1/2563

ชื่อ นายปาล์เมษ เกุรักษา

รหัสนักศึกษา 612110055

ตอนที่ 001

สาขาวิชาแอนนิเมชันและเกม วิทยาลัยศิลปะ สื่อ และเทคโนโลยี มหาวิทยาลัยเชียงใหม่


# เนื้อหา (Contents)

# บทที่ 10

# เสียงเพลงและเสียงประกอบ (Music and Sound Effect)

**เงื่อนไขที่ต้องผ่านก่อน (Pre-requisites)**

1. ปฏิบัติการ การจัดการฉาก (Scenes Management)

**วัตถุประสงค์ (Objectives)**

1. ศึกษาการใช้งานเสียงเพลง (Music)
2. ศึกษาการใช้งานเสียงประกอบ (Sound Effect)

## 10.1 บทนำ (Introduction)

บทนี้ศึกษาเรื่องการใส่เสียงเพลงประกอบหลังฉาก (Background music) และเสียงประกอบ (Sound effect) เพื่อนำมาประยุกต์ใช้เพิ่มความน่าสนใจให้กับเกม

## 10.2 พัฒนาโครงการโดยใช้ต้นแบบจากปฏิบัติการเรื่อง “การจัดการฉาก (Scenes Management)”

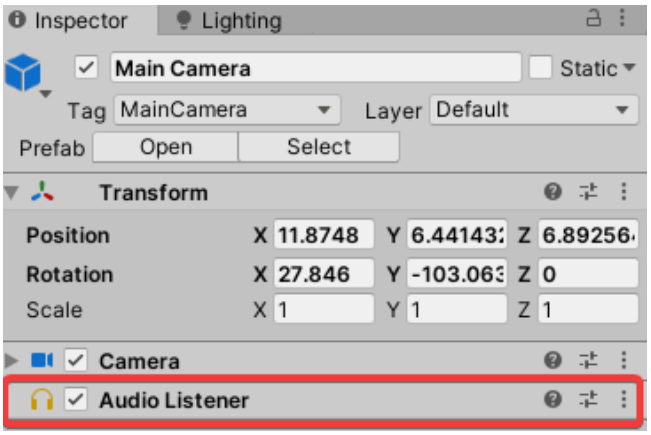
ให้นักศึกษาใช้โครงการเดิมจากปฏิบัติการเรื่อง “การจัดการฉาก (Scenes Management)” หรือสร้างโครงการขึ้นมาใหม่โดยให้มีลักษณะเดียวกับ ปฏิบัติการเรื่อง “การจัดการฉาก” เนื่องจากจำเป็นต้องประยุกต์ใช้การเชื่อมต่อฉากหลายฉาก และฉากที่ประกอบไปด้วยเมนูต่าง ๆ เพื่อใช้ ในการเล่นเสียงเพลงและเสียงประกอบขณะเลือกเมนู อีกทั้งยังใช้ในการกำหนดการตั้งค่าเปิดหรือปิดเสียงเพลงและเสียงประกอบ

## 10.3 องค์ประกอบต่าง ๆ เกี่ยวกับเสียง (Audio Listener, Audio Source, Audio Clip, and Audio Mixer)

องค์ประกอบต่าง ๆ ที่เกี่ยวข้องกับการใช้งานเสียงเพลงและเสียงประกอบใน Unity ประกอบไปด้วย 4 ส่วนคือ ผู้รับฟัง (Audio Listener) ต้นกำเนิดเสียง (Audio Source) ไฟล์เสียง (Audio Clip) และตัวผสมเสียง (Audio Mixer)

### 10.3.1 ผู้รับฟังเสียง (Audio Listener)

Audio Listener หรือผู้รับฟังเสียง ที่จะถ่ายทอดเสียงมาสู่ผู้เล่นเกมนั่นเอง ซึ่งโดยปกติแล้วคอมโพเนนท์ Audio Listener จะถูกติดตั้งให้กับกล้องหลักโดยอัตโนมัติดังรูปที่ 10.1

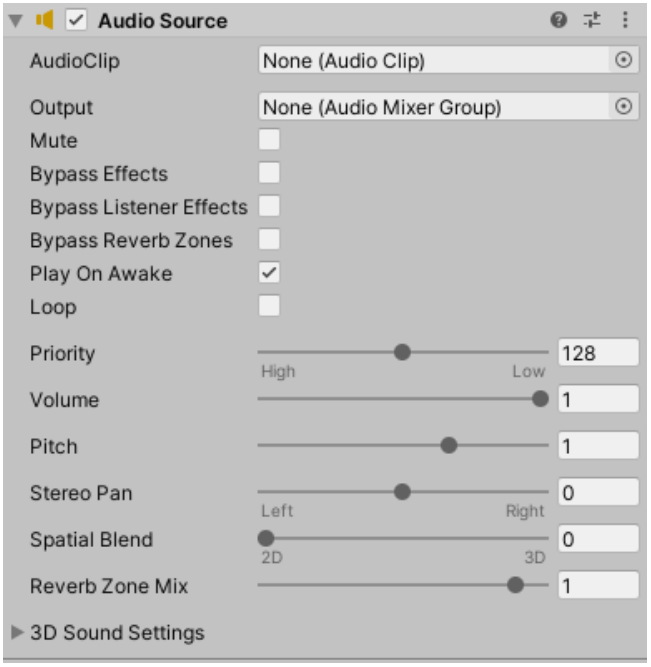



รูปที่ 10.1

### 10.3.2 ต้นกำเนิดเสียง (Audio Source)

เพื่อทำให้เกิดเป็นเสียงในระบบสามมิติ ผู้พัฒนาสามารถติดตั้งคอมโพเนนท์ Audio Source ให้กับวัตถุเกมใด ๆ ในฉาก เพื่อเป็นการกำหนดให้เป็นแหล่งกำเนิดเสียง

รูปที่ 10.2 แสดงคอมโพเนนท์ Audio Source ที่ถูกติดตั้งให้กับวัตถุเกมวัตถุหนึ่ง โดยมีคุณสมบัติหลักที่สำคัญคือ AudioClip คือแหล่งไฟล์เสียง และ Output เป็นการกำหนดช่องทางออกของเสียงไปยังช่องสัญญาณของตัวผสมเสียง (Audio Mixer)

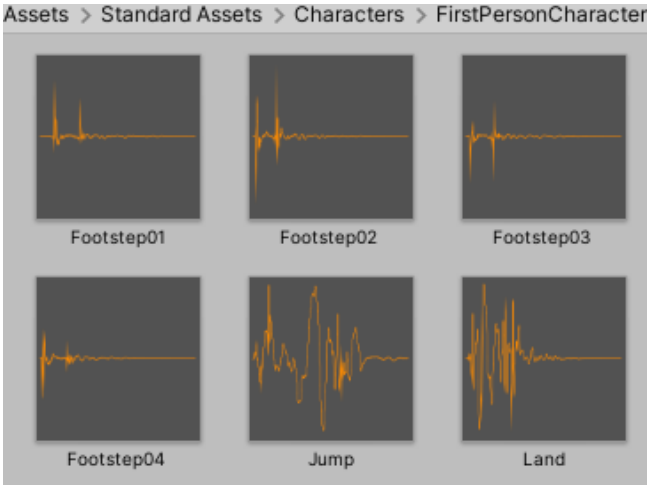



รูปที่ 10.2

### 10.3.3 ไฟล์เสียง (Audio Clip)

Audio Clip คือไฟล์เสียงในรูปแบบต่าง ๆ ที่ Unity สนับสนุน ซึ่งจะถูกนำเข้ามาสู่โครงการในหน้าต่าง Assets เพื่อเตรียมพร้อมสำหรับนำไปติดตั้งกำหนดให้กับคุณสมบัติ Audio Clip ของต้นกำเนิดเสียง (Audio Source) นั่นเอง

