1/2563

ชื่อ Type text here.....

รหัสนักศึกษา.....

ตอนที่.....

สาขาวิชาแอนนิเมชันและเกม วิทยาลัยศิลปะ สื่อ และเทคโนโลยี มหาวิทยาลัยเชียงใหม่

เนื้อหา (Contents)

บทที่ 5

การจัดการฉาก (Scenes Management)

5.1 บทนำ (Introduction)

การพัฒนาเกมที่มีขนาดของเกมประกอบไปด้วยฉาก (Scene) มากกว่าหนึ่งฉาก จำเป็นต้องมีการจัดการฉากต่าง ๆ ที่มีอยู่ในเกม เช่น ฉากเริ่มต้น ฉากเมนู ฉากปรับเปลี่ยนตัวเลือก ฉากตัวเกม ฉากจบเกม เป็นต้น เห็นได้ว่าแต่ละฉากของเกมมีทรัพยากรภาพและเสียงที่ใช้ในฉากที่ไม่เหมือนกัน การบริหารจัดการฉากสามารถยกตัวอย่างเป็นข้อ ๆ ได้ดังต่อไปนี้

- การเปลี่ยนจากฉากหนึ่งไปอีกฉากหนึ่ง
- การกำจัดทรัพยากรของฉากที่ถูกเปลี่ยน
- การโหลดทรัพยากรที่จำเป็นของฉากถัดไป
- การจัดการการเก็บค่าสถานะต่าง ๆ ของเกมที่จำเป็นที่ต้องส่งผ่านต่อ ๆ ไปภายในเกมในฉากอื่น ๆ

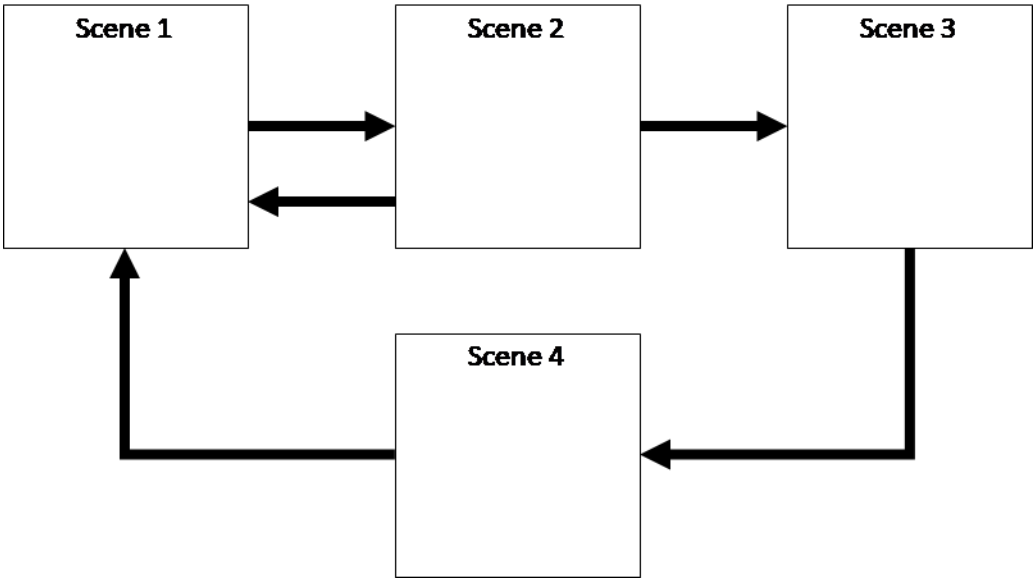
นอกจากการบริหารจัดการฉากต่าง ๆ ภายในเกมแล้ว ภายในฉากหนึ่ง ๆ ยังมีสถานะ ณ ขณะใดขณะหนึ่งที่ไม่เหมือนกัน เช่น ฉากเล่นเกม (Gameplay scene) อาจประกอบไปด้วยสถานะตอนก่อนเริ่มเกม (Counting to start) สถานะของฉากขณะกำลังเล่นอยู่ (Playing) หยุดชั่วคราว (Pause) จบเกม (Ending) ซึ่งแตกต่างกับการจัดการฉากเล็กน้อยที่ผู้พัฒนาอาจจะมองว่าเป็นการเปลี่ยนฉากขนาดย่อม

บทนี้กล่าวถึงเทคนิคการเขียนโปรแกรมเพื่อบริหารจัดการเกมที่ประกอบไปด้วยฉากมากกว่าหนึ่งฉาก (Scene Manager)

5.2 การจัดการฉาก (Scene Management)

ตัวอย่างการเปลี่ยนฉากต่าง ๆ ภายในเกมแสดงดังรูปที่ 5.1 เกมเริ่มที่ฉากที่หนึ่ง (Scene 1) สามารถเปลี่ยนไปยังฉากที่สอง (Scene 2) ได้ ขณะที่ฉากที่สอง สามารถเปลี่ยนกลับไปยังฉากที่หนึ่ง หรือเปลี่ยนต่อไปยังฉากที่สาม (Scene 3)

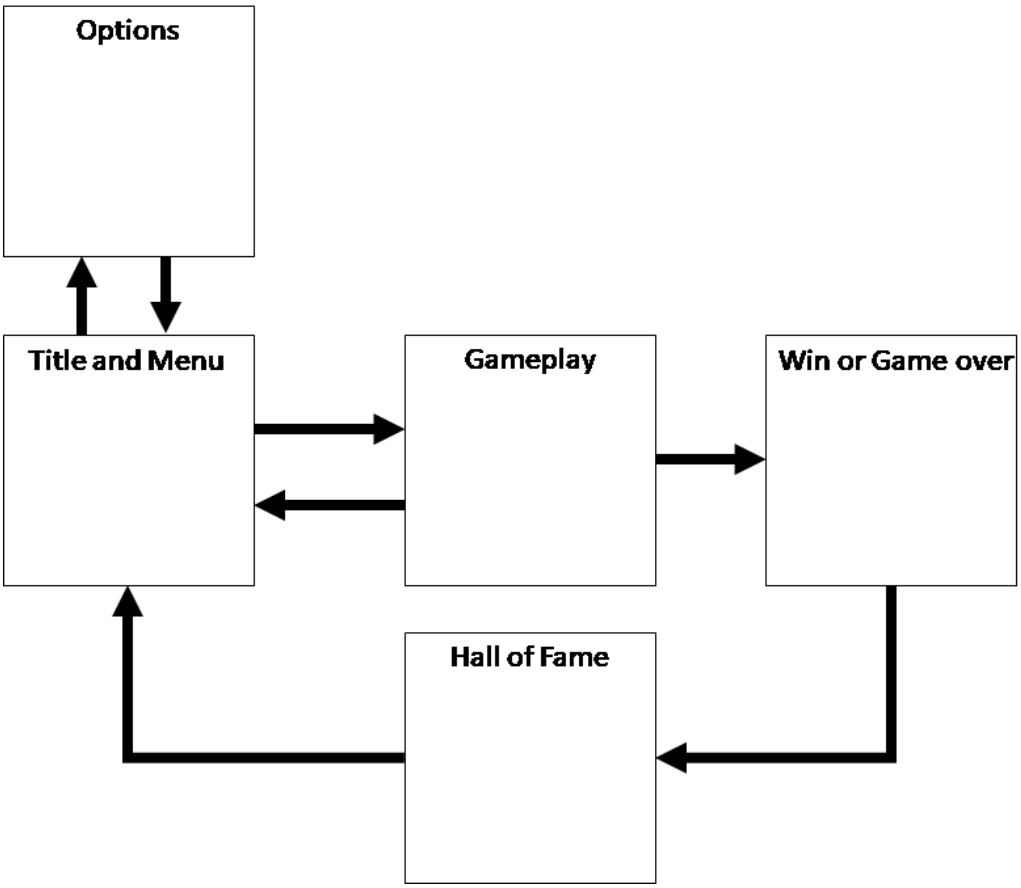
ฉากที่สามไม่สามารถกลับมาฉากที่สองได้ แต่สามารถเปลี่ยนไปฉากที่สี่ (Scene 4) ได้ สุดท้ายฉากที่สี่สามารถเปลี่ยนไปยังฉากที่หนึ่งได้



