게임엔진1 중간고사 과제

2017184002 구건모

목차

1. Assets 자료들 분석

유니티 에셋스토어에서 Tower Defense Template을 검색할 경우 처음으로 나오는 템플릿을 먼저 다운을 받았다.

import를 하여 내용물을 프로젝트로 옮기고 나오게 된 파일들을 살펴보면 애니메이션, 오디오, UI등 게임에 필요한 음성, 이미지 파일들이 있었으며 프리팹 스크립트 등도 폴더별로 잘 정리되어 있었다.

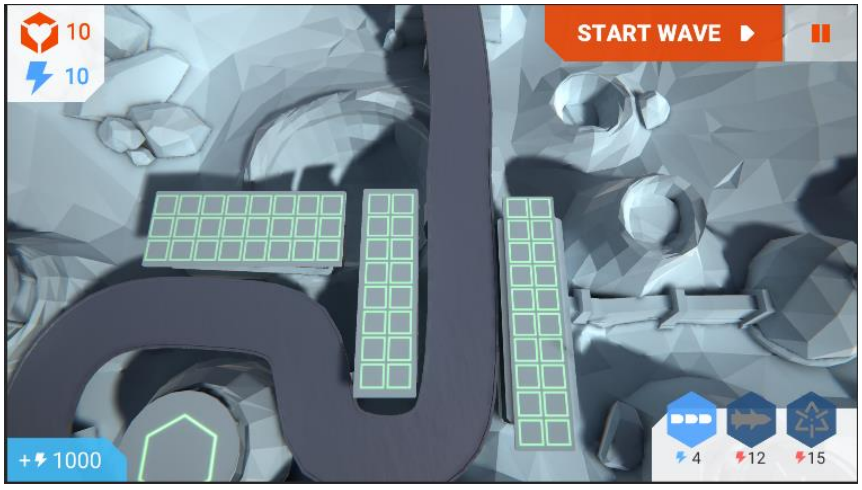
Data 폴더에 asset확장자로 타워와 같은 것들의 데이터가 따로 저장이 되어있다.

2. 게임 진행 분석

처음 게임을 시작하게 되면 3가지의 버튼이 존재하게 된다.

레벨을 선택하는 버튼, 옵션 버튼, 종료 버튼이 있다.

레벨 버튼을 누르게 될 경우 뒷 배경이 움직이면서 5개의 스테이지를 고를수 있게 되어있고 옵션을 누를경우 볼륨 조절과 제작자 사이트 접근이 가능하다. 마지막 종료 버튼을 누르게 될 경우 놀랍게도 아무런 일이 일어나지 않았다.



스테이지를 선택하여 게임을 시작하게 되면 (옵션에서 소리를 전부 줄였지만 스테이지 선택시 소리가 나는 버그가 있었다.) 위 이미지와 같은 화면이 나오게 된다.

좌측상단에는 목숨(라이프)와 타워를 설치할 때 필요한 자원이 표시되어 있고 좌측 하단에는 치트(자원 +1000)증가 버튼이 있다. 우측 상단에는 시작과 일시정지, 우측 하단에는 타워를 설치 할 수 있는 버튼과 소모 자원이 적혀있다.

일정량 이상의 자원을 보유하고 있으면 타워를 끌어다 네모 격자 4칸이나 6각형 격자에 설치가 가능하다.

타워의 종류는 3가지의 종류로 쉽게 소총, 캐논, 레이저로 구분이 되어 있으며 각 타워마다 공격 방법과 비용이 다르다.

마지막 스테이지 기준으로는 총 6개의 타워가 존재한다.

타워를 한번 더 눌러 판매를 하거나 업그레이드를 할 수 있다.

화면을 이동할려면 마우스를 화면 가장자리에 가져다 가거나 키보드 방향키를 이용해 움직일 수 있다.

웨이브 마다 지정된 적들이 몰려나오며 적이 경로를 지나 끝에 도달하게 될 경우 라이프가 깍이게 된다. 모든 라이프를 소모할 경우 게임에서 패배하게 된다.

