2021학년도 1학기

AR/VR이해 및 응용

— AR Avoid Asteroid

이 보고서를 AR/VR이해 및 응용 개인 과제로 제출함.

2021년 6월

한림대학교 SW융합대학

빅데이터학과

20175308 김정웅

목 차

I. 서론

프로젝트 간단 설명

기존 목표 및 변경 점

II. 본론

1. 게임의 기본 알고리즘

2. 장애물 및 아이템의 능력

3. 시연 영상

4. 게임 씬 설명

4.1. Start Scene

4.2. Game Scene

4.3. End Scene

5. 핵심 코드 설명

5.1. 사용스크립트

5.2. 가상버튼의 이용

5.3. 이미지 타겟의 식별

5.4. 아이템 및 장애물 스폰

5.5. 아이템의 장착

5.6. 아이템의 사용

5.7. 데이터 저장

III. 결론

1. 프로젝트 진행 간 문제점
2. 느낀 점 및 배운 점

**I. 서론**

**프로젝트 간단 설명**

휴대폰으로만 조작하는 일차원 적인 게임에서 벗어나 AR마커를 이동시키고 가상버튼을 터치하여 플레이 하는 피하기 장르의 게임

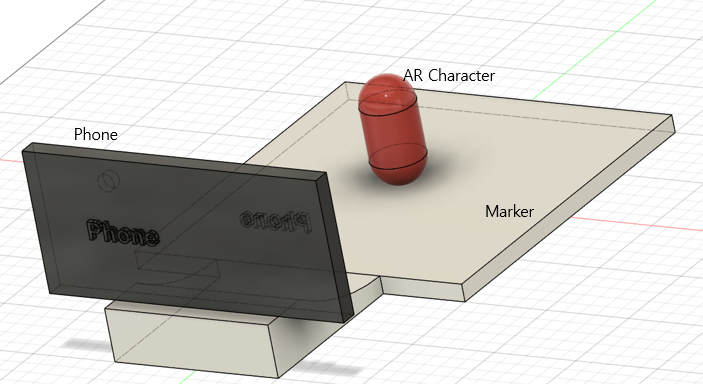
**AR마커**

어디에서나 쉽게 구할 수 있는 천원 지폐를 마커 이미지로 채택

텍스트이(가) 표시된 사진

자동 생성된 설명

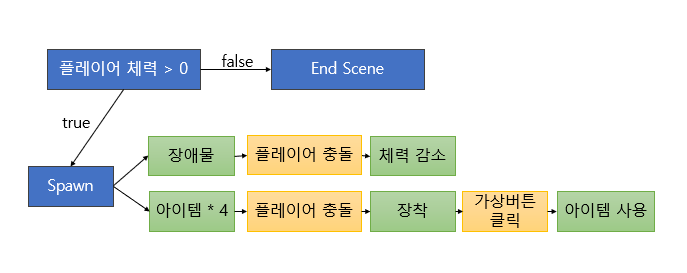
**기존 목표 및 변경 점**

 -> 

* 최초 AR을 이용해 위에서 생성되는 장애물 피하기 게임(똥피하기)을 구현 하려다 Virtual버튼 기능을 발견하고 이 기능을 추가해 아이템을 눌러 사용할 수 있도록 구현하였고, 위에서 장애물이 나오는게 아닌 플레이어의 Forward 방향에서 물체가 생성되어 피하는 비행기 조종형식의 게임으로 수정했다.

