國立臺灣大學

計算機程式設計期末專題成果報告

太鼓達人

作者：大氣一 謝晉維

國企一 范軒榮

中華民國 一百零九 年 六 月

太鼓達人

摘　要

（一）論述重點

做出與太鼓達人功能相同之程序，並且建立在此基礎上新增功能。太鼓達人主要功能包括介面的轉換、音符的判定、音符的生成。新增的功能為闖關模式。報告中會依序用程式碼講解我們如何寫出遊戲的功能，以及透過遊戲實際的截圖畫面輔助說明

（二）方法及程序

利用 Unity 及 C#，完成太鼓達人基本的功能。程序部分會在正文中詳細說明。

目　錄

圖目錄

第一章、研究動機

從前一直覺得寫遊戲是一件遙不可及的事，很想要嘗試看看能不能靠自己完成一個遊戲。因為非常喜歡太鼓，想要做一個類似太鼓機台的遊戲，雖然網路上已經有類似的模擬器，但想要仔細理解每一個步驟該用哪些指令和程式碼達成，並且利用 Unity 和 C#實現預想中的遊戲。初步是要模仿完成類似太鼓機台的程式，完成後就是創意發想的部分，建立在原有太鼓達人的功能上加上一些新功能，例如闖關模式。

關鍵字：Unity、C#、太鼓達人

第二章、相關之課程章節

(一) 第二章　整合發展環境與簡易C#程式

如何使用：了解 IDE 的運作原理，自行探索 Rider（Jetbrains 的 IDE)

(二) 第三章　實值變數與運算式

如何使用：太鼓分數判定、音符生成、碰撞判定等功能

(三) 第四章　流程控制

如何使用：各種迴圈

(四) 第五章　陣列

如何使用：音符生成陣列

(五) 第六章　函式

如何使用：功能（如：分數計算、字符產生）

(六) 第七章　程式規劃與函式導向程式設計

如何使用：版本規劃

(七) 第八章　物件與類別

如何使用：class 及建構式的使用

（八）第九章　物件導向程式設計

如何使用：UML 圖

（九）第十章　繼承與多型

如何使用：繼承 Unity 的 class

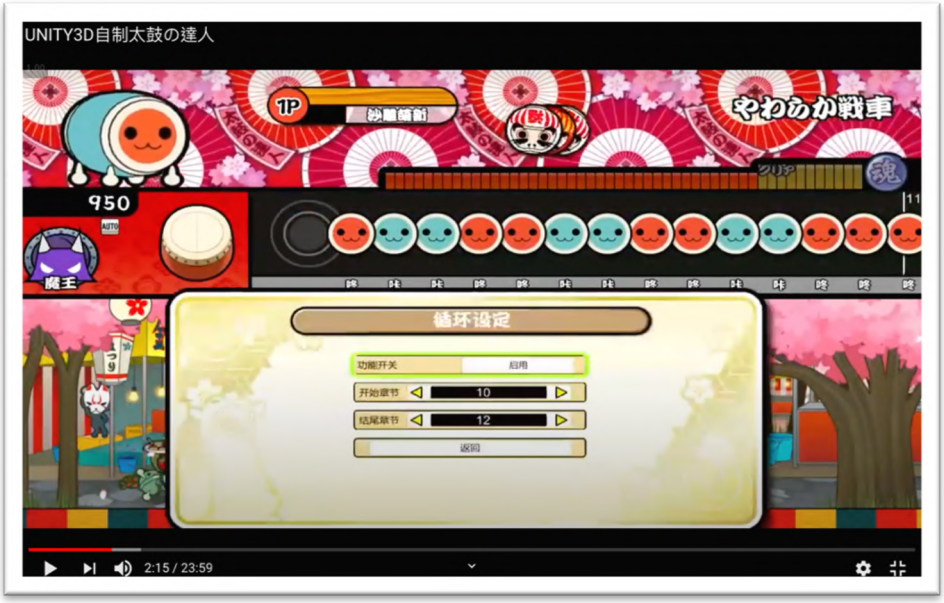
（十）補充影片Unity教學

如何使用：使用 Unity

第三章、文獻回顧理論說明

我們在網路上尋找其他人實作的太鼓達人，發現其實不乏蠻多的版本，如以下幾個都算是功能相對完備的太鼓達人：

1. https://www.youtube.com/watch?v=62FHq5HLvlM&t=117s



【圖（3.1）他人實作太鼓達人】

2. https://www.youtube.com/watch?v=nGuTHeihCn8

【圖（3.2）他人實作太鼓達人】

然而以上兩個用 Unity 實作太鼓達人的案例，其實都仍停留在復刻出一款和原本太鼓達人功能相同的遊戲，並沒有多做創新，因此我們便發現了這個問題並想要加以改善，而我們的改善方法就是新增遊戲功能，因為原先的太鼓達人就是選擇音樂、難度就開始遊戲，我們便想要新增闖關功能，然後在越往後的關卡難度便會自己提升，以增加太鼓達人的遊戲性。