3. 씬 구성요소 분석

먼저 처음 기본이 되는 MainMenu씬부터 살펴 보겠다.

기본적으로 카메라와 조명과 같은 기본요소들은 들어있었다. 카메라의 경우 아이콘을 누를때 마다 이동을 해아하기 때문에 Animation 컴포넌트하고 스크립트가 들어있다.

켄버스에는 화면에 표시되는 UI들이 들어있는데 Main -> Menu Buttons 에 들어가게 되면 3가지 버튼이 들어있다. 각 버튼들은 Image가 설정되어 있으며 게임오브젝터에 UI버튼으로 만들어져 On Click를 이용해 클릭을 하게 될 경우 발생하는 사운드와 스크립트를 지정하였다.

UI로 들어가는 이미지 말고는 WorldObjects의 하의 객체로 들어있다.

Light Probe Group를 이용해 특정 좌표를 지정을 하여 그것을 이용해 빛을 저장하여 객체에 빛을 주고 있다.

그 다음은 Level1씬을 살펴 보겠다.

Level1Configuration 오브젝트를 보면 하위 오브젝트로 Level1WaveManager가 있는데 이 안에 각 레벨별 나오는 적의 종류와 딜레이 타임이 public형테로 정의되어있어 수정을 할 수 있게 되어있다. Nodes에는 이동 경로가 표시되어 있는데 Sphere Collider를 통해 원형으로 경로를 지나갔는지 체크를 하는 것 같다.

Level1PlacementAreas에서는 Grid와 SinglePlacement로 나누어서 격자로 배치할 수 있는 공간과 육각형으로 하나만 배치할 수 있는 공간으로 구분이 되어있다.

UI들은 Game UI에 들어있으며 Canvas Scaler를 사용해서 특정 해상도에서 배치되게 설정을 하였다.

4. 프리팹 분석

여러가지 프리팹들이 있지만 몇 개만 살펴보기로 한다.

Assets -> Prefabs -> Enemies에 있는 Haverbuggy를 적의 대표로 분석해보겠다.

Targetterdp 속도와 같은 정보들이 저장되어 잇으며 Visuals에 적의 이미지가 궤도 부분과 포신 부분으로 나누어져 있으며 이펙트도 저장되어 있다.

그 이외에는 여러 스크립트들과 채력을 표시할 수 있는 부분으로 이루어져 있다.

다음은 Assets -> Prefabs -> Towers에 있는 Rocket 들이다.

로켓 발사체인 RocketProjectile은 0,1,2 로 나누어져 있는데 이 객체 들에는 Splash Damager 스크립트에 public으로 선언된 Damage Amount가 달라 업그레이드에 따른 공격력 변화를 준 것 같다.

Base_RocketTower 들도 3가지로 나누어져 있으며 Meterial을 이용해서 질감을 주었다. 그림자는 Mesh를 통해 나타내었다.

맵 같은 경우에 메쉬가 확장되어 플레이어가 경기 영역 바깥에서 절대로 볼 수 없게되었 있다.

메쉬가 덩어리로 만들어져 많은 구조물이 화면 밖으로 그려지기 때문에 카메라 시야보다 훨씬 큰 메쉬는 비효율적입니다. 드로우 호출과 culling을 최소화하기 위해 적은 메쉬로 결합 된 많은 구조물로 구성이 되어있다.

이외에도 여러 번 사용되는 UI들도 프리팹으로 구성되어 있었다.

5. 알고리즘 분석

이번 파트에서는 스크립트를 분석해서 어떤 역할을 하고 어떤 구조로 짜여 있는지 분석을 할 것이다.

양이 많을 것으로 예상되니 중요한 것 위주로 천천히 분석을 하겠다.

먼저 게임실행 후 가정 저렴한 타워를 설치하면 MachineGunTower(Clone)이 생성된다.

이 안에 각 업그레이드 단계에 따라 MachineGunTower_0 ~ 2 까지의 프리팹이 할당 되는 구조로 되어있다.

이렇게 타워의 기본 기능을 정의하는 타워 구성 요소와 업그레이드 될 때 타워가 작동하는 방법을 정의하는 TowerLevel로 나누어져 있다.