**II. 본론**

**1. 게임의 기본 알고리즘**

****

**플레이어의 체력이 0보다 작을 경우 End Scene으로 이동하고, 0보다 클 경우 장애물 및 아이템을 스폰한다. 장애물이 플레이어인 Ship과 충돌할 경우 Ship의 체력이 감소하고, 스폰 된 아이템이 Ship과 충돌할 경우 Ship의 아이템칸이 비어있는지 확인하고 장착된다. 장착된 아이템 칸의 가상버튼을 클릭할 경우 각 아이템에 맞는 능력이 사용된다. 장애물 및 아이템의 능력 및 스폰 확률은 다음과 같다.**

1. **장애물 및 아이템의 능력**

|  |  |  |  |
| --- | --- | --- | --- |
| 에셋 명 | 이미지 | 스폰확률 | 능력 |
| 장애물Asteroid | 어두운, 검은색, 동굴이(가) 표시된 사진  자동 생성된 설명 | 60% | 플레이어와 충돌 시 체력을 5 만큼 감소시킨다. |
| 아이템Bomb |  | 10% | 사용시 맵에 생성된 모든 장애물들이 제거된다. |
| 아이템  Heal |  | 10% | 사용시 체력을 10 만큼 회복한다. |
| 아이템Invincible | 어두운, 옅은이(가) 표시된 사진  자동 생성된 설명 | 10% | 사용시 4초간 무적상태가 됩니다. |
| 아이템Smaller | 철물, 나사, 옅은이(가) 표시된 사진  자동 생성된 설명 | 10% | 사용시 10초간 크기가 20%작아집니다. |

**3. 시연 영상**

<https://youtu.be/ySSeoStW51I>

\* 자막으로 설명을 대체합니다.

**4. 게임 씬 설명**

**4.1. Start Scene**

**** 마커가 인식되지 않으면 다음과 같은 화면 출력

**** 마커가 인식되면 마커 위 다음과 같은 화면 출력

**** 좌 우측의 버튼을 동시에 누르면 시작씬으로 이동

**4.2. Game Scene**

**** 플레이어의 앞방향에서 장애물 및 아이템 스폰후 플레이어 방향으로 이동



무적아이템 사용시 텍스쳐가 바뀌며(노란색) 체력 고정

 장애물과 충돌 할 경우 체력이 닳고 폭발 효과

****

체력이 0일때 큰 폭발과 텍스쳐 변경(검은색), 3초 후 EndScene으로 이동

**4.3. End Scene**



게임의 최고점수와 내 점수가 표시되고 화면을 길게 눌러 기록을 초기화 할 수 있고, 짧게 누르면 Start Scene으로 이동

**5. 핵심 코드 설명**

**5.1. 사용 스크립트**

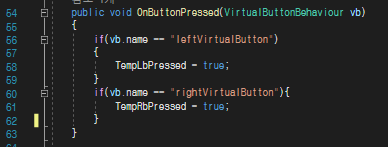
|  |  |  |
| --- | --- | --- |
| **스크립트 이름** | **연결된 오브젝트** | **역할** |
| **SpaceShip** | Ship | 체력, 메시지, Ship의 아이템 상태 확인 |
| Spawn | Ship | Item, Asteroid 생성 |
| Asteroid | Asteroids(Prefab) | 장애물의 이동 및 체력 감소 |
| **Items** | All Items(Prefab) | 아이템 장착, 아이템의 이동 |
| **AllSkill** | Ship | 아이템 스킬 사용 |
| vbButton | ImageTarget | 가상 버튼의 클릭 확인 |
| MydefaultTrack… | ImageTarget | 마커가 인식되었는지 확인 |
| StartGe | StartScene obj | 마커가 인식되었을 때 오브젝트 생성 |
| StartBtn | StartScene obj | 가상버튼의 클릭 확인 |
| EndScene | EndScene obj | 최종 점수 및 최고 점수 반환, 리셋 |

**5.2. 가상버튼의 이용 \*StartBtn.cs , vbButton.cs**

가상버튼은 ImageTarget의 인스펙터창에서 Add Virtual Button을 클릭해 추가할 수 있다.

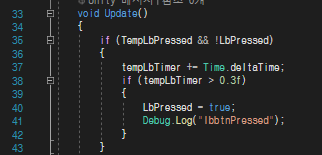
****

스크립트 상단에 Vuforia를 사용한다 선언한다.

