**卡牌遊戲(待命名)**

**軟體設計文件**

SRS-IM-2023-004-1-1

朝陽科技大學資訊管理系

指導老師：吳信成 教授

學　　生：董柏均、周子皓、黃沐卿、徐彬詠、劉俊彬

中 華 民 國 113 年 12 月 8 日

**目錄**

[第一章 簡介 4](#_Toc175239251)

[1.1 文件目標 4](#_Toc175239252)

[1.2 文件範圍 4](#_Toc175239253)

[1.3 定義、縮寫和首字母縮寫 4](#_Toc175239254)

[第二章 軟體架構 5](#_Toc175239255)

[2.1 Unreal Engine GamePlay 框架核心類別 5](#_Toc175239256)

[2.1.1 UObject類別 5](#_Toc175239257)

[2.1.2 AActor類別 6](#_Toc175239258)

[2.1.3 ALevelScriptActor類別 6](#_Toc175239259)

[2.1.4 AController類別 7](#_Toc175239260)

[2.1.5 APawn類別 7](#_Toc175239261)

[2.1.6 AHUD類別 8](#_Toc175239262)

[2.1.7 AInfo類別 8](#_Toc175239263)

[2.1.8 UActorComponent類別 9](#_Toc175239264)

[2.1.9 ULevel類別 9](#_Toc175239265)

[2.1.10 UWorld類別與 FWorldContext 10](#_Toc175239266)

[2.1.11 UGameInstance類別 11](#_Toc175239267)

[2.1.12 UEngine類別 11](#_Toc175239268)

[2.2 Unreal Engine 多人連線架構 12](#_Toc175239269)

[2.2.1玩家託管伺服器與專用伺服器 12](#_Toc175239270)

[2.2.2伺服端與用戶端溝通架構 12](#_Toc175239271)

[2.2.3網路角色 13](#_Toc175239272)

[2.2.4 多人遊戲中GamePlay 框架各類別的用途 14](#_Toc175239273)

[2.3 遊戲架構的具體設計 15](#_Toc175239274)

[第三章 類別細項具體設計 16](#_Toc175239275)

[第四章 專案結構 17](#_Toc175239276)

[第五章 角色設計 18](#_Toc175239277)

[第六章 介面與遊戲形象設計 19](#_Toc175239278)

# 第一章 簡介

## 1.1 文件目標

本文件是根據「VR卡牌遊戲」之軟體需求規格書中所描述的需求，進行遊戲架構、模組、元件、介面、角色等內容的具體設計，本文件的目標是詳細描述「VR卡牌遊戲」的具體設計方案，確保開發團隊能夠正確的根據設計方案，將遊戲完整的建構而成。

## 1.2 文件範圍

本文件涵蓋「VR卡牌遊戲」中所有內容的具體設計方案，在專案上包括遊戲命名、專案結構、資產命名規則、Logo設計，在介面上包括樣式、配色、圓角與使用者互動體驗，在邏輯上包括遊戲架構、行為邏輯樹、模組與組件等內容，角色上包括外觀設計、武器設計、骨架規劃、動作配置以及視覺特效。

## 1.3 定義、縮寫和首字母縮寫

表1.3 1-1-1詞彙定義與說明

|  |  |
| --- | --- |
| **定義** | **說明** |
| 關卡 | 又稱Level，這是遊戲中場景的稱呼方式。 |
| Actor | 遊戲場景中的物件。 |
| RPC | 遠端程序呼叫，它是伺服端與用戶端之間互相溝通所使用的技術 |
| Role | 指的是Actor在裝置上的網路身份 |
| UMG | 它是Unreal Motion Graphics的縮寫，是Unreal 中的使用者介面系統，裡面有大量的的介面部件，例如按鈕、輸入框與文字框等部件，這些部件被稱為 Widget。 |

# 第二章 軟體架構

本遊戲軟體將透過 Epic Games 公司旗下的遊戲引擎 Unreal Engine 以及其提供的視覺化界面工具 Unreal Engine Editor 來進行主要的遊戲開發工作，並且會使用該遊戲引擎中的 GamePlay 框架，作為本遊戲的遊戲基礎架構。

## 2.1 Unreal Engine GamePlay 框架核心類別

遊戲以Unreal Engine 中的 GamePlay 框架作為本遊戲的基礎核心架構，並在此基礎上進行了多層次的擴展和自定義，以滿足遊戲的需求，以下進行這些核心類別的描述與介紹。

