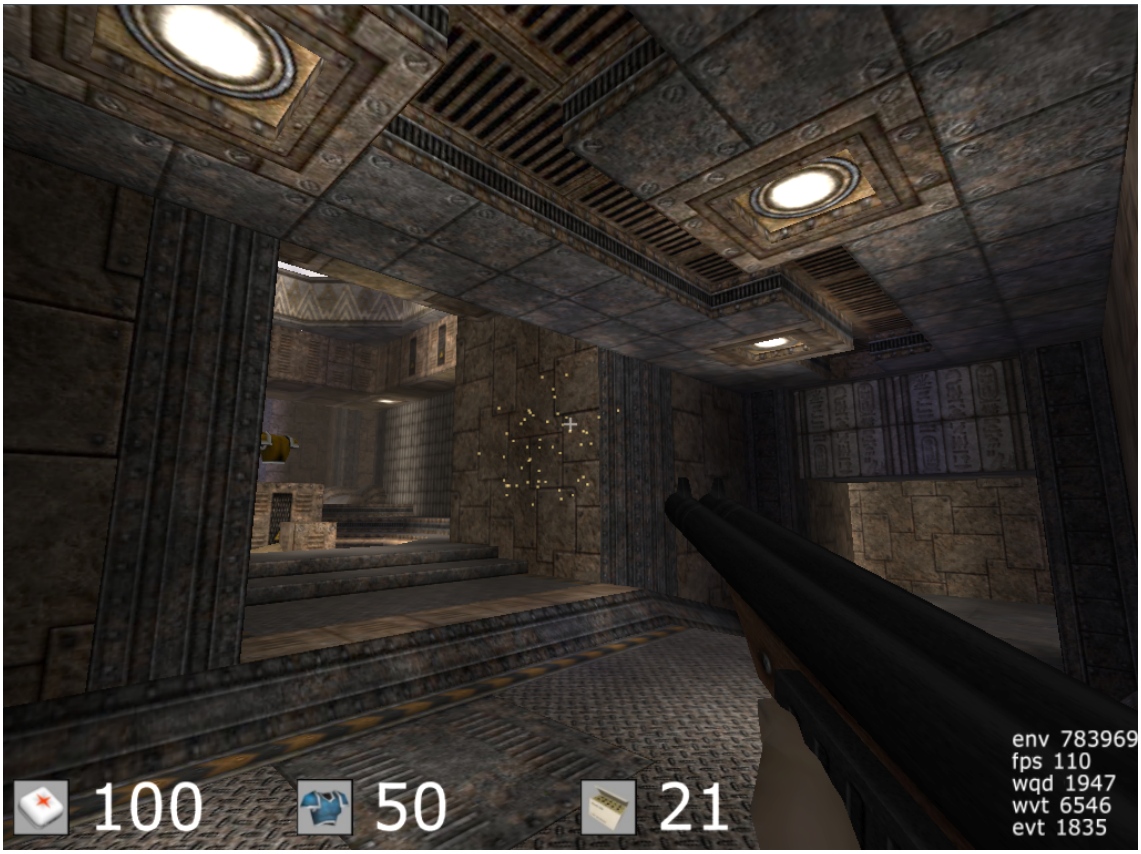
1. Return to Cube, press the **Esc** key and then use the arrow keys on your keyboard to select quit, and press **Enter**.



2. Open Cube once again.

   This time the game will open with the SoundBank you just generated.

3. Click to fire the shotgun.



You now hear the shotgun sound play within Cube.

Congratulations! If you can implement one sound into a game with Wwise, you'll have no problem implementing hundreds, or even thousands! There is certainly much to learn, but you now know the fundamentals of how an event is connected to a sound, and how a SoundBank builds the code to make it work within the game. Even better, you applied those fundamentals and heard the results for yourself. In the next lesson, you'll learn how to get a lot more sound out of that shotgun event you just built!