졸업작품 '우주방어'

1688024 이승원

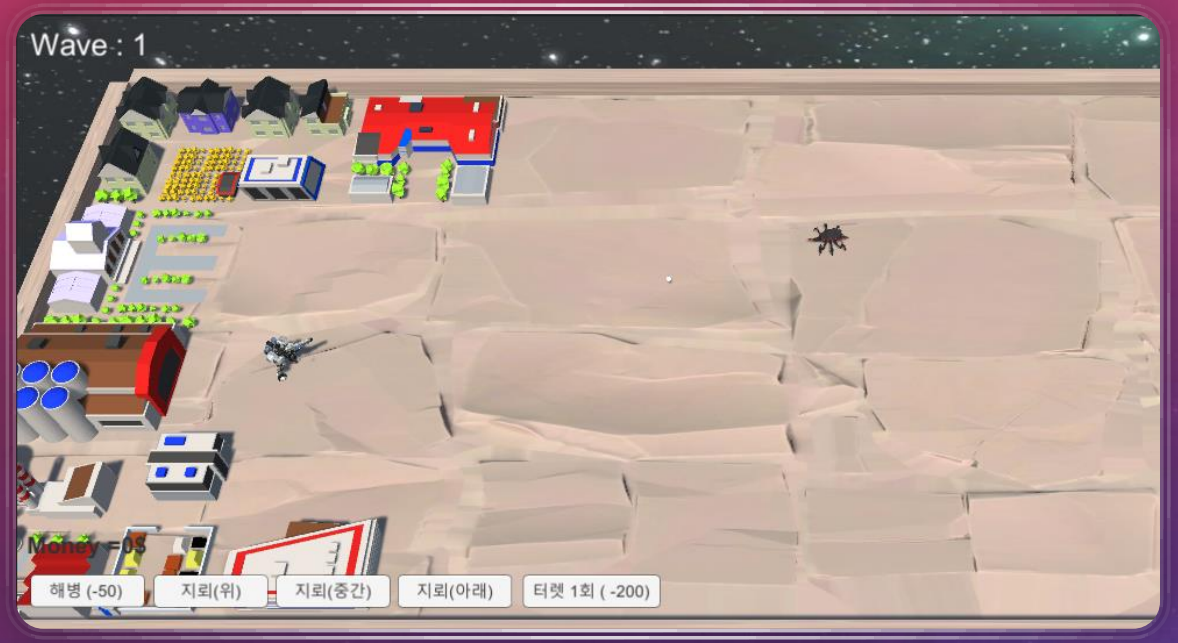
순서

- 개요
- 대표 이미지 소개
- 진행 과정
- 구성 및 구조

개요

- 장르 : RTS, 디펜스
- 우주군이 창설되어 타행성에 정착 하려했으나 외계 생명체들에게 공격을 받고 그것을 막아야 한다는 설정을 가지고 있다.
- 게임은 RTS 조작의 디펜스 형식이고 직접 움직일 수 있는 유닛이 있다는 것이 이 게임의 특징이다. RTS에서 자주 이용 되는 쿼터 뷰 형식의 시점으로 게임을 플레이하며 마지막 스테이지까지 공격을 막아내면 승리하게 되는 조건을 가지고 있다.