รูปที่ 5.1: แผนผังตัวอย่างการเปลี่ยนฉากต่าง ๆ ภายในเกม

จะเห็นว่าการเชื่อมต่อฉากต่าง ๆ ภายในเกมนั้นขึ้นอยู่กับผู้ออกแบบเกม (Game Designer) ว่าจะให้ฉากแต่ละฉากมีความสัมพันธ์กันอย่างไร รวมถึงเงื่อนไข ที่จะทำให้เปลี่ยนจากฉากหนึ่งไปยังอีกฉากหนึ่ง

รูปที่ 5.2 แสดงภาพแผนผังตัวอย่างการเชื่อมต่อของฉากต่าง ๆ ภายในเกมที่อ้างอิงกับชื่อที่มีความหมายของฉากในเกมตามท้องตลาด ประกอบไปด้วยฉากภายในเกม เช่น ฉากชื่อและเมนูเกม (Title and Menu) ฉากตัวเลือก (Options) ฉากเล่นเกม (Gameplay) ฉากจบเกม (Win or Game Over) และฉากตารางคะแนน (Hall of Fame) เป็นต้น



รูปที่ 5.2: แผนผังตัวอย่างการเชื่อมโยงของฉากต่าง ๆ ภายในเกม

5.3 ตัวจัดการฉาก (Scene Manager)

การจัดฉากเกมใน Unity ต้องทำการเขียนซอสโคดเพื่อเพิ่มไลบรารี UnityEngine.SceneManagement เข้ามาที่ส่วนบนของซอสโคดจึงจะสามารถเรียกใช้งาน static method ของคลาส SceneManager เพื่อจัดการการเปลี่ยนฉากได้ การนำเข้าไลบรารีดังกล่าวแสดงดัง Sourcecode 5.1

Source code 5.1: ชื่อเมธอดและพารามิเตอร์สำหรับสร้างวัตถุในฉาก

```
1 using UnityEngine.SceneManagement;
```

การโหลดฉากใหม่และเปลี่ยนฉากหลังจากโหลดฉากเสร็จใช้คำสั่ง LoadScene ซึ่งเป็น static method ของคลาส Scene-Manager ดัง Sourcecode 5.2 จากตัวอย่างนี้ฉากต่อไปมีชื่อว่า “SceneGameplay” ซึ่งถูกสร้างไว้และถูกรวบรวมไว้ในโครงการเกม โดยการถูกกำหนดให้รวมในการสร้างตัวเกม (Build Setting) ซึ่งจะกล่าวถึงในหัวข้อถัดไป

ในทำนองเดียวกันผู้พัฒนาสามารถ unload ฉากได้ด้วยคำสั่ง UnloadSceneAsync ดัง Sourcecode 5.2

Source code 5.2: คำสั่ง load และ unload ฉากพื้นฐาน (LoadScene/Unload scene)

```
1 //□□□□□□□□□□□□□□□□□□□ load □□□
2 SceneManager.LoadScene("SceneGamePlay");
3
4 //□□□□□□□□□□□□□□□□□□□ unload □□□
5 SceneManager.UnloadSceneAsync("SceneGamePlay");
```

โดยลักษณะของการเปลี่ยนฉากมีอยู่สองลักษณะคือ 1) การเปลี่ยนฉากโดยละทิ้งฉากปัจจุบัน 2) การเปลี่ยนฉากโดยการเพิ่มฉากใหม่ซ้อนเข้าไปยังฉากเดิม

เกมเอนจิน Unity สามารถกำหนดลักษณะของการเปลี่ยนฉากด้วยการกำหนดพารามิเตอร์ LoadSceneMode เป็น Single หรือ Additive ดังตัวอย่างคำสั่งใน Sourcecode 5.3

Source code 5.3: การกำหนดพารามิเตอร์ลักษณะของการเปลี่ยนฉาก

```
1 SceneManager.LoadScene(``SceneGamePlay'', LoadSceneMode.Single);
2 SceneManager.LoadScene(``Options'', LoadSceneMode.Additive);
```

5.4 เทคนิคการใช้งานการเขียนโปรแกรมด้วยแนวคิด Singleton เพื่อนำมาประยุกต์ใช้ในงานพัฒนาเกมที่ประกอบไปด้วยฉากหลายฉาก

แนวคิด Singleton คือการบังคับให้คลาสไม่สามารถมี instance ของคลาสได้มากกว่าหนึ่ง instance และจะสามารถเข้าถึง instance นั้นได้ทางเดียวคือผ่านตัวแปรประเภท public static ของคลาสนั้น วิธีการทำให้คลาสไม่สามารถสร้าง instance ของคลาสได้คือการประกาศคอนสตรัคเตอร์ (Constructor) ให้เป็น private ซึ่งทำให้ไม่สามารถสร้าง instance ของคลาสได้ นอกจากการสร้าง instance ขึ้นภายในคลาสเอง

แนวคิด Singleton สะท้อนให้เห็นภาพถึงวัตถุในโลกของความเป็นจริงที่ซึ่งวัตถุแต่ละชิ้นนั้นมีความเป็นเอกลักษณ์ (Unique) ซึ่งเป็นสิ่งที่มีชิ้นเดียวภายในโลกเท่านั้น ไม่สามารถทำสำเนาได้

ประโยชน์อย่างหนึ่งของ Singleton สำหรับการนำมาประยุกต์ใช้พัฒนาเกมคือการรวมศูนย์ของเมธอดการทำงาน และตัวแปรต่าง ๆ ไว้ในที่เดียว หรือเพื่อเก็บสถานะต่าง ๆ ภายในเกม ที่สามารถเข้าถึงจากที่ใดก็ได้ของโปรแกรมเกมด้วยการเรียกผ่านชื่อคลาสและเข้าถึงตัวแปร “instance” ที่ถูกสร้างขึ้นภายในคลาสนั้น

ตัวอย่างการสร้างคลาสด้วยแนวคิด Singleton ใน Unity แสดงดัง Sourcecode 5.4 คลาส GameApplicationManager ซึ่งมีหน้าในการเก็บค่าสถานะต่าง ๆ ที่สามารถเข้าถึงได้ทั่วทั้งเกม เช่น เข้าถึงจากฉากเมนู ฉากเกม ฉากตัวเลือก เป็นต้น

อย่างไรก็ตาม เมื่อมีการเปลี่ยนฉากเกิดขึ้นจากฉากหนึ่งไปยังอีกฉากหนึ่ง โดยปกติแล้ว Unity จะทำลายวัตถุทุก ๆ ชิ้นในฉากก่อนหน้าโดยอัตโนมัติ ทั้งนี้สามารถแก้ปัญหาด้วยการใช้เทคนิคการป้องกันการทำลายวัตถุด้วยเมธอด DontDestroyOnLoad() และมีการตรวจสอบการซ้ำกันของตัวแปร instance เนื่องจากสคริปต์ GameApplicationManager

