Adding Sound Effects to Your Game

Knox Game Design

July 2021

Levi D. Smith

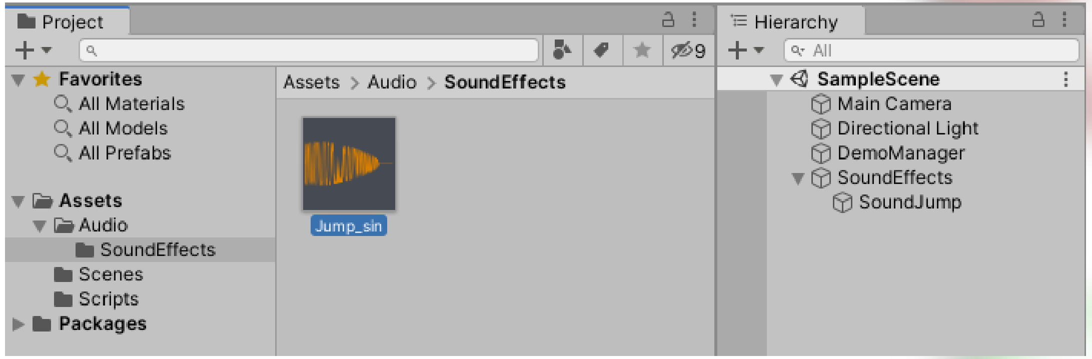
Overview

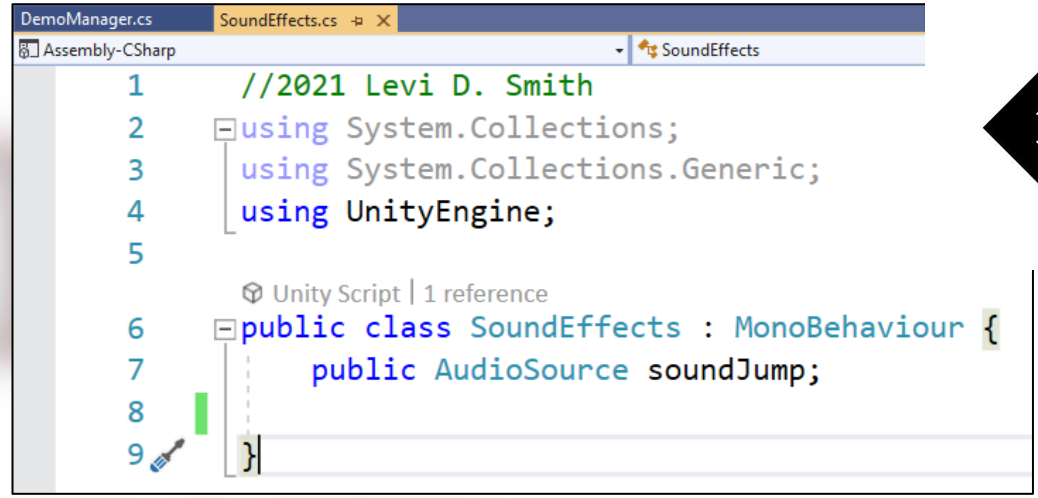
- How to play a sound when pressing a button
- See Knox Game Design June 2021 for creating sound effects with BFXR
- Game development environments
  - Unity
  - GameMaker
  - Godot
  - Unreal Engine
  - Pico-8

Unity

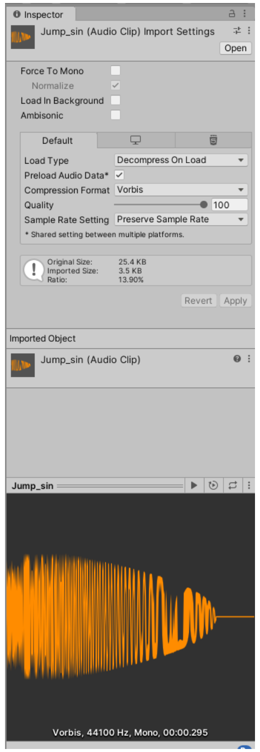
- Under Assets
  - Create Scripts folder
  - Create Audio/SoundEffects folder
- Drag sound effect file (such as .WAV) to Audio/SoundEffects folder
- Create new Empty GameObject called **SoundEffects**
- Create new Empty GameObject called **SoundJump** as a child
- Drag the sound object from Assets/Audio/SoundEffects to **SoundJump**
- Uncheck *Play on Awake*
- Create **SoundEffects** script and assign to **SoundEffects** GameObject
- Create public AudioSource instance variable called **soundJump** for the sound effect
- Drag the **SoundJump** GameObject to the **SoundJump** property of the **SoundEffects** script in the inspector
- Create Empty GameObject called **DemoManager**
- Create DemoManager C# script and assign to the **DemoManager** GameObject
- Create public **SoundEffects** instance variable in the **DemoManager** script
- Assign the **SoundEffects** GameObject to the **SoundEffects** property in the Inspector
- In the *Update* method, check for **Input.GetButtonDown("Jump")**
- Play the sound effect with **soundeffects.jumpSound.Play()**