대표 이미지 소개



- 좌측 상단 : 단계를 표시
- 좌측 아래 : 현재 보유한 자산과 버튼들을 표시

- 점수는 성공/실패 시 누적된 값으로 점수 화면에서 나타남

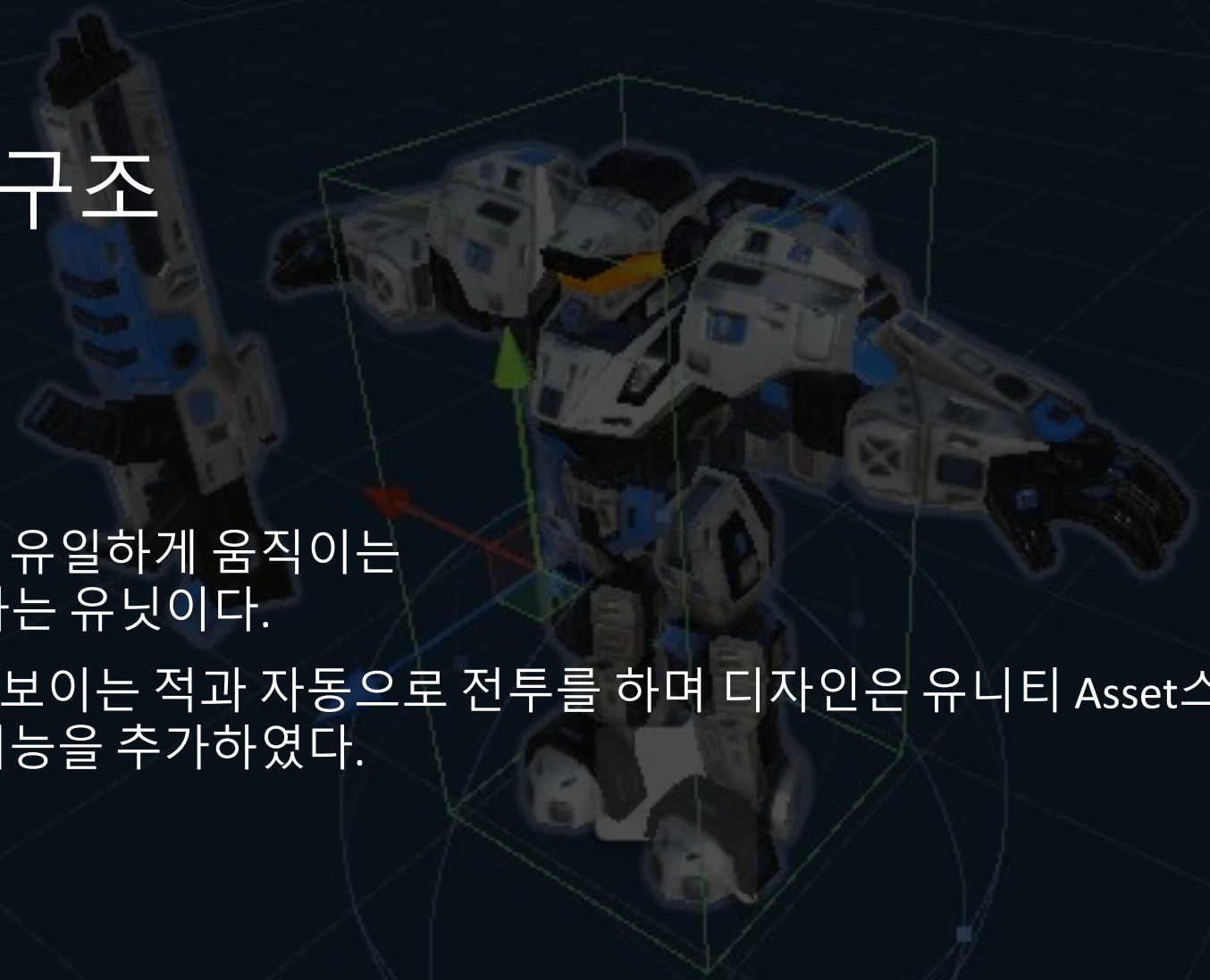
진행과정(1차)

	1주 차	2주 차	3주 차	4주 차	5주 차	6주 차
UI						
맵디자인						
이동						
전투						
단계요소						
게임마무리						

진행과정(2차)

	1주차	2주차	3주차	4주차
맵 디자인 수정				
스크립트 압축				
애니메이션, 사운드				
마무리 작업				

구성 및 구조



- 이 게임에서 유일하게 움직이는 ‘해병 ' 이라는 유닛이다.
- 이동하면서 보이는 적과 자동으로 전투를 하며 디자인은 유니티 Asset스토어에서 가져온 뒤 기능을 추가하였다.