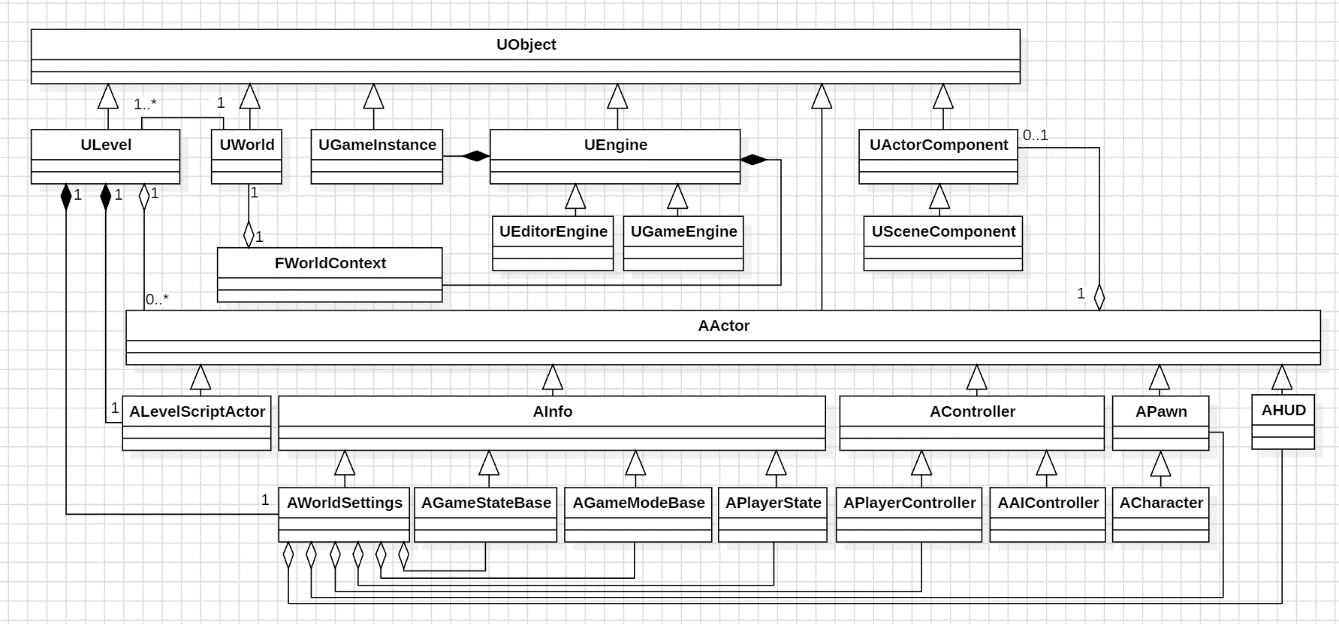


圖2.1 1-1-1GamePlay 框架的類別圖

### 2.1.1 UObject類別

這是 Unreal Engine 中最基礎的類別，幾乎所有的的類別都是繼承於它，它有著最基礎的物件功能，例如: 垃圾回收、引用更新、序列化等底層的基礎功 能。

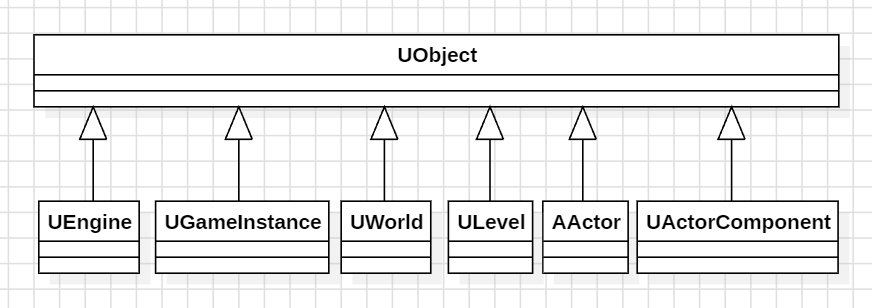


圖2.1 1-1-2 UObject 類別關係圖

### 2.1.2 AActor類別

AActor是專門作用於場景，而從 UObject 派生出來的類別，場景中的萬物皆是由它扮演而成，不管是有形的地板、石頭、動物或太陽，還是無形的規則與狀態全都是由這些演員(Actor) 扮演而成，它們各司其職運作著整個遊戲世界。

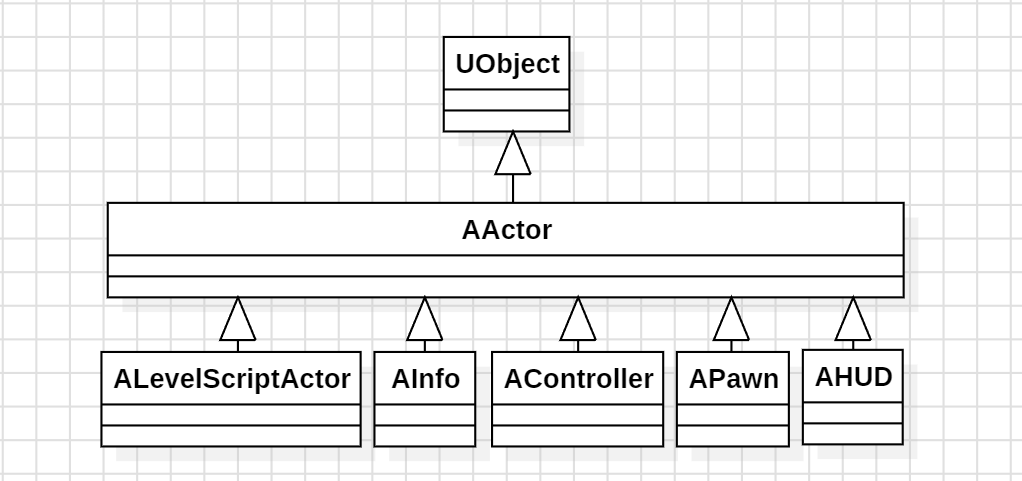


圖2.1 1-1-3 AActor 類別關係圖

### 2.1.3 ALevelScriptActor類別

它是專門向關卡提的供腳本類別，每個關卡都會有一個ALevelScriptActor，可以讓開發者用來處理該關卡中特定的邏輯和事件，而不會影響到其他的關卡。當關卡被創建或刪除時，對應的ALevelScriptActor 也會隨著生成與銷毀。

