2018-2학기 세종창의학기제 주간학습보고서

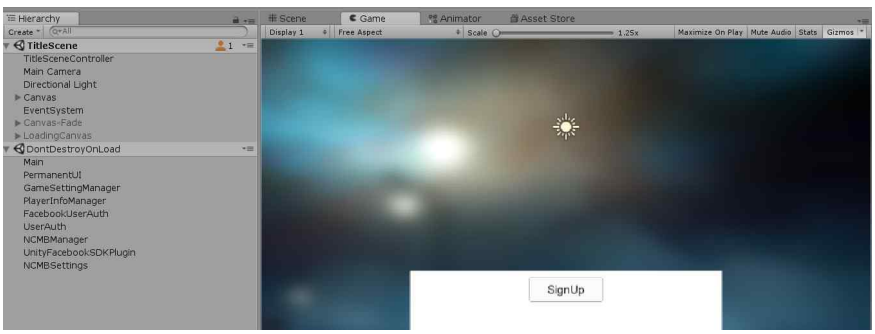
이름	오민규	학과(전공)	컴퓨터공학과		
학번	128206	학년	4		
연락처	010-8843-4757	e-Mail	hotalsrb12@naver.com		
과목명	자기주도 창의전공 Ⅰ,Ⅱ,Ⅲ,Ⅳ	신청학점	12	분반	3
학습기간	2018. 10. 29 ~ 2018. 11. 04	학습주차	8	학습시간	58
창의과제	네트워크를 통해 상대방과의 실감나는 대전형 슈팅 3D 모바일게임 개발				
금주 학습목표	- 데이터베이스 구현 - 씬(Scene) 전환에 필요한 UI 제작 - 네트워크 시스템 변경(UNET -> Photon Cloud)				
학습내용	**1. 게임 데이터베이스 구현** 6주차에 설계한 데이터베이스를 이용하여 백엔드 서버와 연결시켜 구현한다. 이전에 설계했던 유저 데이터 매니저를 이용하여 데이터를 가져오고 추가하는 작업의 내용을 채워 넣기 전에 DataStoreCoroutine 클래스를 설계하여 데이터를 서버에서 찾거나, 전체 데이터를 저장하거나, 가져오는 클래스를 만들어 유저 데이터 매니저의 유연성을 높인다. Coroutine을 이용하여 Data클래스에 있는 함수를 호출하게 된다면 UI가 실행될 때 이 과정 또한 같이 실행될 수 있다. 처음에 로그인 하였을 때 데이터를 전부 가져와야하는 상황과 게임을 종료하였을 때 사용자 데이터를 저장하는 상황과 친구 리스트를 가져오고 친구 한명의 정보를 확인할 때 UI와 같이 사용되어야 한다. DataStoreCoroutine을 구현하여 필요한 부분에 이 스크립트를 자동 추가시켜 사용하도록 작업을 하였다. 데이터 전체를 다루는 DataStoreCoroutine 클래스 구현을 끝마치고 이전에 만들어 놓은 틀에 데이터 한 개가 변경되거나 생성되거나 삭제되었을 때 처리할 수 있는 기능들을 구현해야했다. 처음 유저가 로그인 하였을 때는 정해진 기본 내용을 세팅하였다. 그 후부터는 서버에서 사용자를 탐색하고 기존 사용자라면 그 사용자의 데이터를 가지고 와서 게임이 실행되는 동안 메모리에 올려 사용할 수 있게 구현하였다. 리스트를 사용해야하는 요소와 딕셔너리를 사용해야하는 요소들이 있었는데 이 방법을 어떻게 사용해야할지 처음에는 감이 잡히지 않았다. 그래서 가이드와 커뮤니티에 정보를 찾아보았지만 이런 정보가 많이 부족하였다. 그래서 일단 내 스스로 구현해보자하여 코드를 작성해 나갔다. 하나의 요소를 저장, 갱신, 삭제, 로드 하는 코드를 다 작성한 후 실행해보았지만 다행이도 궁금했던 부분들이 적용이 되는 것이었다. 데이터베이스 구현을 마치고 이 기능을 확인할 수 있도록 UI를 구현해야했다.				

```
namespace Communication
{
    public class DataStoreCoroutine : MonoBehaviour
    {
        // DB에서 오브젝트 리스트 찾는 함수
        public IEnumerator FindAsyncCoroutine(NCMBQuery<NCMBObject> query, UnityAction<NCMBException> errorCallback)
        {
            bool isConnecting = true;

            List<NCMBObject> objectList = new List<NCMBObject>();

            query.FindAsync((List<NCMBObject> _objectList, NCMBException e) =>
            {
                if(e != null)
                {
                    errorCallback(e);
                }
                else
                {
                    objectList = _objectList;
                }

                isConnecting = false;
            });

            while(isConnecting == true)
            {
                yield return null;
            }

            yield return objectList;
        }

        // DB에 오브젝트를 저장하는 함수
        public IEnumerator SaveAsyncCoroutine(NCMBObject _object, UnityAction<NCMBException> errorCallback)
        {
            bool isConnecting = true;

            _object.SaveAsync((NCMBException e) =>
            {
                if(e != null)
```

```
// 비동기 처리에서 개체를 검색하는 함수
public IEnumerator FetchAsyncCoroutine(NCMBObject _object, UnityAction<NCMBException> errorCallback)
{
    bool isConnecting = true;

    _object.FetchAsync((NCMBException e) =>
    {
        if(e != null)
        {
            errorCallback(e);
        }

        isConnecting = false;
    });

    while(isConnecting == true)
    {
        yield return null;
    }
}

// DB에서 해당 객체를 삭제하는 함수
public IEnumerator DeleteAsyncCoroutine(NCMBObject _object, UnityAction<NCMBException> errorCallback)
{
    bool isConnecting = true;

    _object.DeleteAsync((NCMBException e) =>
    {
        if(e != null)
        {
            errorCallback(e);
        }

        isConnecting = false;
    });

    while(isConnecting == true)
    {
        yield return null;
    }
```

