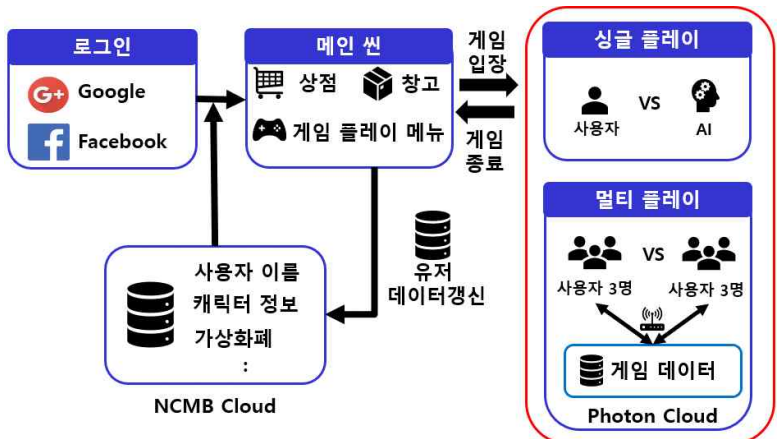
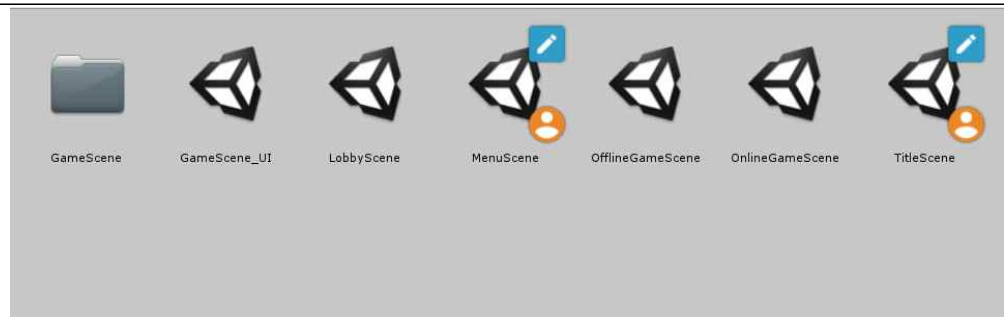


