

# 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. 12 ~ 2018. 11. 18	학습주차	10	학습시간	57
창의과제	네트워크를 통해 상대방과의 실감나는 대전형 슈팅 3D 모바일게임 개발				

## 금주 학습목표

- 게임 네트워크 시스템 적용
- 게임 씬(Scene) UI 구현
- GameManager 구현

## 학습내용

### 1. 게임 네트워크 시스템 적용

이전 UNET으로 작성한 네트워크를 Photon을 이용하여 변경해야 한다. Photon과 UNET의 네트워크 적용방식에 대해서는 큰 차이점이 없지만 상속 관계 또는 통신을 위한 과정의 차이점이 존재한다. 최대한 적은양의 데이터를 전송하는 과정을 구현함으로써 통신량을 줄이는 로직을 개발하여야 했다. 유니티에서 오브젝트의 움직임을 처리할 때 float형 변수 9개를 이용하여 transform, rotation, scale를 처리한다. 하지만 이 9개의 데이터를 모두 전송할 필요가 없기 때문에 네트워크로 연결되어있는 상대방이 내 캐릭터를 보기 위해서는 transform과 rotation의 정보가 필요하다. transform의 경우 플레이어 프리팹에 Photon View를 적용시켜 자동으로 Photon에서 처리하는 방법이 존재하기 때문에 별도로 구현할 필요는 없다. 하지만 rotation의 경우 float형 변수 3개를 넘겨줄 필요가 없었기 때문에 short형 사이즈가 3인 배열을 캐릭터의 회전을 처리한 후 데이터를 저장하는 변수를 선언하였다. 이 변수를 가지고 PunRPC라는 애트리뷰트를 사용한 함수에 매개변수로 넘겨주어 처리를 하도록 하였다. float형 3개를 넘겼을 때 12바이트를 사용하지만 short형을 사용함으로써 6바이트를 사용하므로 6바이트의 데이터를 절약할 수 있는 구조를 구현하여 사용하였다.

```
// 마우스 버튼 클릭시 동적(Dash) 발동 시 공격 불가능
if (enableAttack == true && PlayerPhysics.IsDash == false)
{
    //현재 클라이언트 위치 및 캐릭터 회전을 함께 전송하여 상 위치 동기화
    //또한 추가 대역폭을 절약하기 위해 짧은 배열(x, z - Skip y만 해당)으로 전송
    short[] pos = new short[] { (short)throwPos.position.x, (short)throwPos.position.z };
    this.photonView.RPC("QadThrow", PhotonTargets.AllViaServer, pos, playerRotation);
}

enableAttackIn = true; // 애니메이션 가능
enableAttack = false; // 공격 불가능
}
else
{
    // 공격 대기시간
    attackDelayTime += Time.deltaTime;
    if (attackTime <= attackDelayTime)
    {
        enableAttack = true; // 공격 가능
        attackDelayTime = 0.0f; // 공격 대기시간 초기화
    }
}

// 단지 서버에서 호출되지만 모든 클라이언트로 전달
[PunRPC]
protected void QadThrow(short[] position, short angle)
{
    //이 اسک립트 종류 여러개 일때 이 인덱스로 اسک립트 가져옴
    //int currentDecreamBit = GetView().GetDecreamBit();

    // 전송된 상 위치와 현재 서버 위치 사이의 중심 계산(오인 0.5f ~ 40% 클라이언트, 60% 서버)
    // 이 작업은 네트워크 지연을 보상하고 클라이언트/서버 위치 간에 원활하게 수행하기 위함
    Vector3 throwCenter = Vector3.Lerp(throwPos.position, new Vector3(position[0], throwPos.position.y, position[1]), 0.5f);
    Quaternion syncRot = playerTransform.rotation * Quaternion.Euler(0.0f, angle, 0.0f);

    // 물체를 사용하여 اسک립트 볼 스트림
    GameObject go = new GameObject();
    go.AddComponent<QadThrow>().Throw(throwCenter, syncRot);
    Destroy(go, 2.0f);
}

//이 메서드는 클라이언트 속성이 네트워크에서 변경 될 때마다 호출
public override void OnPhotonPlayerPropertiesChanged(Dictionary<string, object> props)
{
    //이 플레이어의 속성 변경에만 반응
    PhotonPlayer player = playerAndLbdataProps[0] as PhotonPlayer;
    if (player != photonView.owner)
        return;

    // 시각화를 위해 현재까지 변경 될 수 있는 값을 업데이트
    OnHealthChange(GetView().GetHealth());
}

//이 메서드는 총알 여러 번 호출(적어도 10 회 이상)
void OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)
{
    if (stream.isWriting)
    {
        // 여기서 캐릭터 회전 각도를 다른 클라이언트에 전송
        stream.SendNext(playerRotation);
    }
    else
    {
        // 여기에서는 다른 사람으로부터 캐릭터 회전 각을 받고 전송
        this.playerRotation = (short)stream.ReceiveNext();
        OnPlayerRotation();
    }
}

//이 메서드는 플레이어 물리 및 애니메이션 함수
```