Unity





```
//2021 Levi D. Smith
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SoundEffects : MonoBehaviour {
    public AudioSource soundJump;

}
```



Jump_sin (Audio Clip) Import Settings

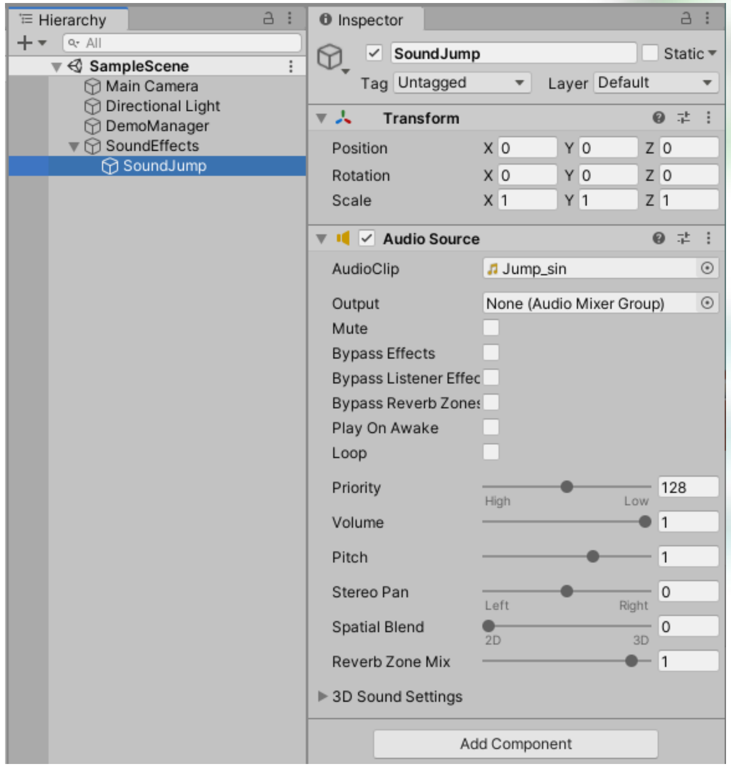
Force To Mono □
Normalize ☑
Load In Background □
Ambisonic □

Load Type: Decompress On Load
Preload Audio Data* ☑
Compression Format: Vorbis
Quality: 100
Sample Rate Setting: Preserve Sample Rate
* Shared setting between multiple platforms.

Original Size: 25.4 KB
Imported Size: 3.5 KB
Ratio: 13.90%

Jump_sin (Audio Clip)

