輔仁大學

軟體工程與數位設計學程

虛擬/擴增實境(VR/AR)實作入門

遊戲製作技術文件

蟑螂捕手

學 生：軟創三 李佩穎 504170303

資管四 蘇亭云 403401263

景觀四 賴鈺儒 403450185

目錄

[概述 2](#_Toc517379814)

[創意源起 2](#_Toc517379815)

[目標對象 2](#_Toc517379816)

[遊戲介紹 2](#_Toc517379817)

[名稱 2](#_Toc517379818)

[類型 2](#_Toc517379819)

[平台 2](#_Toc517379820)

[風格 3](#_Toc517379821)

[遊戲內容與機制 3](#_Toc517379822)

[基本架構 4](#_Toc517379823)

[ 大家玩(四人模式) 4](#_Toc517379824)

[ 繼續玩 4](#_Toc517379825)

[遊戲機制 4](#_Toc517379826)

[ 蟑螂移動 4](#_Toc517379827)

[ 骰骰子 4](#_Toc517379828)

[ 旋轉餐具模組 4](#_Toc517379829)

[ 玩家區域 5](#_Toc517379830)

[遊戲製作詳細說明 5](#_Toc517379831)

[ 蟑螂 5](#_Toc517379832)

[ 骰子 11](#_Toc517379833)

[ 刀叉旋轉 11](#_Toc517379834)

[ 輪到哪位玩家 12](#_Toc517379835)

[(1) 存活玩家4位： 13](#_Toc517379836)

[(2) 存活玩家3位： 13](#_Toc517379837)

[(3) 存活玩家2位： 13](#_Toc517379838)