Tower스크립트 안에는 Max Health와 Starting health가 들어있어 적에게 피격을 당할경우 채력이 줄게된다.

ActionGameFramework.Health 이라는 것이 usnig되어 있는데 이를 보아 프레임워크를 이용해서 처리하는 것 같다.

Alignment 변수는 Player과 enemy중 선택을하여 타워가 적 유닛에 발사되는지, 플레이어 정렬로 무엇이든 발사하는 지를 선택한다.

Tower스크립트가 들어있는 MachineGunTower객체의 레이어는 Towers로 제작자가 새로 만들어 정의를 하였는데 이를 통해 적이 공격하는 대상을 정할 수 있다.

Damage Collider 스크립트에서 Damagers가있는 객체와의 충돌을 처리를 한다

타워에는 스크립트 뿐만이 아니라 Rigidbody 가 들어있는데 IS Kinematic를 활성화 하여 물리력에 반응을 한다. Angular Drag 를 0.05로 주어 각의 속도를 주었다. 타워가 돌아가야 되기 때문이다.

Tower defense Template에는 여러 가지 발사기가 있다.

Ballistic Launcher 스크립트에서 Vector3 등을 이용한 물리를 사용하여 발사체의 탄도를 계산하고 BallisticProjectile를 호출한다.

HitscanLauncher는 즉시 타겟에 HitscanAttack을 배치한다. 그러면 HitscanAttack은 적을 공격하게 된다.

Super Tower Launcher는 예제 프로젝트에 나타나며 일정 시간이 지나면 자동으로 파괴된다.

Fire Particle System 에서 파티클을 선택하고 Tower Life Span는 몇초후에 객체가 사라질지 정하는 것이다. 그리고 Fire Vector X Rotation는 투영 방향에 따라 SuperTower의 터렛이 회전하는 양을 나타낸다.

AttackAffector 스크립트는 타워에서 발사체를 발사하는 역할을 한다.

Projectile에서 발사 종류를 설정을 하고 Projectile Points는 발사체의 위치와 Rotate등 을 지정해 준다. Is Multi Attack을 true로 설정하면 AttackAffector는 범위 내에있는 경우 여러 대상을 찾을 수 있다. false로 설정하면, Affector는 적군이 파괴되거나 적이 범위를 벗어날 때까지 한 번에 한 명의 적을 목표로 한다. Fire Rates는 발사체에게 발사체를 발사하라고 지시하는 딜레이를 정하는 것이다.

발사체를 발사하는게 발사체를 오브젝트로 만들어 발사하는게 아니라 이펙트로만 보여주어 리소스를 적게사용하여 최적화를 하기위해 이렇게 한 것 같다 그래서 실행후 Hierarchy를 보면 총알 오브젝트가 없다.

Targetter스크립트는 터렛의 이동에 관련된걸 정의하고 있다.

Turret 에서 터렛의 메쉬를 표시하고 Turret X Rotation Range 가 타워가 위아래로 볼 수 잇는 거리를 결정하고 Only Y Turret Rotation 을 체크하면 타워가 x축을 중심으로 회전이 가능해진다 Search Rate는 우선공격순위가 되는 적을 선택하게 되고 Idel Rotation Speed는 타워가 공격을 하고 목표물을 못찾아 다시 아이들 상태가 될때까지의 속도를 표시한다.

public List<Targetable> GetAllTargets() 을 통해 List 자료형으로 받아온다는 것을 알 수

있다.

HealthVisualizer 스크립트는 채력을 관리하는 스크립트이다

Health Bar 와 Background Bar를 통해 채력바와 현재 남아있는 채력을 표시할 수 있다. Show when Full 이 활성화 되어있지 않으면 타워가 처음 피해를 입은 때에만 상태가 갱신되게 된다.

protected virtual void Start() 에서

m_CameraToFace = UnityEngine.Camera.main.transform;

통해 현재 유니티에서 쓰고 있는 카메라의 위치를 가져와 저장을 한 다음

Vector3 direction = m_CameraToFace.transform.forward;

transform.forward = -direction;

를 통해 받아온 좌표에 따라 표시 위치를 정해주고 있다.

