The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 23, No. 6, pp.109-117, Dec. 31, 2023. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2023.23.6.109
JIIBC 2023-6-17

랜드마크 기반 체험형 메타버스 증강현실 게임

Augmented Reality Game of Experiential Metaverse based on Landmark

염민규*, 이수민*, 박영훈*, 한경숙**

Min-gyu Yeom*, Su-min Lee*, Young-hoon Park*, Kyung-sook Han**

요 약 최근 들어 현실 세계에 새로운 가치를 창출하는 메타버스에 대한 관심이 높아졌다. 그런 메타버스를 쉽게 체험할 수 있게끔 기존 수집형 AR 게임 형태에서 벗어나서 랜드마크를 기반으로 사용자가 주도적으로 환경을 탐사하고 콘텐츠를 즐기는 체험형 메타버스 AR 게임을 개발하였다. 메타버스의 중요한 특징으로는 현실세계와의 연결성, 상호작용성, 디지털 통화가 있다. 해당 게임은 AR을 통하여 일상의 연장을 추구하고, 실시간 채팅 및 팀 경쟁을 통하여 상호작용성을 만족하려 한다. 마지막으로 상점 시스템을 통하여 디지털 통화를 구축한다. 모바일 게임으로 구현하여 스마트폰만 있다면 언제든 접속할 수 있게 하여 접근성을 높였다.

Abstract Recently, interest in metaverse, which creates new value in the real world, has increased. In order to make it easy to experience such a metaverse, we have developed an experiential metaverse AR game in which users take the lead in exploring the environment and enjoying content based on landmarks. Important features of the metaverse include connectivity with the real world, interactivity, and digital currency. The game seeks to extend daily life through AR and satisfies interaction through real-time chat and team competition. Finally, digital currency is built through a store system. It is implemented as a mobile game and can be accessed at any time if there is a smartphone, increasing accessibility.

Key Words : GPS, Augmented Reality, Metaverse, Landmarks, Game

Ⅰ. 서 론

최근 온라인 기반 사람 간의 네트워킹에 있어 현실적 감정을 강조하는 메타버스에 대한 관심이 높아지고 있다. 메타버스는 3D 기술을 활용하여 가상의 세계에 아바타, 시설 등을 제공하고, 개인화된 아바타를 활용하여 온라인에서 상호 교류 활동을 하도록 돕는 기술로서 비대면 시대에 사회적 연계성과 즐거움을 모두 잡는 기술로 인식되고 있다. 실제로 전 세계 메타버스 시장은 연평균 43% 이상 성장할 것으로 기대되며, 2028년에는 약 829억 달러에 이를 것으로 예측된다. [1]

메타버스와 같은 가상환경을 구축하기 위해서는 HMD(Head Mounted Display)가 필수적이지만 이러한 기기들이 아직까지 가격, 부피, 호환성 등의 부분에서

*준회원, 한국공학대학교 컴퓨터공학부
**정회원, 한국공학대학교 컴퓨터공학부 (교신저자)
접수일자 2023년 9월 18일, 수정완료 2023년 11월 18일
게재확정일자 2023년 12월 8일

Received: 18 September, 2023 / Revised: 18 November, 2023 / Accepted: 8 December, 2023
*Corresponding Author: khan@tukorea.ac.kr
Dept. of Computer Engineering, Tech University Of Korea

보급화가 이루어졌다고 보기는 어렵다. 그러나 반대로 스마트폰 보급률은 굉장히 높으므로 스마트폰으로 체험이 가능한 메타버스 모바일 게임을 개발하였다.

이 게임은 다음과 같은 세 가지 기술적 특징을 가지고 있다. 첫째로 GPS 지도와 카메라 기반 AR을 이용한다. 사용자 위치를 실시간으로 반영하는 지도를 사용하는 필드는 실제 게임의 무대가 되고 무대 내에서 즐기는 컨텐츠는 AR로 즐기며 생동감 있는 게임 플레이가 가능해진다. 둘째로 상호간 소통과 이동을 독려한다. 미션을 수행하여 점수를 올리고 해당 점수로 상대 팀과 포인트 대결을 하는 것은 실제 이동과 소통을 필요로 한다. 셋째로 온라인이라는 점이다. 네트워크로 연결되어 있고 플레이어는 팀에 따라 상대 팀의 목표와 반대되는 목표를 가지는 경쟁지향적인 게임이다.