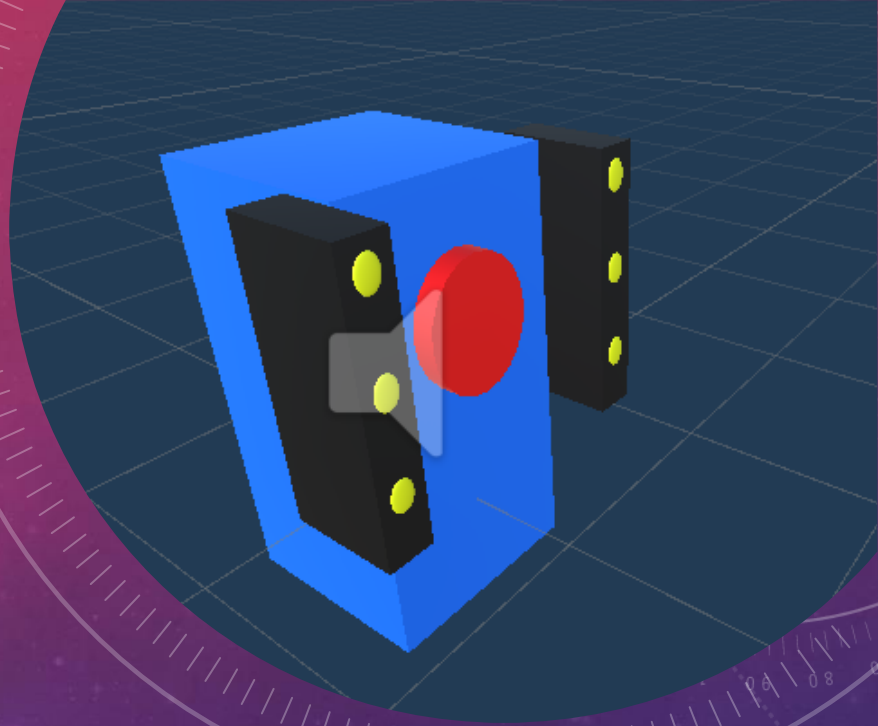
```
//transform.position += dir * (speed * Time.deltaTime;
ray = Camera.main.ScreenPointToRay(Input.mousePosition);

if (Input.GetMouseButtonDown(0))
{
    shooting = true;
    isWalking = true;
    animator.SetBool( name: "Walk",isWalking);
    animator.SetBool( name: "Shoot",shooting);
    //list.RemoveAt(0);
    if (Physics.Raycast(ray, out hit, maxDistance: 10000))
    {
        if (hit.collider.gameObject.tag.Equals("TeamUnit"))
        {
            SelectedUnit = hit.collider.gameObject;
            Debug.Log( message: "UnitSelected");
        }
        else
        {
            Debug.Log( message: "You should hit on TeamUnits");
        }
    }
}
if (Input.GetMouseButton(1))
{
    if (Physics.Raycast(ray, out hit, maxDistance: 10000))
    {
        if (hit.collider.tag.Equals("GameController"))
        {
            //좌표 나타냄
            //Debug.Log("포인트: " + hit.point);
            MoveTo(hit.point, hit);
        }
        else
        {
            Debug.Log( message: "You should hit on the ground");
        }
    }
}
```