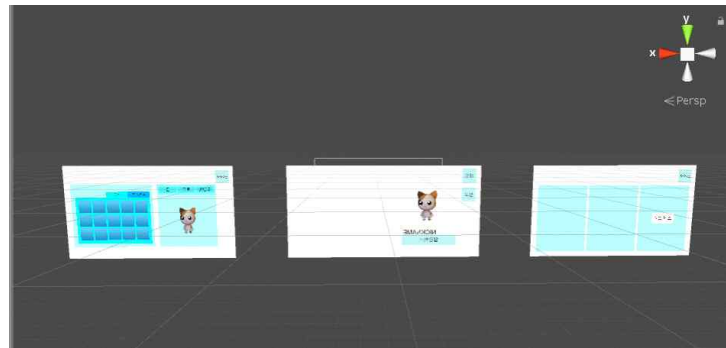
2018-2학기 세종창의학기제 주간학습보고서

이름	오민규	학과(전공)	컴퓨터공학과		
학번	128206	학년	4		
연락처	010-8843-4757	e-Mail	hotalsrb12@naver.com		
과목명	자기주도 창의전공 I, II, III, IV	신청학점	12	분반	3
학습기간	2018. 11. 05 ~ 2018. 11. 11	학습주차	9	학습시간	60
창의과제	네트워크를 통해 상대방과의 실감나는 대전형 슈팅 3D 모바일게임 개발				
금주 학습목표	<ul style="list-style-type: none"> - 게임 시스템 구성도 구현 - 게임 플레이를 위한 메뉴 씬(Scene) UI 제작 - 매치메이킹 구현 				
학습내용	<p>1. 게임 시스템 구성도 구현</p> <p>현재까지 씬(Scene)을 구성하였을 때 독립적으로 해당 씬(Scene)에서 기능을 하도록 구현하였다. 하지만 서버와 데이터베이스가 장착되어 있는 프로젝트이기 때문에 로그인 과정부터 게임을 시작하고 종료되는 과정까지는 게임 시스템 구성도를 구현해야한다. 타이틀 씬(Scene)에서 사용자가 로그인을 하고난 후 화면을 터치하면 메뉴 씬(Scene)으로 넘어가야한다. 메뉴 씬(Scene)에서 게임플레이 UI를 누르고 멀티플레이를 선택하였을 때 매치메이킹이 되어 로비 씬(Scene)으로 넘어가야하고 로비 씬(Scene)에서 다른 플레이어들이 참가하여 적정 플레이어가 참가한다면 게임을 진행을 위해 게임 씬(Scene)으로 넘어가야 한다. 게임이 종료되고 다시 메뉴 씬(Scene)으로 되돌아가야한다.</p>  <p>The flowchart illustrates the game system architecture. It starts with a '로그인' (Login) screen containing Google and Facebook icons. An arrow points to the '메인 씬' (Main Scene), which includes '상점' (Shop), '창고' (Warehouse), and '게임 플레이 메뉴' (Game Play Menu). Below the login screen is a box for '사용자 이름, 캐릭터 정보, 가상화폐' (User Name, Character Info, Virtual Currency) connected to 'NCMB Cloud'. The '메인 씬' is connected to '유저 데이터갱신' (User Data Update). From the '메인 씬', arrows lead to '게임 입장' (Game Entry) and '게임 종료' (Game End). '게임 입장' leads to '싱글 플레이' (Single Play) with '사용자' (User) vs 'AI'. '게임 종료' leads to '멀티 플레이' (Multi Play) with '사용자 3명' (3 Users) vs '사용자 3명' (3 Users). Both play modes are connected to '게임 데이터' (Game Data) and 'Photon Cloud'.</p> <p>이러한 씬 전환이 이루어져야 게임의 기본 틀이 완성이 된다. 이전 작업까지는 타이틀 => 메뉴까지 구현을 완료한 상태이고 나머지 씬(Scene)은 데이터베이스가 적용되어 있지 않아서 사용자의 정보를 가진 게임이 아닌 독립적인 씬(Scene)이었다. 이들을 하나로 묶기 위해서 타이틀 씬(Scene)에서 처음 만들어진 매니저들을 DontDestroyOnLoad 처리하고 특정 이벤트가 처리되었을 때 씬(Scene)을 전환하는 방식으로 처리하여 게임 씬(Scene)까지 전환되는 연결 작업을 하였다.</p>				



2. 게임 플레이를 위한 메뉴 씬(Scene) UI 제작

게임플레이 및 상점, 창고, 뒤로가기 버튼 등 메뉴 씬(Scene)에서 필요한 UI를 구현해야한다. 상점과 창고의 경우 각각 다른 패널을 생성하여 메뉴 씬(Scene)에서 메인 패널에 존재하는 버튼을 누를 경우 각 패널이 Active되는 방식으로 처리하였다. 뒤로가기 버튼의 경우 델리게이트를 사용하여 이벤트를 변경하는 방식을 사용하여 각 패널마다 버튼을 생성 할 필요가 없게 구현하여 메모리 낭비를 줄였다.



메인 패널에서 입장하기 버튼을 누르면 위 그림의 오른쪽에 위치한 패널을 Active시키고 게임 방식을 선택하는 버튼을 패널의 자식으로 두고 사용하였다. 사용자가 싱글플레이 버튼을 클릭할 경우 Offline Scene으로 장면을 전환하도록 이벤트 처리를 하였고, 멀티플레이 버튼을 클릭하면 매치메이킹을 시작하는 이벤트 처리를 하였다. 이 과정에서는 Cloud를 변경하기 전 상태이기 때문에 UNET을 이용하여 매치메이킹이 되는 과정이었다. 앞으로 걱정되는 부분은 UI의 디자인과 패널의 배경을 적용시키는 과정이 어려울 것 같다. 배경의 경우 일러스트를 채워야 하는데 디자이너가 없는 상태여서 외주를 맡겨 진행해야 한다는 점이다. 게임 구현을 완료하고 출시를 할 때쯤 개인사업자를 등록하여 학교에서 지원금을 받아 진행하는 방법을 택해야겠다. 그리고 메인 패널의 UI가 너무 적은 문제가 있어서 앞으로 어떤 기능을 더 추가하고 버튼들을 배치할지 많은 고민이 필요한 상태이다. 이 부분은 출시된 게임들을 많이 찾아보고 플레이를 해보면서 참고해야겠다. 일단 여기까지가 현재 메뉴 씬(Scene)에서 필요한 UI들을 구성한 상태이다.