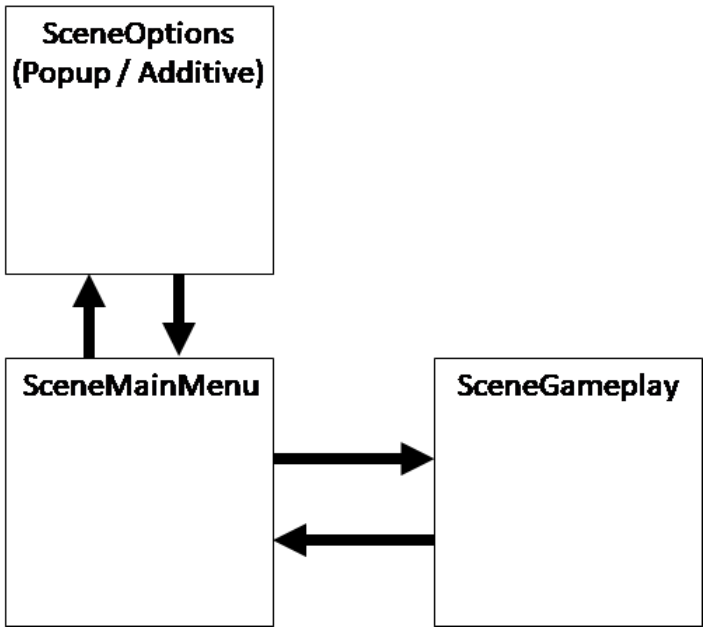
ถูกติดตั้งบนวัตถุเกมเปล่า (Empty Game Object) ที่วางในฉาก ซึ่งจะถูกสร้างขึ้นใหม่ทุก ๆ ครั้งที่เข้าฉากดังกล่าว จึงต้องมีการตรวจสอบและทำลายวัตถุที่สร้างขึ้นซ้ำ แต่จะไม่ทำลาย instance ที่ถูกสร้างขึ้นไว้ในตอนในครั้งแรกสุด

5.5 ทดลองและศึกษาการจัดการการเปลี่ยนฉากของเกม (Scenes Management)

วัตถุประสงค์

1. ศึกษาการจัดการการเปลี่ยนฉาก (Scene management)
2. เรียนรู้เทคนิคการใช้งาน Singleton เพื่อประยุกต์ใช้ในงานพัฒนาเกม
3. เรียนรู้เทคนิคการจัดการกับเหตุการณ์ของ User Interface ใน Unity

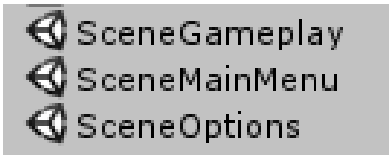
สมมติว่าออกแบบแผนผังการเชื่อมโยงของฉากต่าง ๆ ไว้ดังรูปที่ 5.3



รูปที่ 5.3: แผนผังการเชื่อมโยงระหว่างฉากเกม

5.5.1 เตรียมฉากต่าง ๆ ตามแผนผัง

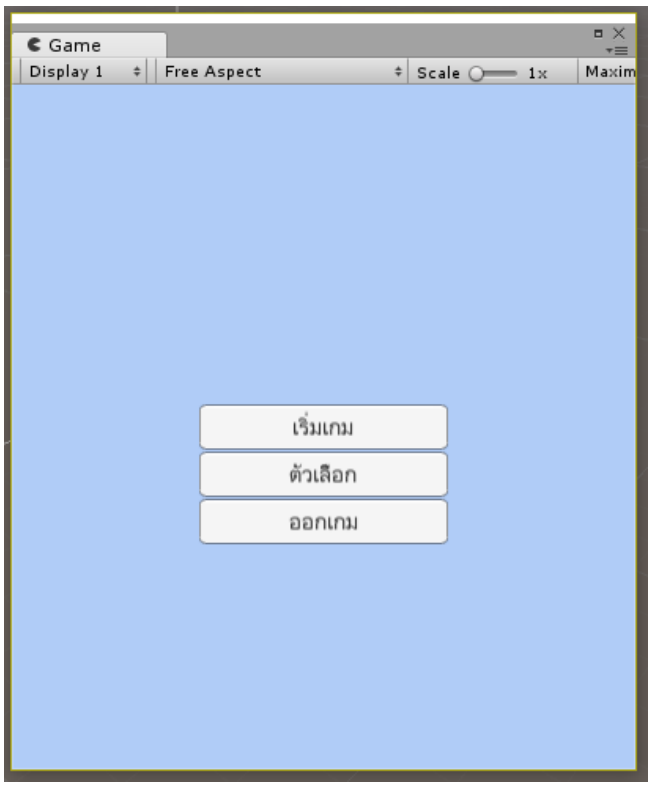
สร้างฉากเกมขึ้นมาเตรียมไว้สามฉาก ด้วยเมนู Assets->Create->Scene ดังรูปที่ 5.4 ตั้งชื่อว่า “SceneMainMenu” “SceneOptions” และ “SceneGameplay”



รูปที่ 5.4: ฉากที่สร้างขึ้นมาเตรียมไว้จำนวนสามฉาก SceneGameplay SceneMainMenu และ SceneOptions ในหน้าต่าง Assets

5.5.2 เขียนสคริปต์ควบคุมเมนูของฉาก SceneMainMenu

เลือกเปิดฉาก SceneMainMenu และสร้างส่วนติดต่อผู้ใช้ (User Interface) โดยอาศัยความรู้การจัดวาง UI ในปฏิบัติการในครั้งก่อน ๆ และจัดวางดังรูปที่ 5.5 โดยใช้ User Interface แบบ Button



รูปที่ 5.5: การจัดวาง UI ใน SceneMainMenu

ตั้งชื่อวัตถุเกมปุ่ม (Button) ทั้งสามดังรูปที่ 5.6 StartButton OptionsButton และ ExitButton สำหรับเมนู เริ่มเกม ตัวเลือก และออกเกม ตามลำดับ

รูปที่ 5.6: วัตถุปุ่มหน้าฉาก SceneMainMenu

5.5.3 สร้าง GameApplicationManager

เปิดฉาก SceneMainMenu ซึ่งเป็นฉากแรกสุดที่จะปรากฏเมื่อเปิดเกม

สร้างสคริปต์ด้วยเมนู Assets->Create->C Sharp Script และตั้งชื่อว่า “GameApplicationManager”

เขียนซอสโคด GameApplicationManager ดัง Sourcecode 5.4

สร้างวัตถุเปล่า GameObject->Create Empty ตั้งชื่อว่า GameApplicationManager และติดตั้งคอมโพเนนท์สคริปต์ GameApplicationManager

Source code 5.4: GameApplicationManager.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class GameApplicationManager : MonoBehaviour {
6      static public GameApplicationManager Instance {
7          get {
8              if (_instance == null)
9              {
10                 _instance = GameObject.FindObjectOfType<GameApplicationManager>();
11                 GameObject container = new GameObject("GameApplicationManager");
12                 _instance = container.AddComponent<GameApplicationManager>();
13             }
14             return _instance;
15         }
16     }
17     static protected GameApplicationManager _instance = null;
18
19     void Awake()
```