第四章、實作流程架構

以下內容按照各個階段完成的進度條列，最後附上 UML 及流程圖

（一）第一次報告

1. 能夠敲打音符（含判定得分、音符的消失）
2. 播放背景音樂及敲打音效
3. 判斷敲擊位置決定分數，並顯示於加分條

（二）第二次報告

1. 譜面生成——含音符生成、移動
2. 選單介面（主選單、選歌、難度選擇）
3. 敲擊時會顯示良、可、不可等字樣
4. 連續敲擊成功時會有 Combo 的累加
5. 音符回收池

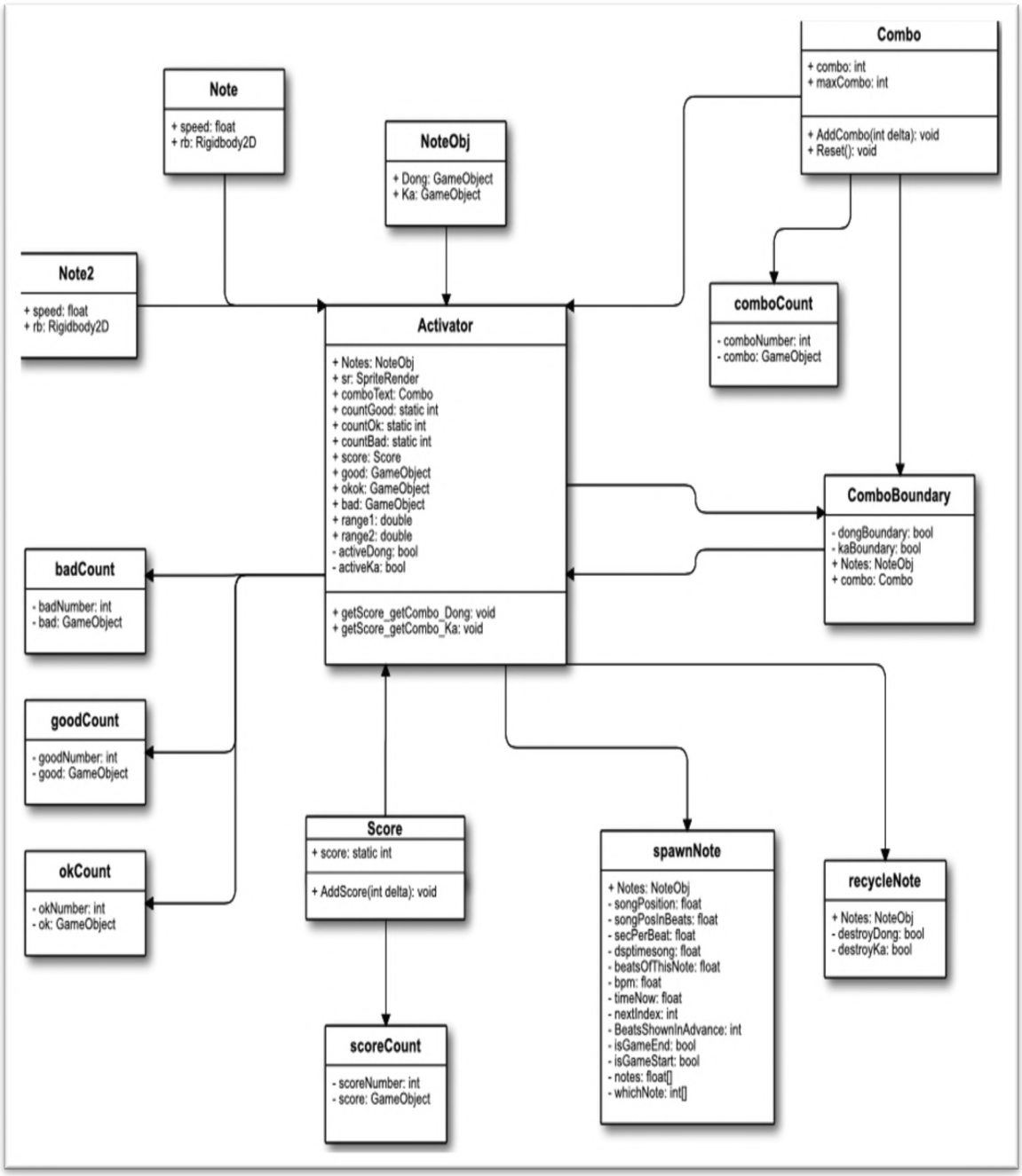
（三）第三次報告

1. 修正之前的bug（Combo、音符生成）
2. 設定譜面
3. 增加更多歌曲的選單
4. 增加不同歌曲的難易度選擇

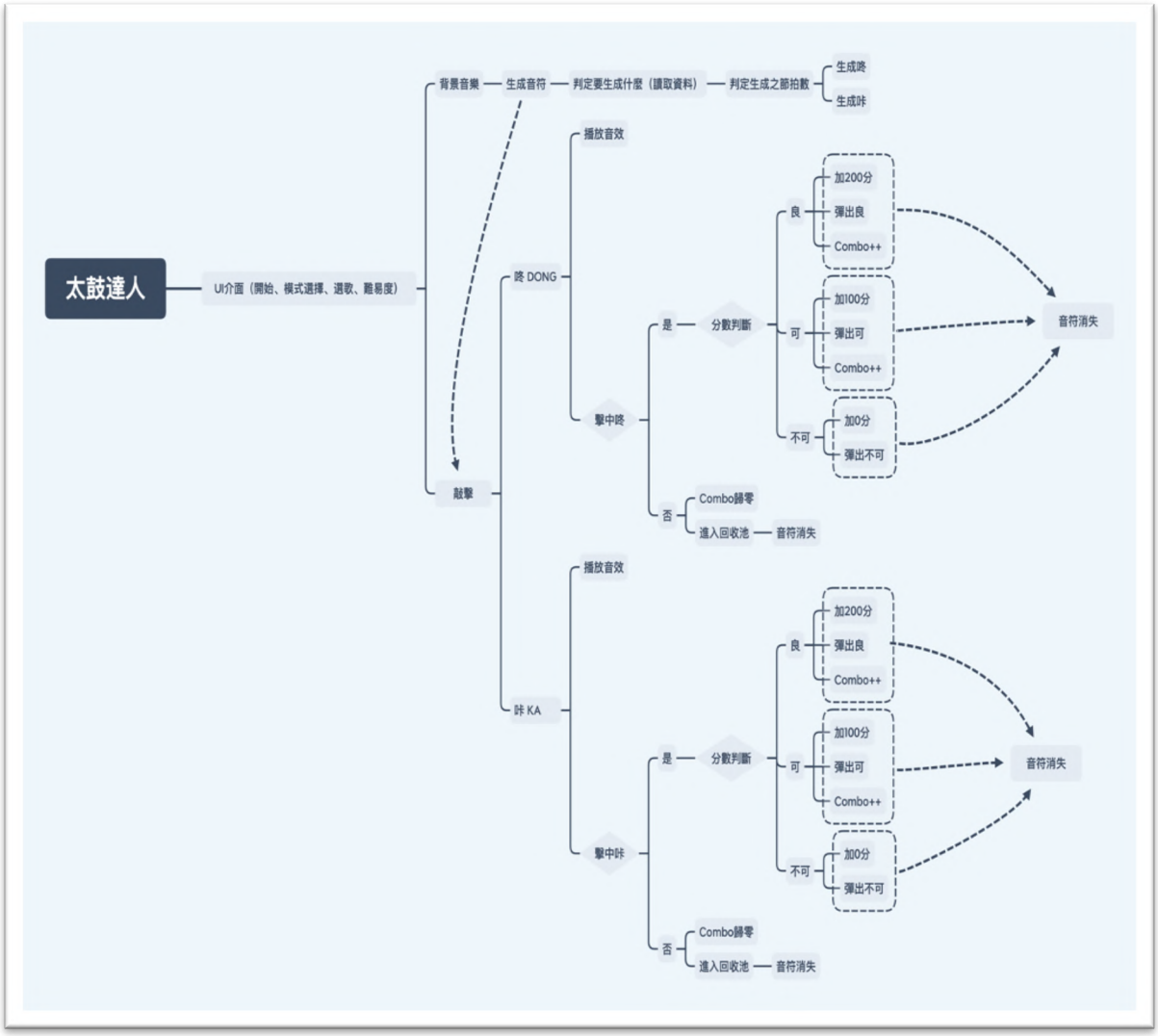
（四）期末報告

1. 闖關版（含配合闖關模式的各種UI介面）

2. 計分板



【圖（4.1）UML 圖】


太鼓達人
UI介面（開始、模式選擇、選歌、難易度）
背景音樂
生成音符
判定要生成什麼（讀取資料）
判定生成之節拍數
生成咚
生成咔
咚 DONG
播放音效
擊中咚
是
分數判斷
良
加200分
彈出良
Combo++
可
加100分
彈出可
Combo++
不可
加0分
彈出不可
音符消失
否
Combo歸零
進入回收池
音符消失
敲擊
咔 KA
播放音效
擊中咔
是
分數判斷
良
加200分
彈出良
Combo++
可
加100分
彈出可
Combo++
不可
加0分
彈出不可
音符消失
否
Combo歸零
進入回收池
音符消失

【圖（4.2）流程圖】

第五章、研究方法及過程

以下放上不同物件的程式碼並在開頭搭配文字解說功能

(一) Activator

1. 功能

a. Dong、Ka的聲音播放

b. 敲打時鼓的顏色變化

c. 得分計算

d. Combo計算

e. 良、可、不可字符顯示

2. 程式碼

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

[Serializable]
public class NoteObj
{
    public GameObject Dong, Ka;
}

public class Activator : MonoBehaviour
{
    public NoteObj Notes;
    public SpriteRenderer sr;
    public KeyCode key_Dong1, key_Dong2, key_Ka1, key_Ka2;

    private bool activeDong = false;
```

```
private bool activeKa = false;

// Combo
public Combo comboText;
public static int countGood;
public static int countOk;
public static int countBad;



public Color old;
public Score score;
public GameObject good;
public GameObject okok;
public GameObject bad;