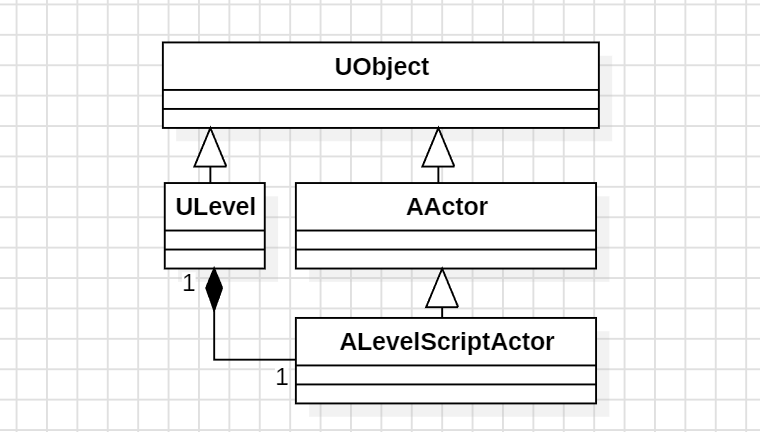


圖2.1 1-1-4 ALevelScriptActor 類別關係圖

### 2.1.4 AController類別

它是專門用來控制遊戲角色的基礎類別，在它之下分化出了兩個控制器類別，分別是處理玩家輸入與互動來操控遊戲角色的 APlayerController(玩家控制器)，以及由遊戲系統自動處理決策邏輯來控制遊戲角色的 AAIController(AI控制器)。

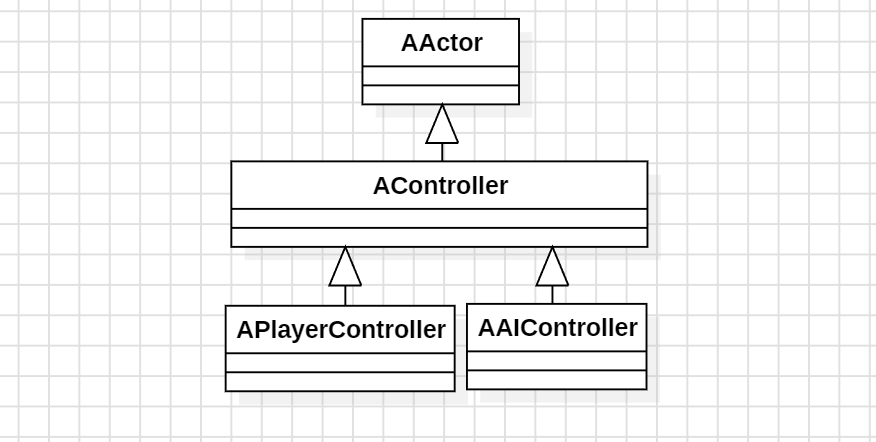


圖2.1 1-1-5 AController類別關係圖

### 2.1.5 APawn類別

它是遊戲中作為遊戲角色，專門用來被操控的類別，可以透過控制器被操控，相當於控制器為靈魂，而 APawn 作為身體的關係，而 APawn 底下還有派生出專門為人型角色而生的 ACharacter 類別，裡面有大量針對人形角色的特殊功能與邏輯。

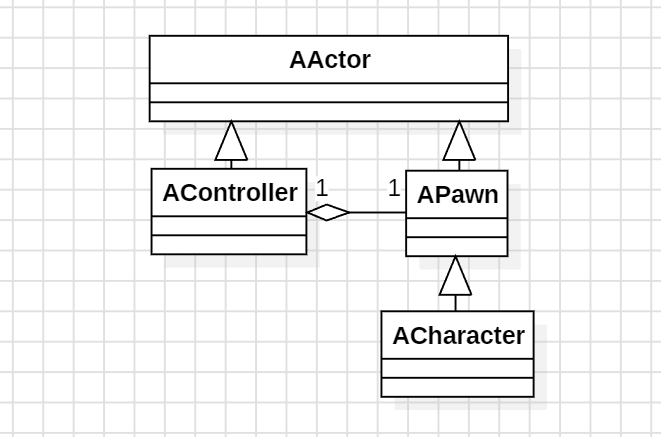


圖2.1 1-1-6 APawn類別關係圖

### 2.1.6 AHUD類別

它是專門用來管理和繪製遊戲介面的類別，通常用來以此呈現各種遊戲資訊，它繼承於 AActor，並且底下沒有派生任何子類別。

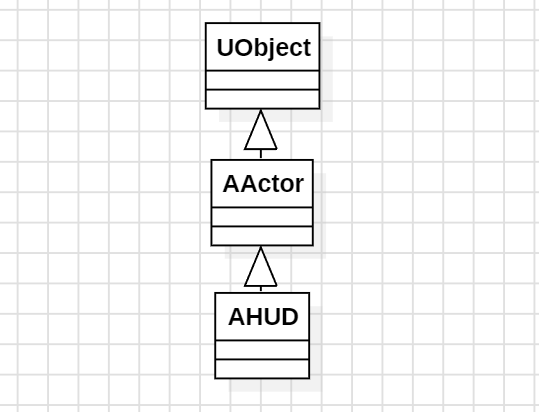


圖2.1 1-1-7 AHUD 類別關係圖

### 2.1.7 AInfo類別

它是專門用來管理和存儲關卡規則與狀態的基礎類別，在它底下派生出了用於管理遊戲規則的 AGameModeBase 類別，和儲存與同步遊戲狀態資訊的 AGameStateBase 類別，還有管理玩家資訊與狀態的 APlayerState 類別，以及用於管理與配置遊戲世界全域設定的 AWorldSettings 類別，雖然 AWorldSettings 是設定在關卡上的，但卻不叫做 ALevelSetting，這是因為關卡的AWorldSettings 會作用於整個遊戲世界上，而不是只限於該關卡當中。