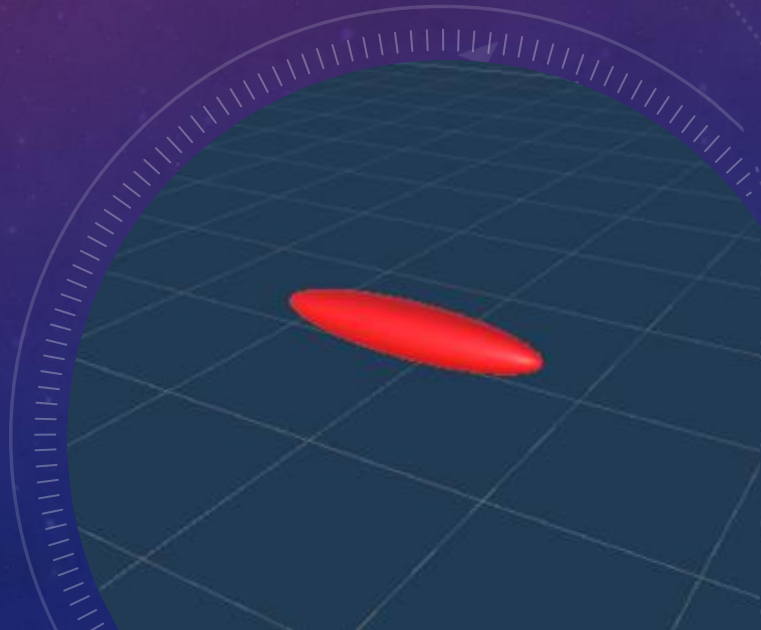
구성 및 구조

- 카메라로 보는 마우스 좌클릭 시 포인터 값에서 ray값 포인트를 찍어 그것이 'TeamUnit' 태그가 등록 되어있는지 감지하여 아군 유닛인지의 여부를 따진다.
  아군 유닛이 맞다면 유닛이 선택(SelectedUnit) 되고 해당 유닛은 좌클릭 시와 같은 원리로 바닥에 우클릭을 지속하는 동안 해당 포인트 위치로 이동하게끔 한다.

구성 및 구조

- 위쪽 사진에 있는 것은 포탑이다. 아래 사진에 있는 대구경탄을 발사하며 일반적으로 알고 있는 타워 디펜스의 기본적인 타워 형태이고 이 게임 내에서는 비싼 값에 한번밖에 설치 하지 못하도록 되어있다. 디자인은 도형의 비율을 맞춰서 직접 만들었다.
- (사진에 보이는 흰색 스피커 모양은 등록 되어 있는 사운드 리소스로 게임 내에서는 보이지 않음)





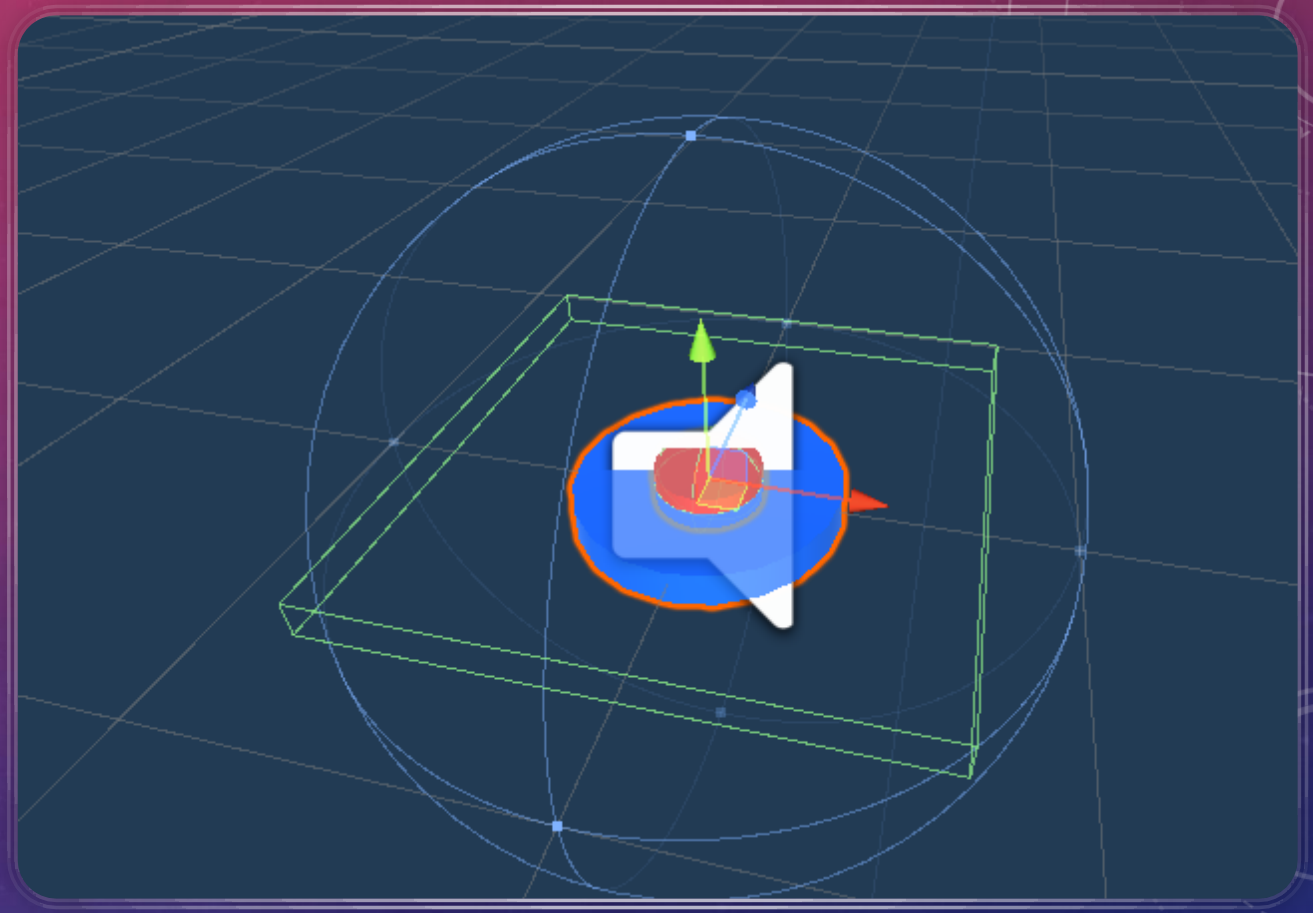
구성 및 구조

```
public class Turret : MonoBehaviour
{
    private float currentTime;

    public float crateTime = 5.0f;  Unchanged
    public GameObject bulletfactory;  TankBullet
    public GameObject firePosition;  GameObject

    public AudioClip shootClip; // 사망시 재생할 오디오 클립  Serializable
    private AudioSource playerAudio;
    // Start is called before the first frame update
    Event function
    void Start()
    {
        playerAudio = GetComponent<AudioSource>();
    }

    // Update is called once per frame
    Event function
    void Update()
    {
        currentTime += Time.deltaTime;

        if (EnemyManager.enemyNum > 1 && currentTime > crateTime)
        {
            GameObject bullet = Instantiate(bulletfactory);

            bullet.transform.position = firePosition.transform.position;

            playerAudio.clip = shootClip;
            playerAudio.Play();

            currentTime = 0;
        }
    }
```