2. 씬(Scene) 전환에 필요한 UI 제작

UI를 디자인까지 신경써가며 제작하지는 않았다. UI의 경우는 사용자가 직접 눈으로 보고 실제 터치를 하며 사용하기 때문에 UI의 디자인 부분은 출시 이전으로 미루기로 계획을 잡았고 유니티에서 제공하는 UI를 사용하여 기능만 구현하였다. 로그인을 하였을 때 타이틀 씬에서 메뉴 씬으로 넘어가기 위해 특정 패널과 로그인을 하며 데이터를 가져오는 시간동안 사용자에게 로딩중이라는 문구가 적힌 패널이 필요하였다. 데이터베이스 구현을 완료하고 로그인이 진행되는 과정과 함께 UI가 활성화되도록 적용시켰다. UI에 대한 개념이 많이 부족하여 책을 보며 학습하고 적용하였기 때문에 시간이 오래 걸렸다.



3. 네트워크 시스템 변경(UNET -> Photon Cloud)

현재 프로젝트에 네트워크는 UNET을 이용하여 구축되어있는 상황이었다. 이전에 Delay 문제점으로 인해서 변경해야하는지 고민을 했었다. Photon을 이용하여 게임회사들이 많이 출시를 하였기 때문에 검증 된 네트워크 시스템이었다. 그래서 프로젝트의 크기가 더 커지기 전에 Photon으로 변경하기로 결정하였다. Photon SDK를 다운받아 프로젝트에 적용시키고 Photon 홈페이지에 프로젝트 등록을 하여 사용할 준비를 마쳤다. 다행이도 UNET과 크게 다르지 않고 한글 가이드가 제공되기 때문에 Photon API를 학습하는데 오랜 시간이 걸리지 않을 것이라고 생각된다. 그리고 UNET보다는 Photon이 더 많은 정보가 인터넷에 많이 있었다. 한국에서도 이 Cloud를 이용하여 출시된 게임이 많이 있어서 한글정보 또한 UNET보다는 비교적 많이 있었다.

	4. 느낀점 이번 주는 데이터베이스 구현에 많은 시간을 사용하였다. 설계한대로 데이터베이스를 구현하면 될 것이라고 생각하였다. 하지만 이를 구현하는 것이 쉽지만은 않았다. 구현을 하며 오류도 많이 발생하였고 제대로 데이터가 저장되는지 디버그를 하는 시간이 상당히 오래 소요되었다. 그리고 DataStoreCoroutine을 설계하고 구현하는데 많은 어려움이 있었다. 백엔드 시스템의 구조가 콜백 함수를 사용하여 서버와 통신하기 때문에 IEnumerable 콜렉션 형태로 함수들을 구현해야했다. 구글링을 통해 자료를 많이 찾아보고 공부를 하였다. 이번 주는 C#의 기능 중 고급기능을 사용하여 코딩을 하였고 코루틴의 개념을 확실히 알고 넘어갈 수 있었다. 개발을 하며 배운 것이 가장 많은 한 주였다. 간단한 게임을 구현하는데 이러한 기능을 사용할 경우가 거의 없지만 비교적 큰 프로젝트를 진행하면서 이런 고급기능을 사용하니 개발 실력이 많이 향상되었다고 느낀다.
참고자료 및 문헌	- 절대강좌! 유니티 - 구글링 - https://doc-api.photonengine.com/ko-kr/pun/current/index.html (Photon 가이드)
학습방법	구글링을 통해 C#의 고급 기법을 익히고 절대강좌! 유니티 문헌을 통하여 UI에 대해 학습하고 적용시킨다. 변경된 네트워크를 프로젝트에 적용시키기 위해 Photon 가이드를 참조한다.
학습성과 및 목표달성도	- 게임 데이터베이스 구현 완료(100%) - 타이틀 씬 UI 기능 구현 - 네트워크 시스템 변경 완료 8주차의 가장 큰 목표인 데이터베이스 구현을 완료했다는 점에서 큰 의미가 있다. 앞으로 UI에 대한 학습이 많이 필요함을 느꼈으며 변경된 네트워크 API를 빨리 습득하여 서버와 클라이언트 부분의 속도를 더 높여야 할 것이다.
내주 계획	로그인과 데이터베이스 구축이 끝났으므로 변경된 네트워크로 Match Making 시스템을 개발하며 게임 전체 구성도를 실제 게임에 적용시키고 각 기능에 필요한 UI를 제작한다.

2018. 11. 05 .

지도교수 **(인)**