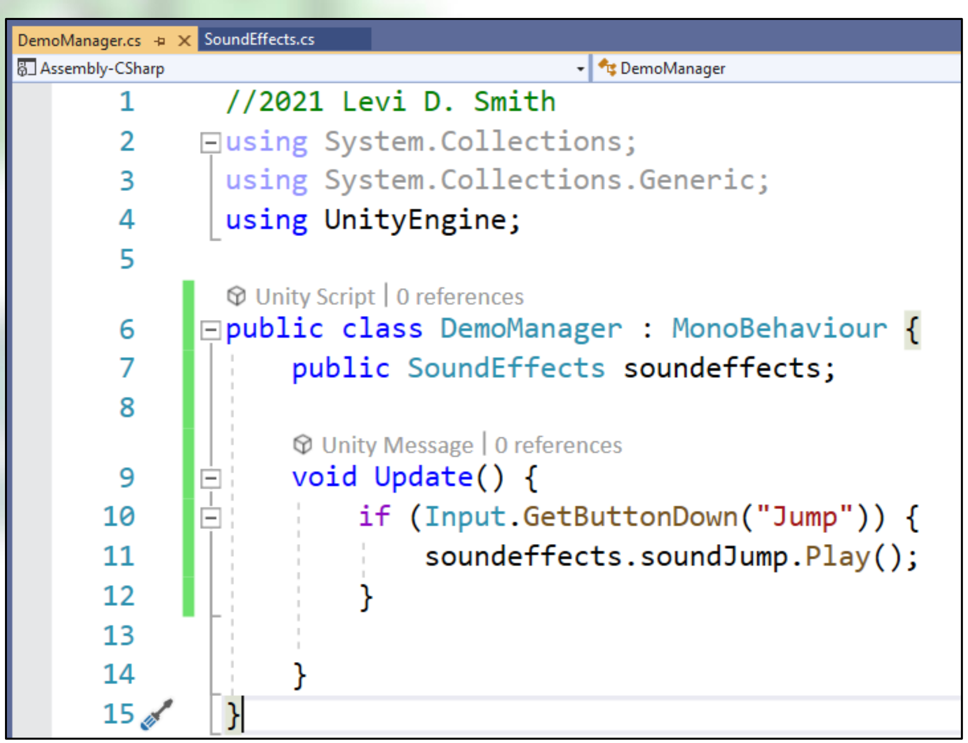
Vorbis, 44100 Hz, Mono, 00:00.295



☑ SoundJump □ Static
Tag: Untagged Layer: Default

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

☑ Audio Source
AudioClip: Jump_sin
Output: None (Audio Mixer Group)
Mute □
Bypass Effects □
Bypass Listener Effec □
Bypass Reverb Zones □
Play On Awake □
Loop □
Priority: 128 (High – Low)
Volume: 1
Pitch: 1
Stereo Pan: 0 (Left – Right)
Spatial Blend: 0 (2D – 3D)
Reverb Zone Mix: 1
3D Sound Settings



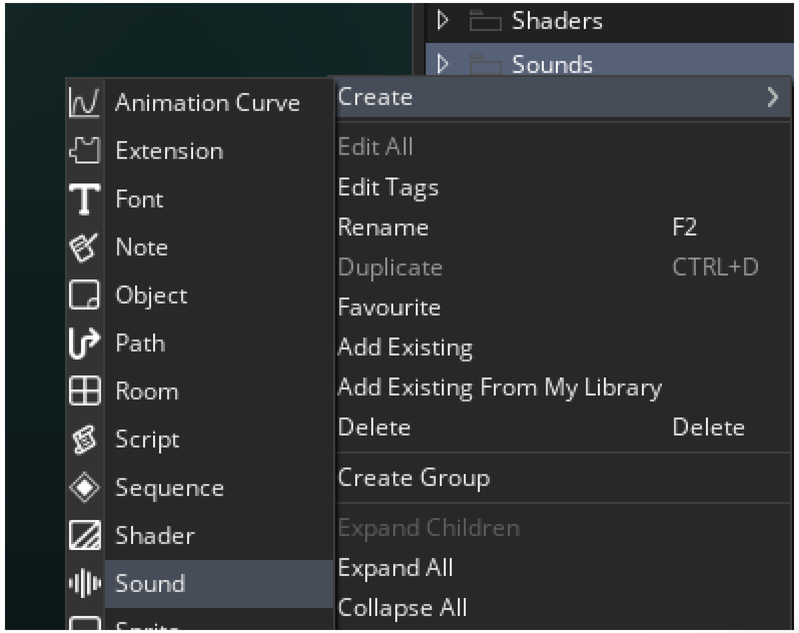
```
//2021 Levi D. Smith
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DemoManager : MonoBehaviour {
    public SoundEffects soundeffects;

    void Update() {
        if (Input.GetButtonDown("Jump")) {
            soundeffects.soundJump.Play();
        }

    }
}
```

