睿決科技股份有限公司

遊戲開發技術的架構

我們的遊戲開發架構可以進一步細分為以下幾個主要模塊

用戶端

用戶端是玩家與遊戲互動的主要窗口，負責所有的遊戲視覺呈現、音效處理、用戶輸入及遊戲邏輯。Unity引擎提供了高度模組化的開發方式，以組件的方式構建遊戲功能。每個遊戲場景都是獨立的模組，包含角色、物理引擎、動畫及其他遊戲物件，便於進行擴展和維護，並可通過場景加載和場景管理器來控制場景間的切換。

資料同步

客戶端與伺服器需要頻繁通信來同步資料，如角色位置、動作及遊戲物件狀態。我們可以使用TCP/IP協議進行傳輸確保數據準確性；或使用UDP協議實現高效的即時通信，適用於對延遲要求較高的功能如角色移動。對於大規模多人遊戲，會使用專門的網路框架或雲端解決方案來應對高並發請求。

伺服器端

伺服端負責管理遊戲的核心邏輯，可以基於Node.js或其他後端技術，處理多個用戶的請求。我們可以選擇無狀態伺服器，處理每次請求結果，並將狀態存儲於外部數據庫中，以提高伺服器的可擴展性。也可以選擇狀態伺服器，持有當前遊戲進程中的玩家狀態，以提供即時的遊戲反應。

網路延遲處理

網路延遲是在多人遊戲開發中常見的挑戰之一。我們可以實施延遲補償技術，將玩家的動作預測渲染，當伺服器確認動作後，再將結果回饋給玩家。這能夠在大多數情況下減少延遲感知，提升玩家體驗。除此之外，我們還可以考慮實現插值和外插技術來平滑角色運動。

遊戲程式開發的技術

遊戲為大型編程的集成，需要用到多項程式語言的整合

Unity引擎

Unity引擎提供了豐富的API及工具來構建遊戲。可以處理角色的碰撞與移動，通過動畫系統實現角色的動畫效果。並且他的加載系統有效減少遊戲啟動時的載入時間，並利用渲染技完成後處理效果提升遊戲畫面的視覺質感。

WebSocket

客戶端與伺服端通信通過WebSocket實現持續的雙向連接，尤其適合即時多人遊戲。並且能讓伺服器在狀態變更時即時通知客戶端，避免客戶端頻繁輪詢。對於較少即時性需求的數據通信，則使用RESTful API。

C#編程

在Unity中，我們使用C#語言來實現遊戲邏輯。C#語言的面向對象特性讓我們能設計出遊戲中的不同模組，如角色控制、AI行為和遊戲規則等。通過事件驅動架構，我們能夠使遊戲更加靈活，且便於維護與擴展。

SQL數據庫

遊戲開發中的數據庫設計是後端的核心。對於玩家的帳號數據、遊戲進度及遊戲內虛擬經濟的數據，使用SQL數據庫進行結構化存儲。關鍵在於確保數據庫能在高併發情況下保持穩定運行，這可通過讀寫分離、緩存策略等技術來實現。