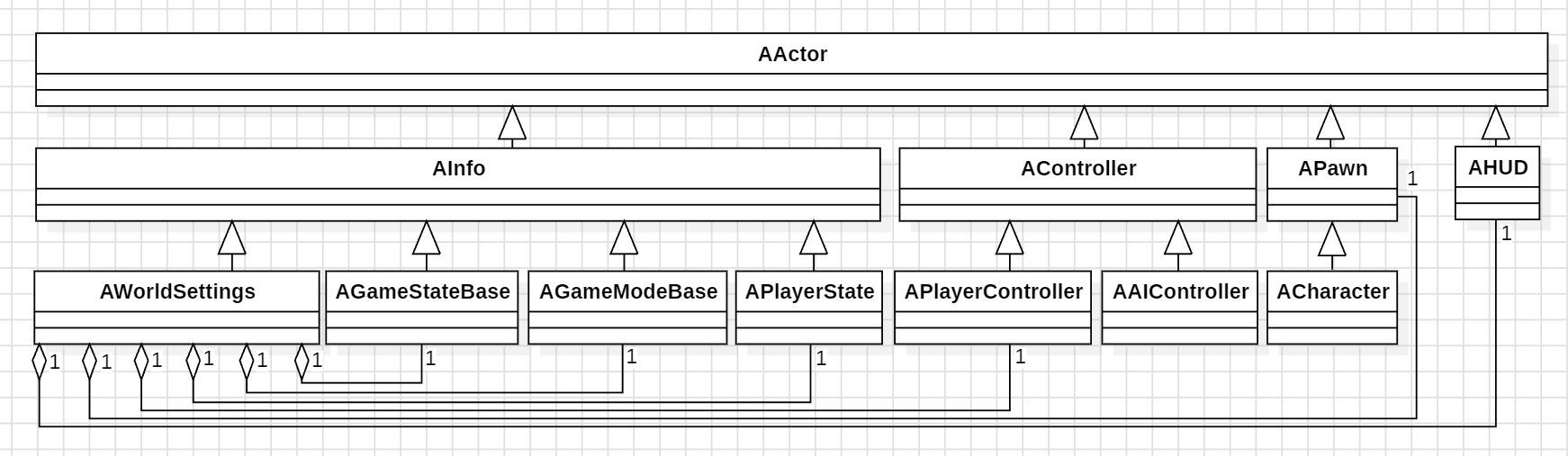


圖2.1 1-1-8 AInfo 類別關係圖

### 2.1.8 UActorComponent類別

Unreal Engine 對於 AActor 的功能並不是完全透過繼承的方式，因為這樣會在繼承過程中累積太多無用的屬性與方法，因此他們採用組件的方式，將各種功能寫在組件上，再將組件添加到需要使用的 Actor 上，而這些組件的基礎類別就是從 UObject 繼承下來的 UActorComponent 類別，它底下派生出了無數的組件，其中最重要的就是 USceneComponent 類別，它賦予了 Actor 能夠放置在場景中的能力，讓 Actor 可以平移、旋轉與縮放。

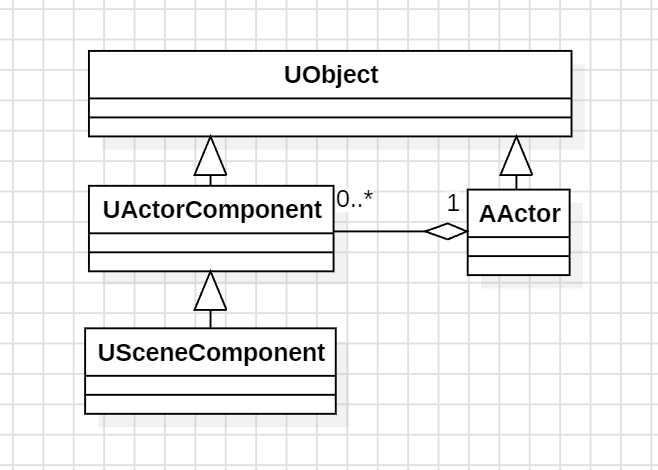


圖2.1 1-1-9 UActorComponent 類別關係圖

### 2.1.9 ULevel類別

它是專門用來放置AActor 的容器，正是所謂的遊戲場景，通常被稱為關卡，關卡被創建時，會連帶生成對應的 ALevelScriptActor，用於編寫該關卡的專屬邏輯，同時關卡也會被分配到一組 AWorldSettings 設定，用於對整個遊戲世界進行全域設定。