// 判定
public double range1, range2; // To judge 良、可、不可

void Awake()
{
    sr = GetComponent<SpriteRenderer>();
    good.SetActive(false);
    okok.SetActive(false);
    bad.SetActive(false);
}

void Update()
{
    // Dong 聲音、顏色
    if (Input.GetKeyDown(key_Dong1) || Input.GetKeyDown(key_Dong2))
    {
        SoundManager.instance.Dong();
        StartCoroutine(Pressed());
    }

    // Ka 聲音、顏色
    if (Input.GetKeyDown(key_Ka1) || Input.GetKeyDown(key_Ka2))
    {
```

```
            SoundManager.instance.Ka();
            StartCoroutine(Pressed());
        }


        // Dong 得分
        if ((Input.GetKeyDown(key_Dong1) || Input.GetKeyDown(key_Dong2)) &&
            (!Input.GetKeyDown(key_Ka1) || !Input.GetKeyDown(key_Ka2)) && activeDong)
        {
            getScore_getCombo_Dong();
            if (Notes.Dong)
            {
                Destroy(Notes.Dong);
            }
            activeDong = false;
        }

        // Ka 得分
        if ((Input.GetKeyDown(key_Ka1) || Input.GetKeyDown(key_Ka2)) &&
            (!Input.GetKeyDown(key_Dong1) || !Input.GetKeyDown(key_Dong2)) && activeKa)
        {
            getScore_getCombo_Ka();
            if (Notes.Ka)
            {
                Destroy(Notes.Ka);
            }
            activeKa = false;
        }
    }

    void OnTriggerEnter2D(Collider2D col)
    {
        if (col.gameObject.tag == "Note")
        {
            activeDong = true;
            Notes.Dong = col.gameObject;
        }
```

```
        if (col.gameObject.tag == "Note2")
        {
            activeKa = true;
            Notes.Ka = col.gameObject;
        }
    }

    void OnTriggerExit2D(Collider2D col)
    {
        if (col.gameObject.tag == "Note")
        {
            activeDong = false;
            Notes.Dong = col.gameObject;
        }

        if (col.gameObject.tag == "Note2")
        {
            activeKa = false;
            Notes.Ka = col.gameObject;
        }

    }

    void getScore_getCombo_Dong()
    {
        // 判斷精準度
        if (Math.Abs(Notes.Dong.transform.position.x - sr.transform.position.x) <= range1) // 良
        {
            //得分
            score.AddScore(100 * 2);
            comboText.AddCombo(1);
            StartCoroutine(ShowGood());
            countGood++; // 計算良
        }
        else if (range1 < Math.Abs(Notes.Dong.transform.position.x - sr.transform.position.x) &&
                 Math.Abs(Notes.Dong.transform.position.x - sr.transform.position.x) < range2)
// 可
        {
```

```
            //得分
            score.AddScore(100);
            comboText.AddCombo(1);
            StartCoroutine(ShowOk());
            countOk++; // 計算可
        }

        else
        {
            comboText.combo = 0;
            StartCoroutine(ShowBad());
            countBad++; // 計算不可
        }
    }


    void getScore_getCombo_Ka()
    {
        // 判斷精準度
        if (Math.Abs(Notes.Ka.transform.position.x - sr.transform.position.x) <= range1) // 良
        {
            //得分
            score.AddScore(100 * 2);
            comboText.AddCombo(1);
            StartCoroutine(ShowGood());
            countGood++; // 計算良
        }
        else if (range1 < Math.Abs(Notes.Ka.transform.position.x - sr.transform.position.x) &&
                Math.Abs(Notes.Ka.transform.position.x - sr.transform.position.x) < range2) //
可
        {
            //得分
            score.AddScore(100);
            comboText.AddCombo(1);
            StartCoroutine(ShowOk());
            countOk++; // 計算可
        }
```

```
            else
            {
                comboText.combo = 0;
                StartCoroutine(ShowBad());
                countBad++; // 計算不可
            }
        }


        IEnumerator Pressed()
        {
            Color old = sr.color;
            sr.color = Color.yellow;
            yield return new WaitForSeconds(0.05f);
            if (sr.color == Color.yellow) sr.color = old;
        }

        IEnumerator ShowGood()
        {
            good.SetActive(true);
            yield return new WaitForSecondsRealtime(0.15f);
            good.SetActive(false);
        }

        IEnumerator ShowOk()
        {
            okok.SetActive(true);
            yield return new WaitForSecondsRealtime(0.15f);
            okok.SetActive(false);
        }

        IEnumerator ShowBad()
        {
            bad.SetActive(true);
            yield return new WaitForSecondsRealtime(0.15f);
            bad.SetActive(false);
        }
    }
```

(二) Combo

1. 功能

   a. 計算Combo的累積

2. 程式碼