```
20      {
21          if (_instance == null)
22          {
23              _instance = this;
24              DontDestroyOnLoad(this.gameObject);
25          }
26          else
27          {
28              if (this != _instance)
29              {
30                  Destroy(this.gameObject);
31              }
32          }
33      }
34
35      public string[] DIFFICULTY_LEVEL_NAMES = { "Easy", "Normal", "Hard", "Extreme" };
36
37      ///////////  Get/Set property declaration  ////////////
38      public bool IsOptionMenuActive
39      {
40          get { return _isOptionMenuActive; }
41          set { _isOptionMenuActive = value; }
42      }
43      protected bool _isOptionMenuActive = false;
44      ////////////////////////////////
45
46      ///////////  Shorter version of Get/Set property declaration  ////////////
47      // The C#  compiler will generate the same as the above for you, automatically.
48      public int DifficultyLevel{get;set;}
49      public bool MusicEnabled{get;set;}
50      public bool SFXEnabled{get;set;}
51      ////////////////////////////////
52  }
```

5.5.4 [ณ ฉาก SceneMainMenu] กำหนดเมธอดที่จะถูกเรียกเมื่อเกิดเหตุการณ์คลิกให้กับปุ่ม UI ต่าง ๆ Start Options และ Exit

สร้างสคริปต์ด้วยเมนู Assets->Create->C Sharp Script และตั้งชื่อว่า “MainMenuControlScript” เขียนสคริปต์ดัง??

สร้างวัตถุเกมเปล่าสำหรับติดตั้งคอมโพเนนท์สคริปต์ที่ใช้ควบคุมเมนูหลัก ตั้งชื่อเป็น MainMenuControl และติดตั้งคอมโพเนนท์สคริปต์ MainMenuControlScript ลงบนวัตถุเปล่าที่สร้างขึ้น

Source code 5.5: MainMenuControlScript.cs

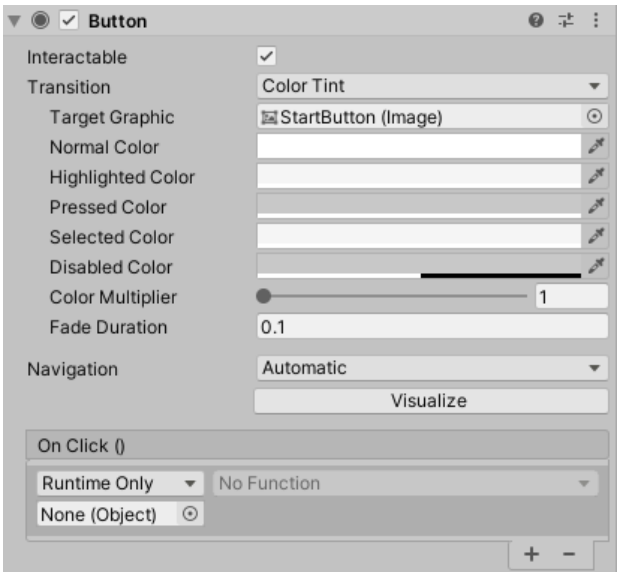
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

using UnityEngine.SceneManagement;

public class MainMenuControlScript : MonoBehaviour {

  // Use this for initialization
  void Start () {
  }

  // Update is called once per frame
  void Update () {
  }

  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 ExitButtonClick(Button button) {
        Application.Quit();
  }
}
```

ส่วนต่อไปนี้ ให้เลือกทำตามวิธีใดวิธีหนึ่ง หรือทดลองทำทั้งสองวิธี คราวละวิธี

วิธีที่ 1 กำหนดผ่านส่วนติดต่อผู้ใช้ (User Interface) ของ Unity (UnityEvent)

คลิกเลือกปุ่ม StartButton บนหน้าต่าง Hierarchy ในส่วนของคอมโพเนนท์ Button ของปุ่ม StartButton คลิกเครื่องหมาย + ที่เหตุการณ์ On Click() เพื่อเพิ่มเป้าหมายของการทำงานเมื่อเกิดการกดปุ่ม

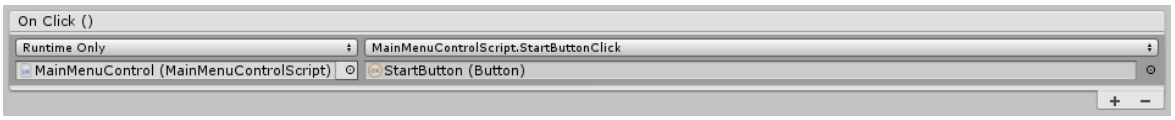
ลากวัตถุ MainMenuControlScript (ถูกติดตั้งให้กับ Canvas) จากหน้าต่าง Hierarchy มาวางที่ None (Object) ของปุ่ม StartButton ดังรูปที่ 5.7



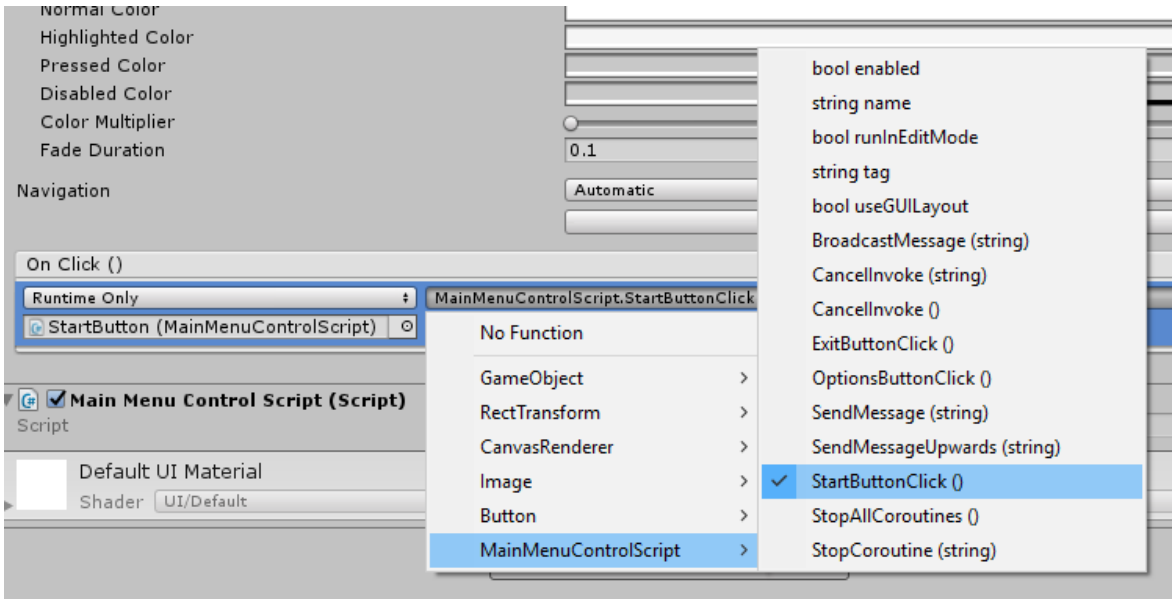
รูปที่ 5.7: Button Component

กำหนดเมธอดที่จะถูกเรียกเมื่อเกิดเหตุการณ์การกดปุ่ม (On Click () event) ของปุ่ม Start ดังรูปที่ 5.8

เลือกเมธอดที่จะกำหนดให้ทำงานเมื่อกดปุ่มดัง รูปที่ 5.9 ด้วยการเริ่มเลือกที่ No Function และเลือกเมธอด Start-ButtonClick() ซึ่งเป็นเมธอด public ที่อยู่ภายในคอมโพเนนท์ MainMenuControlScript