**概述**

此章節介紹團隊創意的起源，以及遊戲的目標對象。

**創意源起**

起初是我們每個成員都有提出想做的桌遊，經由討論其中最有能力製作而成，便訂定此桌遊改為數位化並改良。

**目標對象**

提供給想要打發時間的玩家適合全年齡遊玩。

**遊戲介紹**

此章節介紹本組遊戲的名稱、類型、風格，以及遊戲平台。

**名稱**

本遊戲為市面上所發售的「蟑螂捕手」同名同款製作數位化。

**類型**

本遊戲屬於休閒遊戲，能夠讓人在工作或課業的閒暇之餘，單純的以休閒為目的來進行遊戲。

**平台**

使用Unity製作，只適用於PC上。

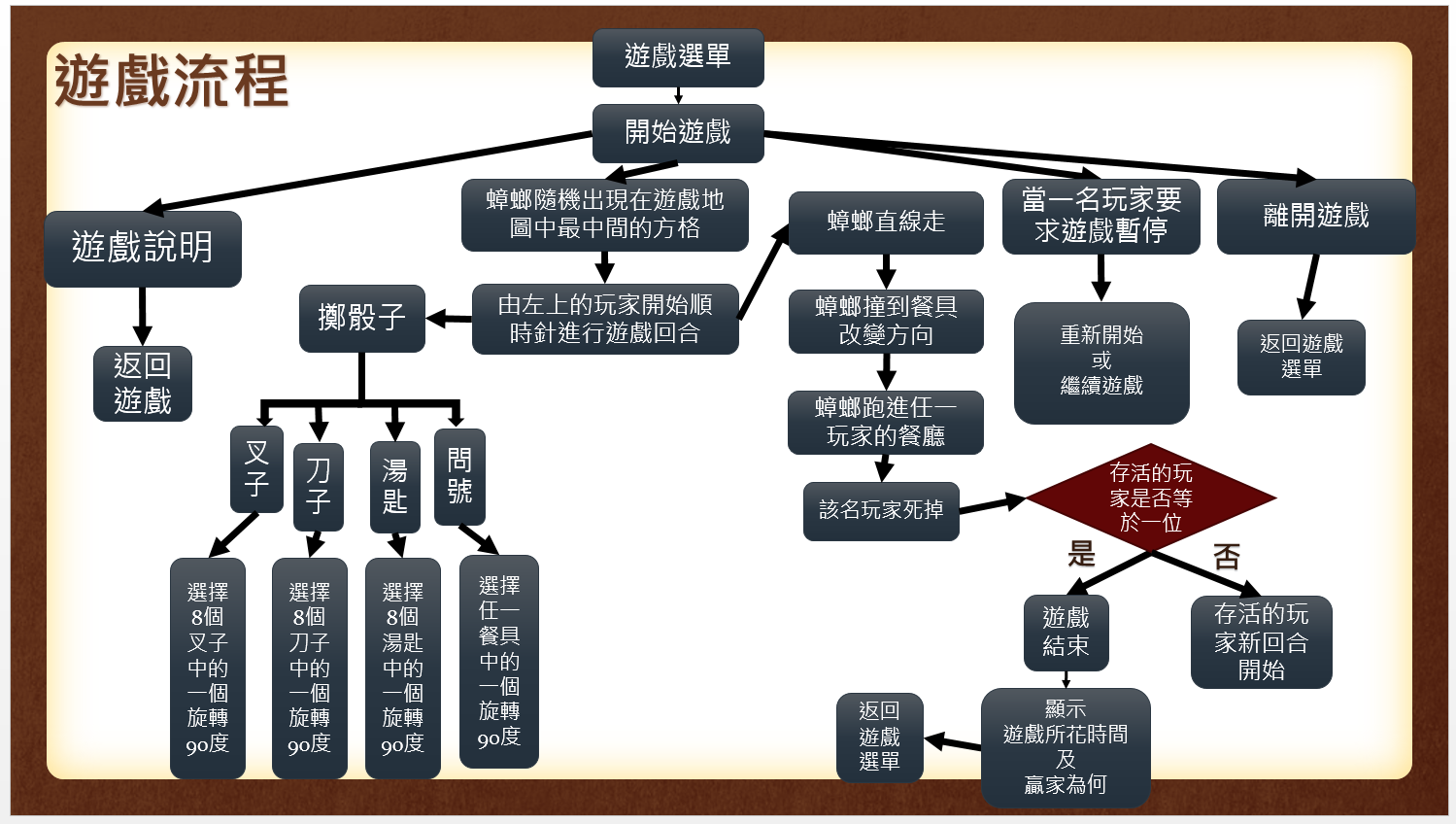
**風格**

本遊戲主要目的是希望玩家能夠輕鬆進行遊戲，所以採用鮮明光亮的遊戲畫面，並使用3D模型製作，使遊戲整體附有動感的效果。

**遊戲內容與機制**

此章節將會介紹本組的遊戲流程、遊戲模式、遊戲機制與遊戲規則

**遊戲流程**



**基本架構**

* 大家玩(四人模式)