ตัวอย่างไฟล์เสียงจาก Unity's Standard Assets แสดงรูปที่ 10.3




รูปที่ 10.3

### 10.3.4 ตัวผสมเสียง (Audio Mixer)

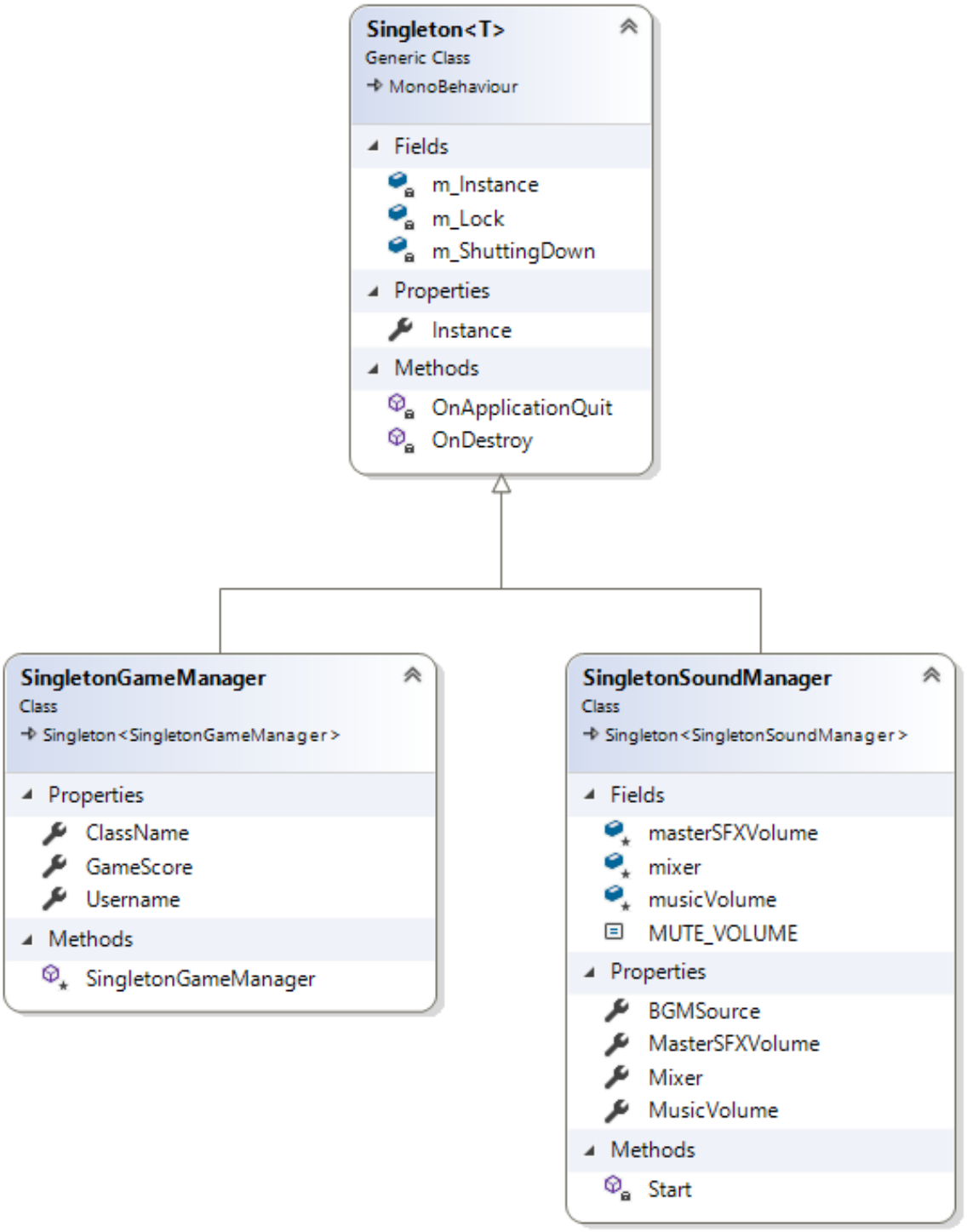
ตัวผสมเสียง (Audio Mixer) เป็นตัวจัดการการผสมเสียงในขั้นตอนสุดท้ายก่อนที่เสียงจะออกสู่ผู้รับฟัง โดยผู้ใช้งานสามารถกำหนดช่องสัญญาณ กลุ่มของสัญญาณเสียง อาทิ เสียงประกอบ เสียงเอฟเฟค์ ได้ตามต้องการ ดังตัวอย่าง รูปที่ 10.9

## 10.4 สคริปต์คอมโพเนนท์สำหรับจัดการเสียงส่วนกลาง (SingletonSoundManager)

เขียนคลาสแม่แบบ Singleton<T> ดัง Sourcecode 10.1 สำหรับคอมโพเนนท์แบบ Singleton

หากเคยเขียนไว้แล้วในบทเรียนเรื่องการจัดการฉาก ไม่จำเป็นต้องเขียนขึ้นใหม่ เพียงเขียนคอมเมนต์บรรทัดที่ 58 เพิ่มเติมลงไป

รูปที่ 10.4 แสดงคลาสไดอะแกรมของคลาสแม่แบบ Singleton<T> และคลาสที่สืบทอดทั้งสองคือ SingletonGameManager และ SingletonSoundManager เพื่อการประยุกต์ใช้งานแบบ Singleton




รูปที่ 10.4: คลาสไดอะแกรม SingletonSoundManager

Source code 10.1: Singleton.cs

```
1 using UnityEngine;
```

```
2
3  public class Singleton<T> : MonoBehaviour where T : MonoBehaviour
4  {
5      // Check to see if we're about to be destroyed.
6      private static bool m_ShuttingDown = false;
7      private static object m_Lock = new object();
8      private static T m_Instance;
9
10     /// <summary>
11     /// Access singleton instance through this propriety.
12     /// </summary>
13     public static T Instance
14     {
15         get
16         {
17             if (m_ShuttingDown)
18             {
19                 Debug.LogWarning("[Singleton] Instance '" + typeof(T) +
20                     "' already destroyed. Returning null.");
21                 return null;
22             }
23
24             lock (m_Lock)
25             {
26                 if (m_Instance == null)
27                 {
28                     // Search for existing instance.
29                     m_Instance = (T)FindObjectOfType(typeof(T));
30
31                     // Create new instance if one doesn't already exist.
32                     if (m_Instance == null)
33                     {
34                         // Need to create a new GameObject to attach the singleton to.
35                         var singletonObject = new GameObject();
36                         m_Instance = singletonObject.AddComponent<T>();
37                         singletonObject.name = typeof(T).ToString() + " (Singleton)";
38
39                         // Make instance persistent.
40                         DontDestroyOnLoad(singletonObject);
41                     }
```

```
                }

                return m_Instance;
            }
        }
    }


    private void OnApplicationQuit()
    {
        m_ShuttingDown = true;
    }


    private void OnDestroy()
    {
        //m_ShuttingDown = true;
    }
}
```