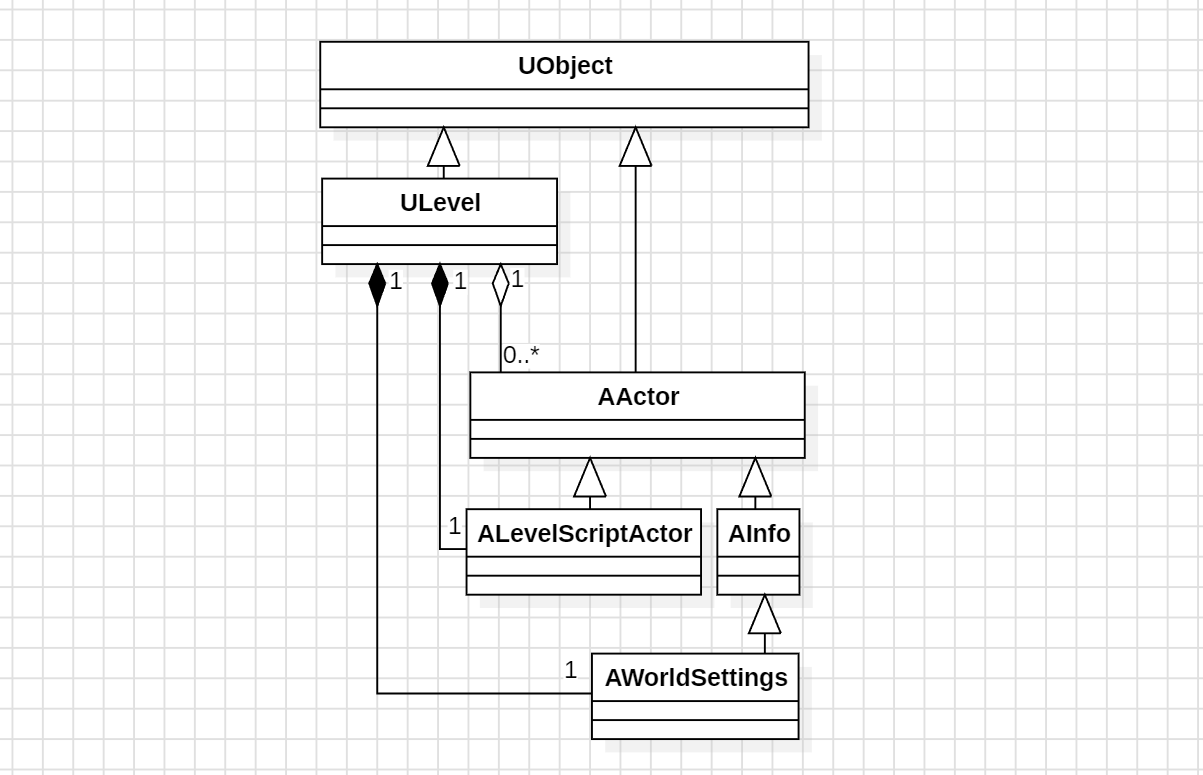


圖2.1 1-1-10 ULevel 類別關係圖

### 2.1.10 UWorld類別與 FWorldContext

在關卡之上還有個容器叫做UWorld ，它是可視化場景的最高級物件，它可以容納複數個場景，但其中必須有一個作為常駐關卡(Persistent Level)，而被選作為常駐關卡的 ULevel 身上所套用的 AWorldSettings 就會作用於整個遊戲世界，所以有多個關卡時，世界會以常駐關卡的世界設定為主。另外它也擁有一組動態載入關卡(Stream Level)，可以依照需求載入多個場景，而 FWorldContext 專門儲存與管理UWorld ，每個 UWorldContext 都對應了一個 UWorld，並且 UEngine 可以使用 UWorldContext 來管理 UWorld，例如 : 切換遊戲場景或同時管理多個遊戲世界。

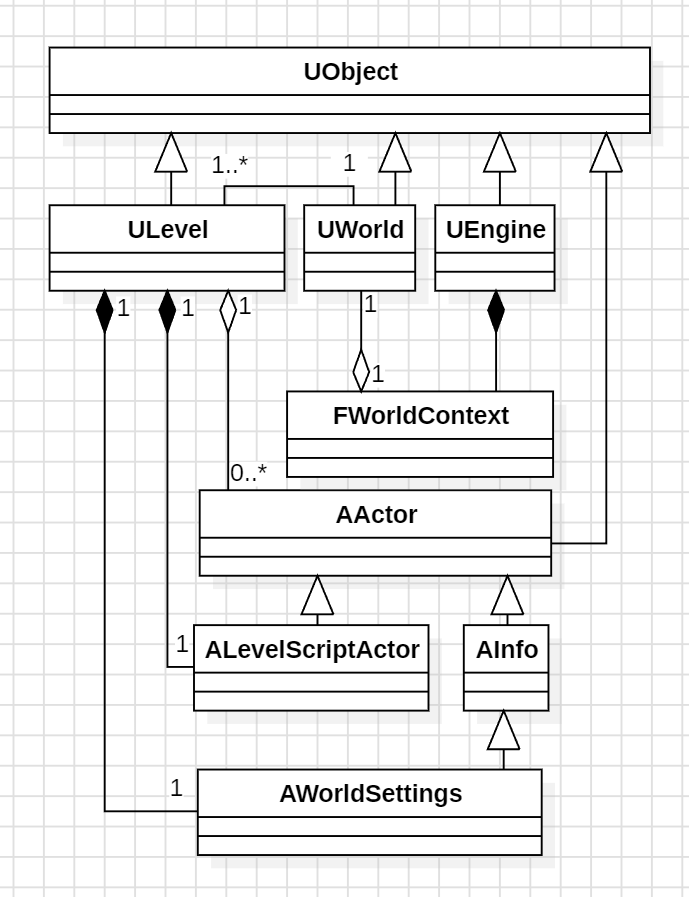


圖2.1 1-1-11 UWorld 類別關係圖

### 2.1.11 UGameInstance類別

它負責存儲和管理跨世界的遊戲狀態和資料，通常用來處理獨立於遊戲世界的功能，但並不直接管理遊戲世界，而是更關注於全域層級的遊戲邏輯。其生命週期由UEngine控制，當UEngine被銷毀時，UGameInstance也會隨之銷毀。

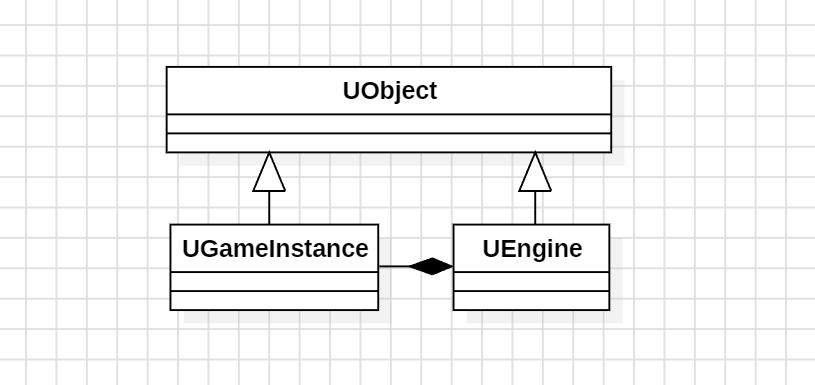


圖2.1 1-1-12 UGameInstance 類別關係圖

### 2.1.12 UEngine類別

它是繼承於 UObject 的引擎基礎類別，並且分化出了編輯器專用的 UEditorEngine 類別，以及運行遊戲專用的 UGameEngine 類別，UEngine與UGameInstance互相配合，UEngine作為整個遊戲引擎的根本管理者，負責管理和調度引擎中的各個子系統，並控制遊戲的整體運行，而UGameInstance做為整個遊戲的邏輯根源，負責管理遊戲的狀態和流程，這種「一個指揮、一個做事」的關係構成了整個遊戲運行的根本架構。