```
using UnityEngine;
using TMPro;
public class Combo : MonoBehaviour
{
    public int combo;
    public static int maxCombo;

    void Update()
    {
        if (maxCombo <= combo) { maxCombo = combo; } // 計算最大Combo
        GetComponent<TMP_Text>().text = combo + "";
    }

    public void AddCombo(int delta)
    {
        combo += delta;
    }

    public void Reset()
    {
        combo = 0;
    }
}
```

(三) comboBoundary

1. 功能

a. 當未擊打到音符時，使combo歸零

2. 程式碼

```
using UnityEngine;

public class comboBoundary : MonoBehaviour
{
    private bool dongBoundary = false;
    private bool kaBoundary = false;

    public NoteObj Notes;
    public Combo combo;

    // Update is called once per frame
    void Update()
    {
        if (dongBoundary || kaBoundary)    combo.Reset();
    }

    void OnTriggerEnter2D(Collider2D col)
    {
        if (col.gameObject.tag == "Note")
        {
            dongBoundary = true;
            Notes.Dong = col.gameObject;
        }

        if (col.gameObject.tag == "Note2")
        {
            kaBoundary = true;
            Notes.Ka = col.gameObject;
        }
    }

    void OnTriggerExit2D(Collider2D col)
```

```
    {
        if (col.gameObject.tag == "Note")
        {
            dongBoundary = false;
            Notes.Dong = col.gameObject;
        }

        if (col.gameObject.tag == "Note2")
        {
            kaBoundary = false;
            Notes.Ka = col.gameObject;
        }
    }
}
```

（四）Note

1. 功能

   a. 使咚變成Rigidbody2D且擁有速度

2. 程式碼

```
using UnityEngine;

public class Note : MonoBehaviour
{
    Rigidbody2D rb;
    public float speed;

    void Awake()
    {
        rb = GetComponent<Rigidbody2D>();
        rb.velocity = new Vector2(-speed,0);
```

```
    }
}
```

（五）Note2

1. 功能

   a. 使咔變成Rigidbody2D且擁有速度

2. 程式碼

```
using UnityEngine;

public class Note2 : MonoBehaviour
{
    Rigidbody2D rb;
    public float speed;

    void Awake()
    {
        rb = GetComponent<Rigidbody2D>();
        rb.velocity = new Vector2(-speed,0);

    }
}
```

（六）recycleNote

1. 功能

   a. 回收未擊中的音符

2. 程式碼

```
using UnityEngine;

public class recycleNote : MonoBehaviour
{
    public NoteObj Notes;

    private bool destroyDong = false;
    private bool destroyKa = false;

    void Update()
    {
        if (destroyDong) { Destroy(Notes.Dong); }

        if (destroyKa) { Destroy(Notes.Ka); }
    }

    void OnTriggerEnter2D(Collider2D col)
    {
        if (col.gameObject.tag == "Note")
        {
            destroyDong = true;
            Notes.Dong = col.gameObject;
        }

        if (col.gameObject.tag == "Note2")
        {
            destroyKa = true;
            Notes.Ka = col.gameObject;
        }
    }
}
```

（七）Score

1. 功能

   a. 顯示分數

2. 程式碼

```
using TMPro;
using UnityEngine;

public class Score : MonoBehaviour
{
    public static int score;
    void Update()
    {
        GetComponent<TMP_Text>().text = score + "";
    }

    public void AddScore(int delta)
    {
        score += delta;
    }
}
```

（八）SoundManager

1. 功能

   a. 管理所有音效

2. 程式碼

```
using UnityEngine;

public class SoundManager : MonoBehaviour
{
    public static SoundManager instance;

    public AudioSource audioSource;
    [SerializeField] private AudioClip dong, ka;
```

```
    private void Awake()
    {
        instance = this;
    }

    public void Dong()
    {
        audioSource.clip = dong;
        audioSource.Play();
    }

    public void Ka()
    {
        audioSource.clip = ka;
        audioSource.Play();
    }
}
```

（九）spawnNote

1. 功能

   a. 生成音符

   b. 遊戲結束後切換場景到計分板

2. 程式碼

```
using System.Collections;
using UnityEngine;
using UnityEngine.SceneManagement;

public class spawnNote : MonoBehaviour
{
    public NoteObj Notes;
```

```
// Song Control
// position track
private float songPosition; // 節拍位置
private float songPosInBeats; // 節拍持續時間
private float secPerBeat;
private float dsptimesong; // 歌曲開始時間
private float beatOfThisNote;

// 判定結束
private bool isGameEnd = false;
private bool isGameStart = true;