用戶端功能的需求

多用戶型態的遊戲，需於用戶間的互動架構完成需求

用戶註冊和登錄

玩家需要能夠創建帳號並安全地登入遊戲。這一功能可以集成第三方認證服務（如OAuth：Google、Facebook等），以簡化用戶體驗並提高安全性。

為了進一步提升帳戶保護，我們還可以實現雙因素認證（MFA）。此外，我們還應提供“忘記密碼”功能，讓玩家能輕鬆恢復帳戶。

遊戲場景管理

我們的多人在線功能不僅需要實現即時的數據同步和玩家交互，還需處理伺服端的同步問題。

舉例來說，在射擊遊戲中，當兩個玩家在不同時間點發射子彈並命中對方時，伺服器需要根據具體情況進行裁決。

我們可以實現“權威伺服器”機制，讓伺服器成為遊戲狀態的最終決策者，避免完全依賴客戶端資料。

社交互動

為遊戲添加社交功能，這是一個提高玩家粘性的關鍵部分。

玩家之間的好友系統、聊天功能以及遊戲內消息通知等都是常見功能。

考慮實現跨伺服器的好友系統，讓玩家無論在哪個伺服器都能與好友保持聯繫。排行榜功能也能提升競技性及玩家互動。

功能需求的延伸

基礎架構上完整的功能，也包含對於遊戲數據的細節處理

遊戲場景管理

包括不同的關卡設計和地圖切換及用戶界面的互動。

Unity提供的場景管理系統可以在場景間切換，並根據玩家的行為動態加載或卸載不同的場景，從而節省資源。

對於場景，我們還可以實現動態光照、物理互動及程序生成的場景物件，進一步提升遊戲的變化性及重玩價值。

數據持久化

保存是關鍵功能，無論是玩家的進度、成就與物品的收集，都需要數據持久化存儲。

我們使用關係型數據庫儲存這些數據，並在設計數據庫架構時，考慮到數據的一致性、持久性及快速查詢能力。

並且可使用NoSQL數據庫來處理一些非結構化數據，如遊戲中的聊天記錄或即時動態。

存儲和版本控制

除了遊戲數據的持久性存儲，也需要即時數據同步。

通過版本控制，我們可以確保每個遊戲版本的數據都能正確保存和恢復。

當遊戲版本更新時，可能會進行數據遷移，因此設計良好的數據架構將提高更新效率及安全性。

安全性技術

用戶身份驗證

為確保用戶創建帳號及登入密碼的安全性，使用安全的哈希算法（如SHA-256或Bcrypt）來存儲密碼，並加入隨機鹽值，防止彩虹表攻擊。並利用雙因素驗證（2FA），在登入時要求玩家提供額外的身份驗證信息（如短信或電子郵件中的驗證碼），以進一步提高帳號安全性。

數據加密

數據傳輸過程需進行加密並保護敏感信息。利用SSL/TLS來加密網路通信可防止數據在傳輸時被竊取。並對其進行加密保護，如玩家個人信息、交易記錄、遊戲中的數據。敏感信息使用AES等對稱加密算法來保護，即使伺服器遭到攻擊，攻擊者也無法直接讀取這些數據。

防作弊與滲透

在多人遊戲中須防止作弊來保護遊戲的公平性。除了檢測和攔截可疑的遊戲行為、分析數據包異常以及伺服器端的遊戲邏輯驗證。伺服器端的權威模型負責所有關鍵的遊戲狀態決策，客戶端僅作為遊戲畫面的呈現和玩家行為的執行。另外並行安全技術來防範DDoS攻擊、SQL注入及XSS等常見的網絡攻擊。

交易系統安全

若遊戲中涉及虛擬經濟或內購系統，可採用加密技術保護交易過程中的數據，確保玩家的支付信息和交易內容不會在傳輸過程中被攔截或篡改。為了防止重放攻擊，我們會設計每筆交易的唯一識別符並進行交易時間戳驗證，確保每次交易都是合法且唯一的。

系統異常檢測

我們的伺服器將設置詳細的日誌系統，記錄伺服器操作及用戶行為。通過日誌，可以追蹤異常行為和可能的安全漏洞。並有異常檢測系統，定期檢查並識別潛在的安全威脅。當檢測到異常數據流或可疑的用戶行為時，系統會自動觸發警報並採取措施進行隔離和處理。

遊戲內的觸發條件設定

針對遊戲內各種條件，須設定上百組遊戲規範來讓遊戲機制平衡和防作弊。

```
137076        bossMapDlg.prototype.updateData = function () {
137077          this.m_name = msMoudle.maplejson['b_name'].split(',');
137078          this.m_map = msMoudle.maplejson['b_map'].split(',');
137079          this.m_mapLimits = msMoudle.maplejson['yz_limit'];
137080          this.lstMap.vScrollBarSkin = '';
137081          this.BSts.text = '今日远征剩余：' + ms.yzcs + '次';
137082          var tArr = [];
137083          for (var i = 0; i < this.m_name.length; i++) {
137084            tArr[i] = new Object();
137085            tArr[i].name = this.m_name[i];
137086            tArr[i].lv = this.m_lv[i];
137087          }
137088          this.lstMap.dataModel = tArr;
137089        };
137090        bossMapDlg.prototype.onLstMapCellClick = function (e, index) {
137091          var _this = this;
137092          var curTime = Sync.serverTime();
137093          if (curTime - this.teamBossClickTime < 2000) {
137094            msMoudle.toast('操作过于频繁');
137095            return;
137096          }
137097          this.teamBossClickTime = curTime;
137098          var mapId = this.m_map[index];
137099          var limit = this.m_mapLimits[mapId];
137100          if (!limit) {
137101            msMoudle.help = null;
137102            msMoudle.tiaotiao_map = mapId + '.img';
137103            console.log('##change map id=', msMoudle.tiaotiao_map);
137104            msMoudle.gameP.gotoScene(msMoudle.tiaotiao_map);
137105            this.close();
137106            return;
137107          }
137108          if (!Sync.partyId) {
137109            msMoudle.toast('请先组建一个队伍');
137110            return;
137111          }
```

