



# Sound, Respawn and Game Over Screen

Steven Harris

# TABLE OF CONTENTS

|  |   |
|--|---|
| TABLE OF CONTENTS.....                 | 2 |
| 1. Adding a Sound .....                | 3 |
| 2. A Jumping Sound Effect .....        | 4 |
| 3. A Character Checkpoint System ..... | 5 |
| 4. Exit & Restart the Game .....       | 6 |
| 5. A Game Over Screen.....             | 7 |

## 1. ADDING A SOUND

You can download various free game sound effects from <https://www.freesound.org/> For example, searching for the keyword: *jump* , will return classic jump sound effects for 2Dplatform games. The best format for importing audio files is the WAV format.

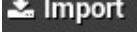
The Unreal Engine documentation for Sound and Audio are here:

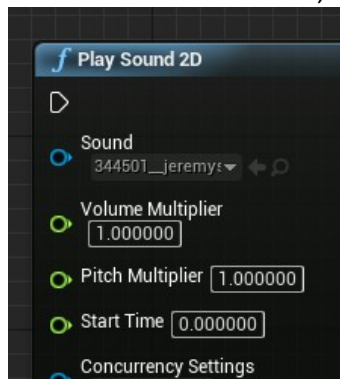
<https://docs.unrealengine.com/latest/INT/Engine/Audio/index.html>


Extra information on importing audio files is here:

<https://docs.unrealengine.com/latest/INT/Engine/Content/ImportingContent/ImportingAudio/>

To add a sound effect:

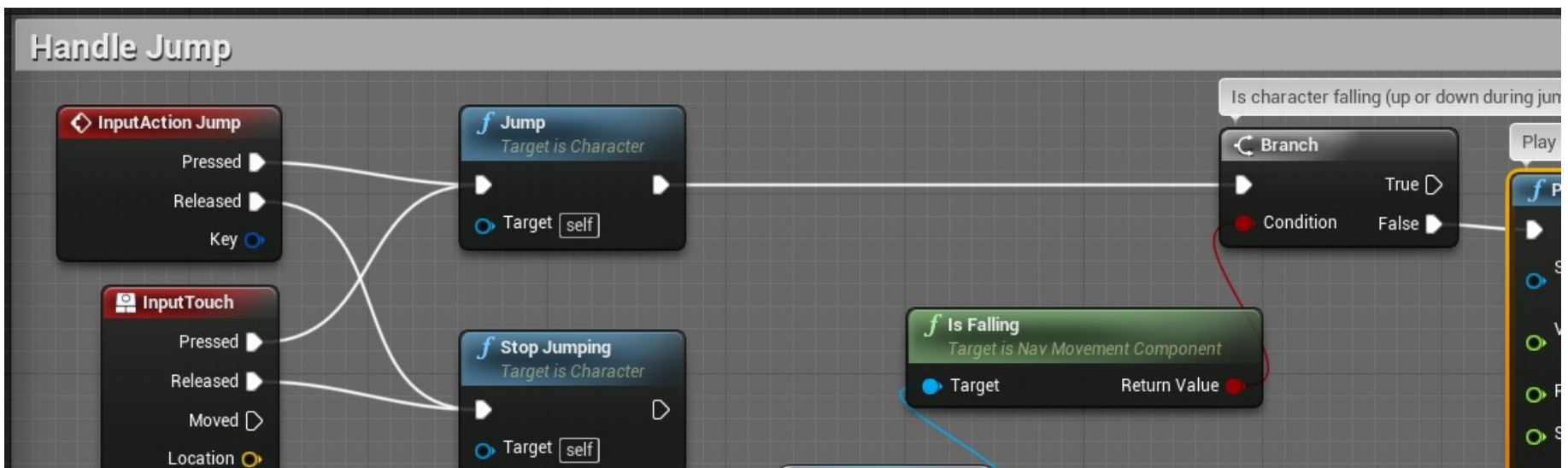
1. In the CONTENT BROWSER, click the IMPORT button: 
2. Select the wav file then click OPEN.
3. This will add the audio file to the CONTENT BROWSER.
4. In the blueprint where you want to add the sound effect, add a PLAY SOUND 2D node.



5.  In the SOUND pin, click the drop down box and select the sound which you just imported to the CONTENT BROWSER.
6. In the SOUND pin, click the drop down box and select the sound which you just imported to the CONTENT BROWSER.