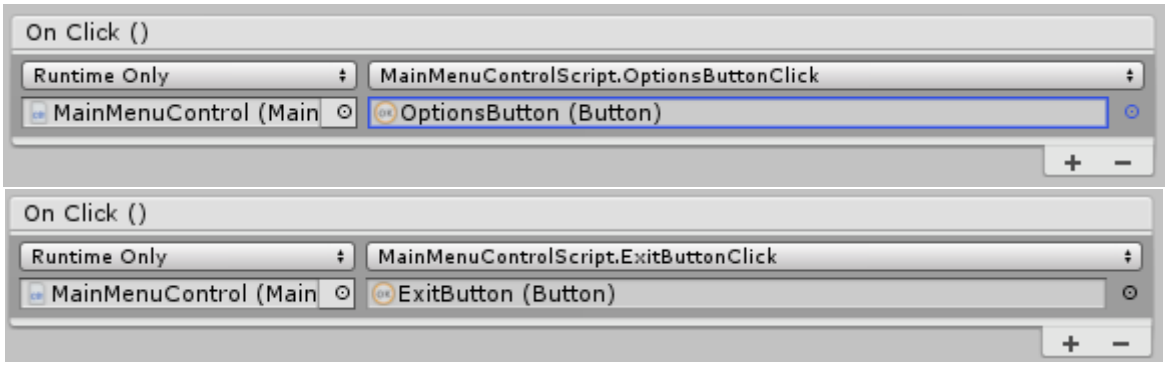


รูปที่ 5.8



รูปที่ 5.9: การเพิ่มเมธอดที่จะถูกเรียกให้กับเหตุการณ์กดปุ่ม หรือ On Click ()

ทำซ้ำ เพื่อกำหนดเมธอดให้กับปุ่ม Options และ Exit ด้วยเมธอด OptionsButtonClick และ ExitButtonClick ตามลำดับดังรูปที่ 5.10



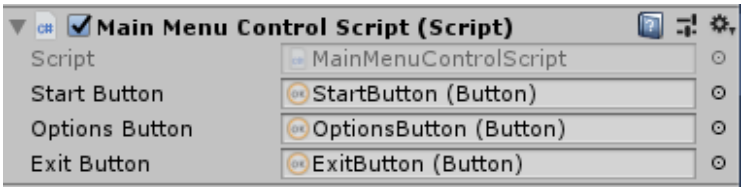
รูปที่ 5.10: การเพิ่มเมธอดที่จะถูกเรียกให้กับเหตุการณ์กดปุ่ม หรือ onClick

เซฟและปิดฉาก SceneMainMenu

วิธีที่ 2 กำหนดด้วยการเขียนโปรแกรมกำหนดเมธอดเหตุการณ์ด้วย event delegate

แก้ไขสคริปต์ MainMenuControlScript.cs โดยเพิ่มส่วนต่าง ๆ ลงไปดัง Sourcecode 5.6 เมื่อแก้ไขสคริปต์เสร็จให้ทำดังต่อไปนี้

- คลิกเลือกวัตถุ MainMenuControl บนหน้าต่าง Hierarchy
- นำปุ่ม StartButton OptionsButton และ ExitButton ลากติดตั้งให้กับตัวแปรบนสคริปต์ MainMenu-ControlScript บนหน้าต่าง Inspector ดังรูปที่ 5.11



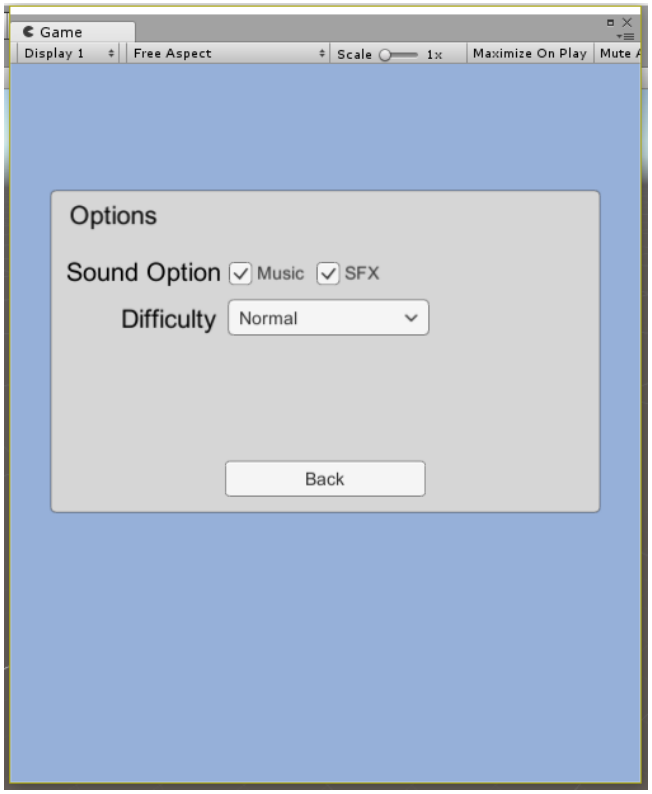
รูปที่ 5.11: กำหนดตัวแปรปุ่มให้กับ MainMenuControl

Source code 5.6: แก้ไข MainMenuControlScript.cs

```
1  //Declarations section
2    [SerializeField] Button _startButton;
3    [SerializeField] Button _optionsButton;
4    [SerializeField] Button _exitButton;
5  //Inside Start() method
6    void Start () {
7      _startButton.onClick.AddListener (
8        delegate{StartButtonClick(_startButton);});
9      _optionsButton.onClick.AddListener (
10       delegate{OptionsButtonClick(_optionsButton);});
11     _exitButton.onClick.AddListener (
12       delegate{ExitButtonClick(_exitButton);});
13     }
```

5.5.5 สร้าง User Interface และสคริปต์ต่าง ๆ ของฉาก SceneOptions

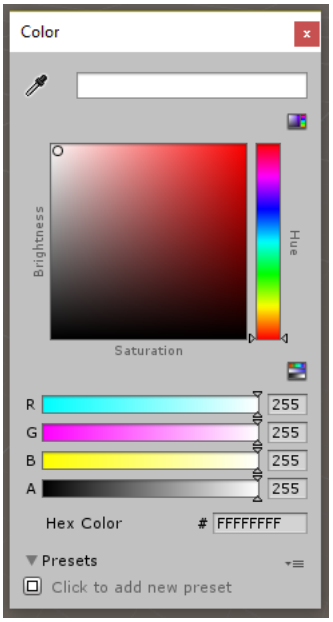
เปิดฉาก SceneOptions เพื่อแก้ไข และสร้าง User Interface ดังรูปที่ 5.12



รูปที่ 5.12: SceneOptions UI design

สร้าง Panel เพื่อจัดวาง UI ต่าง ๆ จากเมนู GameObject->UI->Panel และ Canvas จะถูกสร้างขึ้นมาอัตโนมัติ เปลี่ยนชื่อเป็น CanvasOption