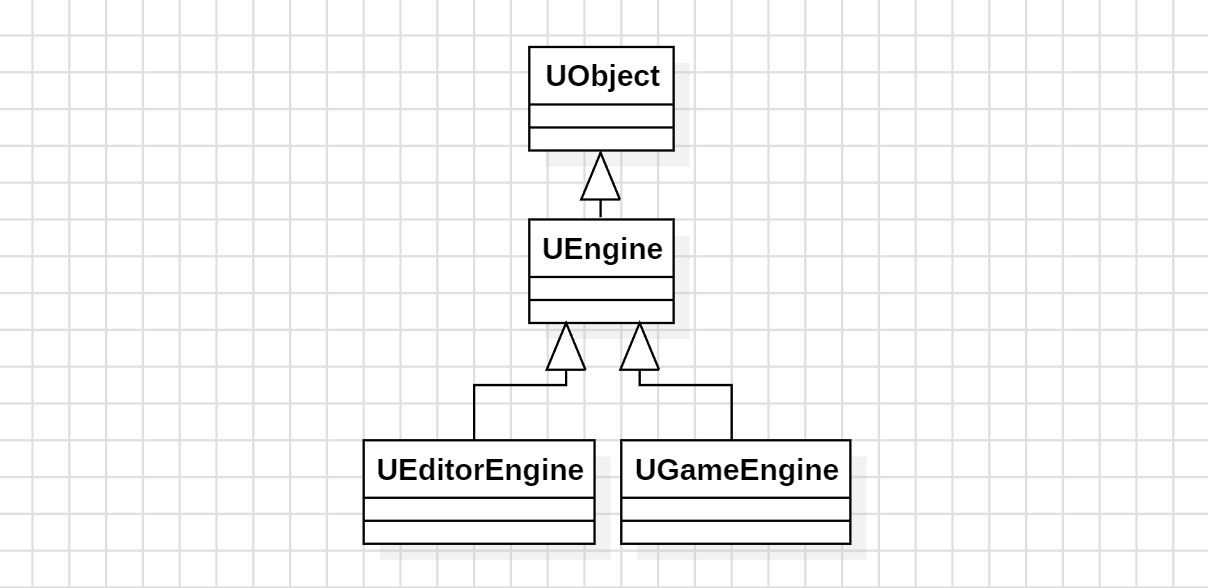


圖2.1 1-1-13 UEngine類別關係圖

## 2.2 Unreal Engine 多人連線架構

多人進行遊戲時，必須將某一方的遊戲執行結果作為基準，並將該結果傳遞給其他玩家，如此一來才能確保遊戲內容的一致性，而此者就是作為遊戲「伺服端」的一方。因此伺服端與用戶端之間的邏輯並不相同，所以開發多人遊戲需要非常注意資料同步與溝通，以下將會詳細描述伺服端與用戶端的關係、架構以及運作模式。

### 2.2.1玩家託管伺服器與專用伺服器

伺服端有兩個種類，其中一種是玩家以自己的用戶端開放其他玩家進行連線，該玩家會成為遊戲的伺服器端，同時也是參與其中的玩家，這一種被稱為「玩家託管伺服器」，另一種則是由第三方作為伺服器，專門用來託管遊戲，不參與遊戲的實際遊玩，這一種被稱為「專用伺服器」，本專案將會兩者皆進行實作，提供給玩家更多種連線選擇。

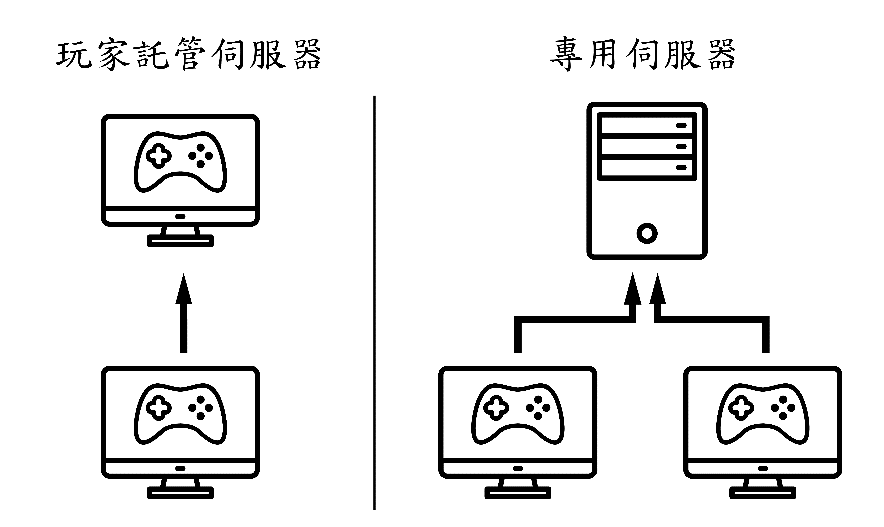


圖2.1 1-2-1 玩家託管伺服器與專用伺服器架構圖

### 2.2.2伺服端與用戶端溝通架構

伺服端中運算的遊戲結果是具備權威性的，例如兩個玩家同時間爭搶同一個道具，究竟是哪個玩家動作更快，是完全交給伺服端去裁決的，而用戶端只需要接收伺服端傳來的結果。整體的溝通方式就是用戶端負責發送遊戲行為的請求給伺服端，以及接收伺服端傳來的資訊並將它同步給玩家，而伺服端則是負責接收並處理用戶端送來的遊戲行為請求，以及將需要同步的遊戲資訊傳送給用戶端。