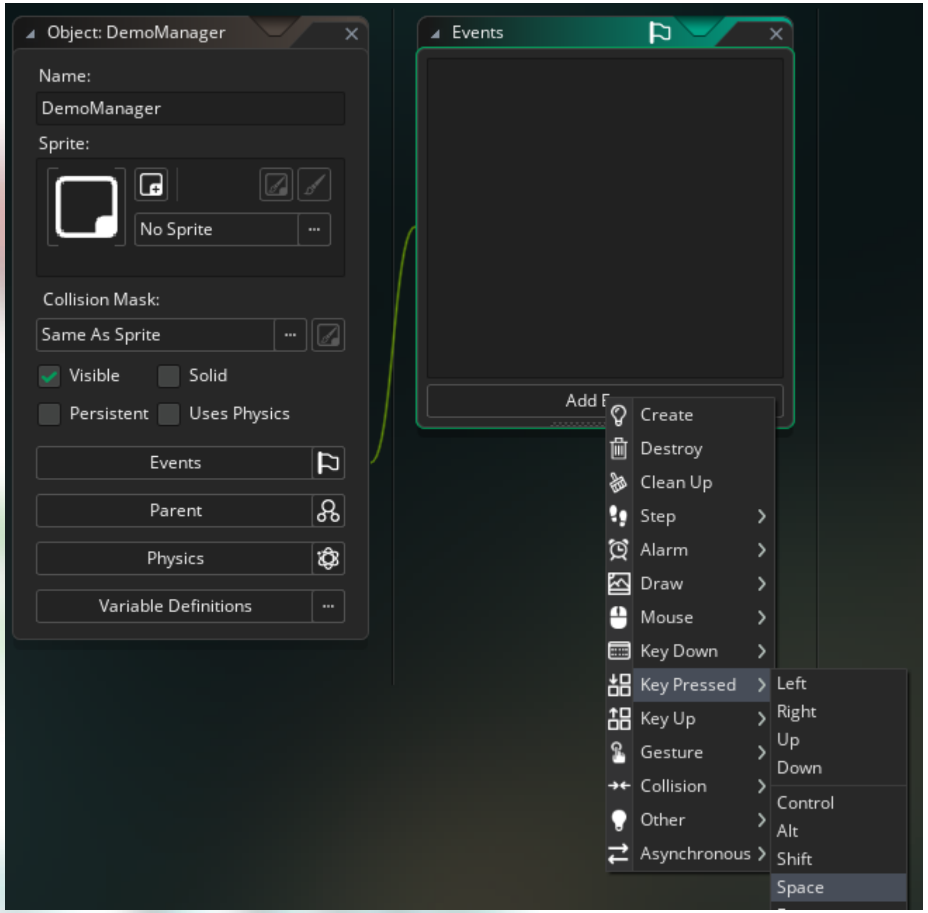
KNOX
GAME
DESIGN

GameMaker

- Right click *Sounds > Create > Sound*
- Name the new sound **soundJump**
- Click the three dot button to select the sound file (such as .WAV)
- Create a new Object called **DemoManager**
- Open **Room1** and drag an instance of **DemoManager** onto the grid
- On the **DemoManager** block, press *Add Event* under *Events*
- Select *KeyPress > Space*
- Play the sound with **audio_play_sound(soundJump, 10, false)**

GameMaker




Object: DemoManager

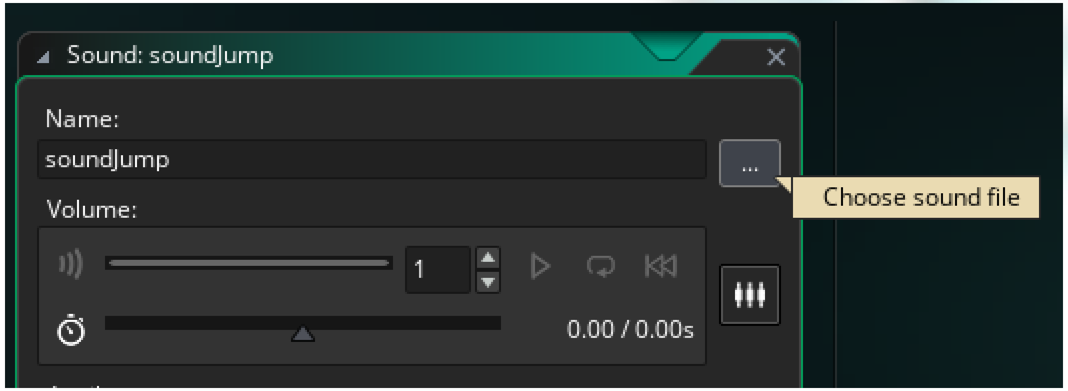
Name: DemoManager

Sprite: No Sprite

Collision Mask: Same As Sprite

☑ Visible ☐ Solid

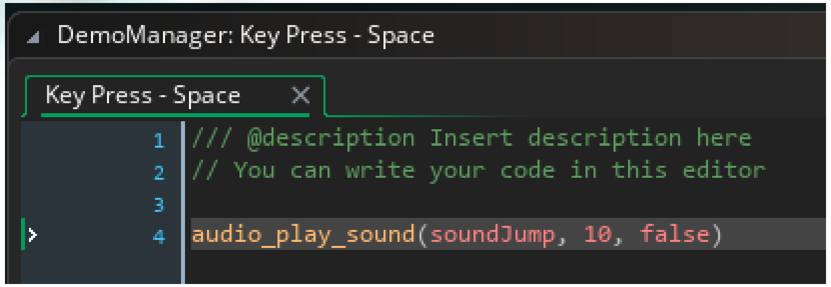
☐ Persistent ☐ Uses Physics


Sound: soundJump

Name: soundJump

Volume: 1

0.00 / 0.00s


DemoManager: Key Press - Space

Key Press - Space

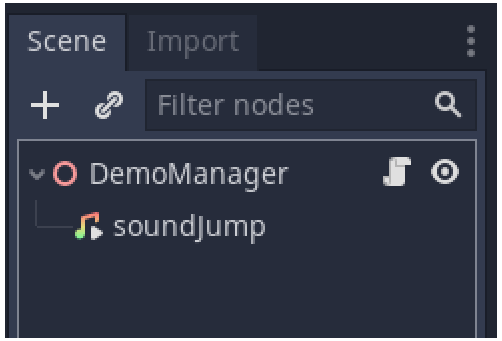
```
/// @description Insert description here
// You can write your code in this editor

audio_play_sound(soundJump, 10, false)
```