จัดวาง Panel ไว้กลางจอและปรับสีของ Panel ส่วนของ Alpha ให้ทึบแสงดังรูปที่ 5.13



รูปที่ 5.13: การปรับ Alpha ของ Panel ให้ทึบแสง

- สร้าง UI ต่อไปนี้ และจัดวางให้เป็น Parent และ Child ดังรูปที่ 5.14

UI ประเภท Text

TextOption

TextSoundOption

TextDifficulty

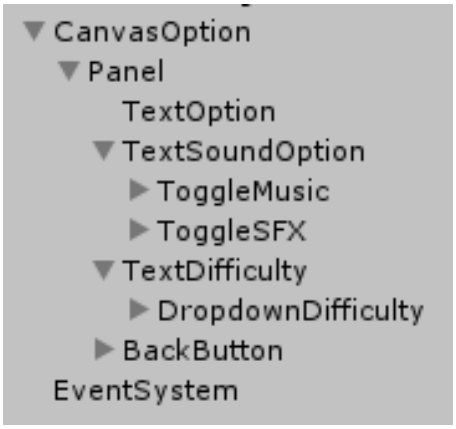
UI ประเภท Toggle

ToggleMusic

ToggleSFX

UI ประเภท Button

BackButton



รูปที่ 5.14: การจัดวาง Parent และ Child สำหรับ UI บนหน้าจอฉาก SceneOptions

สร้างสคริปต์ใหม่ ตั้งชื่อว่า “OptionsMenuControlScript” เขียนสคริปต์ดัง Sourcecode 5.7

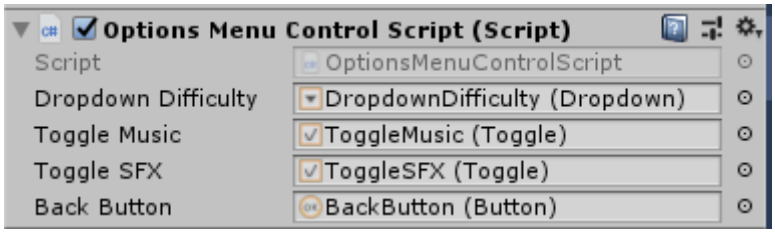
สร้างวัตถุเปล่า ตั้งชื่อเป็น OptionsMenuControl และทำการติดตั้งคอมโพเนนท์สคริปต์ OptionsMenuControlScript ลงบนวัตถุ

ลากวัตถุ UI ต่าง ๆ เพื่อกำหนดให้กับให้กับคอมโพเนนท์สคริปต์ OptionsMenuControlScript ดังรูปที่ 5.15

Source code 5.7: OptionsMenuControlScript.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5  using UnityEngine.SceneManagement;
6
7  public class OptionsMenuControlScript : MonoBehaviour {
8
9    [SerializeField] Dropdown _dropdownDifficulty;
10   [SerializeField] Toggle _toggleMusic;
11   [SerializeField] Toggle _toggleSFX;
12   [SerializeField] Button _backButton;
13
14     // Use this for initialization
15     void Start () {
16   _dropdownDifficulty.value = GameApplicationManager.Instance.DifficultyLevel;
17   _toggleMusic.isOn = GameApplicationManager.Instance.MusicEnabled;
18   _toggleSFX.isOn = GameApplicationManager.Instance.SFXEnabled;
19
```

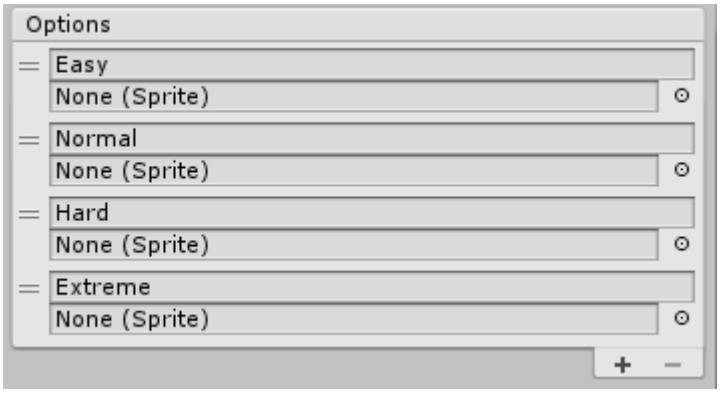
```
20    _dropdownDifficulty.onValueChanged.AddListener(delegate { DropdownDifficultyChanged(
          _dropdownDifficulty); });
21    _toggleMusic.onValueChanged.AddListener( delegate {OnToggleMusic(_toggleMusic);});
22    _toggleSFX.onValueChanged.AddListener( delegate { OnToggleSFX(_toggleSFX);});
23    _backButton.onClick.AddListener(delegate { BackButtonClick(_backButton); });
24      }
25
26      // Update is called once per frame
27      void Update () {
28
29    }
30
31      public void BackButtonClick(Button button) {
32          SceneManager.UnloadSceneAsync("SceneOptions");
33          GameApplicationManager.Instance.IsOptionMenuActive = false;
34      }
35
36      public void DropdownDifficultyChanged(Dropdown dropdown) {
37          GameApplicationManager.Instance.DifficultyLevel = dropdown.value;
38      }
39
40      public void OnToggleMusic(Toggle toggle) {
41          GameApplicationManager.Instance.MusicEnabled = _toggleMusic.isOn;
42      }
43      public void OnToggleSFX(Toggle toggle)
44      {
45          GameApplicationManager.Instance.SFXEnabled = _toggleSFX.isOn;
46      }
47  }
```



รูปที่ 5.15: กำหนดคุณสมบัติให้กับคอมโพเนนท์สคริปต์ OptionsMenuControlScript

กำหนดตัวเลือกของ Dropdown Difficulty ดังรูปที่ 5.16 Easy Normal Hard และ Extreme

เซฟและปิดฉาก SceneOptions

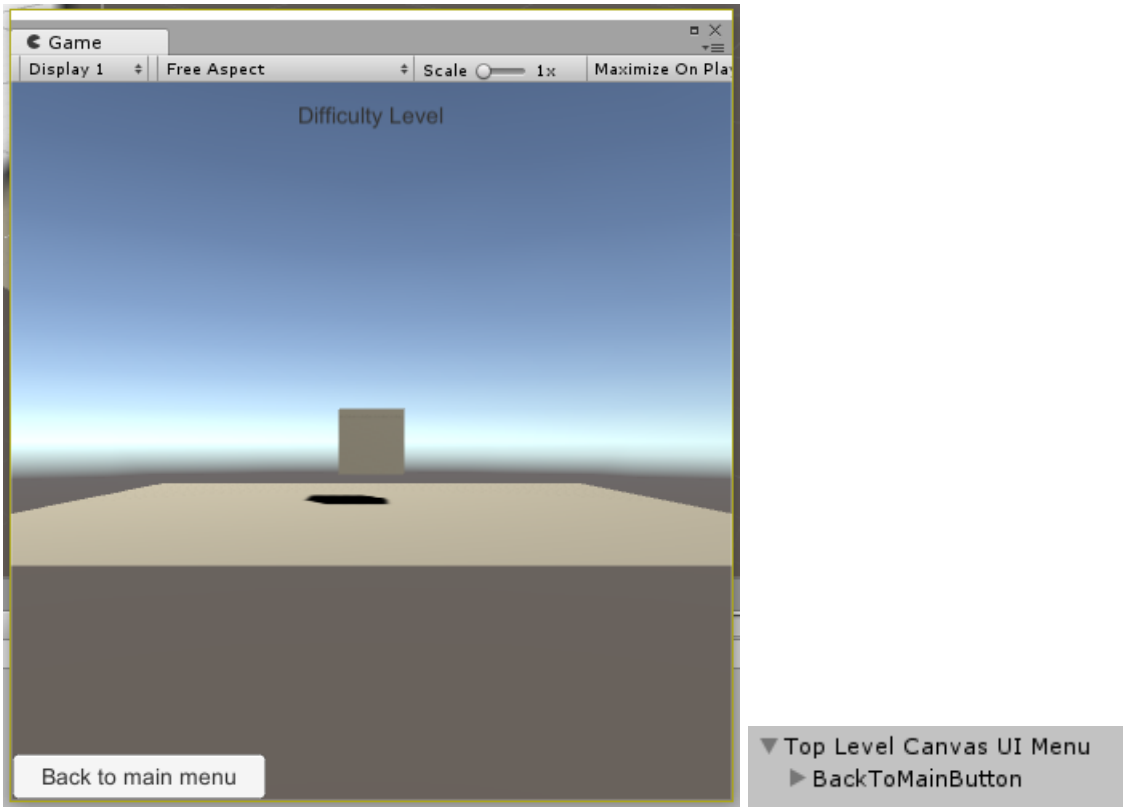


รูปที่ 5.16: กำหนดตัวเลือกของ Dropdown Difficulty

5.5.6 สร้างเนื้อหาในฉากเกมเพล์

เปิดฉาก SceneGameplay

- สร้างวัตถุเกมขึ้นมาตามต้องการ เพื่อให้ทราบว่าเป็นฉากสำหรับเล่นเกม
- สร้าง UI ปุ่มสำหรับกดเพื่อกลับสู่หน้าจอเมนู ดังรูปที่ 5.17 เปลี่ยนชื่อ Canvas เป็น “Top Level Canvas UI Menu”



รูปที่ 5.17: การจัดวางหน้าจอเกม SceneGameplay

เพิ่มสคริปต์ GameplayMenuControlScript.cs เขียนซอสโคดดัง Sourcecode 5.8

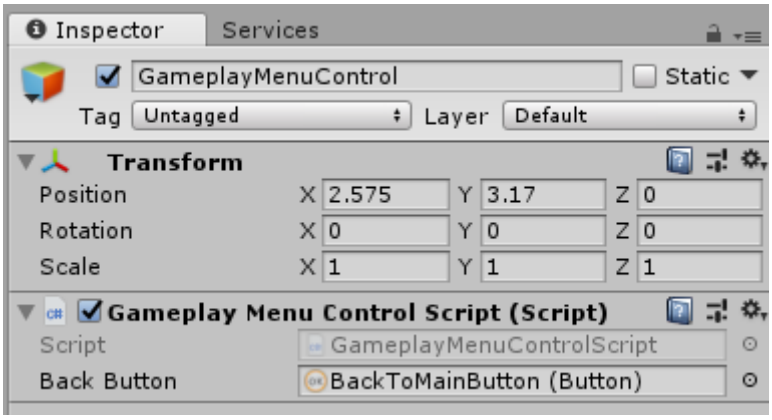
สร้างวัตถุเปล่า ตั้งชื่อว่า GameplayMenuControl และติดตั้งคอมโพเนนท์ GameplayMenuControlScript

นำปุ่ม Back Button กำหนดให้กับคอมโพเนนท์ GameplayMenuControlScript ดัง รูปที่ 5.18

Source code 5.8: GameplayMenuControlScript.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  using UnityEngine.SceneManagement;
6  using UnityEngine.UI;
7
8  public class GameplayMenuControlScript : MonoBehaviour {
9      [SerializeField] Button _backButton;
10     // Use this for initialization
11     void Start () {
```

```
12          _backButton.onClick.AddListener
13              (delegate { BackToMainMenuButtonClick(_backButton); });
14      }
15
16      // Update is called once per frame
17      void Update () {
18
19  }
20
21      public void BackToMainMenuButtonClick(Button button) {
22          SceneManager.UnloadSceneAsync("SceneGameplay");
23          SceneManager.LoadScene("SceneMainMenu");
24      }
25  }
```