สร้างสคริปต์คอมโพเนนท์ในรูปแบบ Singleton เพื่อใช้งานสำหรับการจัดการเสียง SingletonSoundManager

สร้างวัตถุเปล่า (Create Empty) ตั้งชื่อว่า SoundManager เพื่อเป็นที่วางของสคริปต์ SingletonSoundManager ซึ่งเป็นคลาสแบบ singleton เพื่อให้เสียงเพลงสามารถเล่นได้อย่างต่อเนื่อง แม้ฉากจะถูกเปลี่ยนไป

สร้างและติดตั้งสคริปต์ SingletonSoundManager และเขียนซอสโคดตาม Sourcecode 10.2

Source code 10.2: SingletonSoundManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Audio;

public class SingletonSoundManager : Singleton<SingletonSoundManager>
{
    public const float MUTE_VOLUME = -80;

    [SerializeField]
    protected AudioMixer mixer;
    public AudioMixer Mixer { get { return mixer;} set { mixer = value; } }
```

```
13      public AudioSource BGMSource{get;set;}
14
15      #region Music Volume
16
17      public float MusicVolumeDefault{get;set;}
18
19      protected float musicVolume;
20      public float MusicVolume
21      {
22          get
23          {
24              return this.musicVolume;
25          }
26          set
27          {
28              this.musicVolume = value;
29              SingletonSoundManager.Instance.Mixer.SetFloat("MusicVolume", this.musicVolume
                    );
30          }
31      }
32
33      #endregion
34
35      #region SFX Volume
36      protected float masterSFXVolume;
37      public float MasterSFXVolume
38      {
39          get
40          {
41              return this.masterSFXVolume;
42          }
43          set
44          {
45              this.masterSFXVolume = value;
46              SingletonSoundManager.Instance.Mixer.SetFloat("MasterSFXVolume", this.
                    masterSFXVolume);
47          }
48      }
49
50
```

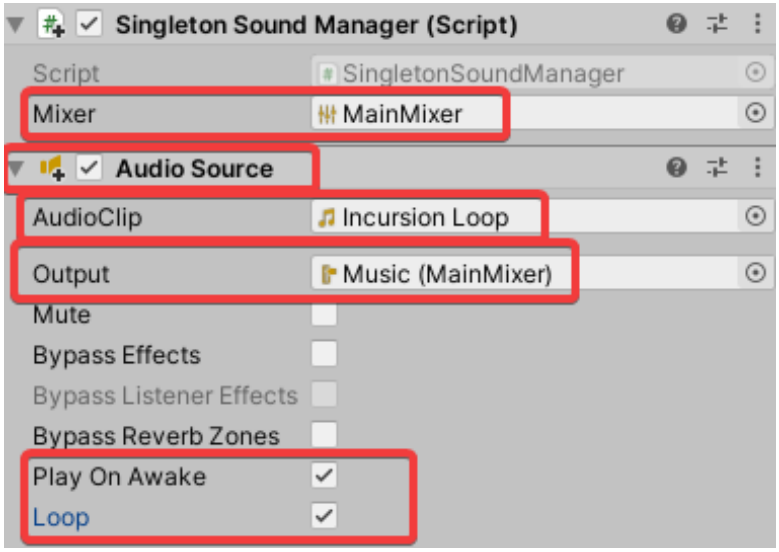
```
51      public float MasterSFXVolumeDefault { get;set; }
52
53
54      #endregion
55
56
57      private void Awake()
58      {
59          this.BGMSource = this.GetComponent<AudioSource>();
60
61          float musicVolume;
62          if (mixer.GetFloat("MusicVolume", out musicVolume))
63              SingletonSoundManager.Instance.MusicVolumeDefault = musicVolume;
64          float masterSFXVolume;
65          if (mixer.GetFloat("MasterSFXVolume", out masterSFXVolume))
66              SingletonSoundManager.Instance.MasterSFXVolumeDefault = masterSFXVolume;
67      }
68
69
70  }
```

กำหนด [SerializeField] Mixer ในหน้าต่าง Inspector ให้เป็น MainMixer โดยลาก MainMixer ที่ทำการสร้างขึ้นในหัวข้อหัวข้อ 10.8 มาวาง

**กำหนดให้ SingletonSoundManager เป็นแหล่งกำเนิดเสียงเพื่อเล่น Background Music**

ติดตั้งคอมโพเนนท์ AudioSource ให้กับวัตถุเปล่า SoundManager กำหนด AudioClip เป็น “Incursion Loop” (สามารถใช้เครื่องมือค้นหาชื่อ AudioClip จากรูปไอคอนแว่นขยาย หรือสามารถเลือกใช้เพลงอื่น ๆ ได้) และกำหนด Output เป็น Music (MainMixer)

เปิดใช้งาน Play On Awake และ Loop ดังรูปที่ 10.5




รูปที่ 10.5: Installing BGM AudioSource on the SoundManager

## 10.5 GameApplicationManager.cs : Hot fixes and Changes

ปรับเปลี่ยน GameApplicationManager จากแลป Scenes Management เป็น SingletonGameApplicationManager (ไม่ใช้ GameApplicationManager) เพื่อให้การเขียนโปรแกรมเป็นไปในรูปแบบเดียวกันคือ คลาส SingletonGameApplicationManager จะสืบทอดมาจาก Singleton ในรูปแบบเดียวกันกับ SingletonSoundManager

เขียน SingletonGameApplicationManager.cs ดัง Sourcecode 10.3

ถอนการติดตั้ง GameApplicationManager บนวัตถุเปล่า GameApplicationManager และทำการติดตั้ง SingletonGameApplicationManager เข้าไปแทนที่

Source code 10.3: SingletonGameApplicationManager.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class SingletonGameApplicationManager : Singleton<SingletonGameApplicationManager>
       {
6      public string[] DIFFICULTY_LEVEL_NAMES = { "Easy", "Normal", "Hard", "Extreme" };
7
8      ////////////  Get/Set property declaration  ////////////
9      public bool IsOptionMenuActive
10     {
11         get { return _isOptionMenuActive; }
```

```
12          set { _isOptionMenuActive = value; }
13      }
14      protected bool _isOptionMenuActive = false;
15      ////////////////////////////////
16
17      ////////////  Shorter version of Get/Set property declaration  ////////////
18      // The C#  compiler will generate the same as the above for you, automatically.
19      public int DifficultyLevel{get;set;}
20      public bool MusicEnabled { get; set; } = true;
21      public bool SFXEnabled{get;set;} = true;
22      ////////////////////////////////
23  }
```