II. 관련 연구

상용화 되었던 AR 게임으로는 엠게임의 〈캐치몬〉[2]과 한빛소프트의 〈소울캐처〉[3]가 존재한다. 〈캐치몬〉은 수집과 보드게임 요소를 특징으로 하는 게임이나 수집 욕구를 이루어내지 못했고, 보드게임을 제대로 즐기기 위해서는 오랜 시간 수집 활동을 해야 한다는 점 때문에 큰 인기를 얻지 못하였다. 〈소울캐처〉는 GPS 서비스를 이용해 유적지와 관광지, 지역 축제 현장과 같은 명소에서 역사적인 영웅을 수집하는 AR 모바일 게임이다. 유적지와 관광지를 게임에 포함시켜 역사적 지식을 쌓을 수 있으나 Niantic, Inc.의 〈포켓몬 GO〉와 완전히 똑같은 콘텐츠를 가져 플레이어들의 이목을 끌지 못하였다.

정리하자면 현재 시장에 분포한 대다수의 AR 게임이 선두주자인 〈포켓몬 GO〉와 비슷한 수준으로 머물러 있고, 이러한 틀에서 벗어나 AR 게임 시장의 다양성을 위한 새로운 AR 게임 콘텐츠의 필요성을 느꼈다.

III. 본 론

1. 개발환경

본 논문의 게임은 윈도우 10 환경에서 유니티 엔진을 이용해 개발되었으며 GPS 기반 맵은 Mapbox SDK[4]를 사용하였고, AR 게임은 AR foundation 2.1.8과 AR Core xr plugin 2.1.8을 사용하여 구현하였다. 서버는 뒤끝 API[5]를 통해 개발하였다. 테스트에 사용한 단말기는 SM-N975(Andriod 12), SM-N971N(Android 12), SM-F711N(Android 13), SM-X700(Android 13)이다. 파편화 되어 있는 안드로이드 스마트폰의 다양한 환경을 염두에 두어 해상도와 하드웨어 성능, 운영체제가 조금씩 다른 4개의 모델을 선택, 테스트를 진행하였다.

2. 시스템 구성도

이 게임의 기본적인 동작방식은 서로 다른 클라이언트를 서버가 중계해주는 방식이다. 그림 1는 게임 시스템 구성도를 요약한 그림이다. 서버에는 유저 데이터와 랜드마크 데이터 등의 정보를 가진 DB가 포함되어 있으며 클라이언트는 서버를 통해 게임 타이머, 점수, 미니게임 위치 등을 주고받는다. 클라이언트는 ARCore[6]와 Unity[7]를 통해 구현하였고 서버는 뒤끝 API를 통해 구현하였다.

그림 1. 시스템 구성도
Fig. 1. System Configuration Chart

3. 구현

가. GPS 맵 구현

그림 2과 같은 GPS를 기반으로 한 맵을 구현하기 위해 GameManager 클래스, LoadingSceneManager 클래스, GPSManager 클래스를 작성하였다.

그림 3은 메인화면을 구현하기 위해 작성된 클래스들의 구성도이다. GameManager 클래스는 현재 플레이어 객체를 생성한다. 싱글톤 패턴에 사용할 Awake 함수는 'DontDestroyOnLoad'를 호출하여 싱글톤 개체가 장면 간에 유지되도록 한다. 이렇게 하면 새 장면(scene)이 로드될 때 싱글톤 오브젝트가 소멸하지 않으며, 이는 전역 상태를 유지하거나 게임 자원을 효율적으로 관리할 수 있다.

그림 2. GPS 기반 메인화면
Fig. 2. GPS-based main screen

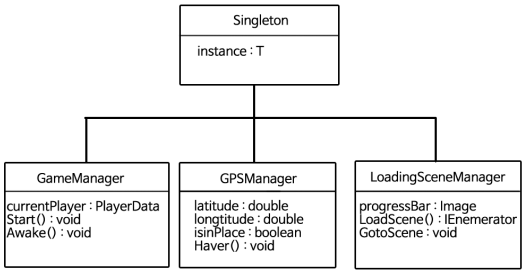

그림 3. 메인화면 시스템 구성도
Fig. 3. Main Screen System Configuration Chart

LoadingSceneManager 클래스는 로딩 화면을 관리하는 클래스이다. 코루틴은 Mathf를 사용하여 로딩 게이지를 채운다. 장면 로드 작업의 진행률을 기반으로 0.9 미만이면 그에 따라 채우기 양이 업데이트되고, 0.9 이상이면 채우기 양이 1.0으로 설정되어 장면 활성화가 허용된다. 장면이 완전히 로드되면 코루틴이 종료된다.