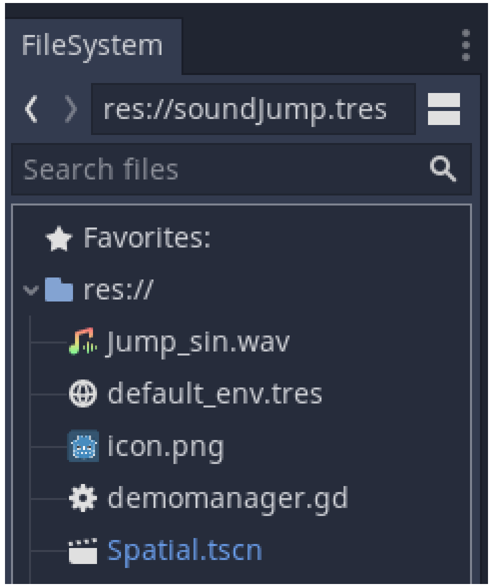
KNOX GAME DESIGN

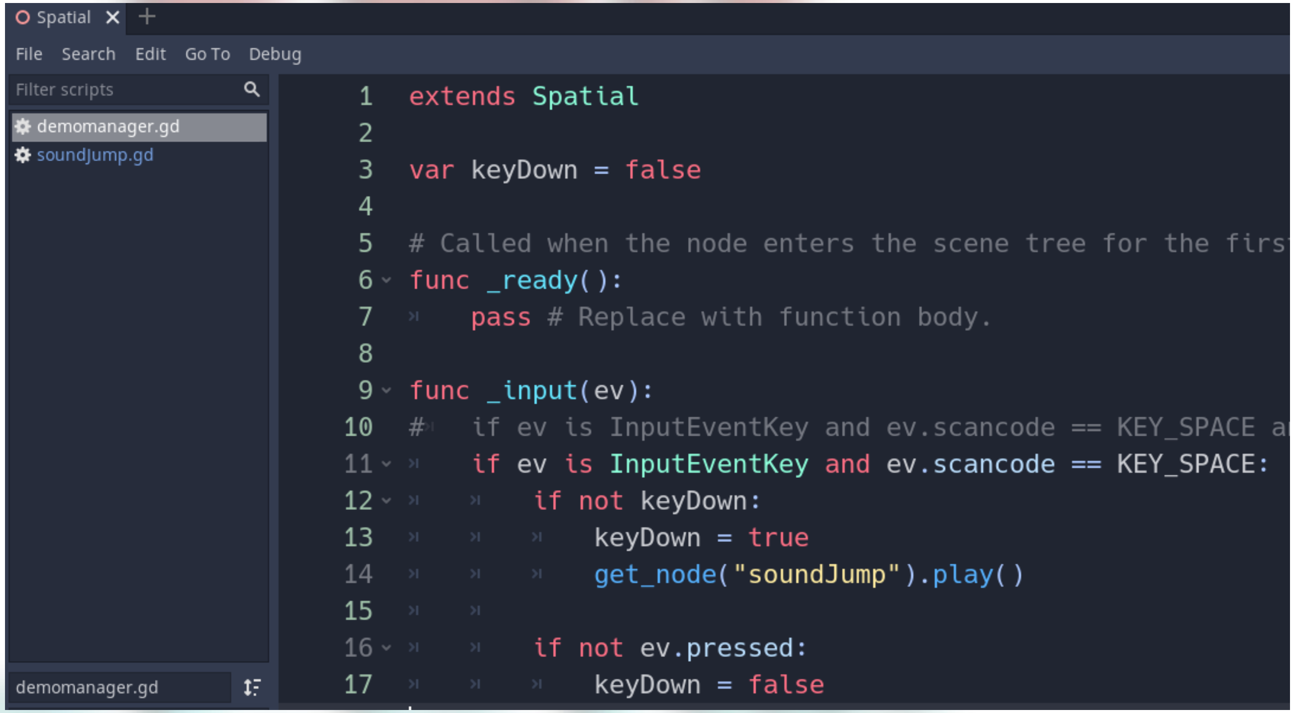
Godot

- Drag sound file (.WAV) into resource area
- Create new Spatial Node called **DemoManager**
- Create child node type *AudioStreamPlayer* called **soundJump**
- *Attach Script* on **DemoManager** node
- Open and edit **demomanager.gd** script
  - Add variable called **keyDown** and set to *false*
  - Add function *_input(ev)*
  - If event is key event and is jump key and **keyDown** is false
    - Play sound effect with **get_node("soundJump").play()**
    - Set **keyDown** to true
  - If key is not pressed (The key was released. Otherwise sound will play on key press and release))
    - Set **keyDown** to false

Godot







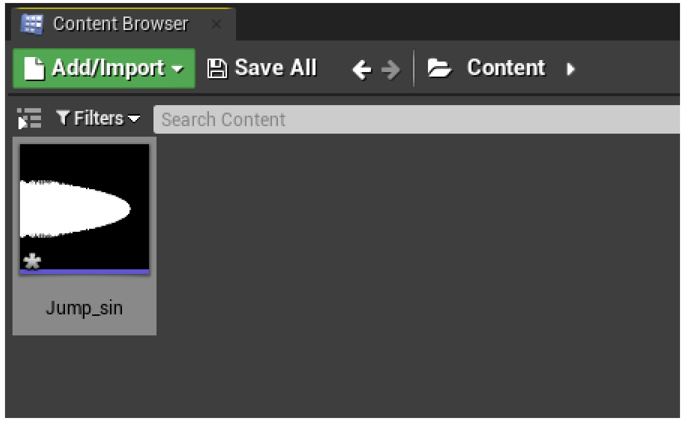
```
extends Spatial

var keyDown = false

# Called when the node enters the scene tree for the firs
func _ready():
	pass # Replace with function body.

func _input(ev):
#	if ev is InputEventKey and ev.scancode == KEY_SPACE a
	if ev is InputEventKey and ev.scancode == KEY_SPACE:
		if not keyDown:
			keyDown = true
			get_node("soundJump").play()
			
		if not ev.pressed:
			keyDown = false
```