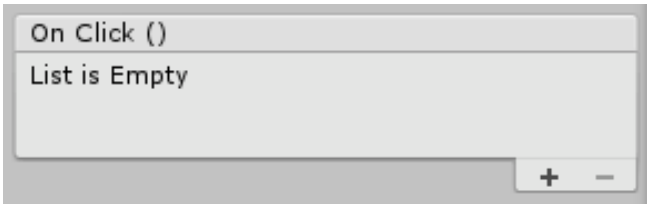
## 10.6 ปรับเปลี่ยนระบบเหตุการณ์ของหน้าเมนู ให้เป็นระบบกำหนดเมธอดเหตุการณ์ที่เกี่ยวข้องเมื่อเกิดเหตุการณ์ในแบบ delegate

***ข้อแนะนำ*** *ผู้พัฒนาสามารถข้ามหัวข้อนี้ไปได้ หากลักษณะการเขียนโปรแกรมเพื่อจัดการเหตุการณ์การคลิกของปุ่มเมนูต่าง ๆ บนฉาก SceneMainMenu เป็นในลักษณะรูปแบบวิธีที่ 2 "กำหนดด้วยการเขียนโปรแกรมกำหนดเมธอดเหตุการณ์" อยู่แล้ว เพียงแต่ ให้สังเกตุซอสโคดใน Sourcecode 10.4 ว่ามีส่วนใดแตกต่าง ให้เขียนโคดที่แตกต่างนั้น โดยเฉพาะในส่วนของการสืบทอดจากอินเทอเฟส IPointerEnterHandler*

หัวข้อนี้จะกระทำการปรับรูปแบบการเรียกเมธอดเหตุการณ์ของปุ่ม (Button) เมนูต่าง ๆ บนฉาก SceneMainMenu จากการกำหนดเมธอดเหตุการณ์ด้วยการเลือก เมธอด บน Inspector มาเป็นระบบที่เรียกเมธอดเหตุการณ์ด้วยวิธี "กำหนดด้วยการเขียนโปรแกรมกำหนดเมธอดเหตุการณ์"

***ขั้นตอนการเปลี่ยนแปลง***

1. ให้ทำการถอนการติดตั้งคอมโพเนนท์สคริปต์ MainMenuControlScript บนทุก ๆ วัตถุบนฉาก (เช่น MainMenu-ControlScript ได้ถูกติดตั้งบนปุ่ม Start จากปฏิบัติการ "Scenes Management")
2. ลบเหตุการณ์ On Click() ออกจากทุก ๆ เมนูดังรูปที่ 10.6
3. ปรับเปลี่ยนสคริปต์ควบคุมเมนู MainMenuControlScript เป็น MainMenuControlScriptBGMSFX ตาม Source-code 10.4
4. ติดตั้งสคริปต์ MainMenuControlScriptBGMSFX ให้กับ Canvas จากนั้นลากปุ่มเมนูต่าง ๆ มาวางดังรูปที่ 10.7



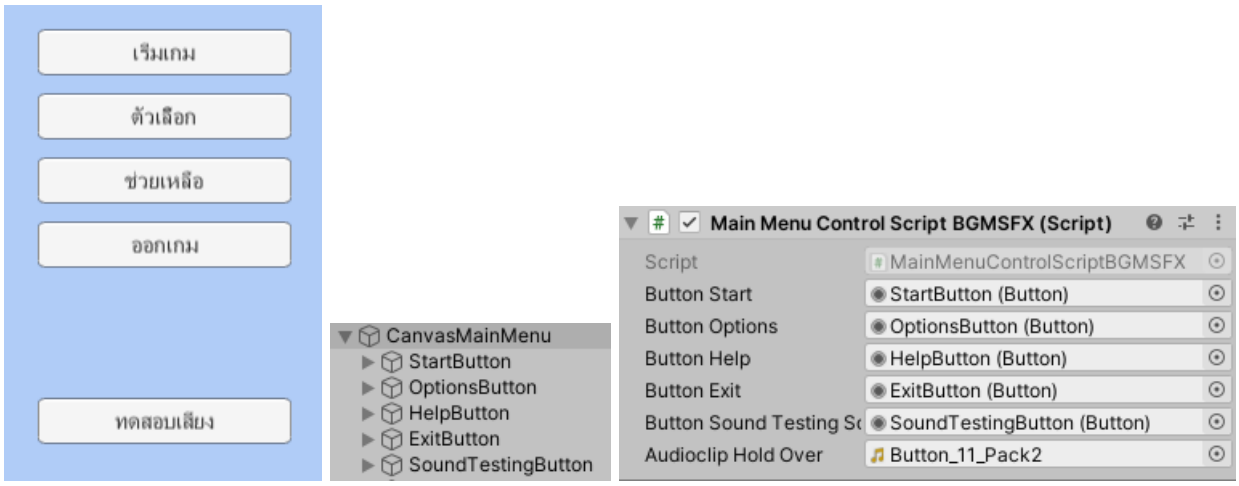

รูปที่ 10.6: ลบเหตุการณ์ On Click ()

Source code 10.4: Updated - MainMenuControlScriptBGMSFX.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class MainMenuControlScriptBGMSFX : MonoBehaviour, IPointerEnterHandler{
  [SerializeField] Button buttonStart;
  [SerializeField] Button buttonOptions;
  [SerializeField] Button buttonHelp;
  [SerializeField] Button buttonExit;
  [SerializeField] Button buttonSoundTestingScene;

  AudioSource audiosourceButtonUI;
  [SerializeField] AudioClip audioclipHoldOver;

  // Use this for initialization
  void Start () {
    this.audiosourceButtonUI = this.gameObject.AddComponent<AudioSource> ();
    this.audiosourceButtonUI.outputAudioMixerGroup = SingletonSoundManager.Instance.Mixer
        .FindMatchingGroups("UI")[0];

        SetupButtonsDelegate ();

        if (!SingletonSoundManager.Instance.BGMSource.isPlaying)
      SingletonSoundManager.Instance.BGMSource.Play();
    }

  public void OnPointerEnter(PointerEventData eventData)
```

```
{
  if (audiosourceButtonUI.isPlaying)
    audiosourceButtonUI.Stop ();

  audiosourceButtonUI.PlayOneShot (audioclipHoldOver);
}

void SetupButtonsDelegate(){
      buttonStart.onClick.AddListener(delegate{StartButtonClick(buttonStart);});
  buttonOptions.onClick.AddListener ( delegate {OptionsButtonClick(buttonOptions);});
  buttonHelp.onClick.AddListener ( delegate {HelpButtonClick(buttonHelp);});
      buttonExit.onClick.AddListener ( delegate {ExitButtonClick(buttonExit);});
      buttonSoundTestingScene.onClick.AddListener ( delegate {SoundTestingButtonClick(
          buttonSoundTestingScene);});
}

public void StartButtonClick(Button button) {
      SceneManager.LoadScene("SceneGameplay");
  }

public void OptionsButtonClick(Button button) {
      if (!GameApplicationManager.Instance.IsOptionMenuActive)
      {
          SceneManager.LoadScene("SceneOptions", LoadSceneMode.Additive);
          GameApplicationManager.Instance.IsOptionMenuActive = true;
      }
  }

public void HelpButtonClick(Button button) {
  SceneManager.LoadScene("SceneHelp");
}

public void SoundTestingButtonClick(Button button) {

      if (SingletonSoundManager.Instance.BGMSource.isPlaying)
    SingletonSoundManager.Instance.BGMSource.Stop();