private float[] notes = {5f, 6f, 6.5f, 7f, 8f, 9f, 10f, 11f,
                                    11.5f, 12f, 12.5f, 13f, 14f, 14.5f, 15f,
                                    16f, 17.0f, 18f, 19f,
                                20f, 21f, 22f, 23f, 23.5f, 24f, 24.5f,
                                25f, 25.5f, 27f, 28f, 29f, 30f, 30.5f, 31f,
                                32f, 33f, 33.5f, 34f, 35f, 35.5f, 36f, 37f,
                                37.5f, 38f, 39, 39.5f, 40f, 41f, 42f, 43f,
                                44.5f, 45f, 45.5f, 46f, 47f, 47.5f, 48f,
                               49f, 50f, 50.5f, 51f, 51.5f, 53f, 54f, 55f,
                                56f, 56.5f, 57f, 58f, 59f, 59.5f, 60f, 61f,
                                62f, 62.5f, 63f, 64f, 65f, 65.5f, 66f, 67f,
                                67.5f, 68f, 69f, 69.5f, 70f, 72f, 74f, 75f,
                                75.5f, 77f, 79f, 80f, 81f, 81.5f, 82f, 83f,
                                84f, 86f, 88f, 90f, 91f, 92f, 92.5f, 93f,
                                94f, 94.5f, 95f, 97f, 99f, 100f, 101f, 102f,
                                102.5f, 103f, 105f, 107f, 109f, 110f, 111f,
                                113f, 116f, 117f, 117.5f, 118f, 119f, 120f,
                                121f, 122f, 122.5f, 124f, 125f, 126.5f,
                          127f, 129f, 131f, 132.5f, 133f, 135f, 136.5f, 138f,
                                139f, 140f, 141f, 141.5f, 142f, 144f, 148f,
                     149f, 150f, 151f, 152f, 154f, 154.5f, 155f, 157f, 158f,
                                159f, 161f, 164f, 165f, 165.5f, 168f, 170f,
                   171.5f, 172f, 173f, 174f, 174.5f, 176f, 178f, 179f, 180f,
                          181f, 182f, 183f, 184.5f, 186f, 188f, 189f, 191f,
                           193.5f, 194f, 195f, 196f, 197f, 199f, 201f, 202f,
                            204f, 206f, 209f, 210f, 211f, 212f, 213f, 214f,
```

```
                215f, 217f, 219f, 221f, 223f, 223.5f, 225f, 227.5f,
                    229f, 231f, 232f, 233.5f, 234f, 236f, 238f, 240f,
                        241f, 242f, 243f, 244f, 245f, 246f, 247f};

private int[] whichNote = {0, 0, 1, 0, 1, 1, 0, 0, 1,
                           0, 1, 1, 0, 1, 1, 0, 1, 0, 1,
                        0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1,
                        0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1,
                        0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1,
                        1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1,
                        0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1,
                        0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1,
                        1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1,
                        0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1,
                        0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1,
                        0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1,
                        1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1,
                        0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1,
                       0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1};

// song messages
private float bpm = 147f;
private float timeNow;
private int nextIndex;
private int BeatsShownInAdvance = 4;

void Awake()
{
    secPerBeat = 60f / bpm;
    dsptimesong = Time.time;
}

void Update()
{
    // 產生音符
    timeNow += Time.deltaTime;
    songPosition = (float) (timeNow - dsptimesong); // 現在的秒數
    songPosInBeats = songPosition / secPerBeat; // 現在的拍數
```

```
            // Dong 生成
            if (nextIndex < notes.Length && notes[nextIndex] <= songPosInBeats +
BeatsShownInAdvance)
            {
                if (whichNote[nextIndex] == 1) // Dong 生成
                {
                    Instantiate(Notes.Dong.transform.gameObject, new Vector3(140.0f, 10.0f, 4f),
Notes.Dong.transform.rotation);
                }

                if (whichNote[nextIndex] == 0) // Ka 生成
                {
                    Instantiate(Notes.Ka.transform.gameObject, new Vector3(140.0f, 10.0f, 4f),
Notes.Ka.transform.rotation);
                }
                nextIndex++;
            }

            StartCoroutine(GameJudge());

            if (isGameEnd == true && isGameStart == false)
            {
                SceneManager.LoadScene(7);
            }

        }

        IEnumerator GameJudge()
        {
            if (notes.Length <= nextIndex)
            {
                isGameEnd = true;
                yield return new WaitForSecondsRealtime(7f);
                isGameStart = false;
            }
        }
}
```

（十）goodcount、okcount、badcount、combocount、scorecount

（含有五段程式碼）

1. 功能

   a. 顯示良數目、可數目、不可數目、最大combo數目、分數

2. 程式碼

```
using UnityEngine;
using TMPro;

public class goodcount : MonoBehaviour
{
    private int goodNumber;
    private GameObject good;

    // Update is called once per frame
    void Update()
    {
        goodNumber = Activator.countGood;
        GetComponent<TMP_Text>().text = goodNumber + "";
    }
}
```

```
using UnityEngine;
using TMPro;
public class okcount : MonoBehaviour
{
    private int okNumber;
    private GameObject ok;

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

```
            okNumber = Activator.countOk;
            GetComponent<TMP_Text>().text = okNumber + "";
        }
}

using UnityEngine;
using TMPro;
public class badcount : MonoBehaviour
{
    private int badNumber;
    private GameObject bad;

    // Update is called once per frame
    void Update()
    {
        badNumber = Activator.countBad;
        GetComponent<TMP_Text>().text = badNumber + "";
    }
}

using UnityEngine;
using TMPro;
public class combocount : MonoBehaviour
{
    private int comboNumber;
    private GameObject combo;

    // Update is called once per frame
    void Update()
    {
        comboNumber = Combo.maxCombo;
        GetComponent<TMP_Text>().text = comboNumber + "";
    }
}

using UnityEngine;
using TMPro;
```

```
public class scorecount : MonoBehaviour
{
    private int scoreNumber;
    private GameObject score;

    // Update is called once per frame
    void Update()
    {
        scoreNumber = Score.score;
        GetComponent<TMP_Text>().text = scoreNumber + "";
    }
}
```