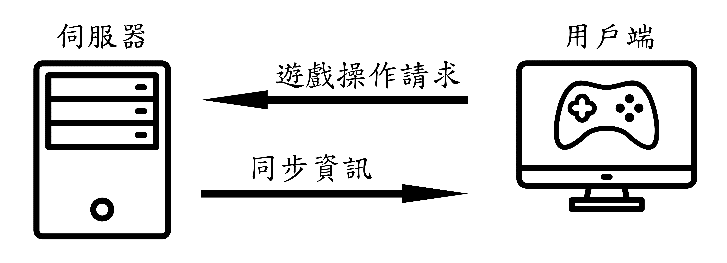


圖2.1 1-2-2 伺服端與用戶端的溝通關係

### 2.2.3網路角色

在多人遊戲中，Actor 具備一個叫做 Role(網路角色) 的概念，可以在網路同步時，描述該 Actor 在不同機器(例如伺服器、玩家電腦)時的網路狀態，如此一來就可以藉此進行不同的同步邏輯處理。Unreal Engine 使用了四種不同的網路角色來表示物件在網路中的狀態。

表2.2 1-2-1網路角色說明

|  |  |
| --- | --- |
| 狀態 | 說明 |
| None | 不參與網路同步 |
| SimulatedProxy | 擁有該狀態表示它是用戶端電腦中的Actor，並且其行為完全由伺服器控制，用戶端只能對它進行模擬，可以說它就只是個不能操控的複製品。 |
| AutonomousProxy | 擁有該狀態表示它是用戶端電腦中的Actor，並且玩家在自己的用戶端上對該 Actor擁有直接的控制權，可以說它是個可以操控真品的複製品。 |
| Authority | 擁有該狀態表示它是伺服端電腦中的Actor，伺服端擁有對它的完整控制權，它的資訊會被同步給用戶端，可以說它就是該遊戲物件的真品。 |