- ‘currentTime += Time.deltaTime’을 이용해 함수를 초 단위로 업데이트를 하고 적이 감지 될 때마다 bullet을 생성하여 발사 되게 한다. 앞서 설명한 해병도 소총에서 같은 원리로 총알을 발사한다.

구성 및 구조

- 지뢰는 3개의 버튼(위, 중간, 아래)을 통해 설치가 가능하도록 구성 되어있다. 초록색 직육면체로 보이는 부분이 감지 반경이다. 충돌이 일어날 때 적과 함께 같이 사라지도록 설계 되어있다.



구성 및 구조

- 위 사진은 ‘바퀴’ 아래 사진은 ‘타이탄’이라는 유닛으로 바퀴는 빠른 속도로 도시에 파고 들어 게임에서의 라이프를 사라지게 한다. 타이탄은 느리지만 체력이 많고 포탑이나 지뢰로 처리하는 것이 요구될 만큼 높은 체력을 지니고 있다.

```
Event function
private void OnCollisionStay(Collision col)
{
    if (col.gameObject.tag == "TeamUnit")
    {
        Debug.Log(message: "아군 유닛 공격");

        InvokeRepeating(methodName: "Damaged", time: 0, repeatRate: attackSpeed*Time.deltaTime);

        if (hp == 0.0f)
        {
            Destroy(gameObject);
            CancelInvoke();
        }
    }
}
Frequently called  1 usage
void Damaged()
{
    hp -= attackPower;
}
```

구성 및 구조

- 적 유닛은 목표 지점을 향해 계속 이동을 하는데 그 과정에서 마주친 아군 유닛을 충돌 감지에 따라 공격을 하게 된다. 충돌이 감지되었을 때 attackSpeed만큼 아군 유닛에게 데미지를 주고 적 유닛 자신도 데미지를 받는다.
- 이런식으로 블로킹으로 막는 방법도 있지만 많이 해서 소모가 크게 되면 전투 유닛이 부족해질 수 있다.