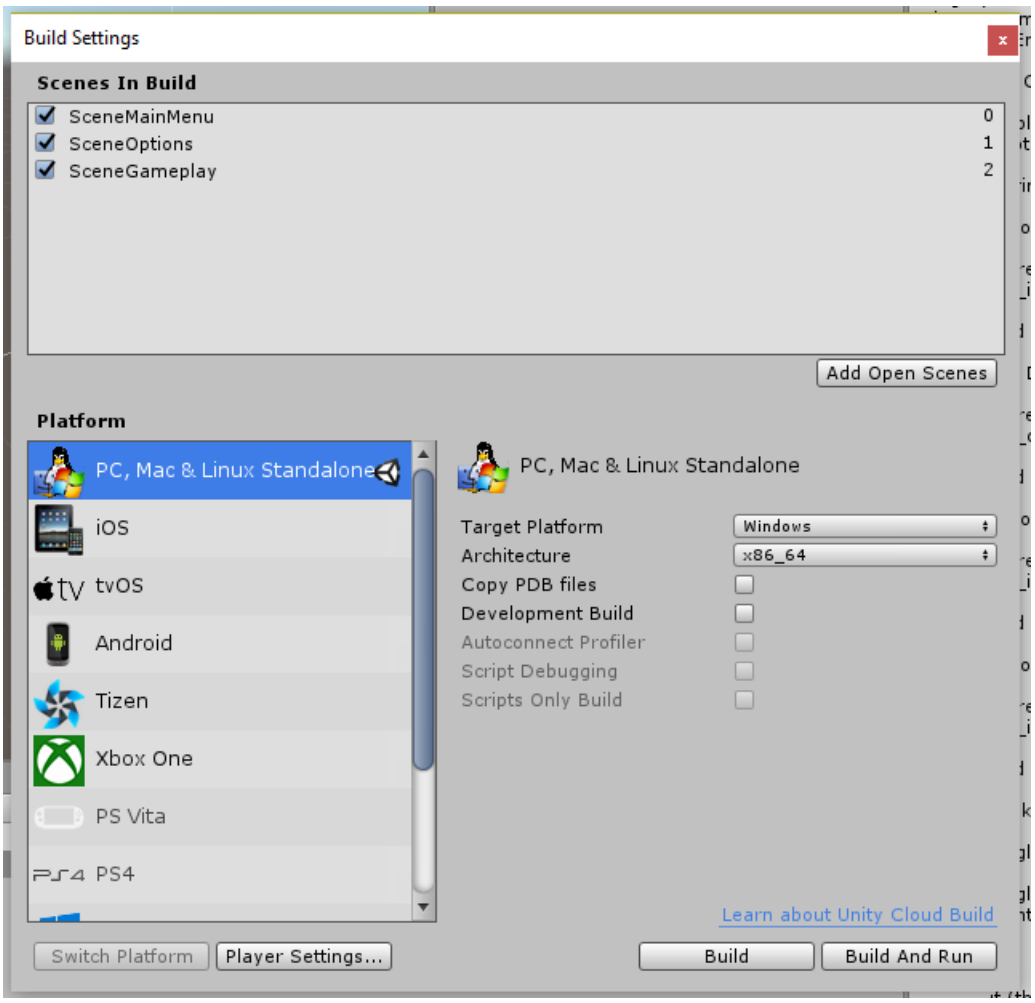
รูปที่ 5.18: กำหนดคุณสมบัติ Back Buttonให้สคริปต์ GameplayMenuControlScript

5.6 กำหนดฉากที่จะถูกรวบรวมเพื่อสร้างไฟล์เกม (Building a Game Package)

กำหนดฉากที่จะถูกรวบรวมเพื่อสร้างไฟล์เกมด้วยการเข้าเมนู File->Build Settings... ดัง รูปที่ 5.19

ลากฉาก SceneMainMenu SceneOptions และ SceneGameplay เข้าไปวางในช่อง Scenes In Build โดยลำดับของฉากที่จะปรากฏเป็นฉากแรกคือฉากบนสุด ซึ่งในกรณีนี้ควรจะเป็นฉาก SceneMainMenu

กดปุ่ม Build หรือ Build And Run เพื่อสร้างไฟล์เกม จะปรากฏหน้าต่างให้กำหนดชื่อไฟล์ Executable (*.exe) ให้กำหนดเป็นชื่อ ScenesManagement.exe หรือตามที่ผู้พัฒนาต้องการ



รูปที่ 5.19: กำหนดฉากที่จะรวบรวมเข้าในการสร้างไฟล์เกม (Executable Game)

คำถามสำหรับ chapter 5 การจัดการฉาก (Scenes Management)

ทดสอบการทำงานของโปรแกรม และอธิบาย Flow ของการเปลี่ยนฉาก กลไกการเปลี่ยนฉากต่าง ๆ ความแตกต่างของการเปลี่ยนฉากแบบธรรมดาและแบบ Additive
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .