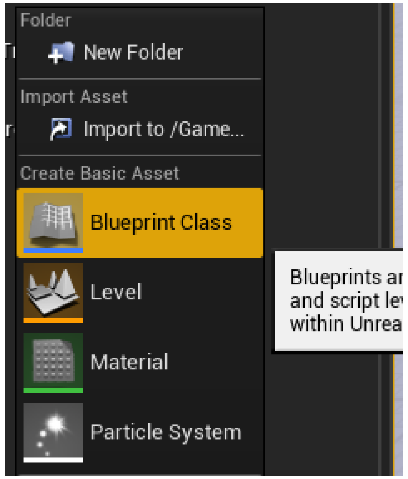
KNOX
GAME
DESIGN

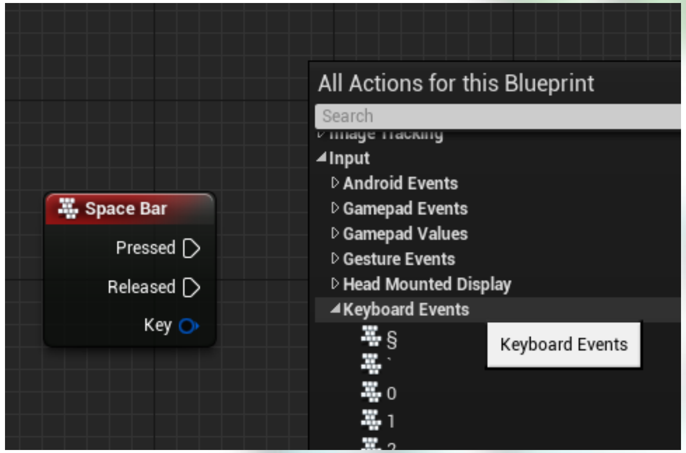
Unreal Engine (Blueprints)

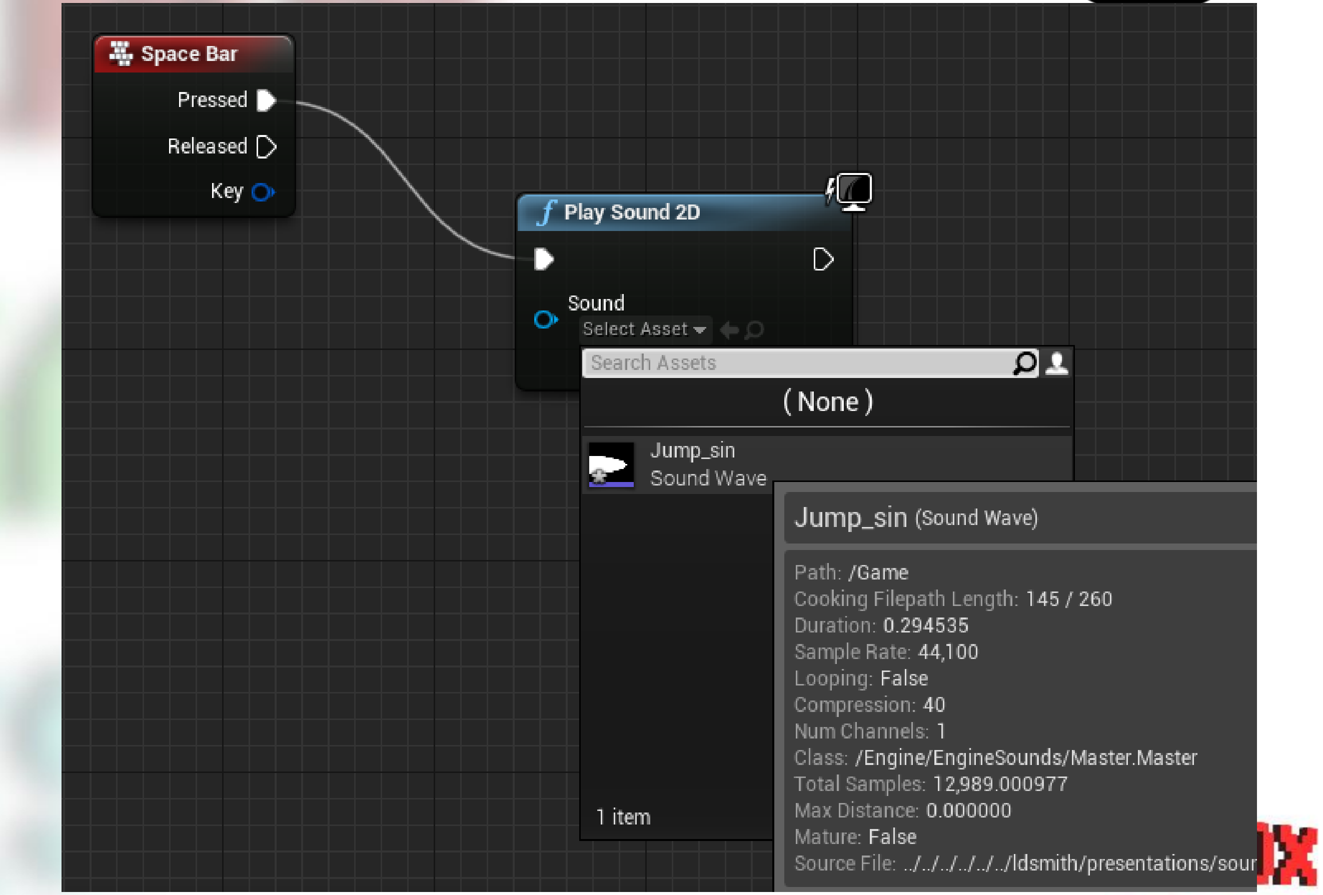
- Drag sound file (.WAV) from explorer to Unreal *Content Browser*
- Right click and create a new BluePrint class and call it **DemoManager**
- Drag the **DemoManager** from the *Content Browser* into the scene
- Select **DemoManager** in the *World Outliner*
- Set *Auto Receive Input* to *Player 0*
- Press *Edit Blueprint*, Select *Open Blueprint Editor* then *Full Blueprint Editor*
- Right click, expand *Input*, expand *Keyboard Events*, select *Space Bar*
- Right click, expand *Audio,* select *Play Sound 2D*
- Select the jump sound asset in the *Sound* dropdown on the *Play Sound 2D* block
- Connect the *Space Bar Pressed* output with the *Play Sound 2D block*
- Press the *Play* button on the main Unreal window. Click in the scene view for the keypress to register