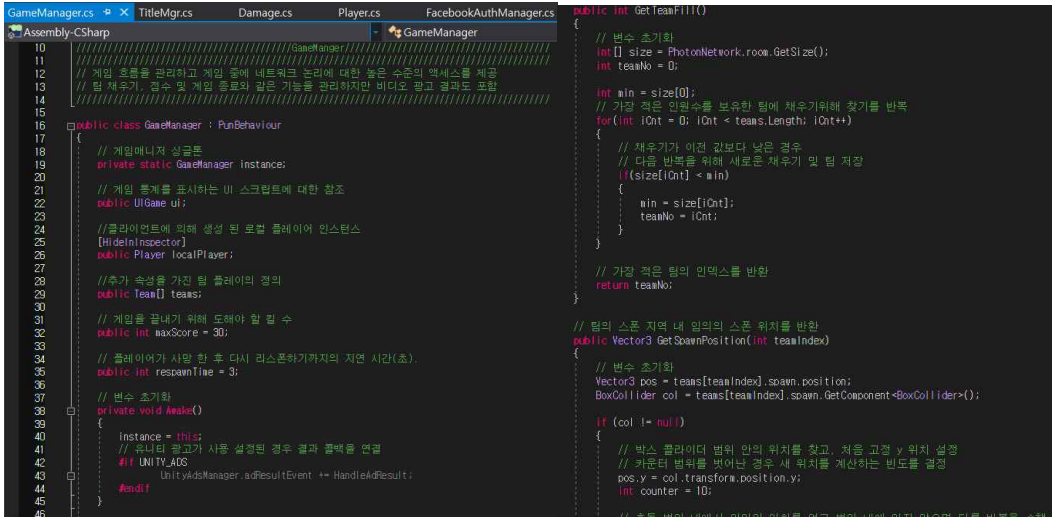
## 2. 게임 씬(Scene) UI 구현

게임에서 공통 데이터를 플레이어들에게 표시해주기 위해 UI를 제작해야한다. 리소스를 구입하지 않은 상태여서 기본 색상, 캔버스, 버튼 등을 사용하여 구현하였다. 게임에서 필요한 UI는 모바일 기기에서 캐릭터를 조작하기 위한 조이스틱, 팀의 인원수를 표시해주는 UI, 팀 점수, 자신의 킬 / 데스 수, 조준점, 게임 종료 메뉴, 플레이어가 상대방에게 공격을 받아 죽었을 때 리스폰을 위한 시간 등 많은 작업이 필요하였다. 조이스틱을 제외하고는 유니티에서 제공하는 UI를 가지고 제작하는 작업이라서 코드를 구현하고 게임 사이즈에 맞게 패널들을 조작하는 작업만 하면 되기 때문에 단순 마우스로 조정하는 시간이 소요되었다. 하지만 조이스틱을 구현할 때는 고려해야할 부분이 많이 있었다. 사용자가 조이스틱을 가지고 이동하는 경우와 조이스틱을 조작하는 동그란 버튼(이하 엄지)이 있는데 엄지가 움직일 수 있는 범위, 엄지를 드래그 시작했고 언제 드래그를 그만하였는지를 구현해야하는데 고려해야할 부분이 너무 많이 있었다. 움직일 수 있는 범위나 조작을 할 때 각을 이용하여 계산을 해야 하기 때문에 수학적 계산을 많았다. 앞으로 기능적으로 추가할 것은 공격이나 점프 버튼을 추가하는 작업이 남아있고 디자인적 추가할 것은 리소스 구입을 통해 UI를 꾸미는 작업이 남아있다.



## 3. GameManager 구현

대전에 시작되었을 때 모든 플레이어들이 공통으로 사용되는 데이터가 존재한다. 게임 시간, 킬 카운트, 팀 정보, 리스폰 시간 등의 데이터를 모든 사용자들이 공유하여 사용해야한다. 이 부분은 매니저를 생성하여 플레이어들에게 전달해야한다. 네트워크가 적용된 싱글톤 패턴을 사용해야하므로 이전에 생성한 싱글톤 클래스를 사용하지 않고 포톤의 PunBehavior를 상속받아 인스턴스를 생성하여 다시 제작하였다. 제작한 게임 씬(Scene) UI를 전체 게임 데이터를 관리할 수 있도록 스크립트를 포함시켜 내용을 작성하였다. 구글링을 통해 오픈소스를 찾아 많은 프로그래머들이 게임매니저를 설계한 방법을 보고 참고를 하여 구현하였더니 효율적인 설계를 할 수 있었고 많은 매니저를 구현했던 경험이 있어서 구현하는데 어려움이나 시간을 줄일 수 있었다. 싱글플레이 또한 이 매니저를 이용하여 게임을 관리하면 되기 때문에 앞으로 게임을 구현하는데 있어서 문제가 없다고 판단된다.

	
<b>참고자료 및 문헌</b>	<ul style="list-style-type: none"> <li>- 절대강작! 유니티</li> <li>- 구글링</li> <li>- <a href="https://doc-api.photonengine.com/ko-kr/pun/current/index.html">https://doc-api.photonengine.com/ko-kr/pun/current/index.html</a> (Photon 가이드)</li> </ul>
<b>학습방법</b>	<p>절대강작! 유니티 문헌을 통하여 UI의 패널 제작 방법을 학습하여 적용시킨다. 구글링을 통하여 오픈소스를 찾아 게임 매니저를 효율적으로 설계하고 구현할 수 있는 방법들을 찾았으며 네트워크를 효율적으로 설계하기 위해 Photon 가이드를 이용하여 네트워크로 전달하는 세부적인 요소를 학습하고 커스텀으로 직접 구현한다.</p>
<b>학습성과 및 목표달성도</b>	<ul style="list-style-type: none"> <li>- 게임 네트워크 시스템 구현 (60%)</li> <li>- GameManager 구현 완료 (100%)</li> <li>- 게임 씬(Scene) UI 구현 (80%)</li> </ul>
<b>내주 계획</b>	<p>11월 30일에 진행되는 세종 ICT 창업 아이디어 경진대회에 참가를 위한 사업계획서 및 발표 자료를 작성해야한다. 1차 서류평가에 통과한 후 메일로 멘토를 배정받았기 때문에 멘토분과 만나 사업계획서와 발표 자료를 지도받는다.</p>

2018. 11. 19 .

지도교수

(인)