      SceneManager.LoadScene("SceneSoundTesting");
}

```

```
69    public void ExitButtonClick(Button button) {
70          Application.Quit();
71      }
72  }
```

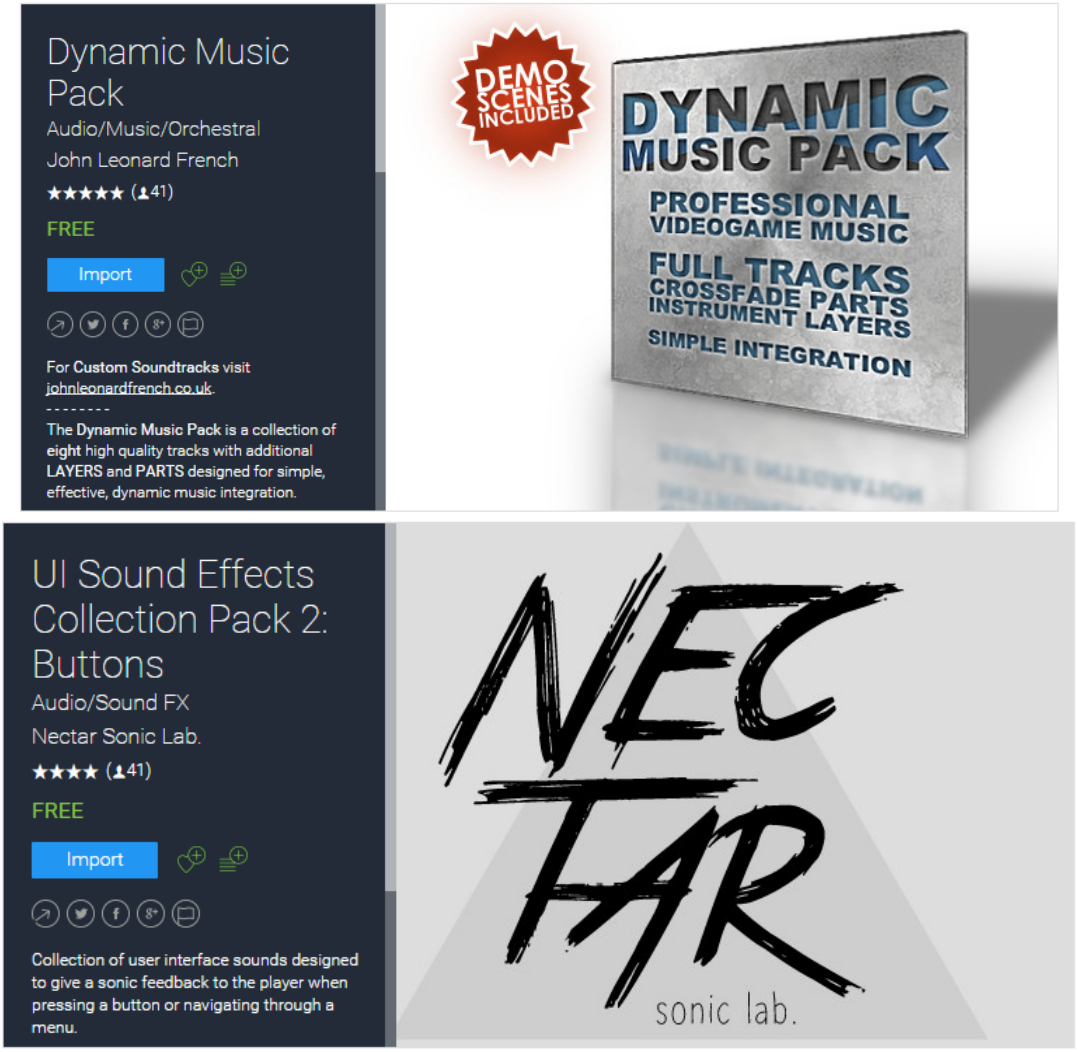



รูปที่ 10.7: การกำหนดปุ่มตัวแปรต่าง ๆ ให้กับ MainMenuControlScriptBGMSFX

## 10.7 ไฟล์เสียงเพลงและเสียงประกอบ

ให้ค้นหาและดาวน์โหลดเสียงเพลง เสียงประกอบ เพื่อนำมาใช้ในปฏิบัติการนี้ ทั้งนี้สามารถหาได้จาก Asset Store หรือจากแหล่งอื่น ๆ หรือสามารถดาวน์โหลด Asset ตามตัวอย่างดังรูปที่ 10.8

ข้อควรระวัง - ไฟล์ Asset มีขนาดใหญ่ อาจจะต้องใช้เวลานานในการดาวน์โหลด



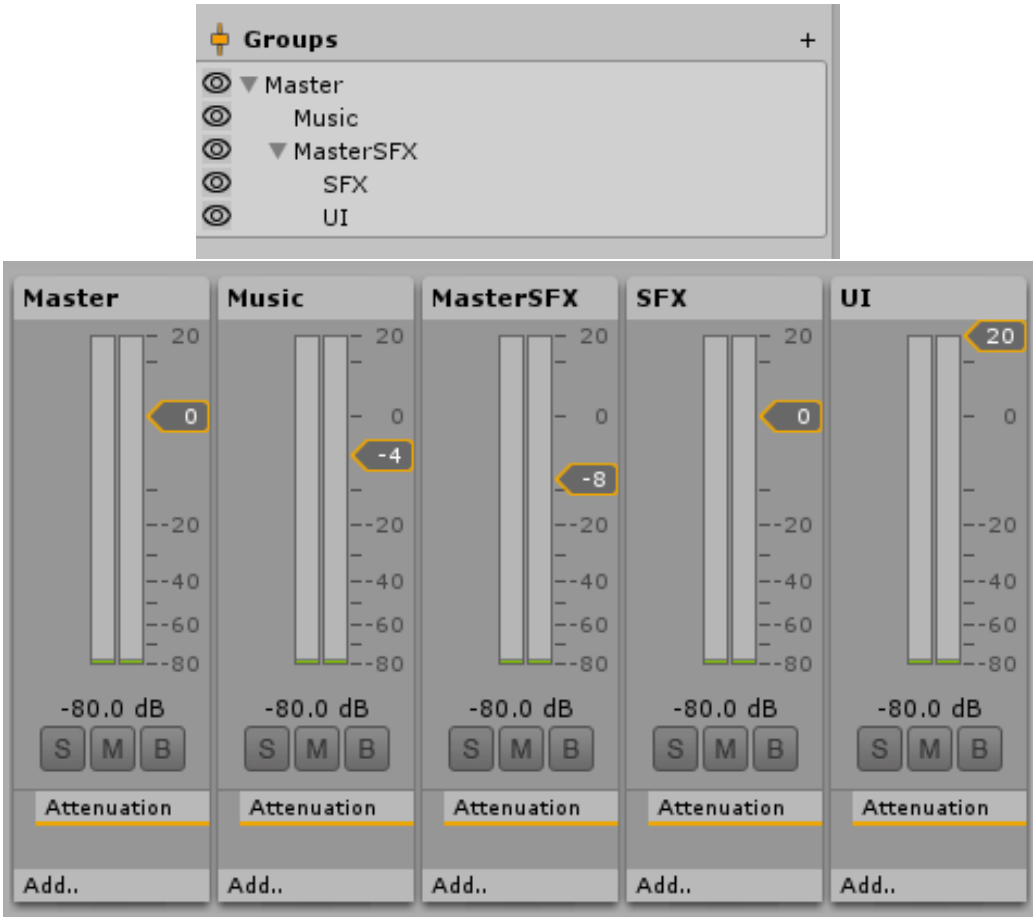

รูปที่ 10.8: Music and SFX Packs

## 10.8 ตัวผสมเสียง (Audio Mixer)

สร้าง AudioMixer ลงบนหน้าต่างโครงการ (Project->Assets) จากเมนู Assets->Create->Audio Mixer เปลี่ยนชื่อเป็น MainMixer

ดับเบิลคลิกที่ MainMixer เพื่อเปิดหน้าต่าง Audio Mixer

เพิ่มหมวดหมู่ (Groups) การจัดการเสียงจากหน้าต่าง Mixers ดังรูปที่ 10.9 โดยการคลิกที่เครื่องหมาย + เพิ่มกลุ่มการจัดการเสียงให้ได้ตามรูป และปรับแต่ง Volume ของเสียงแต่ละหมู่ดังรูปที่ 10.9




รูปที่ 10.9: การปรับแต่ง Audio Mixer

## 10.9 การใช้งาน Audio Mixer สำหรับควบคุมการเล่นเสียงเพลงและเสียงประกอบ

### 10.9.1 กำหนดต้นกำเนิดเสียง (AudioSource) ให้กับ Canvas

เพิ่มคอมโพเนนท์ Audio Source ให้กับ Canvas ของเมนูในฉาก SceneMainMenu และกำหนด Output ให้เป็น UI โดยเปิด AudioMixer ที่สร้างขึ้นในข้อหัวข้อ 10.8 MainMixer และลากหมวด UI มาวางดังรูปที่ 10.10 เป็นการกำหนดเส้นทางของเสียงที่จะเล่นให้ผ่านตัวผสมเสียง (MainMixer) ให้ออกที่ช่องทาง UI


รูปที่ 10.10: กำหนด Output ของ Audio Source ให้ออกที่ MainMixer->UI

### 10.9.2 กำหนดเสียงประกอบ Hold Over Clip ให้กับคอมโพเนนท์สคริปต์ MainMenuControlScript

ให้ทำการกำหนด Audio Clip ชื่อ UI_Buttons_Pack2->Button_11_Pack2 (สามารถใช้คลิปเสียงอื่นได้) ให้กับสคริปต์ MainMenuControlScript ดังรูปที่ 10.7 ”Hold Over Clip” ซึ่งเป็นการกำหนดคลิปเสียงที่จะถูกนำไปใช้ในการเล่นเมื่อเมาส์วางทาบอยู่บนปุ่ม

### 10.9.3 ทดสอบรันโปรแกรม

ทดลอง เปิด / ปิด คอมโพเนนท์ Audio Listener ซึ่งติดตั้งอยู่ที่ Main Camera และแนะนำให้เปิดหน้าต่าง Audio Mixer ด้วยการดับเบิลคลิกที่ “MainMixer” จะเห็นการทำงานของ Mixer ขณะที่โปรแกรมกำลังทำงาน

**คำถามสำหรับ chapter 10 เสียงเพลงและเสียงประกอบ (Music and Sound Effect)**

| **อธิบายการทำงานของโปรแกรม หน้าที่ของ Audio Listener, Audio Source, Audio Clip และ Audio Mixer** |
| --- |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |

## 10.10 สร้างฉากเพิ่มเติม (Create additional scenes)

สร้างฉากเพิ่มเติมขึ้นมาอีกจำนวนสองฉาก คือ SceneHelp และ SceneSoundTesting จากนั้นนำไปเพิ่มรวมเข้าใน Build Setting

สร้าง SceneHelp และ SceneSoundTesting (**ชื่อของฉากต้องตรงกันกับการเรียกเปลี่ยนฉากในสคริปต์ MainMenu-ContorlScript.cs**)

นำไปรวมใน Build Setting...

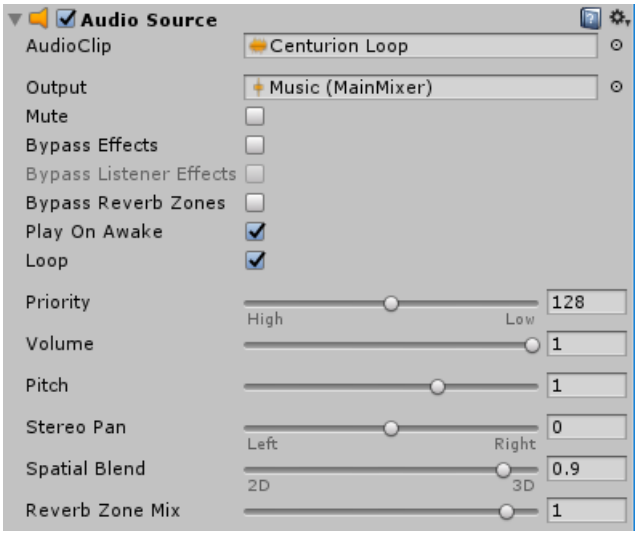
เปิดฉากทั้งสองขึ้นมาแก้ไข และทำการเพิ่ม UI ปุ่มสำหรับรองรับการคลิกเพื่อกลับสู่หน้าเมนูหลัก (SceneMainMenu)