그림 4. 랜드마크 이벤트 예시
Fig. 4. Examples of landmark events

GPSManager 클래스는 사용자의 스마트폰에서 위치 정보를 수신받는 클래스이다. 사용자의 위도와 경도를 받아 저장한다. 해당 변수는 랜드마크 오브젝트와의 거리 계산과 사용자의 현재 위치를 파악하는 데에 사용된다. 위도와 경도로 랜드마크 오브젝트와의 거리를 Haver 메소드를 통해 계산하여 충분히 가까운 거리라면 그림 4와 같은 미니게임을 즐길 수 있는 랜드마크 이벤트가 활성화되는 방식이다. 주변에 있다면 isInPlace = true, 주변에 위치하지 않는다면 isInPlace = false로 랜드마크 이벤트를 활성화할지 처리한다.

Haver 클래스는 하버사인 거리 계산을 하는 함수이다. 두 지점 사이의 거리를 계산하기 위해 사용한다. 지구의 구조는 완벽한 구가 아니어서 하버사인 공식을 사용하여 계산할 경우 0.3%(100km당, 300m)의 오차가 발생한다.

해당 오차를 해결하기 위한 더욱 정밀한 공식으로 빈센티 공식(Vincenty's formula)이 있으나 수식의 복잡도와 추가로 필요한 각도계산이슈, 그리고 모바일 기기의 연산처리 문제로 배제하였다.[8]

나. 미니게임 5종 구현

(1) RC카

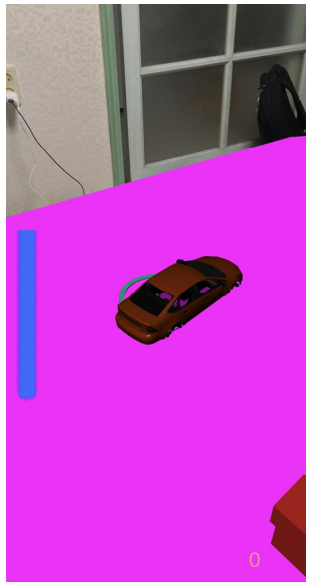

그림 5. RC카 AR 게임 화면
Fig. 5. RC Car AR-Game Screen

그림 5와 같이 증강현실 기술을 기반으로 자동차 오브젝트를 조종해 선물박스 오브젝트와 충돌시켜 점수를 얻

는 게임을 구현하기 위해 CarManager 클래스, DrivingSurfaceManager 클래스, CarBehaviour 클래스, PackageManager 클래스, ReticleBehaviour 클래스를 작성하였다.

CarManager 클래스는 크로스헤어가 찍힌 위치를 터치할 시, 자동차 오브젝트를 배치하는 클래스이다. Update 함수를 통해 자동차 오브젝트가 존재하지 않고, 현재 지면이 인식된 상태라면 크로스헤어가 존재하는 위치에 자동차 오브젝트를 설치하게 된다. 그리고 현재 지면을 고정시킨다.

지면 고정 기능은 DrivingSurfaceManager 클래스에서 담당하고 있다. 해당 클래스의 LockPlane 함수는 현재 지면 정보를 가져와 만약 새로 인식된 지면이 현재 지면과 다르다면 새로 인식된 지면을 비활성화 한다. 이런 식으로 현재 지면의 크기만 늘어날 뿐 완전히 새로운 지면은 인식되지 않게 한다.

CarBehaviour 클래스는 자동차의 행동을 정의하는 스크립트 클래스이다. Update 함수에서 크로스헤어의 위치 값을 계속 추적한다. 이렇게 해서 distance 값이 0.1 이상일 경우 Quaternion 값을 조정하여 자동차 오브젝트가 크로스헤어의 방향으로 회전시키고 크로스헤어를 목표로 전진시킨다.

다음은 자동차가 선물박스 오브젝트에 부딪혀 점수를 얻는 로직 관리를 위한 PackageManager 클래스이다. 지면이 인식됐고 Package가 null 값이면 SpawnPackage 함수를 실행한다. SpawnPackage 함수는 점수 오브젝트를 생성하는데 생성 위치값은 FindRandomLocation 함수를 통해 계산한다. 여기서 랜덤 값은 삼각 난수 공식을 이용하였다.

다음은 ReticleBehaviour 클래스이다. 지면에 크로스헤어를 표시해주는 클래스이다. 카메라를 통해 지면이 인식됐다면 해당 정보를 변수에 저장한다. 이미 인식된 지면이 존재한다면 새로 인식된 지면과 trackableID를 비교하여, 같을 때에만 지면을 인식하고 다르다면 인식하지 않는다. 이런 식으로 지면 인식의 일관성을 부여하여 자동차가 지면 위를 이동하도록 한다.

