

서버 기획서

김용혁

201911166

**서버 기획서**

|  |
| --- |
| **목차**  **1. 프로젝트 개요**  **2. 서버 아키텍쳐**  **3. 주요 기능**  **3.1. 로그인 및 계정**  **3.2. 매치메이킹**  **3.3. 게임 로직**  **4. 보안 및 안정성**  **4.1. 접근 제어**  **4.2. 서버 안정성**  **5. 업데이트 및 유지보수** |

**1. 프로젝트 개요**

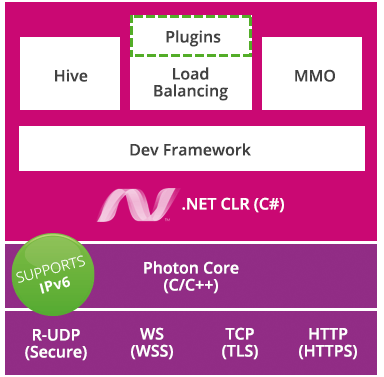
• 프로젝트 명: TRASHES

• 게임 장르: 멀티 배틀로얄

• 사용 서버: Photon 서버

• 플랫폼: PC & Steam

**2. 서버 아키텍쳐**



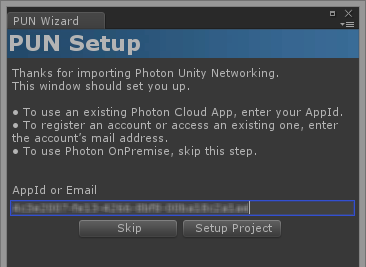
• 서버 유형: 실시간 멀티플레이어

• 네트워크 모델: Client-Server

• 클라우드 서비스 활용: Photon 클라우드 서비스 활용

**3. 주요기능**

**3.1. 로그인 및 계정**



Photon Server 가입 후 AppId 발급, 이후 Photon Unity Networking(PUN) 프로젝트 임포트 AppId 등록

**3.2. 매치메이킹**

**OnConnectedToMaster** 내에서 기존에 있는 룸에 참여하거나 룸을 생성할 수 있다. 다음 코드는 게임을 시작하거나 참여하기 위한 메소드 가능한 호출을 나타낸다.

1. PhotonNetwork.JoinRoom("someRoom");
2. PhotonNetwork.JoinRandomRoom();
3. PhotonNetwork.CreateRoom("MyMatch");

친구들이 같이 플레이하기를 원하고 PUN 외부에서 소통할 방법을 가지고 있을 때, 룸 이름을 만들어 JoinOrCreateRoom 을 사용할 수 있다.

RoomOptions roomOptions = new RoomOptions();

roomOptions.IsVisible = false;

roomOptions.MaxPlayers = 4;

PhotonNetwork.JoinOrCreateRoom(nameEveryFriendKnows, roomOptions, TypedLobby.Default);

무작위 매치: 특정 맵이나 모드 임의의 사용자 지정 룸 속성을 설정하여 JoinRandomRoom 요청에서 필터로 사용할 수 있다.

빠른 매치: 플레이어들은 매치에 바로 참여하기를 원할 수 있다. 그래서 플레이어가 긴 룸 목록에서 임의로 하나를 선택하도록 요청하지 않고 바로 룸으로 이동할 수 있도록 수행

**3.3. 게임 로직**

GameObject들은 PhotonView 컴포넌트에서 "networked GameObjects" 로써 인스턴스화 될 수 있다. 객체와 소유자(또는 컨트롤러)를 식별한다. 제어를 하고 있는 플레이어가 다른 모든 사용자를 업데이트를 한다. 전형적으로, PhotonView를 프리팹에 추가하여 Observed 컴포넌트를 선택하고 인스턴스 생성을 위해 PhotonNetwork.Instantiate를 사용할 수 있다. 스크립트는 반드시 OnPhotonSerializeView를 정의하는 IPunObservable을 구현해야 한다.

텍스트, 스크린샷, 폰트, 대수학이(가) 표시된 사진

자동 생성된 설명

**4. 보안 및 안정성**

**4.1. 접근 제어**

1. Network/DNS 이슈

체크하기 위해서 만약 Licence Monitor에 연결할 수 있으면 curl 과 같은 툴을 사용할 수 있다.

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

두 요청 모두 실패한다면 네트워크 관리자에게 연락하여 네트워크 환경설정/DNS 이슈를 해결해야 한다.

1. 방화벽 설정

PhotonSocketServer.exe 가 License Monitor URI의 443 포트(HTTPS)에 아웃바운드 연결 성립의 권한이 있는지 확인해야 한다. 그리고 Windows 방화벽과 호스팅 업체에서 제공하는 하드웨어/소프트웨어 방화벽도 점검해야 한다.

1. 인증서 설정

Internet Explorer에서 https://licensesp.exitgames.com를 오픈 한다. 만약 오류 메시지를 보게 된다면 서버에 신뢰받은 Root CA에서 발급받은 SSL 인증서를 설치해야 한다.

**4.2. 서버 안정성**



연결형 서비스를 지원하는 전송계층 프로토콜인터넷 환경에서 기본으로 사용한다. 호스트간 신뢰성 있는 데이터 전달과 흐름제어 및 혼잡제어 등을 제공하는 전송계층이다.

**5. 업데이트 및 유지보수**

RPC 태그가 붙은 함수 이름을 짧게 한다.

public void PlayerMoveAndRotationSetRPC() { } X, public void UpdateTransform() { } O

이유는 간단하다. 문자열은 네트워크를 통한 전송에서 가장 비효율 적이기 때문임, 그렇기에 포톤에서는 해당 함수 문자열을 줄여서 보내는 트릭을 사용하기 때문에 함수 이름을 짧게 사용하는 것이 좋다

RPC 호출 후 바로 패킷을 보낸다.

RPC는 호출 시 바로 패킷을 전송하는 것이 아니라 버퍼에 담은 후 오브젝트의 변화가 있을 때, 전송을 하기 때문에 해당 RPC를 바로 보내고 싶을 경우 다음과 같은 함수를 사용하면 된다.

Ex) PhotonNetwork.SendAllOutgoingCommands();

해당 함수를 호출하게 된다면 RPC로 실행하고 버퍼에 담긴 함수를 강제로 실행할 수 있다. 하지만 패킷을 계속 보내게 된다면 결국 속도는 느려지게 될 것이니 적절한 타이밍에 사용하는 것이 좋다.

패킷 전송을 줄인다.

당연한 말이지만 지연 속도를 줄이기 위해서는 패킷 보내는 양을 줄이면 된다.

객체 갱신을 위해 OnPhotonSerializeView와 FixedUpdate 사용한다.

두 번째 옵션은 OnPhotonSerializeView 구현 외에 Unity의 FixedUpdate 함수를 사용하는 것이다.

다시 빈 OnPhotonSerializeView 함수로 시작한 후 이 접근법에서 보낸 사람은 동일한 작업을 수행한다. Rigidbody의 위치, 회전 및 속도를 공유하는 것이기 때문에 수신자의 작업은 이전의 접근법과 다른 것이다.