ในกระบวนการทำการเพิ่มปุ่ม “กลับสู่หน้าเมนูหลัก” ให้กับฉากทั้งสองนั้น แนะนำให้ สร้าง Canvas และปุ่ม จากนั้นทำให้เป็น Prefab เพื่อให้สามารถนำมาใช้ใหม่ได้ ฉากใด ๆ ก็ตามที่ต้องการให้มีปุ่ม “กลับสู่หน้าเมนูหลัก”

## 10.11 แหล่งกำเนิดเสียงในฉากสามมิติ (Audio Source in 3D Space) และการตั้งค่าแหล่งกำเนิดเสียงสามมิติ (3D Sound Settings)

สร้างพื้นที่สำหรับทดลองงาน (Playground)

สร้างวัตถุทรงสี่เหลี่ยมวางไว้กลางฉาก ตั้งชื่อว่า Stereo ติดตั้งคอมโพเนนท์ AudioSource กำหนด AudioClip และ Output ดังรูปที่ 10.11




รูปที่ 10.11: การตั้งค่าคอมโพเนนท์ AudioSource ให้กับวัตถุ Stereo

จำลองตัวละครที่สามารถเคลื่อนที่ได้อิสระในฉาก (เลือกหัวข้อใดหัวข้อหนึ่งจากหัวข้อด้านล่าง)

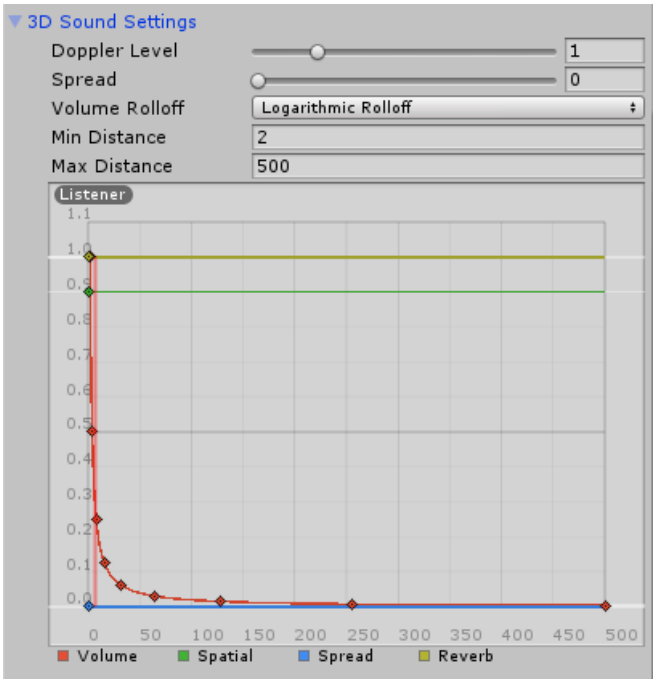
สร้างสคริปต์เพื่อควบคุมการเคลื่อนไหวของกล้องหลัก (Main Camera) หรือ

สร้าง Capsule ตั้งชื่อว่า Player และนำ FPSController หรือ ThirdPersonController จาก “Unity Standard Assets” มาติดตั้งเป็น Child และลบ Main Camera ออก หรือ

ใช้ตัวละครที่สร้างขึ้นจากปฏิบัติการ “Character Animation and Control” ตั้งชื่อว่า Player และนำ FPSController หรือ ThirdPersonController จาก “Unity Standard Assets” มาติดตั้งเป็น Child และลบ Main Camera ออก

ทดสอบรันโปรแกรม โดยเคลื่อนที่ตัวละคร เข้าใกล้ และออกห่างจากแหล่งกำเนิดเสียง

ทดลองปรับค่า Spatial Blend (รูปที่ 10.11) Doppler Level และโหมด Volume Rolloff (รูปที่ 10.12)




รูปที่ 10.12: หน้าต่างตั้งค่าเสียงสามมิติ (3D Sound Settings)

**คำถามสำหรับ chapter 10 เสียงเพลงและเสียงประกอบ (Music and Sound Effect)**

| **อธิบายถึงผลลัพธ์และความสัมพันธ์ของตำแหน่งผู้ฟัง (Audio Listener) แหล่งกำเนิดเสียง (Audio Source) การตั้งค่า Spatial Blend, Doppler Level, โหมด Volume Rolloff** |
| --- |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |

**โจทย์เพิ่มเติม : ให้นักศึกษาทำให้ตัวละครสามารถยิงกระสุนวิถีโค้ง ออกจากตัวละคร ให้เคลื่อนที่ตามหลักทางฟิสิกส์ และหากกระสุนชนกับกำแพง หรือสิ่งกีดขวางบนฉาก ให้เล่นเสียงเอฟเฟ็ก ณ ตำแหน่งที่ชน**

(Hints: สร้าง Prefab กระสุนที่ติดตั้ง AudioSource และตั้งค่า Output ไปที่ Mixer->SFX)

อธิบายสิ่งที่สร้างขึ้น . . . . . .

## 10.12 การควบคุม Volume เพื่อ เปิด / ปิด เสียงเพลงและเสียงเอฟเฟ็ก ภายในเกมจากหน้าฉากตัวเลือก (SceneOptions)

เปิดฉากตัวเลือก (SceneOptions) ขึ้นมาแก้ไข

ทำการลบเหตุการณ์ On Value Changed (Boolean) ของปุ่ม ToggleMusic และ ToggleSFX ออก ดังเช่นการทำในหัวข้อ 10.6

เพิ่มซอสโคดเพื่อตอบรับเหตุการณ์ด้วยการเขียนโคด ตาม Sourcecode 10.5 ซึ่งอยู่ในส่วนของเมธอด Start() แทนที่การลบเหตุการณ์ข้างต้น

Source code 10.5: OptionsMenuControlScript.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5  using UnityEngine.SceneManagement;
6
7  using KSCamt;
8
9  public class OptionsMenuControlScript : MonoBehaviour {
10
11     [SerializeField] Dropdown _dropdownDifficulty;
12     [SerializeField] Toggle _toggleMusic;
13     [SerializeField] Toggle _toggleSFX;
14     [SerializeField] Button _backButton;
15
```