（十一）SongSelect、Song_Dive（含有兩段程式碼）

1. 功能

   a. 點選歌曲選單時，發出音效

2. 程式碼

```
using UnityEngine;
using UnityEngine.SceneManagement;
public class SongSelect : MonoBehaviour
{
    public void Song()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }
    void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            SoundManager.instance.Dong();
        }
```

```
    }
}

using UnityEngine;
using UnityEngine.SceneManagement;
public class Song_Dive : MonoBehaviour
{
    public void Song()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }
    void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            SoundManager.instance.Dong();
        }
    }
}
```

（十二）LevelSelect

1. 功能

   a. 切換場景

   b. 播放音效

2. 程式碼

```
using UnityEngine;
using UnityEngine.SceneManagement;
public class LevelSelect : MonoBehaviour
{
    public void Level()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
```

```
}
public void Level2()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 2);
}
public void Level4()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 4);
}
public void Level0()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex - 1);
}
public void Level00()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex - 2);
}
public void Level04()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex - 4);
}
public void Level05()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex - 5);
}
public void Level07()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex - 7);
}
public void Menu()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex - 3);
}

public void Exit()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex - 4);
}
```

```
    void Update()
    {
        if (Input.GetMouseButtonDown(0)) {SoundManager.instance.Dong();}
    }
}
```

（十三）Menu

1. 功能

   a. 返回主選單

   b. 播放音效

2. 程式碼

```
using UnityEngine;
using UnityEngine.SceneManagement;
public class Menu : MonoBehaviour
{
    public void PlayGame()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }

    public void UIEnable()
    {
        GameObject.Find("Canvas/MainMenu/UI").SetActive(true);
    }
    void Update()
    {
        if (Input.GetMouseButtonDown(0)) { SoundManager.instance.Dong(); }
    }
}
```

第六章、實作結果與討論

可直接觀看影片了解功能：https://youtu.be/xDNb-kOZd1s

1. 我們的 UI 介面有分成：【圖（6.1）初始頁面】、【圖（6.2）模式選擇】、【圖（6.3、6.4）歌曲選單】、【圖（6.5）難度頁面】、【圖（6.6）遊戲頁面】、【圖（6.7）計分頁面】、【圖（6.8、6.9）關卡選單】