(2) 풍선 맞추기

그림 6과 같이 카메라 시야각 안에서 풍선 오브젝트가 무작위 생성되고 풍선 오브젝트를 크로스헤어에 맞춰 Shoot 버튼을 누르면 풍선이 파괴되고 점수를 얻는 AR 게임을 구현하기 위해 ShootBalloons 클래스와 SpawnBalloons 클래스를 작성하였다.

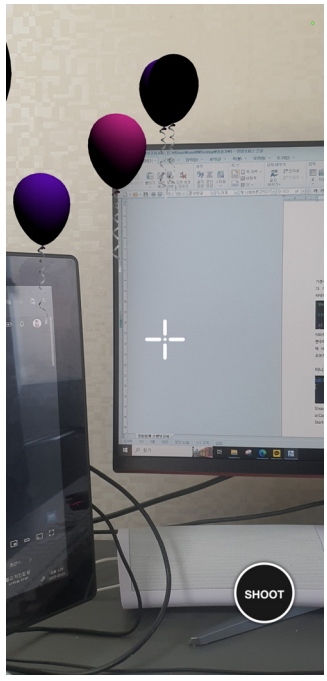

그림 6. 풍선 맞추기 AR 게임 화면
Fig. 6. Balloon hitting AR-game screen

ShootBalloons 클래스는 풍선을 쏘고 파괴시키는 데에 사용된다. arCamera, Smoke, point의 세 개의 변수를 선언되었고, Start 함수와 Shoot 함수를 선언하였다. Start 함수에서는 ScoreCounter.score 변수를 0으로 설정한다. Shoot 함수는 플레이어가 스마트폰 화면을 터치할 때 호출되며, arCamera 객체의 위치에서 마주보는 방향으로 광선을 쏜다. 광선을 통해 Balloon 태그를 가진 게임 오브젝트와 충돌할 경우, 풍선을 파괴하고 point 변수를 증가시킨다.

SpawnBalloons 클래스는 게임 내의 풍선의 위치를 제어하는 기능을 한다. 클래스 내의 StartSpawning 함수는 For문을 사용하여 Instantiate()를 호출하여 각기 다른 세 개의 풍선을 생성한다. For문은 풍선 배열을 순회하여 세 가지 풍선 범위 중 하나에서 무작위로 생성할 풍선을 선택한다. For문이 끝나면 연속적으로 풍선을 생성하기 위해 코루틴 StartSpawning()이 다시 호출된다.

GetRandomSpawnPosition 함수는 AR 카메라의 위치와 회전을 먼저 얻은 후 정의된 범위 내에서 각도와 거리를 무작위로 선택하여 이 범위 내에서 임의의 위치에 풍선 오브젝트를 생성한다. 카메라의 FOV(시야)와 위치를 사용하여 풍선이 무작위 생성된다. 이 클래스를 통해 풍선 오브젝트는 플레이어의 카메라 시야각 내에서만

생성되며 게임의 몰입감을 한층 높여준다.

(3) 공 던지기

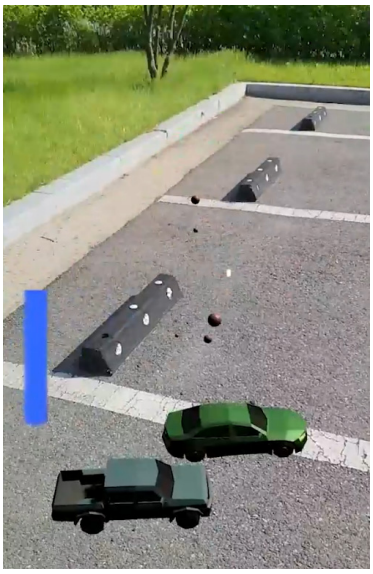

그림 7. 공 던지기 AR 게임 화면
Fig. 7. Ball Throwing AR-Game Screen

그림 7과 같이 카메라를 통해 인식된 지면 위에 무작위적으로 자동차 오브젝트가 생성되고 해당 오브젝트를 터치하면 발사되는 공으로 맞춰 파괴시키고 점수를 얻는 AR 게임을 구현하기 위해 ARPlacement 클래스, Shoot 클래스, Explode 클래스를 작성하였다.

ARPlacement 클래스는 공을 맞출 대상을 지면 위에 설치하는 클래스이다. 우선 Raycast를 통해 지면을 인식하고 플레이어가 터치를 했다면 타겟이 되는 오브젝트를 설치한다. 여기서 UpdatePose 함수를 통해 현재 추적하는 지면 위치가 유효한지 확인하여 유효하다면 PlacementPose 변수에 해당 지면의 값을 저장한다. 만약 지면은 인식했지만 생성된 타겟 오브젝트가 없다면 UpdateIndicator를 통해 지면 인식용 크로스헤어를 생성한다.