設定玩家每日可以挑戰BOSS的次數，以及限定需要組隊和人數設定才可以進入關卡。

判斷防作弊的機制設定，限定兩次滑鼠點擊時間不得低於2000毫秒，如果低於2000毫秒，則跳出警告。

遊戲測試與優化延伸

開發遊戲的過程，正式發布前最主要的重要測試

1. 單元測試與自動化測試

在遊戲開發過程做單元測試，確保每個模組在能夠正確運行。Unity提供了Test Runner工具來幫助編寫和運行自動化測試腳本，藉此測試核心遊戲邏輯、用戶界面以及網絡通信功能。並利用自動化測試在每次更新遊戲時，快速檢測並修復潛在的問題。

2. 性能優化

通過優化遊戲的渲染流水線，確保僅渲染可見物體，並使用LOD（細節層次）技術降低遠距離物體的多邊形數。另針對移動設備優化控制記憶體使用、減少過多的運行時物件實例化，並利用資源池技術來重用物件，減少垃圾回收的頻率。

3. 網絡性能測試與壓力測試：

針對多人遊戲的網絡進行性能測試，模擬不同網絡環境下的延遲、丟包等情況，測試遊戲的網絡延遲補償技術和數據同步機制的可靠性。並進行壓力測試，模擬大量玩家同時連接伺服器時伺服器能夠穩定應對高併發請求。

4. 用戶反饋測試：

遊戲的內測階段邀請玩家參與測試，根據玩家的反饋幫助我們識別遊戲中的潛在問題，無論是遊戲性、用戶體驗還是技術問題。並針對關鍵功能進行多次迭代測試，以確保最終的遊戲版本能夠滿足玩家的需求。

遊戲發佈與運維延伸

遊戲發佈準備

遊戲進入開發最終階段，準備遊戲的正式發佈。我們會確保所有遊戲模組已經通過測試，並且伺服器架構能夠支持大規模的玩家登入。並制定詳細的發佈計劃，包括宣傳活動、玩家引導以及版本更新計劃，確保遊戲的發佈順利進行。

伺服器監控與維護

遊戲發佈後實施24/7的伺服器監控，利用監控工具來追蹤伺服器的性能數據，包括CPU、記憶體使用率、網絡流量等，並設置自動化警報系統，當伺服器負載過高或出現異常時及時通知維護人員。定期的伺服器維護和更新將確保遊戲持續穩定運行。

定期更新與內容擴展

我們將根據遊戲的發展進行定期更新，不僅會修復潛在的漏洞和性能問題，還會推出新的遊戲內容，如關卡、角色和道具等，來保持玩家的新鮮感和參與度。另制定長期的更新計劃，確保遊戲能夠不斷演進並符合玩家的期望。

版本控制與持續集成

使用Git或其他版本控制系統來管理代碼變更，並確保更新時代碼的穩定性。通過持續集成（CI）工具，自動化編譯和測試流程，快速檢測和修復潛在的問題，從而提高開發效率。設置開發、測試和生產環境，能確保每個更新能夠在安全的情況下部署。

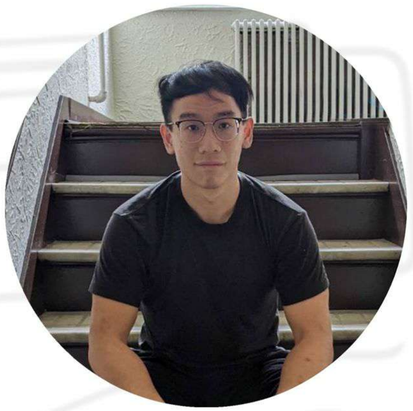
用戶支持與回饋機制

發佈後將提供多種管道的玩家支持服務，包括在線客服、社群支持以及故障報告系統。定期收集玩家的反饋，並根據玩家的需求進行遊戲內容的優化與更新。通過快速回應玩家的問題和反饋，我們能夠提升用戶滿意度並增加遊戲的長期活躍度。

核心團隊介紹
Core Members

Vance Lin / 林廷翰
Founder & CEO

台灣科技大學資訊管理所 碩士
中國 浙江大學 區塊鏈實驗室訪問研究
唯數娛樂 區塊鏈專案負責人
5年區塊鏈產業經驗



Kobe Chao
Co-Founder & CTO

政治大學 資訊科學
新加坡商 Xfers 區塊鏈開發主管
多年區塊鏈開發經驗
參與兩國穩定幣發行專案

Kilin Chen
CMO

十年電商經驗
Groupon、Klook、
foodpanda 資深業務經理