****

버튼이 눌리면 다음 메소드가 동작한다 : OnButtonPressed

버튼에 대한 정보가 vb에 저장된다. 눌린 버튼을 판별한다.

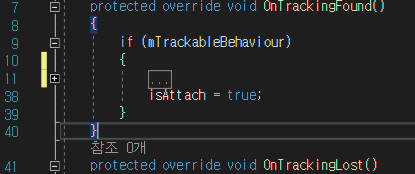


조명환경과 마커의 구겨짐에 따라 버튼이 쉽게 눌리기 때문에 0.3초 이상 눌렸을 때만 실제 눌린눌린 것 판단했다.

**5.3. 이미지 타겟의 식별 \*MyDefaultTrackableEventHandler.cs**

****

기존 ImageTarget에 붙어있는 DefaultTrackableEventHandler스크립트를 수정하려했는데 알수없는 이유로 프로그램이 종료되었다. 기존 스크립트를 오버라이드 한 새로운 스크립트를 작성해 이미지의 식별 상태를 확인했다.

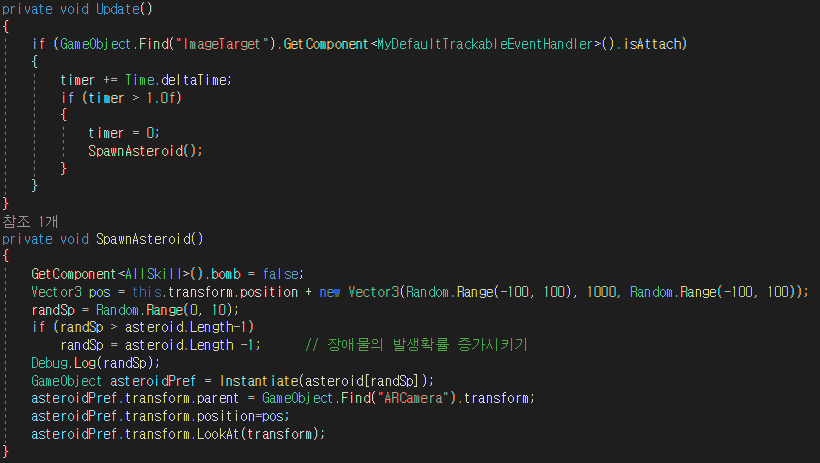


Tracking되었을 때 isAttach = true, Lost되었을 때 false

**5.4. 아이템 및 장애물 스폰 \*Spawn.cs**



인스펙터 창에서 asteroid에 총 5개의 프리팹을 연결한다.



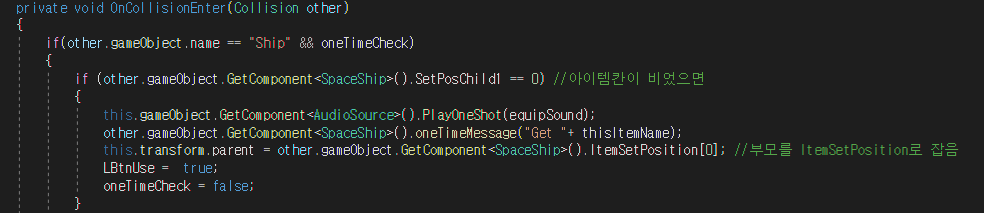
Tracking 되었을 때 (isAttach) 스폰하며, 1초마다 반복한다.

스폰 위치는 현재 플레이어의 위치의 y방향으로 1000만큼 떨어진 x,z (200\*200,200\*200)의 공간에서 생성된다.

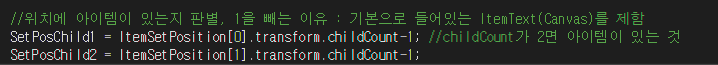
장애물의 발생확률을 증가시키기 위해 randSp엔 0~4가 아닌 0~9까지의 랜덤한 수가 들어가며 5~9가 나왔을 때 4(asteroid의 크기(5) -1)를 저장하고 Instantiate한다.

