3D 슈팅 멀티 PC 게임

오동호[O] 강성준 이원찬 이보경
한국공학대학교 컴퓨터공학과
{pipity03[O], kevin9809, getchan0119, bklee}@tukorea.ac..kr

3D Shooting Multi PC Game

Dong-Ho Oh[0] Seong-Jun Kang Won-Chan Lee Bo-Kyung Lee
Dept. of Computer Engineering, Tech University of Korea

요 약

대작이라 불리는 트리플 A 타이틀 게임은 너무나 많은 콘텐츠로 부담이 되고 있고, 간편하게 게임을 즐기고 싶은 라이트 유저가 즐길 수 있는 슈팅 게임이 수요가 증가하고 있다. 본 논문에서는 유니티와 포톤을 기반으로 멀티 플레이로 유저 간 협력을 할 수 있고, 스테이지 구성으로 짧은 플레이 타임으로 간편하게 즐길 수 있다. 포톤 클라우드 서버를 이용하여 개발 유지 비용을 줄이고, 저사양 컴퓨터에도 즐길 수 있도록 하였다. 이로써 멀티 플레이 기반으로 협동 전술을 구사할 수 있는 3D 멀티 슈팅 게임을 개발하여 라이트 유저도 간편하게 즐길 수 있는 게임을 제공한다.

1. 서론

“트리플 A 타이틀”이라고 부르는 게임은 대작이라 불릴 만큼 수많은 컨텐츠와 즐거움을 제공해준다. 하지만 대작을 만들기 위해선 개발 비용이 천문학적인 제작비가 들어간다. 성공이 보장되지 않는 게임에 개발 비용을 투자하는 것은 수지타산에 맞지 않게 보일 수 있다. 계속 이어지는 흐름으로 인해 출시되는 대작들은 성공을 위해 너무나 많은 콘텐츠를 제공하기 때문에 간단히 즐길 라이트 유저들이 대상으로 잡지 않고 숙달된 헤비 유저들을 대상으로 개발한다. 라이트 유저들이 즐기기엔 적합하지 않다고 판단하여, 남녀노소 간편하게 즐길 수 있는 게임을 개발하고자 한다. 본 프로젝트는 라이트 유저들을 대상으로 잡은 슈팅 게임을 제공한다.

2. 관련 연구

2.1 기존 유사 게임과의 비교

항목	기존 사례 특징	차별 및 개선점
그래픽	카툰 렌더링의 그래픽 스타일을 이용	카툰 렌더링 및 실사 그래픽을 **혼합하여 사용**
플레이어 수	1인 싱글 플레이	**1~4인 멀티 환경을 제공** 협동 및 재미 제공
물리 효과	(관련 내용 없음)	**물리적인 파괴 기능 추가** 시각적인 재미 추가 전략 및 전술 활용

표 1. 유사 게임과의 비교

위의 표 1에서는 라이트 유저를 대상으로 하는 비교군 게임을 보면 1시간이란 짧은 플레이 타임과 1인 플레이를 중심으로 개발된 속도감 있고 화려한 전투를 즐길 수 있는 게임이다. 본 프로젝트에서 개발하고자 하는 ‘3D 슈팅 멀티 PC 게임 구현’은 다른 게임들과 비교하였을 때, 카툰 렌더링과 실사 그래픽을 융합된 그래픽을 제공하는 점, 1인~4인 멀티 플레이가 제공되는 점, 물리 효과를 부여하여 스테이지가 동적으로 변하여 상황에 따라 전략 전술을 바꿔서 사용할 수 있는 차이가 있었다.

3. 세부 설계 및 구현

3.1 개발환경

본 논문의 3D 슈팅 멀티 게임은 윈도우 10 및 11에서 개발되었으며, **유티니(Unity) 2021.3.1.f1 [1]과 포톤(Photon) [2]**을 이용하였다. 라이트 유저를 위해 저사양 컴퓨터도 충분히 즐길 수 있도록 염두에 두어 저사양 컴퓨터 및 일반적인 사양의 컴퓨터로 테스트를 진행하였다.

3.2 게임 진행 시나리오

유니티 엔진을 기반으로 동작하는 게임이며, 포톤을 이용해 서로 다른 클라이언트를 클라우드 서버인 포톤이 제어한다. 유니티와 3Ds Max2021를 이용하여 스테이지, 플레이어블 캐릭터의 모델, 적 캐릭터의 모델, 무기 모델 및 애니메이션을 제작하고 유니티를 이용해 인게임에서의 플레이어, 적, 스테이지, 이동 및 상호작용을 제어할 수 있다. 서버는 포톤(Photon)을 이용하여 클라우드 서버를 생성하고 Pun2을 이용하여 로비의 구성 및 멀티플레이어 매칭, 인게임에서의 동기화를 하였다. 포톤 서버는 유저의 닉네임 외에는 정보를 수집하지 않으며, 방을 생성하여 원하는 방에서 게임을 즐길 수 있다. 인게임에선 지하 주차장 스테이지에서 시작하여 사무 빌딩 스테이지, 옥상 스테이지로 총 3가지 스테이지를 주파한다. 유저는 적 AI를 상대한다. 옥상 스테이지에서는 보스전을 진행한다. 이때 플레이어들은 포톤(Photon) 서버를 통해 정보

를 주고받으며 실시간으로 상호작용 및 협력할 수 있다. 또한 스테이지에선 엄폐물과 파괴 효과가 있는 사물이 있다. 이를 이용하여 지형지물을 이용한 공략적인 게임이 가능하다. 마지막 스테이지까지 보스 공략을 한 경우 클리어와 함께 정산 화면이 출력되며 게임은 타이틀 화면으로 돌아간다. 모든 진행 내용은 아래 그림 1을 통하여 게임 진행 시나리오를 정리하였다.

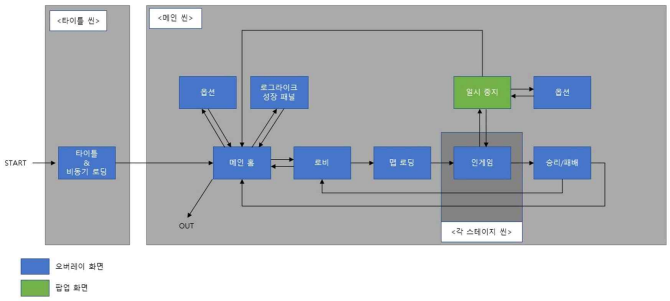

그림 1. 게임 진행 시나리오

3.3 핵심 시스템

3.3.1 스테이지 구성

모든 스테이지는 지하 주차장, 사무 빌딩, 옥상 스테이지로 구분된다. 스테이지 내에는 안전 구역과 전투 구역으로 나뉘며, 안전 구역은 적 AI가 접근과 공격이 불가한 지역이다. 전투 구역 스테이지를 통과하여 새로운 안전 구역으로 진입하는 경우 스트리밍 레벨 작업을 진행하여 메모리 관리와 최적화를 진행하였다. 이는 지하 주차장에서 사무 빌딩 스테이지를 넘어갈 때, 사무 빌딩에서 옥상 스테이지로 이동할 때 **스트리밍 레벨 작업[3]**이 진행된다. 또한 지하 주차장 스테이지 전투 구역 내에 배치된 기둥을 향해 마련된 무기를 발사하면 중력에 의해 무너지도록 구현이 됐다. 기둥에 총알의 착탄이 확인되면 파괴 효과가 일어난 후 기둥 조각의 물리 컴포넌트가 활성화되어 중력에 의해 떨어진다. 플레이어는 총으로 기둥을 부수는 등 사격 행위를 통해 실제 총기를 다루는 느낌을 받을 수 있다.

3.3.2 무기 시스템

게임 내에서 플레이어는 3가지 종류의 총기를 사용할 수 있다. 각 총기는 기획단계에서 실제 총기 에서 영감을 얻어 만들었다. 게임 내 총기는 권총, 샷건, 라이플이 존재한다. 각 총기는 서로 다른 탄퍼짐과 **반동[6]**, 유효 사거리를 구현했다. 또한 이러한 총기의 특징에 맞추어 서로다른 파괴력를 총기들이 가지고 있다. 권총의 예시를 들면 적은 반동과 탄퍼짐이 존재하여 빠른 속도로 가까이 다가온 적을 제압할 수 있으며 하지만 적은 탄창과 유효사거리를 가지고 있어 멀리 있는 적을 제압하기 힘들게 설계하였다. 반대로 라이플의 경우 넓은 유효 사거리를 가지고 권총보다 강한 반동으로 사용자가 먼 거리의 적을 상대할 수 있지만 총기의 반동으로 인해 조준점을 적에게 맞추기 어렵게 설계하였다. **그림 2**에서 실제 총기의 탄퍼짐을 자세히 볼 수 있다.

총기의 재장전 방식[4] 또한 다양한 방식으로 구현되어있다. 재장전의 애니메이션 모션은 펌프식 장전과 자동식 장전 방식이 있다. 펌프식은 샷건에 장전 애니메이션에 적용이 되어 있고 라이플과 권총은 자동식 장전방식이 적용되어 있다. 또한 재장전도중 위급상황시 현제 총기를 바꾸어서 재장전을 취소할 수 있다.

총을 사용한 데미지 전달에도 다양한 작용이 있다. 총에서 발사 되는 총알 오브젝트는 다양한 방식으로 상호작용한다. 적에게 총알이 접촉하면 피격 이벤트가 발생하여 적의 HP가 줄어든다. 또한 이 이벤트를 같은 매칭 사용자에게 모두 알림으로써 모든 사용자가 적의 HP가 감소하도록 설계되었다. 이와 반대로 적이 플레이어를 공격하면 플레이어의 HP가 감소한다. **그림 3**은 이 이벤트가 발생하면 실행되는 함수이다.

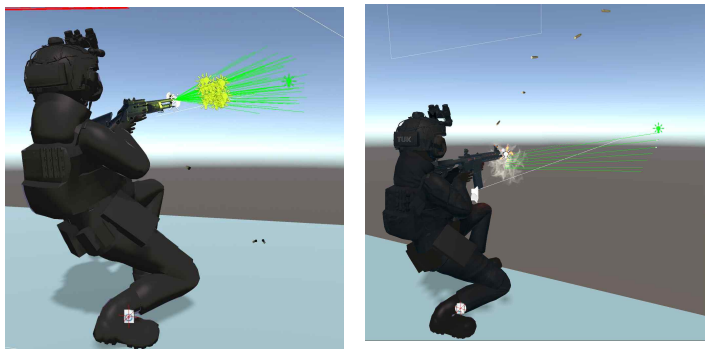

그림 2. 총기별 탄퍼짐

```
7 references
public override void TakeDamage(DamageMessage _damageMessage, HitParts _hitPart)
{
    base.TakeDamage(_damageMessage, _hitPart);

    base.photonView.RPC("GetDamge",RpcTarget.All,this.curHealth);
    PhotonNetwork.IsMessageQueueRunning = false;
    PhotonNetwork.IsMessageQueueRunning = true;
    // 마지막 공격자를 저장
    var player = _damageMessage.attacker.GetComponent<PlayerController>();
    if (player != null)
    {
        lastAttacker = player.ID;

        // 플레이어 정보가 없으면 플레이어 저장
        if (players[lastAttacker - 1] == null) players[lastAttacker - 1] = _damageMessage.attacker;
    }
}
```

그림 3. 피격시 실행되는 함수

그 외에 사용자가 물체 오브젝트와 상호작용하기도 한다. 사용자가 발사한 총알이 특정 오브젝트와 만나면 일어나는 이벤트이다. 총알이 건물내 벽이나 기둥에 만나면 벽이 금이 가거나 부서지는 이벤트가 있다. 총알이 특정 벽과 만나서 **그림 4**처럼 벽에 물체를 파괴할 수 있다.

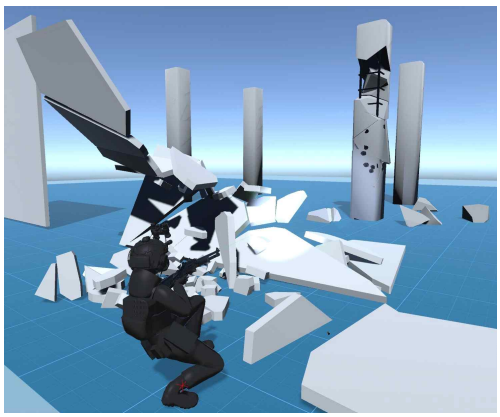

그림 4. 벽과 기둥을 파괴하는 모습

3.3.2 적 AI 시스템

적AI는 플레이어를 감지하고 자신만의 체력을 가지고 있으며 또한 정해진 리스폰 구역이 존재한다. 적 AI는 머리, 팔, 몸통, 다리로 자신

만의 부위가 존재하고 각각의 부위마다 서로 다른 피격 데미지를 가진다.

AI는 정해진 구역에서 일정 시간마다 리스폰되여 플레이어에게 공격을 가한다. 플레이어가 그 구역을 지나가면 리스폰은 활성화가 되지 않고 꺼지며 또한 적 AI가 리스폰 될 때마다 일정 개체수 이상 리스폰이 되지 못하도록 제한을 주어서 무한으로 증식하는 것을 막았다. 그리고 **레그돌 상태 [6]**가 되면 바닥에 쓰러지고 **그림 5처**럼 캐릭터들이 쌓여서 죽게 된다. 이 오브젝트들은 해당 스테이지가 끝나면 사라지며 또한 자신의 스테이지 밖으로는 나가지 못하도록 설계되어있다.

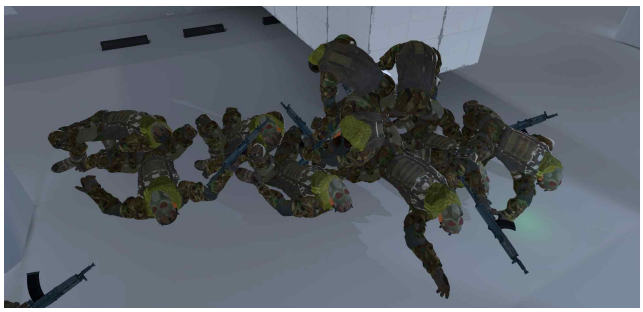

그림 5. AI 오브젝트 레그돌

마지막으로 적AI가 플레이어를 감지하는 부분이다. 플레이어를 유니티의 레이저 포인트 기능을 활용하여 범위에 플레이어가 존재하면 이를 바라보는 방향으로 전진한다. 그리고 플레이어를 추적하여 공격 사거리 안에 플레이어가 존재하면 공격하고 아니면 플레이어를 계속 추적하면서 따라간다. 플레이어를 공격할 때 위험한 정도에 따라 우선순위가 결정된다. AI가 공격당한 대상과 피격당한 데미지를 계산하여 공격할 플레이어 대상을 선정한다.

3.3.2 플레이어 캐릭터 시스템

플레이어 캐릭터는 이동 시스템과 카메라 시스템이 있다. 이동 시스템의 경우 기본적인 8방향에서 이동이 가능하며 마우스에 따라서 플레이어의 화면이 변화된다. 또한 달리기, 걷기, 뛰기에 따라서 캐릭터 애니메이션 모션과 이동속도가 다르게 되어있다. 그 외에 **그림 6**처럼 계단이나 벽을 만나면 카메라가 벽을 감지하여 영향을 받지 않는 위치로 이동하며 경로상에 경사나 계단이 존재할시 이동속도 감소와 애니메이션 변환을 구현했다. 마지막으로 총을 줌인 하여 사격할시 카메라가 총으로 줌인 되어서 플레이어가 AI를 조준하기 쉽게 하도록 했다.

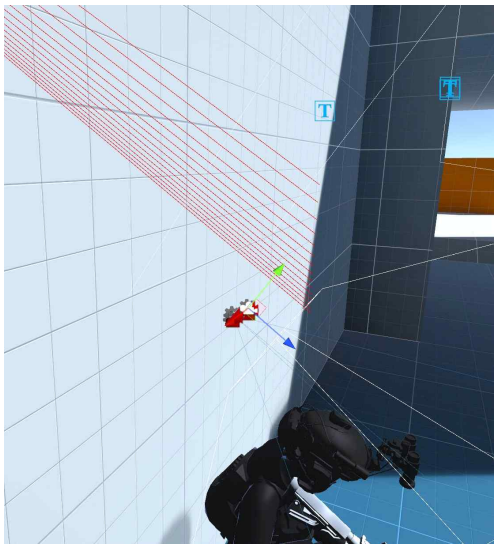

그림 6 카메라의 벽 감지 후 자동 전환

3.3.2 네트워크 시스템

네트워크는 크게 게임 매치메이킹 시스템과 게임 내 동기화가 있다. Photon 클라우드 서버를 이용하여 게임 내의 네트워크를 구현했다. 게임 매치메이킹은 Photon SDK의 API를 활용하면 게임룸 생성, 채팅, 게임룸 실행을 구현하였다. 그리고 게임 내에서 핵심적인 동기화들은 **Photn의 RPC와 RaiseEvent[7]**를 사용하여 구현하였다. 적AI의 대한 위치, Hp, 상태, 에니메이션을 동기화 했다. 사용가는 이벤트를 게임 내에서 발생시키면 이러한 정보가 다른 사용자들에게 전송되여 발생된다.

그림 7 게임 내 멀티 플레이

4. 결론 및 향후 연구과제

유니티를 이용하여 최대 4인까지 플레이가 가능한 TPS슈팅게임을 개발하였고, 품질과 성능 검증을 위해 1인 게임 플레이와 멀티 게임플레이를 진행하였다.

플레이어, 적AI,스테이지 관리, 보스의 기능 및 소프트웨어의 성능을 테스트하였고, FPS(Frame Per Second)부문에서 평균 60fps를 만족하지 못하는 결과를 확인하였다. 문제를 확인하기 위해서 유니티 프로파일러를 통해 cpu에서 gpu로의 병목현상을 발견하였고 batch 수를 줄이고, Built-in Renderer Pipeline에서 **Universal Rendering Pipeline[8]**으로 렌더러를 변경하여 대부분의 상황에서 120fps이상을 유지하게 되었다.

또한 건물 기둥을 향해 무기를 발사하면 중력에 의해 무너지도록 구현이 되어있으며, 총알의 착탄이 확인되면 기둥조각의 물리 컴포넌트가 활성화되어 중력에 의해 떨어지는 방식이다. 그러나 라이트맵을 굽는 과정에서 제외되어 정적인 배경과 동화되지 못하며, 조각의 수가 많을수록 연산량에 큰 부담을 주게되므로, 미리 물리 연산을 하여 애니메이션을 저장 및 플레이하거나 강조된 이펙트를 통해 파괴된 부분이 삭제되는 것을 가리는 등 자연스러운 연출을 위하여 더 연구가 필요한 부분이다.

멀티 게임플레이에서도 개선 사항이 존재했다. 클라우드 서비스를 이용한 서버 개발을 진행해서 예상보다 동기화적인 부분이 문제가 많았다. 특히 클라우드에서 PhotonRPC와 PhotnRaiseEvent를 활용한 게임 내 동기화를 진행했는데 RPC추상화의 문제가 발생했다. 특히 게임내 플레이어 간의 공정성 문제와 RPC의 파라미터에서 문제가 있다. 추후 이러한 부분은 플레이어간의 타이밍을 같은 속도로 맞추는것이나 파라미터 객체를 쪼개는 연구가 더 필요해 보인다.

참고문헌

[1] 유니티 공식 사이트 https://unity.com/kr
[2] 포톤 공식 사이트 https://www.photonengine.com/ko-KR/

[3] 스트리밍 레벨 기능 구현하기 | 유니티 . (2020). https://www.youtube.com/watch?v=MeNOdL2mg18
[4] [유니티 3D게임] FPS Prototype #16 무기 교체 시스템 . (2021).https://www.youtube.com/watch?v=Petq5lS68gw
[5] 이런 캐릭터 물리 효과는 어떻게 만들까? 래돌과 캐릭터 조인트 | 유니티 . (2020). https://www.youtube.com/watch?v=cTHceZpwGt4.
[6] Procedural Recoil System | Unity Tutorial . (2021). https://www.youtube.com/watch?v=geieixA4Mqc.
[7] RPCs 와 RaiseEvent https://doc.photonengine.com/ko-kr/pun/current/gameplay/rpcsandraiseevent
[8] Universal Rendering Pipeline https://docs.unity3d.com/kr/2020.3/Manual/universal-render-pipeline.html