상점

참고



NICKNAME
입장하기

3. 매치메이킹 구현

UNET으로 구현된 매치메이킹 시스템을 Photon을 이용하여 변경해야한다. 네트워크를 변경하고 가이드를 통해 학습을 해보니 UNET보다 Photon을 사용할 경우에 모바일 기기에서 장점이 더 많이 있었다. 모바일 기기에서 플레이어는 게임도 중 언제든 게임을 종료할 수 있다. 우리는 호스트 방식을 사용한 네트워크 시스템이기 때문에 마스터 클라이언트의 역할이 중요하다. 이 클라이언트가 매치메이킹이 완료된 후 게임 플레이를 대기하는 상태에서 도중에 나가버리게 된다면 문제가 발생하게 된다. 게임이 시작된 후에도 마찬가지이다. 하지만 Photon의 경우 마이그레이션이 가능하기 때문에 마스터 클라이언트가 매치메이킹이 시작된 후 나가버리게 된다면 다른 클라이언트에게 이 권한을 인계할 수 있는 기능이 탑재되어있다. UNET의 경우도 이를 지원하기는 하지만 문제가 아직 많이 있어서 많은 개발자들이 사용하지 않는다고 하였다. 네트워크 시스템을 변경한 것은 좋은 선택이었던 것이다. 가이드를 학습하고 매치메이킹 구조를 UNET과는 다르게 만들고 코딩을 시작하였다. PunBehavior를 상속받아 기본적으로 구현되어있는 API를 사용하지 않고 매치메이킹에 사용되는 함수를 오버라이딩 시켜 Custom으로 직접 기능을 구현하였다. 사용자가 멀티플레이를 선택하였을 때 매치메이킹을 시작하는데 만약 만들어진 게임방 중 이미 게임이 시작되어서 참가가 불가능한 경우와 게임방 로비에 사람이 가득 찬 경우 게임방에 참가하는 불가능하므로 이 때 사용자가 직접 방을 생성하여 대기하는 방식을 구현하였다. 생성, 참가, 나가기 등 매칭에 필요한 요소들을 구현하였고 앞으로 매칭이 되고나서 로비의 UI와 게임 시작하기 전 이벤트들을 구현해야한다.

```

// 로비 서버
// Photon 네트워크에 접속
void AwakePhotonConnect()
{
    PhotonNetwork.ConnectUsingSettings(appVersion);
}

// 로비 방 생성 함수
public static void StartMatch(NetworkMode mode)
{
    // 같은 버전의 클라이언트끼리 모드
    PhotonNetwork.autoacticallySyncScene = true;
    PhotonNetwork.playerName = UserAuth.Instance.GetNickName();

    switch (mode)
    {
        // Photon 서버에서 사용할 수 있는 클라우드 게임에 연결
        case NetworkMode.Online:
            PhotonNetwork.PhotonServerSettings.HostType = ServerSettings.HostingOption.BestRegion;
            PhotonNetwork.ConnectToBestCloudServer(appVersion);
            break;
        case NetworkMode.Offline:
            PhotonNetwork.offlineMode = true;
            break;
    }

    // 연결이 설정되기 전에 Photon 서버에 대한 연결 호출이 실패한 경우 호출
    public override void OnFailedToConnectToPhoton(DisconnectCause cause)
    {
        if (connectionFailedEvent != null)
            connectionFailedEvent();
    }

    // 무언가가 연결을 실패하게 만들 때 호출 (설정된 후)
    public override void OnConnectionFailed(DisconnectCause cause)
    {
        if (connectionFailedEvent != null)
            connectionFailedEvent();
    }
}

// 로비 클라이언트
// 방장 서버의 로비에 입장 할 때 호출
// 최초 서버에 연결 되었을 때 (이전 로그인하고 Main에서 호출되어야함)
public override void OnJoinedLobby()
{
    // 방장에 연결할 때 방에 들어 가려고함
    PhotonNetwork.JoinRandomRoom();
}

// 클라이언트가 방을 만들고 들어 갈 때 호출
public override void OnCreatedRoom()
{
    // 방을 만들것으로 초기 방 속성을 할 채우기 및 점수 배열 채우기와 비슷하게 설정
    Hashtable roomProps = new Hashtable();
    roomProps.Add(RoomExtensions.size, new int[RoomExtensions.initialArrayLength]);
    roomProps.Add(RoomExtensions.score, new int[RoomExtensions.initialArrayLength]);
    PhotonNetwork.room.SetCustomProperties(roomProps);
}

// 로비 씬 로드
PhotonNetwork.LoadLevel(lobbySceneIndex);

// 방에 들어갈 때 호출(만들거나 참가함으로써)
public override void OnJoinedRoom()
{
    // 게임이 끝난방에 들어간다면, 즉시 연결을 끊는다.
    if (GameManager.GetInstance() && GameManager.GetInstance().IsGameOver())
    {
        PhotonNetwork.Disconnect();
        return;
    }

    if (!PhotonNetwork.IsMasterClient)
        return;

    // 게임에 자신을 추가
    // 마스터 클라이언트에서만 호출
    // 다른 클라이언트가 OnPhotonPlayerConnected 콜백을 직접 트리거하기 때문
}