จะเกิดอะไรขึ้นหากเปลี่ยนไปยังฉาก SceneOptions โดยไม่ตรวจสอบถึงตัวแปร IsOptionMenuActive ในคลาส GameApplicationManger (Additive Scene Loading)

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

สังเกตลักษณะ พฤติกรรมของ GameApplicationManager อธิบายตามความเข้าใจ และอธิบายการทำงานในเมธอด Awake() ของ GameApplicationManager

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

อธิบายแนวคิด Singleton ตามที่นักศึกษาเข้าใจ

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

เพิ่มฉากที่เคยทำในคาบก่อน ๆ หน้า อีกอย่างน้อย 2 ฉาก (เช่น Game Objects Management หรือในวิชา 951302) พร้อมปุ่มและกลไกในการเข้า-ออกจากฉากที่เพิ่มเข้าไป อธิบายว่าเพิ่มอะไรเข้าไปและ กลไกเปลี่ยนฉากเป็นอย่างไร

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

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

5.7 หัวข้อเพิ่มเติมเรื่องการ Implement Singleton ให้สามารถใช้งานได้ง่ายยิ่งขึ้น

จะเห็นได้ว่าจากหัวข้อ หัวข้อ 5.5.3 ในการสร้างคลาส GameApplicationManager แบบ Singleton มีส่วนของการเขียนโปรแกรมที่ซับซ้อนพอสมควร อย่างไรก็ตามชุมชนนักพัฒนา Unity (Unity Community Developer) ได้สร้างคลาสต้นแบบ Singleton ที่ทำให้ผู้ใช้งานสามารถสืบทอดและนำไปใช้งานได้โดยง่าย แสดงดัง Sourcecode 5.9

และตัวอย่างการสืบทอดจากคลาส Singleton ดังกล่าวเพื่อการนำไปใช้งานแสดงดัง Sourcecode 5.10

Source code 5.9: 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                      }
42                  }
43
44                  return m_Instance;
45              }
46          }
47      }
48
49
50      private void OnApplicationQuit()
51      {
52          m_ShuttingDown = true;
53      }
54
55
56      private void OnDestroy()
57      {
58          m_ShuttingDown = true;
59      }
60  }
```

Source code 5.10: SingletonGameManager.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class SingletonGameManager : Singleton<SingletonGameManager>
6  {
7      protected SingletonGameManager(){}
8
9      public string ClassName{get;} = "SingletonGameManager";
10
11     public int GameScore{get;set;} = 0;
12 }
```

คำถามสำหรับ หัวข้อ 5.7 หัวข้อเพิ่มเติมเรื่องการ Implement Singleton ให้สามารถใช้งานได้ง่ายยิ่งขึ้น

Implement เทมเพลท Singleton และสร้างคลาสของตนเองเพื่อนำไปใช้ในการเก็บข้อมูลชื่อของผู้เล่น และแสดงในทุก ๆ ฉากที่มุมซ้ายบนของจอภาพ

ชื่อไฟล์วิดีโอ . . . . . .

1/2563

ชื่อ...นายปาล์เมษ เกตุรักษา.....

รหัสนักศึกษา.....6212110055.....

ตอนที่.....001.....

สาขาวิชาแอนนิเมชันและเกม วิทยาลัยศิลปะ สื่อ และเทคโนโลยี มหาวิทยาลัยเชียงใหม่

5.8 คำถามและปัญหาชวนคิด (Questions and Problems)

คำถามสำหรับ chapter 5 การจัดการฉาก (Scenes Management)

ทดสอบการทำงานของโปรแกรม และอธิบาย Flow ของการเปลี่ยนฉาก กลไกการเปลี่ยนฉากต่าง ๆ ความแตกต่างของการเปลี่ยนฉากแบบธรรมดาและแบบ Additive

ใช้event ของปุ่มแบบ onClick ในการเปลียนแปลงฉาก ด้วยคำสั่ง loadScene

โดยมีหน้าแรกไว้รวบทางเข้าสู่หน้าต่างๆ และจากหน้าอื่นๆสามารถกลับมาหน้าแรกได้

แบบธรรมดาจะไม่สามารถย้ายค่าหรือส่งค่าให้ฉากอื่นได้

additiveจะสามารถส่งค่าให้ฉากอื่นๆได้

จะเกิดอะไรขึ้นหากเปลี่ยนไปยังฉาก SceneOptions โดยไม่ตรวจสอบถึงตัวแปร IsOptionMenuActive ในคลาส GameApplicationManger (Additive Scene Loading)

มีไว้เพื่อบอกว่ากำลังใช้งานหน้าoption อยู่หรือไม่

สังเกตลักษณะ พฤติกรรมของ GameApplicationManager อธิบายตามความเข้าใจ และอธิบายการทำงานในเมธอด Awake() ของ GameApplicationManager

ถ้าหากไม่มี instanceของตัวเองให้สร้างมาใหม่ ถ้ามีซ้ำให้ลบตัวนี้ เพราะinstance มีซ้ำกันไม่ได้

อธิบายแนวคิด Singleton ตามที่นักศึกษาเข้าใจ

สร้าง class แบบ มีลักษณะเฉพาะตัวทำให้มีได้แค่ตัวเดียวเหมือนคนที่ไม่เหมือนกัน

เพิ่มฉากที่เคยทำในคาบก่อน ๆ หน้า อีกอย่างน้อย 2 ฉาก (เช่น Game Objects Management หรือในวิชา 951302) พร้อมปุ่มและกลไกในการเข้า-ออกจากฉากที่เพิ่มเข้าไป อธิบายว่าเพิ่มอะไรเข้าไปและ กลไกเปลี่ยนฉากเป็นอย่างไร

สร้างCode ใหม่ เพื่อ รับปุ่มกดและชื่อของScene ที่จะโหลด พอกดก็จะโหลดมา

สร้างcode เปลี่ยน ฉากแบบตั้งเวลา พอเวลาลนลงจนเหลือ0 จะเปลี่ยนหน้าไป

คำถามสำหรับ หัวข้อ 5.7 หัวข้อเพิ่มเติมเรื่องการ Implement Singleton ให้สามารถใช้งานได้ง่ายยิ่งขึ้น

Implement เทมเพลท Singleton และสร้างคลาสของตนเองเพื่อนำไปใช้ในการเก็บข้อมูลชื่อของผู้เล่น และแสดงในทุก ๆ ฉากที่มุมซ้ายบนของจอภาพ

ชื่อไฟล์วิดีโอ . . . . .001. . . . . .