只有支援四人遊玩。

* 繼續玩

把遊戲時間做暫停，按繼續按鈕可依上次暫停時間點繼續遊玩。

* 離開遊戲

遊戲進行時可以直接離開。

**遊戲機制**

此節介紹遊戲玩法與遊戲規則。

本遊戲建立在所有玩家處於相同的限制下，在此局內必須設法存活不讓蟑螂進入玩家區域，成為最後贏家。

* 蟑螂移動

蟑螂在開始遊戲時會一直持續移動，直到碰到玩家區域回到原始位置，碰到剩下最後一名玩家結束遊戲。

* 骰骰子

共有四種圖式，三種餐具與一種隨機問號圖示，當出現圖示會顯示黃光限制玩家只可旋轉此餐具模組。

* 旋轉餐具模組

當出現黃光時滑鼠移到上方出現紅光即可旋轉。

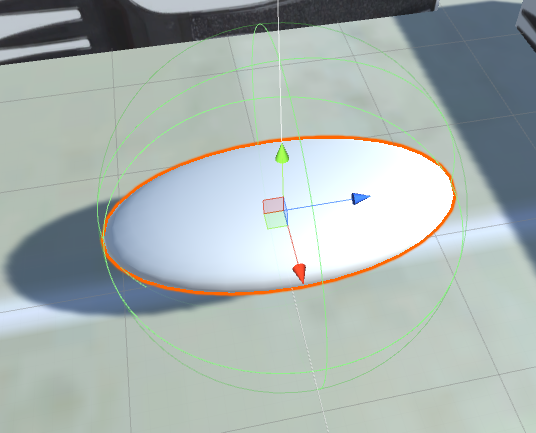
* 玩家區域

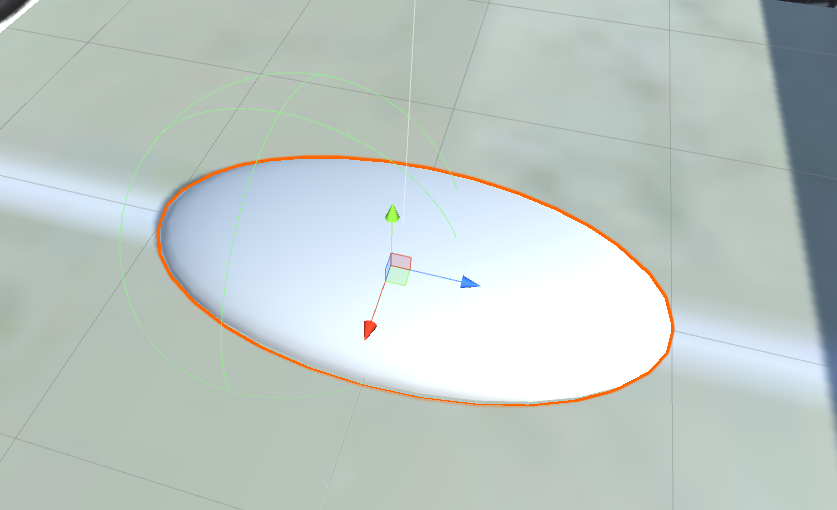
當輪到此玩家時會顯示青綠光，代表此時為此玩家回合，蟑螂跑到玩家區域時為此玩家(此時也可以不是此玩家回合)失敗。

**遊戲製作詳細說明**

此章節將會介紹本組的遊戲製作方式與所碰到的困難處如何加以突破更改。

* 蟑螂





**變更**

一開始我們這組最明顯的問題是當碰觸到餐具模組時會卡進去模組裡面就出不來了，原因為當碰到餐具模組時會執行直走隨機旋轉的程式碼，導致一直前進並卡進去餐具模組裡，一開始此蟑螂模組使用的是Capsule Collider 就會持續碰撞到餐具模組的Box Collider 導致一直前進旋轉無法出來，之後把蟑螂模組的Capsule Collider改為Sphere Collider 並改制前端使蟑螂模組尾端不會有大面積碰撞，就可以降低卡住的機率當卡住時也會自行脫困。

首先先設定蟑螂的速度與移動方向再次設定Z軸為畫面往下走的方向transform.Translate(0, 0, -5 \* Time.deltaTime);並記錄蟑螂上次位置會自動回上次位置，避免卡在”Cube”裡面

public Vector3 prePos;

void Start () {

prePos = this.transform.position;

}

void Update()

{

prePos = this.transform.position;

transform.Translate(0, 0, -5 \*

Time.deltaTime);

}

再來是設定當蟑螂碰到”Cube”和”Wall”時要隨機改變方向(隨機左180度和隨機右180度)

void OnTriggerStay(Collider other)

{

if (other.name == "Cube" || other.transform.tag == "wall")

{

this.transform.position = prePos;

radius = Random.Range(-180f, 180f);

gameObject.transform.Rotate(new

Vector3(0, radius, 0));

}

}

當蟑螂碰到”Player”區域內時，會出現”CloseDoor”物件public GameObject closeDoor;擋住玩家區域，當蟑螂碰到玩家時玩家會消失玩家數-1，然後蟑螂回到啟始位置Vector3 (0, 0, 0);

void OnTriggerEnter(Collider other) {

if (other.name == "Player1")

{

Destroy (other.gameObject);

Player.live\_player1 = false;

Player.live\_Num--;

gameObject.transform.position = new

Vector3 (0, 0, 0);

Instantiate(closeDoor, new Vector3

(-13.96f, 0f, 11.72f),

closeDoor.transform.rotation);

}

if (other.name == "Player2")

{

Destroy (other.gameObject);

Player.live\_player2 = false;

Player.live\_Num--;

gameObject.transform.position = new

Vector3 (0, 0, 0);

Instantiate(closeDoor, new Vector3 (-

13.96f, 0f, -11.64f),

closeDoor.transform.rotation);