Unreal Engine (Blueprints)









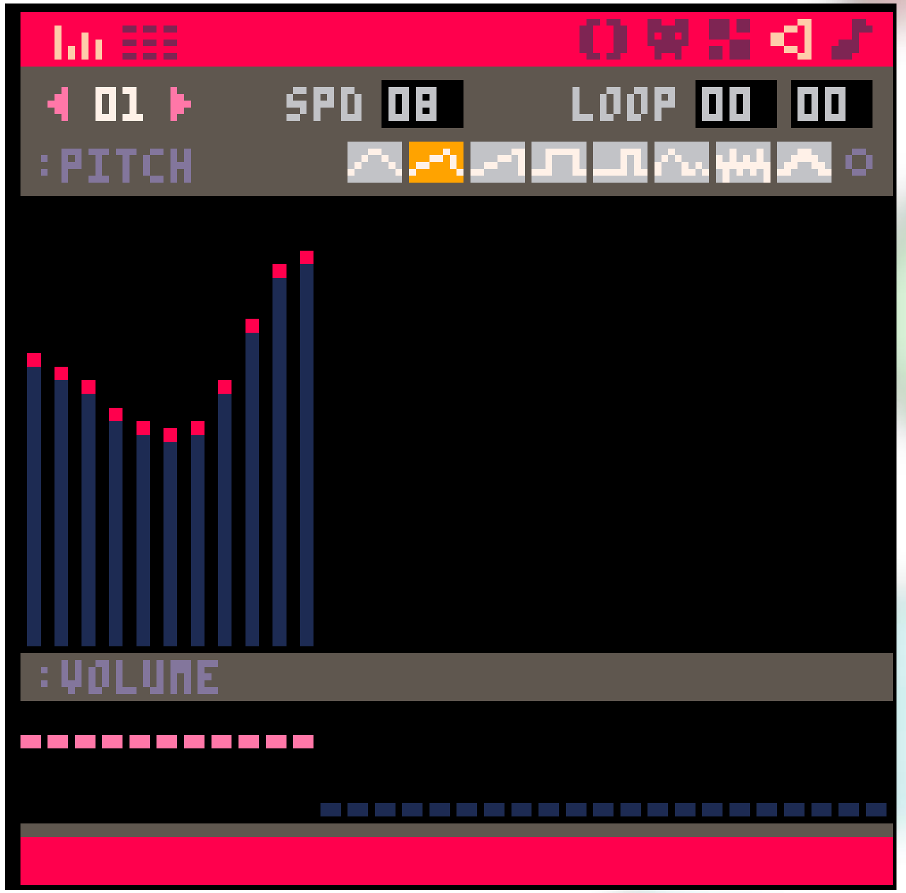
Jump_sin (Sound Wave)