구성 및 구조

- 맵에 있는 장소의 세부 정보
- - 빨간색으로 나오는 점선 길은 적들이 오는 길이다.
  위, 중간, 아래 중 적이 생성 되는 루트는 랜덤으로 구성 되어있다.
  - 기지 가장 가까이에 있는 파란색 사각형은 포탑이 설치 되는 곳
  - 하늘색 동그라미가 있는 곳이 유닛이 생산 되는 곳(둘 중 랜덤 위치로 생성)
  - 초록색 사각형이 지뢰가 깔리는 지점이다.

구성 및 구조

- 카메라의 스크립트이고 키 입력 “W,S,A,D”를 이용하여 각각 확대, 축소, 왼쪽, 오른쪽으로 이동하게 구성되어있다.
  마우스 휠을 통해 카메라 기준 높이를 위, 아래로 조절할 수 있다. 아래로 돌리면 위로 당겨지고 위로 돌리면 아래로 당겨진다.

```
public class RTSCam : MonoBehaviour
{
    //camera move speed
    public int cameraSpeed;    "10"

    Event function
    void Update ()
    {
        //translating on the x, z and Y axes using WASD
        if(Input.GetKey( name: "w"))
        {
            transform.Translate( translation: Vector3.forward * cameraSpeed * Time.deltaTime );
        }

        if(Input.GetKey( name: "s"))
        {
            transform.Translate( translation: Vector3.back * cameraSpeed * Time.deltaTime );
        }

        if(Input.GetKey( name: "d"))
        {
            transform.Translate( translation: Vector3.right * cameraSpeed * Time.deltaTime);
        }

        if(Input.GetKey( name: "a"))
        {
            transform.Translate( translation: Vector3.left * cameraSpeed * Time.deltaTime );
        }


        //zooming up and down with the scrollWheel

        if(Input.GetAxis("Mouse ScrollWheel") < 0  && !UnityEngine.EventSystems.EventSystem.current.IsPointerOverGameObject())
        {
            transform.Translate( translation: (Vector3.up * 5) * cameraSpeed * Time.deltaTime );
        }

        if(Input.GetAxis("Mouse ScrollWheel") > 0 && !UnityEngine.EventSystems.EventSystem.current.IsPointerOverGameObject())
        {
            transform.Translate( translation: (Vector3.down * 5) * cameraSpeed * Time.deltaTime );
        }
    }
```

구성 및 구조

- 게임 내에 있는 버튼의 구조이다.
- 버튼 내의 클릭이 감지되면 (AddListner) MarineProduct 함수를 실행 시켜 자원이 있는지 없는지의 여부를 따져 생산 또는 함수를 빠져나가게 된다. 밑에 있는 randValue를 통해 나오는 유닛의 포지션을 정하게 된다.

```
{
    public GameObject Marine;  Marin1

    public Button MarineButton;  MarineBtn (MonoBehaviour)

    public GameObject UnitPos1;  UnitPosition1
    public GameObject UnitPos2;  UnitPosition2
    //public GameObject UnitPosition2;

    // Start is called before the first frame update
    Event function
    void Start()
    {
        MarineButton.onClick.AddListener(MarineProduct);
    }

    // Update is called once per frame
    1 usage
    void MarineProduct()
    {
        if (EnemyManager.money < 50)
        {
            Debug.Log( message: "Not enough money");
        }
        else
        {
            int randValue = UnityEngine.Random.Range(0, 9);

            GameObject Marine1 = Instantiate(Marine);

            EnemyManager.money = EnemyManager.money - 50;
            if (randValue < 5)
            {
                Marine1.transform.position = UnitPos1.transform.position;
            }
            else
            {
                Marine1.transform.position = UnitPos2.transform.position;
            }
        }
```

이상입니다. 감사합니다.

- https://revalues.github.io/
  1차 진행 과정 깃허브 주소