Camera Rig 스크립트에는 카메라의 작동 방식을 조정하는 여러 가지 방법을 제공한다.

Look and Movement Damp Factors 카메라 움직임이 얼마나 탄력적인지를 제어한다. 이 값이 높을수록 카메라 움직임이 부드럽게 된다.

Nearest Zoom, Furthest Zoom, Max Zoom 이 3개는 카메라의 줌 범위를 제어한다.

최대 확대 / 축소는 Springy Zoom이 아래에 선택되어있는 경우에만 관련이 있다. 그것은 줌이 전체적으로 클램핑되기 전에 플레이어가 줌 아웃 할 수있는 가장 멀리 떨어진 줌보다 얼마나 더 멀리 제어 하는지를 제어한다.

Zoom Log Factor는 Springy Zoom이 활성화 되어 있을 경우 고무줄 같이 따라오는 화면이 얼마나 적용할지를 결정하게 된다.

Zoom Recover Factor 는 고무줄 효과가 얼마나 빨리 복구되는지를 나타낸다.

Floor Y는 그냥 Translate y이다.

바닥평면 초기화를 하기 위해 protected virtual void Awake()이런 함수를 썼다

Awake를 쓰는 이유는 씬이 호출되고 나서 Awake, OnEnable, Start 순서대로 호출이 되기 때문에 바닥 초기화는 맨 처음에 할 일이기 때문이다.

데미지와 관련된 스크립트는 DamageableBehaviour여기서 관리하고 있다

DamageableListener 여기서는 UnityEvent를 쉽게 관리하여 이 스크립트에서 수정하면 쉽게 바뀔수 있게 구성 하기 위해 스크립트가 제작되었다.

DamageCollider 는 Box Collider이용해 충돌했을 때 DamageableBehaviour에 데이지를 보내는 역할을 한다.

Damager는 말 그대로 데미지를 준다. void OnCollisionEnter(Collision other) 함수가 있는데 여기서 충돌체크를 하여 충돌된 객체가 other로 넘어와서 이걸 가지고

var pfx = Poolable.TryGetPoolable<ParticleSystem>(collisionParticles.gameObject)

이렇게 다른 녀석들을 불러서 써먹는다.

Damager스크립트 변수 중

[Range(0, 1)]

public float chanceToSpawnCollisionPrefab = 1.0f;

이렇게 선언이 되어있는데 public이라 유니티에서 보면

이렇게 바로 조절하게 만들어져 있다.

여기 있는 관련 스크립트 들은 서로 다 연결되있다. 하나나 두개로 합치는게 더 보기 좋아 보이고 유지보수하기 편해보이는 것 같긴하다.

6. 마무리

처음 이 게임을 봤을 때는 간단하게 만들어 졌을 것이라고 생각하였다. 하지만 실제로 프로젝트를 열어보아 봤을 때에는 생각보다 많은 코드와 그래픽 데이터에 압도당해 버렸다.

예를 들어 데미지를 관리하는 코드가 있다고 하면 공격에 따른 데미지를 저장하는 스크립트 따로, 넘겨주는 스크립트 따로 등 한가지 일을 하다 라도 여러 가지 스크립트로 분류하여 코드가 짜여 있기 때문에 분석하는데 시간이 오래 걸리고 난이도도 어려웠다.

그리고 스크립트 뿐 만이 아니라 재질이나, 모델링 데이터나 UI이미지등 생각보다 자료가 많아 이정도는 만들어야 게임 하나를 제대로 만들 수 있구나 하고 생각이 들었다.

신기 했던 부분에는 타워의 경우 각 업그레이드마다 타워 프리팹을 따로 만드는 것이 아닌 공통된 부분에 관한 프리팹을 미리 만들어 둔 다음 거기에 타워 업그레이드 단계에 따른 다른 프리팹을 부모 자식 관계로 붙어서 리소스를 쉽게 관리하는 것에 공부가 되었던 것 같다.