}

if (other.name == "Player3")

{

Destroy (other.gameObject);

Player.live\_player3 = false;

Player.live\_Num--;

gameObject.transform.position = new

Vector3 (0, 0, 0);

Instantiate(closeDoor, new Vector3

(14.04f, 0f, -11.64f),

closeDoor.transform.rotation);

}

if (other.name == "Player4")

{

Destroy (other.gameObject);

Player.live\_player4 = false;

Player.live\_Num--;

gameObject.transform.position = new

Vector3 (0, 0, 0);

Instantiate(closeDoor, new Vector3

(14.04f, 0f, 11.72f),

closeDoor.transform.rotation);

}

}

* 骰子

在Hierarchy新增一個empty object並給這個empty object一個Image，當按下Throw Dice按鈕，產生1-7的隨機數random\_dic = (int)(Random.Range (1, 8));，當隨機數為1及2時顯示叉子；當隨機數為3及4時顯示湯匙；當隨機數為5及6時顯示刀子；當隨機數為7時顯示問號；

Spr\_DiceIcon = Resources.LoadAll<Sprite>("dice\_icon");來抓取圖片。

* 刀叉旋轉

用變數Tableware\_Name 紀錄骰到餐具的名稱來判斷玩家可以選擇旋轉的物件( 以物件的tag與Tableware\_Name做比對來判斷 )，利用FindGameObjectsWithTag 將骰到餐具的物件的布林CanTurned 設為true，並把CanTurned 為true的餐具物件顏色設為黃色，來方便玩家看出哪些物件是可以轉的，並利用布林變數MouseCtrl.canRotate及RaycastHit;來判斷玩家是否做出選擇，MouseCtrl.canRotate = false 代表玩家已選擇旋轉的物件，不能再次旋轉，全部餐具物件的布林CanTurned 設為false( 餐具物件顏色設為白色 )；而RaycastHit 透過滑鼠得點選知道哪個餐具物件被選擇並將其選轉90度，且當滑鼠移到可轉物件Hitted為true，餐具物件顏色設為紅色，Hitted為true則餐具物件恢復為黃色。

void whichTablewareCanTurned(string tablewareName, bool canTurned)

{

Tableware\_Name = tablewareName;

tableware = GameObject.FindGameObjectsWithTag (tablewareName);

for (int i = 0; i < tableware.Length; i++) {

tableware [i].transform.GetComponent<tablewareScript> ().CanTurned = canTurned;

}

}

* 輪到哪位玩家

遊戲一開始統一由左上玩家開始回合，故將左上放置Hierarchy的物件Player下的子物件Player1(Cube)，以逆時鐘方向依序放置左下物件Player2(Cube)；右下物件Player3(Cube)，右上物件Player4(Cube)。

每當按下Throw Dice按鈕，會將Player ( empty object )下存在的玩家( Cube )( 玩家死亡Cube消失 )的.material.color 用迴圈設為白色，再利用Round\_str % live\_Num【 ( 回合數 / 存活人數 ) 的餘數 】來判斷輪到哪位玩家，將material.color設為青綠色。

(1) 存活玩家4位：

餘數為0為Player 下第一個子物件；餘數為1為Player下第-二個子物件；餘數2為Player 下第三個子物件；餘數為3為Player下第四個子物件。

(2) 存活玩家3位：

餘數為0為Player 下第一個子物件；餘數為1為Player下第-二個子物件；餘數2為Player 下第三個子物件。

(3) 存活玩家2位：

餘數為0為Player 下第一個子物件；餘數為1為Player下第-二個子物件。

for (int i = 0; i < this.transform.childCount; i++)

{

this.transform.GetChild(i).GetComponent<MeshRenderer>().material.color = Color.white;

}

this.transform.GetChild(Round\_str%live\_Num).GetComponent<MeshRenderer>().material.color = Color.cyan;

玩家生死：

利用bool變數( live\_player1、live\_player2、live\_player3、live\_player4 )紀錄玩家生死，當蟑螂Sphere與玩家 Player( Cube )，產生碰撞，即玩家死亡，被撞到的玩家bool變數變為false，變數 live\_Num 存活人數 -1。