2. 假如在模式選擇中直接選擇一般版，則會開始選擇歌曲；選擇闖關版時，將會直接進入關卡選單的部分，且在遊戲結束後闖過的關卡將以黃色標示。

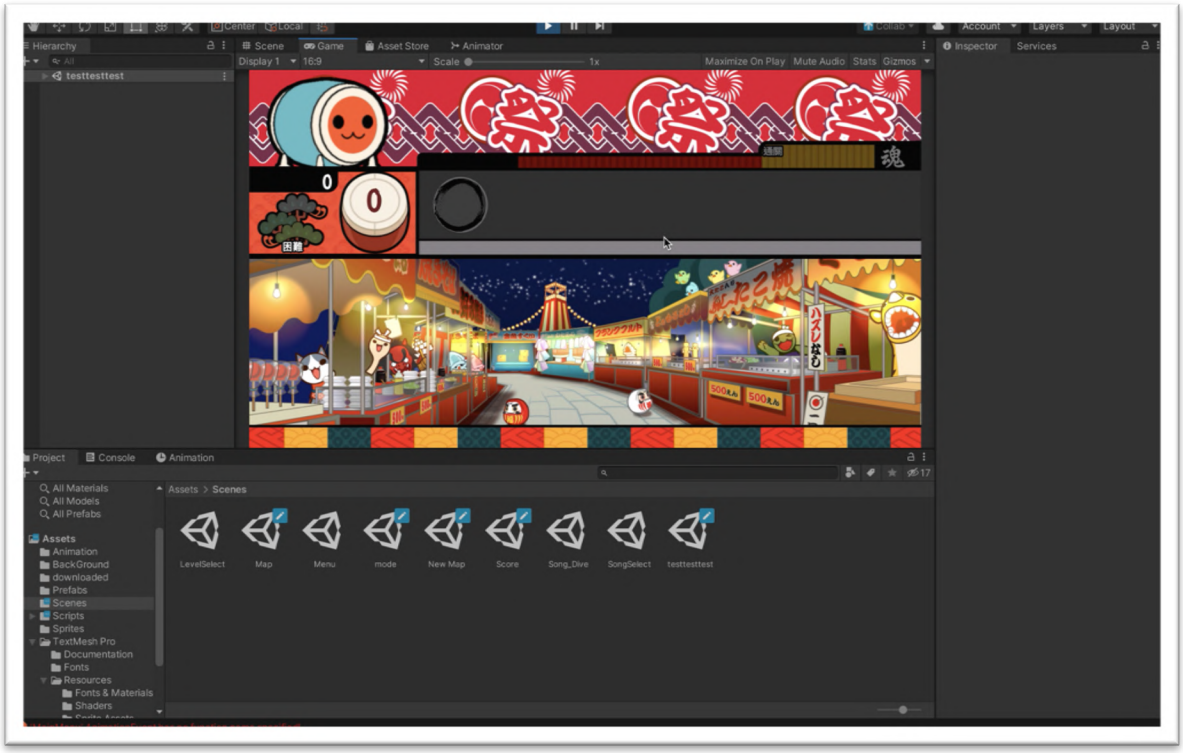
【圖（6.1）初始頁面】

【圖（6.2）模式選單】

【圖（6.3）歌曲選單】

【圖（6.4）歌曲選單】

【圖（6.5）難度頁面】



【圖（6.6）遊戲頁面】

【圖（6.7）計分頁面】

【圖（6.8）關卡選單】

【圖（6.9）關卡選單】

3. 在進入遊戲介面後，我們將音符設置成紅色的咚與藍色的咔，在敲擊時分別會顯示良、可、不可的字樣，並累計 combo 以及算分顯示在一旁的計分欄位上。

4. 其中擊中良（200 分）、可（100 分）、不可（0 分），且只有在良、可的時候會持續累加 combo 數，若擊出不可時，則會將 combo 數歸零。

5. 音符的生成及消失可以從圖片左方的物件欄觀察到。

【圖（6.10）良 顯示】

【圖（6.11）可 顯示】

【圖（6.12）不可 顯示】

第七章、結論與優化

（一）結論

經由一學期的實作，完成了一系列太鼓達人的基礎功能以及加上自己的創意設計。基礎功能包括咚咔功能（分數、Combo、字符顯示）、音效、選單、簡易動畫、譜面製作、模式選擇，創意設計為闖關模式。

（二）優化

即使已經完成基本遊戲，因為時間不足的關係，還有許多功能可以優化以及增加。例如歌曲的數目可以增加、加上更多的新功能（原本計劃的 RPG、交友功能等）。在主遊戲的部分，判斷音符的擊打如果在音符十分密集的時候會出現小問題，如果將來要設計更難的譜面時，需要修正這部分的問題。

參考文獻

素材參考區

[1] 太鼓背景、音效 https://taiko.bui.pm

[2] 咚圖片 https://www.hiclipart.com/free-transparent-background-png-clipart-nxiyh

[3] 咔圖片 https://www.hiclipart.com/free-transparent-background-png-clipart-nxiyh

（用photoshop把咚的顏色改掉）

[4] 碰撞器圖片

https://www.vippng.com/preview/Thwbx_png-file-size-transparent-circle-border-png/

[5] 捲軸圖片 http://699pic.com/tupian-401015685.html

[6] 歌曲——Dive to world（利用Audacity剪輯片段）

https://www.youtube.com/watch?v=HYB8x17Ijl4

程式內容參考區

[1] 了解如何判定優、良、不可 http://burningxempires.blogspot.com/2016/09/

[2] Unity 自學影片

https://www.youtube.com/playlist?list=PL_Pb2I110MfGAsoqtDs8-6kEU55wU8CnE

[3] 節奏生成 https://www.zhihu.com/question/26133992

[4] 音符同步 https://gameinstitute.qq.com/community/detail/118731

學習心得

我們從最開始只有簡單的概念發想，到一步步讓遊戲功能陸續完備——包括基本遊戲的咚咔功能、音效、選單、簡易動畫、譜面製作、模式選擇，過程中真的遇到很多不同的問題，也有很多次其實根本不知道該如何下手，只能不停在網路上查找影片資源或是文本資料，再自己慢慢去試能不能成功解決問題，其實有時候也算是蠻挫折的，但當最後能成功新增功能或 debug 的時候卻會很有成就感，因此一路以來也算是很開心地開發我們太鼓達人的遊戲。

我們的期末報告讓太鼓達人增加了新的闖關模式，也是以往太鼓達人所沒有的部分，都讓我們覺得真的有部分實現了一款與眾不同的太鼓達人，能慢慢自己一步一步開發出現在的這個樣子，實在是覺得十分開心且欣慰，雖然我們期初希望能增加的公會、交友功能仍沒有實現，超出了目前我們的能力範疇，但仍希望未來能夠繼續發展到更多功能！

最後，我們希望往後能繼續去開發更多遊戲，也能繼續開發太鼓達人使其成為一款具有劇情主線的 RPG 節奏遊戲，這樣便與原先單純的音樂遊戲有所不同，我們也認為這樣的遊戲性會更高，也會有更好的遊戲體驗。除了在既有的遊戲上進行擴展，我們也希望自己能夠去獨立開發出其他遊戲，也在這門課中發現開發遊戲真的是一件很好玩的事，所以也很開心這學期有機會可以自己寫出一款太鼓達人，也希望可以持續在這條路上繼續努力！