```

	<p>4. 느낀점</p> <p>게임의 흐름을 엔진을 사용하여 구현해보니 함수 하나를 이용하여 전환을 할 수 있다는 점에서 편리하게 구현할 수 있었다. 데이터베이스를 연동시키는 과정이 조금 까다로웠지만 그래도 메뉴 씬(Scene)에서 사용자의 정보와 렌더타겟을 이용하여 사용자의 캐릭터를 렌더링 시키는 부분에서 큰 만족감을 얻었다. 앞으로 가장 걱정되는 부분이 디자인적인 요소인데 컴퓨터전공생 2명이서 만들다보니 아무래도 아트적인 감각이 부족하다. 많은 게임들을 찾아보고 에셋스토어에서 우리 게임과 어울리는 디자인들을 찾아봐야겠다. 매치메이킹을 구현하면서 Photon의 강점을 이용하고 편리한 API로부터 문제없이 구현을 할 수 있었다. Photon은 한국에 서버가 존재하여 나라 별로 서버를 두어 사용자가 선택할 수 있도록 구현이 가능하며 같은 버전을 사용하는 사용자와 매칭이 되기 때문에 UNET을 사용하는 것보다 훨씬 좋은 기능들이 들어있다. 이번 프로젝트를 진행하면서 많은 플랫폼을 사용해 보는 경험을 쌓았고 앞으로 지금 개발하는 게임이 잘 되어 창업의 성공까지 이어지는 결과가 나타나게 된다면 추후 게임개발과 동시에 회사가 독자적으로 기술력을 바탕으로 게임을 구현하는데 필요한 플랫폼을 개발하여 개발자들에게 제공해주고 싶다는 생각을 하게 되었다.</p>
참고자료 및 문헌	<ul style="list-style-type: none"> - 절대강좌! 유니티 - 구글링 - https://doc-api.photonengine.com/ko-kr/pun/current/index.html (Photon 가이드)
학습방법	<p>절대강좌! 유니티 문헌을 통하여 UI에 대해 학습하고 적용시킨다. 구글링을 통하여 효율적인 씬(Scene) 전환방법을 찾고 Photon 가이드에서 함수의 기능에 관하여 Custom으로 제작하였을 때 그 함수가 하는 역할에 대해 세부적인 내용을 찾아 학습한다. Photon 가이드를 통하여 매치메이킹을 구현하기 위해 API를 학습하고 구현한다.</p>
학습성과 및 목표달성도	<ul style="list-style-type: none"> - 게임 시스템 구성도 구현 (70%) - 게임 플레이를 위한 메뉴 씬(Scene) UI 제작 (30%) - 매치메이킹 구현(80%) <p>게임 시스템 구성도 구현은 아직 게임이 종료되었을 때 메뉴 씬(Scene)으로 되돌아가는 루프가 존재하지 않기 때문에 완성되었다고 할 수 없고 로비 씬(Scene)으로 넘어가 다른 플레이어들을 기다리고 게임이 시작되는 루프가 남아있다. UI의 경우는 상점, 창고, 게임 플레이만 존재하기 때문에 메뉴UI를 구현하기까지는 아직 많이 남아있고 게임에서 UI가 가장 많이 적용되는 부분이기 때문에 한 주에 완료할 수 없었다. 매칭이 되도록 구현을 하였지만 매칭이 되고 여러 가지 작은 요소들을 구현하는 단계가 남아있다.</p>
내주 계획	<p>매치메이킹 구현이 완료되었으므로 게임을 플레이 했을 시 멀티플레이가 가능하도록 인 게임에 Photon을 이용하여 네트워크가 적용 된 이동, 공격, 게임 방식을 구현할 것이다. 게임에서 사용자들에게 게임 정보를 표시해주는 UI도 개발할 계획이다.</p>

2018. 11. 12 .

지도교수

(인)