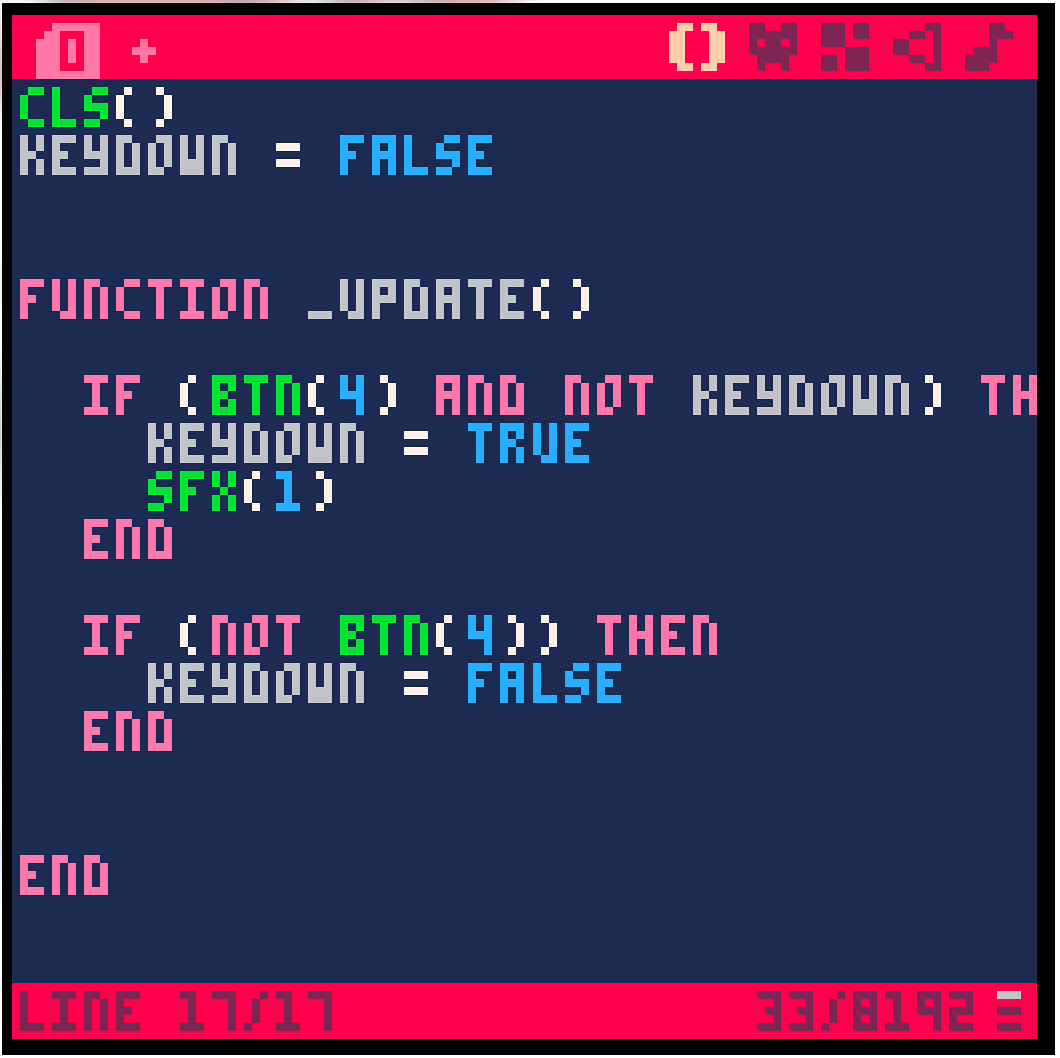
Path: /Game
Cooking Filepath Length: 145 / 260
Duration: 0.294535
Sample Rate: 44,100
Looping: False
Compression: 40
Num Channels: 1
Class: /Engine/EngineSounds/Master.Master
Total Samples: 12,989.000977
Max Distance: 0.000000
Mature: False
Source File: ../../../../../../ldsmith/presentations/sour

Pico-8

- Create sound effect in sound editor
- Create **keydown** variable and set it to **false**
- Create _*update* function
- If button is down and **keydown** variable is **false**
  - Set **keydown** variable to **true** (this is needed so that the sound effect only plays once)
  - Play sound with **SFX(1)** (where 1 is the index of the sound effect)
- If button is not down
  - Set **keydown** to **false**

Pico-8





```
CLS()
KEYDOWN = FALSE


FUNCTION _UPDATE()

  IF (BTN(4) AND NOT KEYDOWN) TH
    KEYDOWN = TRUE
    SFX(1)
  END

  IF (NOT BTN(4)) THEN
    KEYDOWN = FALSE
  END


END
```

LINE 17/17 33/8192

KNOX GAME DESIGN