Shoot 클래스는 사용자가 화면을 터치했을 때 총알 오브젝트가 발사되게끔 하는 클래스이다. 사용자가 터치를 하는 순간 카메라의 정중앙에 총알 오브젝트를 생성한다. 그리고 AddForce를 통해 총알을 발사한다.

다음은 해당 게임의 핵심이 되는 Explode 클래스이다. 해당 클래스는 총알 프리팹과 충돌하는 객체가 타겟 오브젝트가 맞다면 파괴하고 파괴될 때마다 타겟 오브젝트를 무작위 위치에 생성하는 클래스이다. OnCollisionEnter를 통해 만약 총알 프리팹과 충돌한 객체가 타겟 오브젝트라면 파괴하고 점수를 얻는다. KillPos 위치 변수에 AddRandomOffset 함수를 통해 랜덤한 위치 값을 도출해낸다. 그리고 SpawnEnemyAgain 함수를 통해 랜덤한 타겟을 해당 위치 값에 생성한다. 여기서 무작위 위치 값은 특정 영역 범위를 벗어나선 안 된다. AR 게임의 특성상 너무 멀리 떨어져있는 오브젝트는 보기 힘들기 때문이다. 그래서 범위 내에서 랜덤한 위치를 가진 벡터값을 구한 후 서로의 벡터값을 더하는 방식을 사용하였다. 해당 코드는 설정된 범위 내에서 랜덤한 위치의 Vector3 값을 반환하는 코드이다. 우선 rangeObject의 현재 오브젝트(여기선 플레이어가 파괴시킨 오브젝트이다.)의 위치 값을 가진 Vector 변수인 originPosition를 선언한다. 그리고 좌표 방향의 x과 z 위치 값 변수를 할당한다. 그리고 Random.Range() 함수를 이용해 현재 위치 값을 반으로 나눈 값에 -1을 곱한 값부터 곱하지 않은 값이 랜덤으로 나오게 하였고 이를 이용해 새로운 Vector3 변수인 RandomPosition을 선언하였다. 이로써 특정 범위 내에 타겟 오브젝트가 랜덤하게 생성되는 것을 볼 수 있다.

(4) 음식 먹기

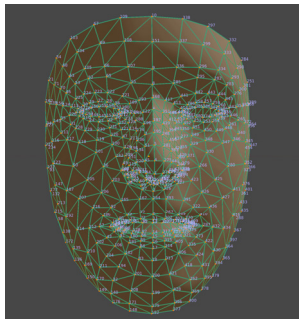

그림 8. AR Face의 인식 범위
Fig. 8. AR Face's range of recognition

기본적으로 해당 게임은 ARCore에서 제공하는 AR Face Api[9]를 사용한다. 그림 8에서 보듯이 해당 Api는 사람의 얼굴을 468개의 점으로 인식하여 얼굴을 감지해 얼굴 프리팹을 렌더링한다.

해당 기술을 통하여 그림 9과 같이 랜덤하게 떨어지는 음식 오브젝트에 입을 가져다대면 점수를 얻는 게임을 설계하였다. 카메라 위에서부터 음식 오브젝트가 랜덤하게 내려오는 동작을 구현하기 위해 Spawner 클래스, Movement 클래스를 작성하였다.

그림 9. 음식 먹기 AR 게임 화면
Fig. 9. Eating AR-Game Screen

Spawner 클래스는 화면 위에서 랜덤하게 내려오는 음식 오브젝트의 동작에 관한 클래스이다. Awake 함수에서는 spawnTimer와 generatedSeconds가 랜덤 값을 받아 결정된다. 이후 Update 함수에서 랜덤 값을 가지는 generatedSeconds가 spawnTimer보다 작거나 같다면 스스로 파괴되는 음식 오브젝트를 생성한다. SpawnTimer는 deltaTime을 이용하여 현실 시간동안 1씩 더해진다. 다음은 음식 오브젝트의 크기를 결정하는 Scale 클래스이다. 여기서는 Mathf.PingPong을 사용할 것인데 이것은 어떤 값이 최대값에 도달하면 그 뒤부터는 0까지 줄어드는 값이다. 값이 0이 되면 다시 최대값까지 늘어난다.

Movement 클래스는 음식 오브젝트의 움직임을 결정한다. 음식 오브젝트의 transform.position에 deltaTime값을 계속 곱해줌으로서 시간이 지날수록 빠르게 떨어지도록 설계하였다. 또한 OnTriggerEnter를 추가하여 만약 충돌한 대상이 ARFace를 통해 인식된 얼굴 오브젝트라면 점수를 1점 올리고 해당 음식 오브젝트는 파괴한다.