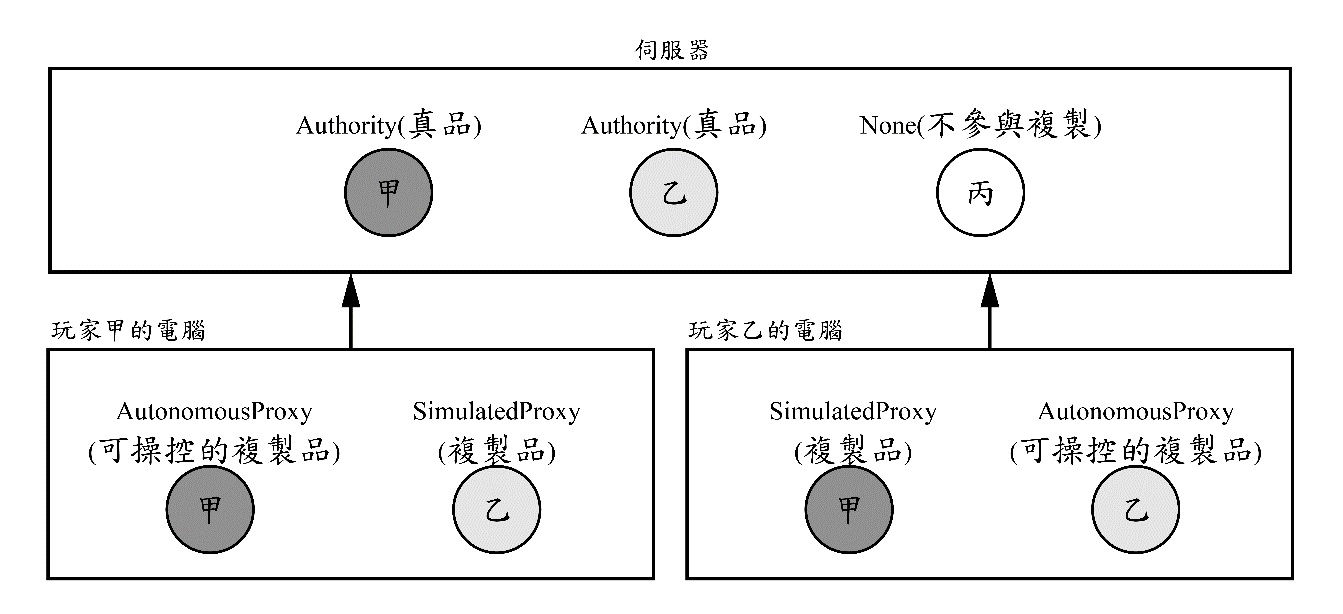


圖2.1 1-2-2 網路角色關係圖

### 2.2.4 多人遊戲中GamePlay 框架各類別的用途

在多人遊戲中，Unreal Engine 的 GamePlay 框架各類別有著各自負責的工作範圍，以下將會詳細描述每個類別的用處與彼此間的合作關係。

表2.2 1-2-2 GamePlay框架各類別在多人遊戲中的用途

|  |  |
| --- | --- |
| 類別 | 說明 |
| GameInstance | 用來儲存全域的遊戲資訊，例如遊戲的音量、按鍵設定與多人連線邏輯等功能，並且它只存在於自己的裝置中，也就是其他玩家或甚至伺服器都無法取得你的 GameInstance。 |
| GameMode | 負責管理玩家登入、生成任何需要同步的遊戲物件(包括玩家的控制器與玩家的遊戲角色)、管理遊戲暫停或重啟等邏輯，且擁有所有玩家的PlayerController ，同時它只存在於伺服端中，用戶端是無法取得它的。 |
| GameState | 用來儲存共用的遊戲資料，例如目前的回合數、遊戲時間、遊戲階段等資料，同時也擁有所有玩家的PlayerState，用戶端與伺服端中都會有它，以便於同步共用的資料。 |
| PlayerState | 儲存玩家的個人遊戲資料，例如目前自己的分數、獲得的金幣、擊殺敵人的數量等個人的遊戲資訊，當玩家斷線時它還可以保留在遊戲中，直到玩家重新連線時，可以用來還原玩家的遊戲進度。 |
| PlayerController | 它擁有玩家自己的遊戲角色(Pawn)，並且用來設定玩家輸入如何去操控遊戲角色，用戶端中玩家只擁有自己的控制器，並不存在其他人的。同時，它關聯了用戶端與伺服器，所有的網路同步都是透過它，去向伺服器進行溝通的，所以在用戶端中需要進行同步的遊戲物件都需要讓PlayerController 去擔任它們的擁有者，才能成功進行同步。 |
| AIController | 用來控制非玩家的遊戲角色，可以設置其操控角色的行為樹，使該遊戲角色會依照事先編寫好的行為來進行活動，並且它只存在於伺服端，用戶端是無法取得它的。 |
| Pawn | 用來撰寫遊戲角色的能力邏輯，例如施放魔法、撿拾物品等遊戲角色的專屬邏輯，伺服端與用戶端它都存在，並且伺服端上的Pawn會即時同步到用戶端的Pawn上。 |
| HUD | 這是用來繪製簡易介面元素用的，比起 UMG 的介面更節省效能，但是更加複雜，並且它只存在於自己的用戶端中，其他的用戶端中也只會他們自己的HUD。 |

## 2.3 遊戲架構的具體設計

綜合以上Unreal Engine 的 GamePlay 框架設計，以及其中類別的關係、負責的部分、網路複製的概念等內容，以下將會詳細描述整個 VR卡牌遊戲的遊戲架構。

### 2.3.1 開始場景的整體運作流程概述

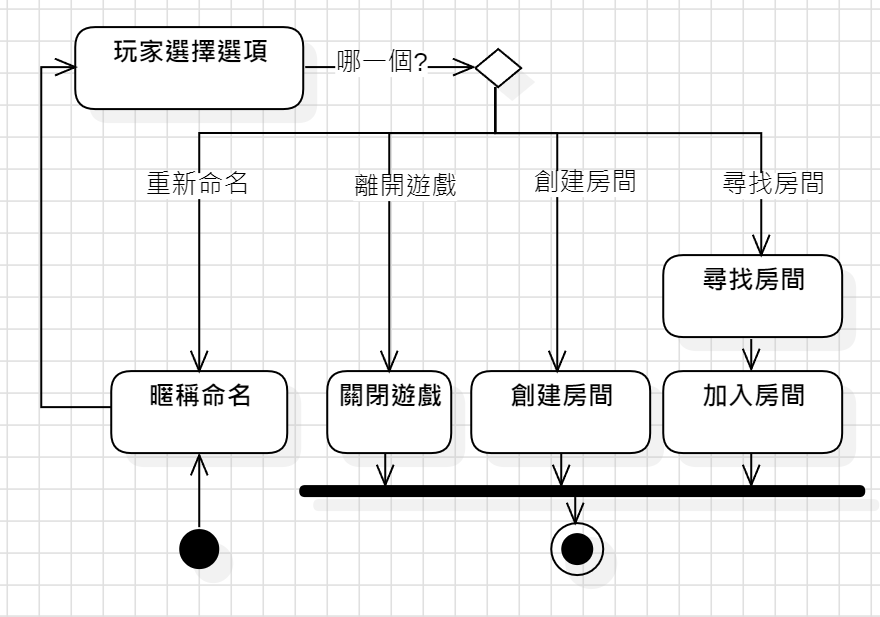


圖2.1 1-3-1 開始場景流程概略描述

### 2.3.2 開始場景的整體運作流程詳述

圖2.1 1-3-2 開始場景流程詳細描述

### 2.3.3 對戰場景的整體運作流程概述

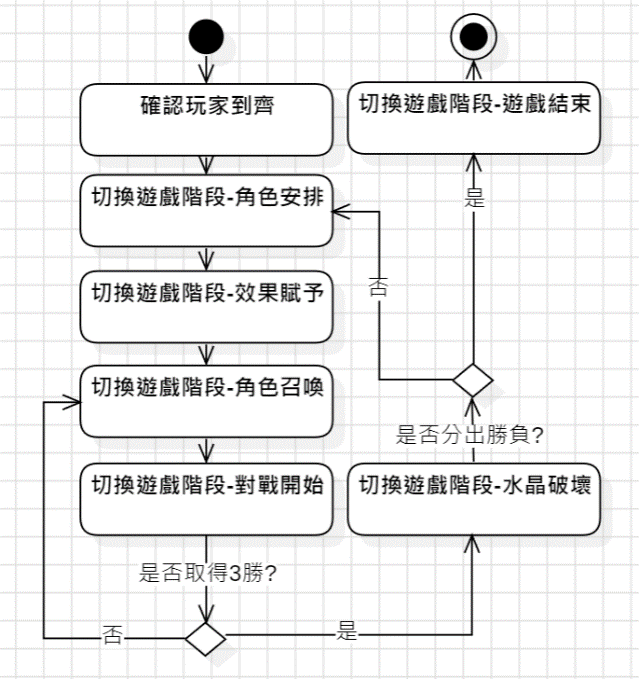


圖2.1 1-3-3 對戰場景流程簡略描述

# 第三章 類別細項具體設計

Qweqrqr

# 第四章 專案結構

Qweqrqr

# 第五章 角色設計

Qweqrqr

# 第六章 介面與遊戲形象設計

Qweqrqr