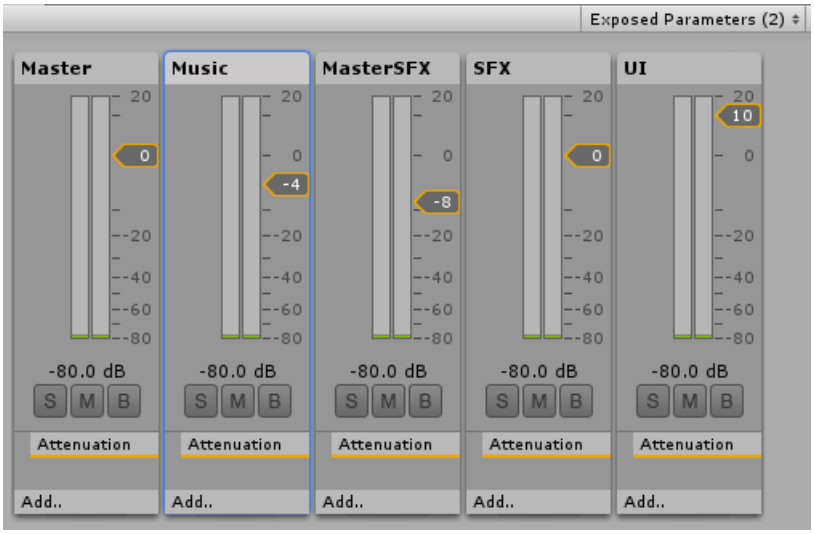
```
16      // Use this for initialization
17      void Start () {
18          _dropdownDifficulty.value = SingletonGameApplicationManager.Instance.
                DifficultyLevel;
19          _toggleMusic.isOn = SingletonGameApplicationManager.Instance.MusicEnabled;
20          _toggleSFX.isOn = SingletonGameApplicationManager.Instance.SFXEnabled;
21
22          _dropdownDifficulty.onValueChanged.AddListener(delegate {
                DropdownDifficultyChanged(_dropdownDifficulty);});
23          _toggleMusic.onValueChanged.AddListener(delegate { OnToggleMusic(_toggleMusic);})
                ;
24          _toggleSFX.onValueChanged.AddListener(delegate { OnToggleSFX(_toggleSFX);});
25          _backButton.onClick.AddListener(delegate { BackButtonClick(_backButton);});
26      }
27
28      // Update is called once per frame
29      void Update () {
30
31  }
32
33      public void BackButtonClick(Button button) {
34          SceneManager.UnloadSceneAsync("SceneOptions");
35          SingletonGameApplicationManager.Instance.IsOptionMenuActive = false;
36      }
37
38      public void DropdownDifficultyChanged(Dropdown dropdown) {
39          SingletonGameApplicationManager.Instance.DifficultyLevel = dropdown.value;
40      }
41
42      public void OnToggleMusic(Toggle toggle) {
43          SingletonGameApplicationManager.Instance.MusicEnabled = _toggleMusic.isOn;
44          if (SingletonGameApplicationManager.Instance.MusicEnabled)
45              SingletonSoundManager.Instance.MusicVolume = SingletonSoundManager.Instance.
                    MusicVolumeDefault;
46          else
47              SingletonSoundManager.Instance.MusicVolume = SingletonSoundManager.
                    MUTE_VOLUME;
48      }
49      public void OnToggleSFX(Toggle toggle)
50      {
```

```
51          SingletonGameApplicationManager.Instance.SFXEnabled = _toggleSFX.isOn;
52          if (SingletonGameApplicationManager.Instance.SFXEnabled)
53              SingletonSoundManager.Instance.MasterSFXVolume = SingletonSoundManager.
                    Instance.MasterSFXVolumeDefault;
54          else
55              SingletonSoundManager.Instance.MasterSFXVolume = SingletonSoundManager.
                    MUTE_VOLUME;
56      }
57  }
```