## 2. A JUMPING SOUND EFFECT

When the PLAY SOUND 2D effect is connected to a blueprint sequence, it will play the sound every time the node is activated. If you wish to add a sound to the jumping action of your character, then the sound will play every time you hit space bar, even if the character is in mid air. To avoid this you can check to see if the character is falling. A character is considered to be 'falling' even when it is on the upwards part of the jump ! See the sample code below which can be added to the 2DSideScrollingCharacter blueprint.

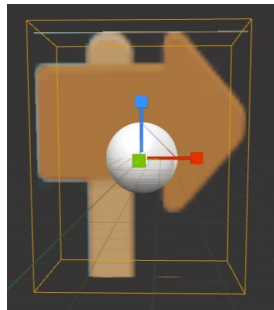


However, this code will need enhancing, as it will not work correctly when the player is within the ladder volume, as per the previous ladder tutorial. You should add a check to record when the player is in the ladder volume and then only play the sound when the character is NOT in the volume.

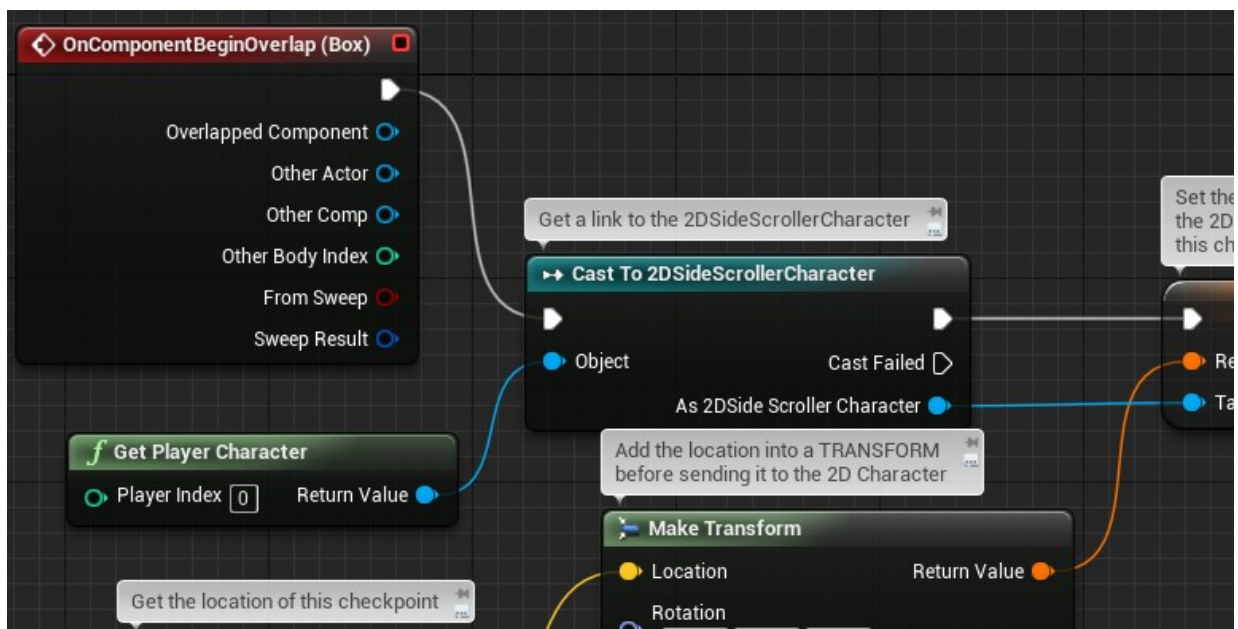
### 3. A CHARACTER CHECKPOINT SYSTEM

To add the ability for a player to respawn at a previous checkpoint the following scripts will provide a basic working system.

1. In the 2DSideScrollerCharacter blueprint, add a variable called RespawnLocation of a type TRANSFORM. This will store the location of the last checkpoint the character touched.
2. Create a new Blueprint, add a 2D Sprite to act as the checkpoint. In the COLLISION section of the details tab for the sprite, set COLLISION PRESET to: NoCollision. This will stop the collision detection and allow the character to overlap the sprite.
3. Add a collision box to over the sprite as below.