(5) 농구

그림 10. 농구 AR 게임 화면
Fig. 10. Basketball AR-Game Screen

그림 10과 같이 골대 오브젝트를 인식된 지면 위에 설치하고, 공 오브젝트를 투척해 골대 오브젝트와 충돌하면 점수를 얻는 AR 게임을 구현하기 위해 BallController 클래스와 manager 클래스를 작성하였다.

BallController 클래스를 간단히 정리하면 사용자가 화면에 손가락을 터치한 상태에서 아래에서 위쪽으로 드래그하면 손가락이 이동한 거리에 비례하여 Ball 오브젝트를 전방 위쪽 45도 각도로 발사한다. 만일 골대 오브젝트의 Collider 영역과 충돌했을 시 점수를 얻는다. 물체와의 충돌이 감지되지 않는다면 3초 후에 공이 제자리로 돌아와 다시 던질 수 있게 된다.

SetBallPosition은 공의 위치를 항상 사용자 카메라의 전방 0.5m, 하단 0.2m 지점에 위치시키는 함수이다. 여기서 중요한 것은 공이 날아가는 도중에는 공의 위치를 카메라 앞에 고정시키면 안 되므로 isReady 변수로 현재 공의 상태를 확인한다.

Update 함수에선 공을 발사할 힘을 화면과 터치된 채로 밀어낸 간격만큼으로 계산하여 발사한다. 단, 농구 게임 특성상 직선이 아닌 정면 위쪽 45도 방향 포물선으로 날아가야 하므로 카메라의 forward 방향과 up 방향을 더해 설정한다. OnCollisionEnter 함수를 통해 충돌을 감지하여 점수를 얻고 공을 초기화한다.

다음은 manager 클래스이다. 해당 클래스를 통해 사용자가 직접 골대의 위치를 정하여 설치한다. AR 공간에

서의 상호작용을 위해 Raycast를 이용하였다.

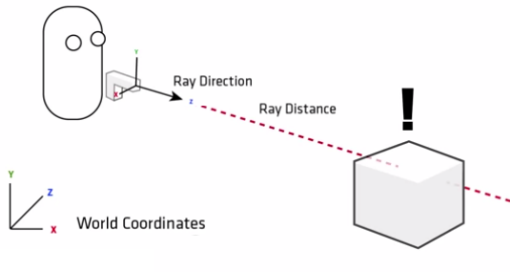

그림 11. Raycast 동작 과정
Fig. 11. Raycast Operation Process

그림 11처럼 기본적으로 Raycast는 화면상에서는 보이지 않는 직선 광선을 쏘아서 오브젝트와 충돌하는지 체크한다. 이 과정에서 Plane 타입 오브젝트를 추적하는데 여기서 Plane 타입은 카메라를 통해 인식한 바닥면을 의미한다. 다시 말해 Raycast는 실제 바닥면을 인식하는 광선을 쏘는 과정이다. hitInfos 변수에 Raycast의 충돌 정보를 저장한다. Raycast 과정을 거치면 앞서 생성한 hitInfos 변수에 충돌 결과가 유니티 월드 좌표 형태로 저장되고 해당 위치가 바닥이라면 크로스헤어를 생성해 사용자가 설치할 위치를 미리 알 수 있다. 여기서 터치를 하는 순간 그 위치에 농구 골대 오브젝트를 생성되며 게임이 시작된다.

다. 매칭 시스템 구현

그림 12에서 보여주는 시퀀스 다이어그램을 바탕으로 한 매칭 시스템을 구현하기 위해, 클라이언트 쪽에서는 MatchingManager 클래스를 구현하였고, 서버와의 통신을 위한 API로는 뒤끝 API를 사용하였다. Matching Manager 클래스에는 서버와의 통신 과정에 필요한 이벤트 핸들러 모음인 MatchMakingHandler, 매칭의 각 과정을 수행할 Init, GetMatchInfo, JoinMatchMakingServer, MatchingProcess, JoinGame의 메소드와, 매칭 이후 인게임 상황에서 지속적인 데이터 조율을 위한 Update, IncreaseTeamScore 메소드로 구성된다.

클라이언트에서는 로그인과 동시에 매칭 서버와의 연결을 확인하고, 매칭을 위한 기본적인 Indate값을 확보한다. 이후 사용자가 매칭을 요청할 경우, 기존에 확보된 Indate값을 활용해 MatchingProcess로 서버와 통신을 시도한다. 충분한 매칭 인원이 모였을 때, 서버는 매칭을 시도한 클라이언트의 정보를 각 클라이언트에 공유한 뒤, 게임 시작 트리거를 발생시킨다. 각 클라이언트는 서버에서 공유받은 클라이언트의 정보를 저장하고, 게임 시작 트리거에 반응하여 인게임 장면으로 이동하게 된다. 10초 동안 충분한 인원이 모이지 않았을 경우, 현재 매칭된 인원만으로 위와 같은 프로세스를 수행한다.

매칭 이후, 인게임 상황에서 서버는 특정 클라이언트의 채팅, 점수 변동, 게임 종료 트리거에 반응하고, 이를 전 클라이언트에게 브로드캐스팅 하는 역할을 수행한다. 클라이언트의 MatchingManager는 서버측에 메시지를 바이트배열로 인코딩 하여 전송하거나, 서버에서 받아온 바이트배열을 디코딩 한 뒤 클라이언트의 데이터를 업데이트하는 역할을 수행한다.

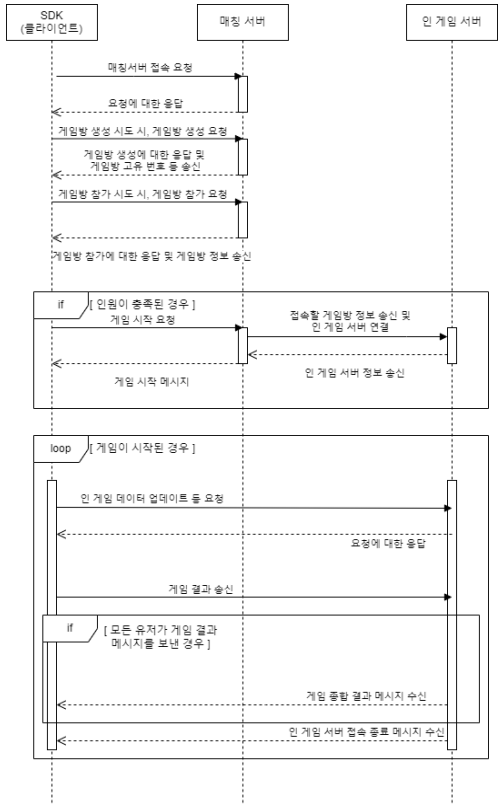

그림 12. 매칭 시스템 시퀀스 다이어그램
Fig. 12. Matching System Sequence Diagram

게임 종료 트리거의 발생 시, 각 클라이언트의 MatchingManager는 현재까지 서버와 통신한 데이터를 기반으로 승패를 결정하고, 이를 서버에게 통지한다. 서버는 각 클라이언트에서 통지받은 데이터를 검증하고, 그림 13와 같은 게임 종료 화면을 각 사용자들에게 보여주고, 이를 랭킹 시스템에 등재한다. 일련의 과정이 끝나게 되면 서버와 클라이언트의 연결을 끊어진다.

그림 13. 게임 종료 화면
Fig. 13. Game Ending Screen

라. 랭킹 시스템 구현

랭킹 시스템을 구현하기 위해 클라이언트 쪽에서는 RankSceneManager 클래스를 구현하였고, 서버와의 통신을 위한 API로는 뒤끝 API를 사용하였다. 각 미니게임 수행 후, 각 플레이어가 얻은 점수가 하루 동안 획득한 점수 중, 최고점수라면 서버에 랭킹을 새로 갱신되도록 하였고, 서버로부터 랭킹 정보를 가져오는 기능 또한 구성하였다. 위의 두 기능들은 MinigameDataManager 클래스에 기능을 추가로 구현한다. RankSceneManager 클래스는 서버에 등록된 각 랭킹 정보들을 Json 파일 형태로 받아와 화면에 띄운다.

랭킹 장면에서는 총 6개의 랭킹 정보가 있다. 누적 획득 경험치를 기준으로 랭킹을 확인할 수 있는 종합 랭킹, 각 미니게임의 랭킹을 확인할 수 있는 자동차 게임 랭킹, 슈팅 게임 랭킹, 농구 게임 랭킹, 음식 먹기 랭킹, 사격 랭킹이다.

마. 상점 구현

상점 시스템을 구현하기 위해 ShopSceneManager 클래스를 구현하였고, 서버와의 통신을 위한 API로는 뒤끝 API를 사용하였다. 그리고 상점에서 판매하는 아이템들의 정보를 담기 위한 ItemSource 클래스를 선언하였다. 각 아이템들의 목록 정보는 뒤끝 콘솔에 csv 파일 형태로 업로드 하였으며, 아이템을 추가하게 될 경우, 뒤끝 콘솔에서 추가 가능 및 기존 아이템 정보 수정 또한 뒤끝 콘솔에서 조작이 가능하다. 위와 같이 세팅함으로써 뒤끝 콘솔로부터 아이템 정보들을 Json 파일 형태로 받아와, 생성한 ItemSource 클래스의 객체에 각 아이템의 고유 정보를 매핑시킬 수 있도록 로직을 구성하였다.