สังเกตการปรับแก้ไขเพิ่มเติมซอสโคดของเมธอด OnToggleMusic และ OnToggleSFX ตาม Sourcecode 10.5

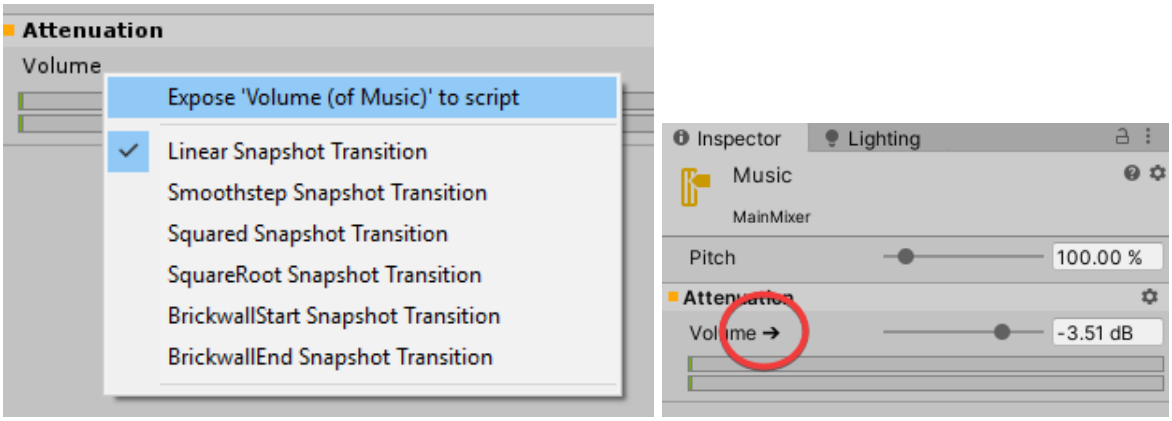
เปิด MainMixer ขึ้นมาแก้ไข

คลิกแถบปรับค่า Music บนหน้าต่าง Audio Mixer เพื่อ Activate ให้ขึ้นกรอบสีฟ้าดังรูปที่ 10.13




รูปที่ 10.13: เลือก Music channel

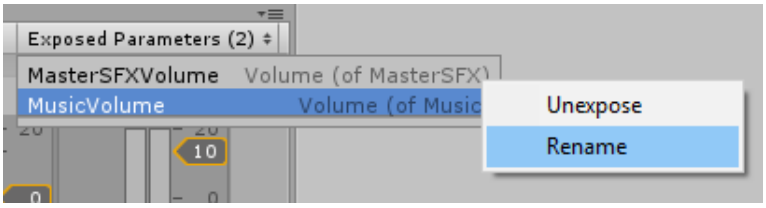
คลิกขวาที่คำว่า “Volume” Attenuation->Volume บนหน้าต่าง Inspector เลือก Expose....to script ดังรูปที่ 10.14 เมื่อทำเสร็จจะมีเครื่องหมายลูกศรขึ้นที่หลังคำว่า Volume




รูปที่ 10.14: การกำหนดให้คุณสมบัติเสียงสามารถเข้าถึงผ่านสคริปต์

ให้ทำการในแบบเดียวกันกับแถบ mixer MasterSFX

ตั้งชื่อพารามิเตอร์ที่จะใช้อ้างอิงดังรูปที่ 10.15 โดยเลือก Rename และเปลี่ยนชื่อเป็น MusicVolume และ MasterSFXVolume สำหรับแถบ Music และ MasterSFX ตามลำดับ โดยชื่อนี้จะเป็นชื่อสำหรับอ้างอิง เพื่อกำหนดค่าระดับของเสียง (Volume) ภายในสคริปต์




รูปที่ 10.15: การกำหนดชื่อให้กับพารามิเตอร์ทางเสียงที่จะนำไปอ้างอิงในสคริปต์

ส่วนของการควบคุมพารามิเตอร์ทั้งสอง ได้ถูกเขียนไว้แล้ว อยู่ในซอสโคด SoundManager.cs เช่น

Source code 10.6: ตัวอย่างการกำหนดค่าให้กับตัวแปรของ Audio Mixer

```
1 SingletonSoundManager.Instance.Mixer.SetFloat("MusicVolume", musicVolume);
2 SingletonSoundManager.Instance.Mixer.SetFloat("MasterSFXVolume", masterSFXVolume);
```

รันโปรแกรม และตรวจสอบว่า สามารถเปิด / ปิดเสียงเพลงและเสียงเอฟเฟค จากหน้าตัวเลือกได้หรือไม่ โดยให้ทดสอบเปิด/ปิด ย้ายฉาก และกลับมา เปิด / ปิด

1/2563

ชื่อ.....

รหัสนักศึกษา.....

ตอนที่.....

สาขาวิชาแอนนิเมชันและเกม วิทยาลัยศิลปะ สื่อ และเทคโนโลยี มหาวิทยาลัยเชียงใหม่


## 10.13 คำถามและปัญหาชวนคิด (Questions and Problems)

**คำถามสำหรับ chapter 10 เสียงเพลงและเสียงประกอบ (Music and Sound Effect)**

| **อธิบายการทำงานของโปรแกรม หน้าที่ของ Audio Listener, Audio Source, Audio Clip และ Audio Mixer** |
| --- |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |
| . . . . . |

**คำถามสำหรับ chapter 10 เสียงเพลงและเสียงประกอบ (Music and Sound Effect)**

**อธิบายถึงผลลัพธ์และความสัมพันธ์ของตำแหน่งผู้ฟัง (Audio Listener) แหล่งกำเนิดเสียง (Audio Source) การตั้งค่า Spatial Blend, Doppler Level, โหมด Volume Rolloff**

. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .

**โจทย์เพิ่มเติม : ให้นักศึกษาทำให้ตัวละครสามารถยิงกระสุนวิถีโค้ง ออกจากตัวละคร ให้เคลื่อนที่ตามหลักทางฟิสิกส์ และหากกระสุนชนกับกำแพง หรือสิ่งกีดขวางบนฉาก ให้เล่นเสียงเอฟเฟ็ก ณ ตำแหน่งที่ชน**

(Hints: สร้าง Prefab กระสุนที่ติดตั้ง AudioSource และตั้งค่า Output ไปที่ Mixer->SFX)

อธิบายสิ่งที่สร้างขึ้น . . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .