https://youtu.be/HWFyZt96ekg

Unity3D 기능활용

1) Coroutine, Vector – 플레이어나 몬스터의 상태변화 및 시야부여에 사용했습니다

```
public IEnumerator State_Control() // 몬스터의 상태변화를 컨트롤합니다
{
         while(true)
        {
            yield return new WaitForSeconds(.1f);
            switch (current_state)
            {
                case MONSTER_STATE.idle:
                    break;
                case MONSTER_STATE.walk:
                    rayDirection = playerTr.position - transform.position;
                    if ((Vector3.Angle(rayDirection, transform.forward)) < 45)
                   // 몬스터의 45도 시야안에 있고
                    {
                        if (Physics.Raycast(transform.position, rayDirection, out hit))
                        {
                            if (hit.collider.name == "Player" &&
                                Vector3.Distance(transform.position, playerTr.position) < 20.0f)
                                // 상대가 플레이어이고 거리가 20.0f 이하라면
                            {
                                current_state = MONSTER_STATE.trace; // 추격한다
                            }
                        }
                    }
                    if (Vector3.Distance(transform.position, tarPos) <= 0.5f)
                        GetNextPosition(); // 몬스터가 랜덤한위치로 이동완료했을때 다음위치를 찾는다
                    nav.SetDestination(tarPos);
                    break;
                case MONSTER_STATE.trace:
                    anim.SetTrigger("trace");
                    nav.SetDestination(playerTr.position);
                    if(Vector3.Distance(transform.position, playerTr.position) < 5f)
                            // 플레이어와 몬스터의 위치가 가까우면
                    {
                        current_state = MONSTER_STATE.attack; // 어택모드
                    }
                    break;
                case MONSTER_STATE.attack:
                    attack();
                    break;
            }
        }
    }

void GetNextPosition() // 랜덤으로 다음위치를 찾는다.
    {
        tarPos = new Vector3(Random.Range(minX, maxX), 0.5f, Random.Range(minZ, maxZ));
    }
```

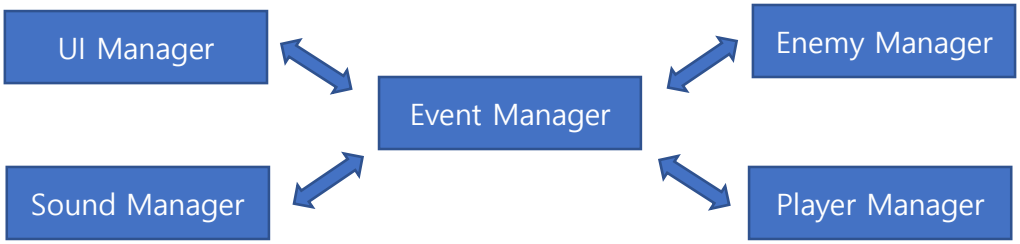
2) UGUI

유니티에 내장된 UGUI를 이용하여 이동버튼, 공격버튼, 옵션버튼 및 체력게이지를 넣었으며, 해당하는 UI를 누르면 EventManager로 이벤트를 전달해 다른 객체가 반응할 수 있도록 구현하였습니다.

객체간 통신방법

1) Delegate Event를 이용한 방법을 사용했습니다

예를 들어 플레이어가 공격받으면 이벤트 Event Manager로 발생시킵니다. 그것을 반응을 필요로 하는 클래스로 이벤트를 보냅니다. UI(체력 게이지), Sound(맞았을 때 효과음), Enemy에서 이벤트를 받아 각각 구현된 부분을 실행합니다.



2) 싱글턴

```
public static EventManager Instance
    {
        get { return instance; }
        set { }
    }

    public static EventManager instance = null;

void Awake ()
{
                if(instance == null)
        {
            instance = this;
            DontDestroyOnLoad(gameObject);
        }
        else
        {
            DestroyImmediate(this);
        }
}
```

EventManager 클래스는 이벤트 수신, 송신의 이유로 모든 스크립트에서 사용이 가능해야합니다. 그래서 싱글턴패턴을 활용하여 객체가 파괴되는 것을 막고 이미 선언되어 있을경우 instance를 리턴하여 단 1개만 생성되는 클래스로 지정하였습니다.

사용한 자료구조

1) Dictionary

EventManager 클래스에서 사용하였습니다.

```
Dictionary<EVENT_TYPE, List<IListener>> Listeners = new Dictionary<EVENT_TYPE, List<IListener>>();
```

현재 이벤트타입과 이벤트 수신을 위해 등록 되어있는 이벤트의 집합입니다.

Dictionary를 사용하여 이벤트 타입을 키로하여 리스너들에게 이벤트를 알립니다.