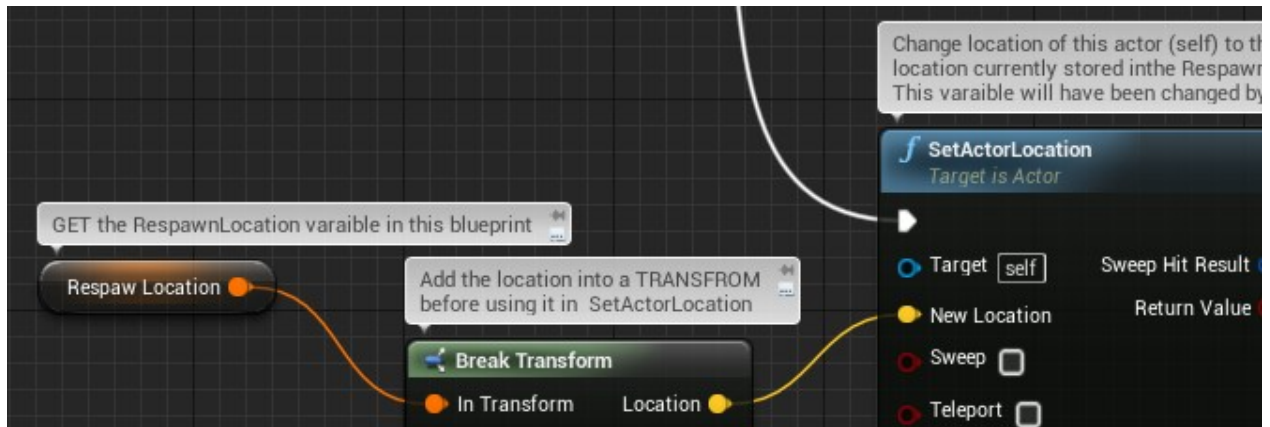


4. In the VIEWPORT tab of the checkpoint actor, select the COLLISION BOX and then go to the EVENT GRAPH tab and add an OnComponentBeginOverlap event for the collision box and then add the code below.



5. The above script will activate whenever the player touches the collision box around the checkpoint sprite. It will then access the RespawnLocation variable in the 2DSideScrollerCharacter and change it to be the location of this checkpoint actor. This ensures that the character knows the location of the last checkpoint which it passed.
6. You now have to decide how you want to trigger the respawn. This could happen when your health variable reaches zero, or perhaps when the player enters a collision box which is designed to kill the player. To force the character to respawn, simply change the

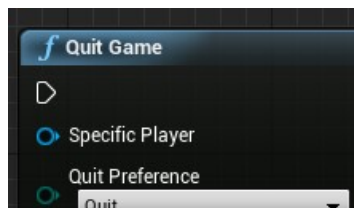
'location' of the player character. This would usually take place within the 2DSideScrollerCharacter blueprint, as below.



This basic script can be enhanced. For example, when the game starts, the RespawnLocation variable will not have a value, so it will respawn at (0,0,0). You should therefore setup an initial respawn location for when the game starts. You could do this by giving the RespawnLocation variable a default value in the DETAILS tab. This default value could be the location of the player start location. This would ensure that if the player is forced to respawn before reaching the first checkpoint, then it will respawn back where the player started the game.

#### 4. EXIT & RESTART THE GAME

To exit the game, use the QUIT GAME node.



To restart the game use the RESTART GAME node. When using this node, drag off the TARGET pin and search for the GET GAME MODE node, and ensure this is connected to the TARGET PIN, as below.



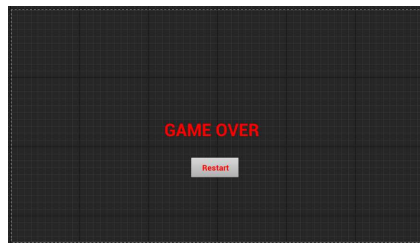
This action is the same as starting the game manually, so all your score\health\timer values and the player location will have their original values.

## 5. A GAME OVER SCREEN

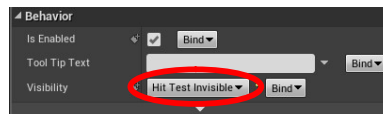
To add a game over screen, we use UMG, as we did for creating the health bar HUD. Using UMG, you can create a new layout for your game over screen, as you did with the Health bar. When the game end state is achieved (ie health= 0) you can display this UMG HUD over the game.