각 아이템들은 '구매' 버튼과 '설명' 버튼이 존재한다. '구매' 버튼은 해당 아이템을 구매하여 해당 플레이어가 해당 아이템을 여러 개 구매할 수 있다면 보유 개수가 1 증가하고, 한 개만 구매할 수 있다면 해당 플레이어가 구매하여 보유 중인 아이템인지 확인 후, 구매하지 않은 아이템이라면 구매 가능, 구매했던 아이템이라면 구매할 수 없도록 알고리즘을 구성하였다. 추가로 아이템을 성공적으로 구매했다면 플레이어가 소지한 게임 머니를 일정 차감하고, 만약 게임 머니가 부족하다면 구매가 불가능하다. 또한 각 아이템들에 대한 가격 및 설명은 '설명' 버튼을 클릭하여 확인할 수 있도록 화면을 구성하였다. 해당 기능들을 제공하는 메소드는 BuyBtnListener이다.

Ⅳ. 결 론

개발한 애플리케이션으로 이용자들에게 학교 부지 한정으로 메타버스 및 AR 기능을 체험할 수 있도록 하였고, 그 과정에서 재미 요소를 자극하기 위해 매칭 시스템을 통해 팀 대항 시스템 또한 즐길 수 있도록 하였다. 더 나아가 게임 플레이 과정에서 한국공학대학교 부지를 돌아다니도록 유도하여 캠퍼스를 탐방할 수 있도록 하였다. 이를 통해 한국공학대학교를 배경으로 여러 복합적인 재미 요소를 함유하고 있는 메타버스 개발에 대한 발전 가능성을 기대할 수 있다.

지금까지 구현된 것들 이외에도, iOS 기종에서의 작동을 위한 추가적인 모바일 빌드와 해당 과정에서 생길 수 있는 시행착오에 대한 고려, 그리고 UX 측면에서의 추가적인 고려가 필요하다. 또한 해당 애플리케이션을 이용하는 앱 이용자들에게 더 나은 재미 요소를 위해 추가 미니게임들을 고려해볼 수 있겠다. 부가적으로 상점 시스템에 대해 더 활발히 연구하고 개발하여 이용자들이 아이템 구매를 하고 싶게 유도할 수 있다면 수익 또한 기대할 수 있을 것이다.

References

[1] Inho-Hwang, "The Effects of Media Richness of Metaverse on Intention to Offer Support of User Through Presence and Flow", Journal of Korea

Academia-Industrial cooperation Society, Vol. 23, No. 7 pp. 192-205, 2022
DOI:http://doi.org/10.5762/KAIS.2022.23.7.192

[2] Hyuk-woo Nam. “Catchmon's official service. Is it beyond Pokemon GO?”. ZDNET Korea. March 30th, 2017

[3] Hee-kang Shin,“Hanbit Soft Launches 'Soulcatcher AR', an AR game using GPS, in Korea in Q1”. Aju Korea Daily. January 25th, 2017

[4] Mapbox SDK. "https://www.mapbox.com/".

[5] thebackend, thebackend SDK. "https://developer.thebackend.io/unity3d/main/".

[6] Google. ARCore. "https://developers.google.com/ar?hl=ko".

[7] Unity. Unity Engine. "https://unity.com/kr".

[8] Jae-hyun Choi, "Location-based area set up method and optimization technique for deviation detection", The Korea Contents Association, Vol. 14, No. 4, pp. 19-28. 2014.
DOI:https://doi.org/10.5392/JKCA.2014.14.04.019

[9] Google. Augmented face introduction. “https://developers.google.com/ar/develop/augmented-faces?hl=ko”.

저 자 소 개

염 민 규(준회원)

- 한국공학대학교 컴퓨터공학부 재학
- 관심분야 : 서버, 데이터베이스

이 수 민(준회원)

- 한국공학대학교 컴퓨터공학부 재학
- 관심분야 : 게임 프로그래밍, 서버

박 영 훈(준회원)

- 한국공학대학교 컴퓨터공학부 재학
- 관심분야 : 증강현실, 메타버스

한 경 숙(정회원)

- (학사) 홍익대학교 컴퓨터공학과
- (석사) 홍익대학교 전자계산학과
- (박사) 홍익대학교 전자계산학과
- 한국공학대학교 컴퓨터공학부 교수
- 관심분야 : 컴파일러 최적화, 프로그래밍 언어