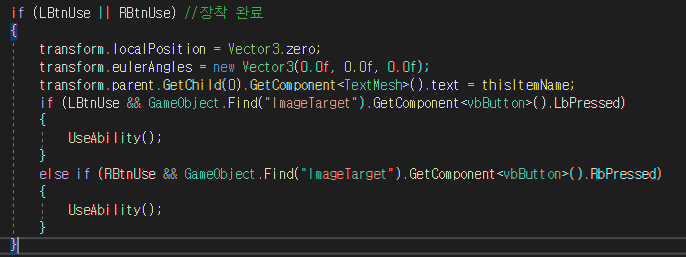
카메라와 같이 아이템도 이동하기 위해 아이템 객체의 부모를 AR카메라로 잡는다.

**5.5. 아이템의 장착 \*Items.cs**



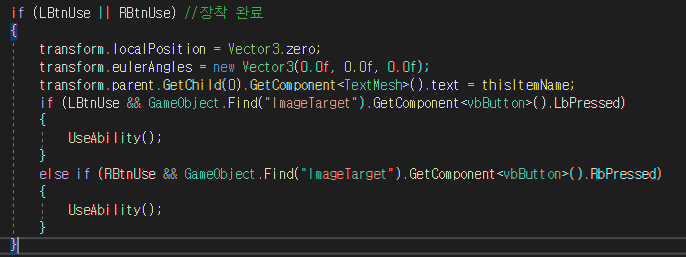
처음 플레이어 방향으로 날아오게 하기 위해 LookAt() 사용

SpaceShip : 

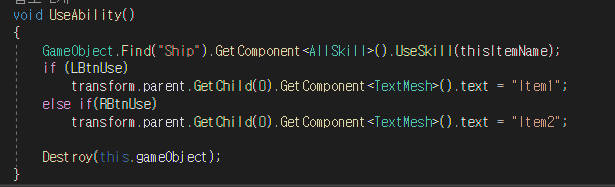
SpaceShip에서 아이템의 빈칸이 있으면 아이템 장착

장착 완료 시 포지션과 각도를 고정하고, 스킬창의 텍스트를 아이템 이름으로 지정

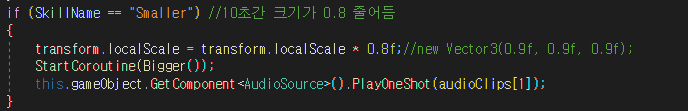
**5.6. 아이템의 사용 \*Items.cs, \*AllSkill.cs**