Follow the steps below:

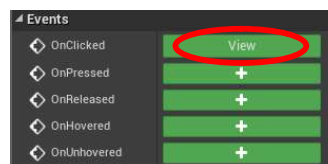
1. In the CONTENT BROWSER, click the ADD NEW button and select USER INTERFACE then WIDGET BLUEPRINT. This will add a new UMG Widget. DO NOT add your game over screen into your existing health bar HUD.
2. Double click this new UMG Widget to open it. Then add a TEXT widget, and a BUTTON as below. Then add another TEXT widget over the button to act as the text on the button.



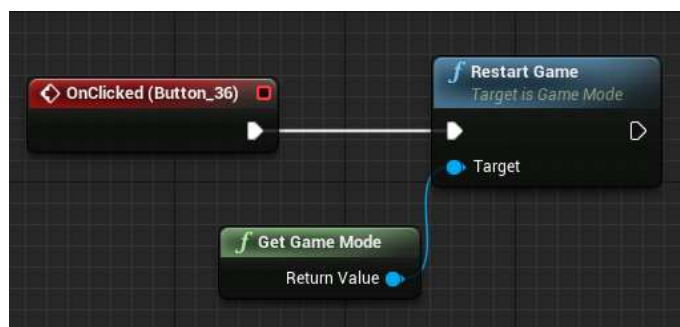
3. For both the TEXT items, select the text item on the workspace, and change the VISIBILITY setting to HIT TEST INVISIBLE, as below. This will ensure that both text items will ignore the mouse if the player clicks on them. This in turn allows the user to click through the text, onto the button.



4. We now need to add some code to the button, so when it is clicked, it will restart the game. Select the button widget. In the DETAILS tab, scroll down and click the green button next to the ONCLICKED event, as below.



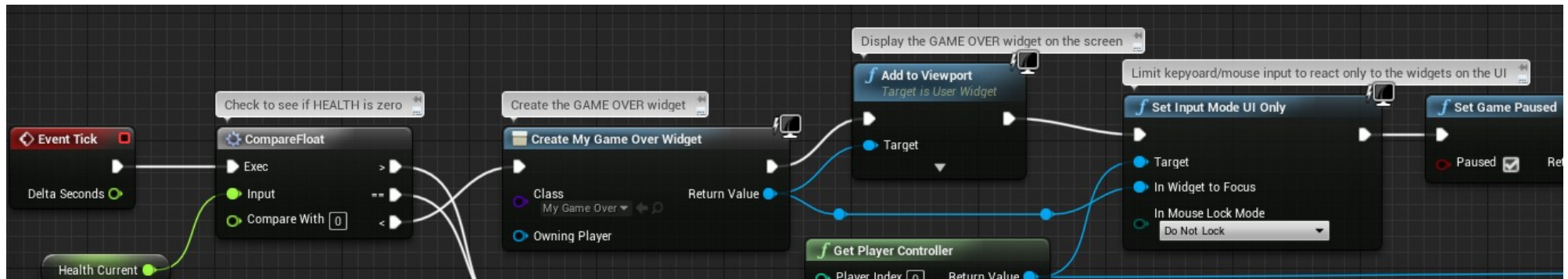
5. This will take you to the EVENT GRAPH. An ON CLICKED event for the button will have been created for you. From this event, add the code below, to restart the game.



6. We now have a HUD with a clickable button, which will restart the game when clicked. We now need to display this HUD at the appropriate time.



7. Go to the 2DSideScrollerCharacter blueprint. Find the EVENT TICK node. This will already be connected to some other code, but you can insert the code below in-between.



8. The code above runs off the EVENT TICK event, so will execute every frame of the game. The code checks to see if the health is zero (the Health variable is in the 2DSideScrollerCharacter blueprint. The same blueprint as this code). The next two nodes create the widget and add it to the screen, as we have seen in the previous notes on adding a health bar. In the CREATE WIDGET node, select your GAME OVER widget from the CLASS dropdown list. The code then restricts input to just the UI (ie the widgets). Input will then be ignored for the character, thus preventing the character from moving. The game is also paused as an extra precaution. Finally, the mouse cursor is made visible, so the player can click the button.
9. We now need to reverse the above code when the game is restarted. To do this, find the EVENT BEGIN PLAY node in the 2DSideScrollerCharacter blueprint. This will already be connected to some other code, but you can insert the code below in-between. This code un-pauses the game, and removes the GAME OVER widget from the screen. The input is returned to the game, and the mouse cursor is hidden. This means the character can now receive keyboard input and start to move again.