장착이 되었으며, 마커 인식중인 상태에서 해당되는 버튼 클릭시 아이템 사용

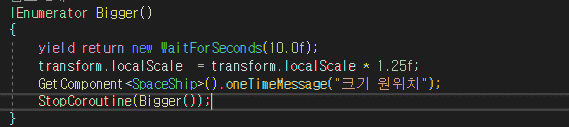


Ship의 AllSkill스크립트에서 전체 스킬 관여



<AllSkill.cs> SkillName이 Smaller일경우 다음 코드 실행

localScale을 0.8배로 조정하고 Bigger() 코루틴 실행



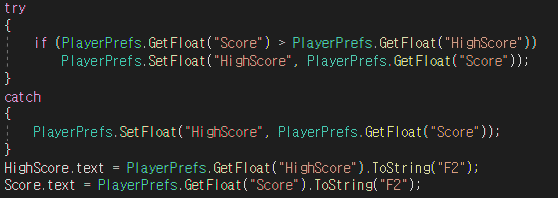
10초 후 1.25배 (0.8배의 역) 실행

**5.7. 데이터 저장 \*SpaceShip.cs, \*EndSceneGE.cs**



<SpaceShip>에서 죽었을 경우 PlayerPrefs SetFlot를 이용해 timer변수(게임시간)를 Score에

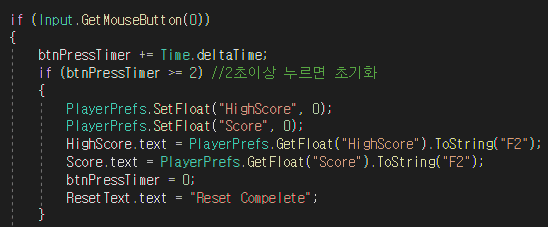
저장



<EndSceneGE>에서 try catch문으로 playerprefs GetFloat의 변수를 읽고, (highscore가 비었는

지 판별) 만약 받아온 Score값이 HighScore보다 더 크면 HighScore를 갱신한다. F2는 소수점

2번째 자리까지



길게 화면을 터치할 경우 HighScore와 Score을 0으로 초기화한다.

**III. 결론**

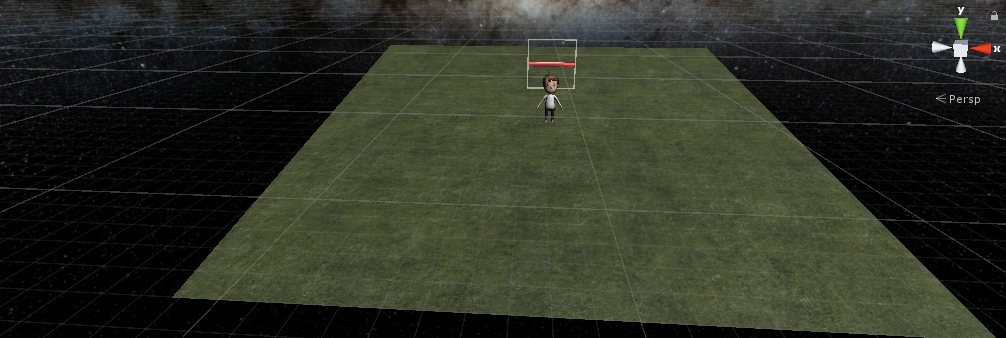
1. **프로젝트 진행간 문제점**

AR마커가 제대로 구분되는 이미지가 아니어서 이미지가 튀는 현상이 자주 발생했다.



* + 정상(왼쪽)과 이미지가 튀는 현상 발생(오른쪽)

가장 처음 제작하려 했던 AR똥피하기게임을 제작하며 캐릭터의 시선 방향이 지형의 기울어진 방향을 보게 만들려 했는데 기술적인 부족으로 실패했다. (이미지 타겟과 동일한 크기의 바닥에서 모든 이미지 타겟으로 Ray를 쏴 가장 먼저 맞는 위치를 캐릭터가 바라보기) -> up축으로 피하는게 아닌 forward축으로 피하기 게임으로 변경하며 필요가 없어졌다.



* + 처음 만들던 게임

Vuforia가 아닌 유니티 2020.2.7f버전에서 지원하는 AR이 더 편하고, 기능적인 면에서 좋아 보였는데 이미 제작한 부분이 많아서 아쉬웠다.

1. **느낀 점 및 배운 점**

처음 유니티를 시작할 땐 게임제작은 힘들 것이라 지레짐작하고 유니티를 어렵게 생각했지만 매 수업을 배울수록 게임제작이 다른 분야보다 쉽고, 재미있으며, 완성되어 가는 과정을 시각적으로 보는 것을 좋아하는 제 적성에 맞다고 느끼고 있습니다.

AR이란것에 생소했지만 수업 내용을 통해서, 프로젝트를 제작하면서 AR외에도 유니티의 기능 뿐만 아니라, 로직을 찾거나 생각하며 코딩으로도 많은 부분 배우고 성장할 수 있었습니다.

**감사합니다.**

* **마침 -**