# ICEBERG

## 환경 살리기 게임 프로젝트

지원자: 최윤호

프로젝트 GIT 저장소:
https://github.com/yyyoonho/Iceberg

포트폴리오 영상 링크:
https://youtu.be/i58oosaSKxw?t=229

# 게임시스템

## 내부 DB

- Transform
- Time
- Status

게임 데이터
Save & Load

Iceberg

환경 선택
Save & Load

랭킹 요청
점수 Save & Load

## Firebase

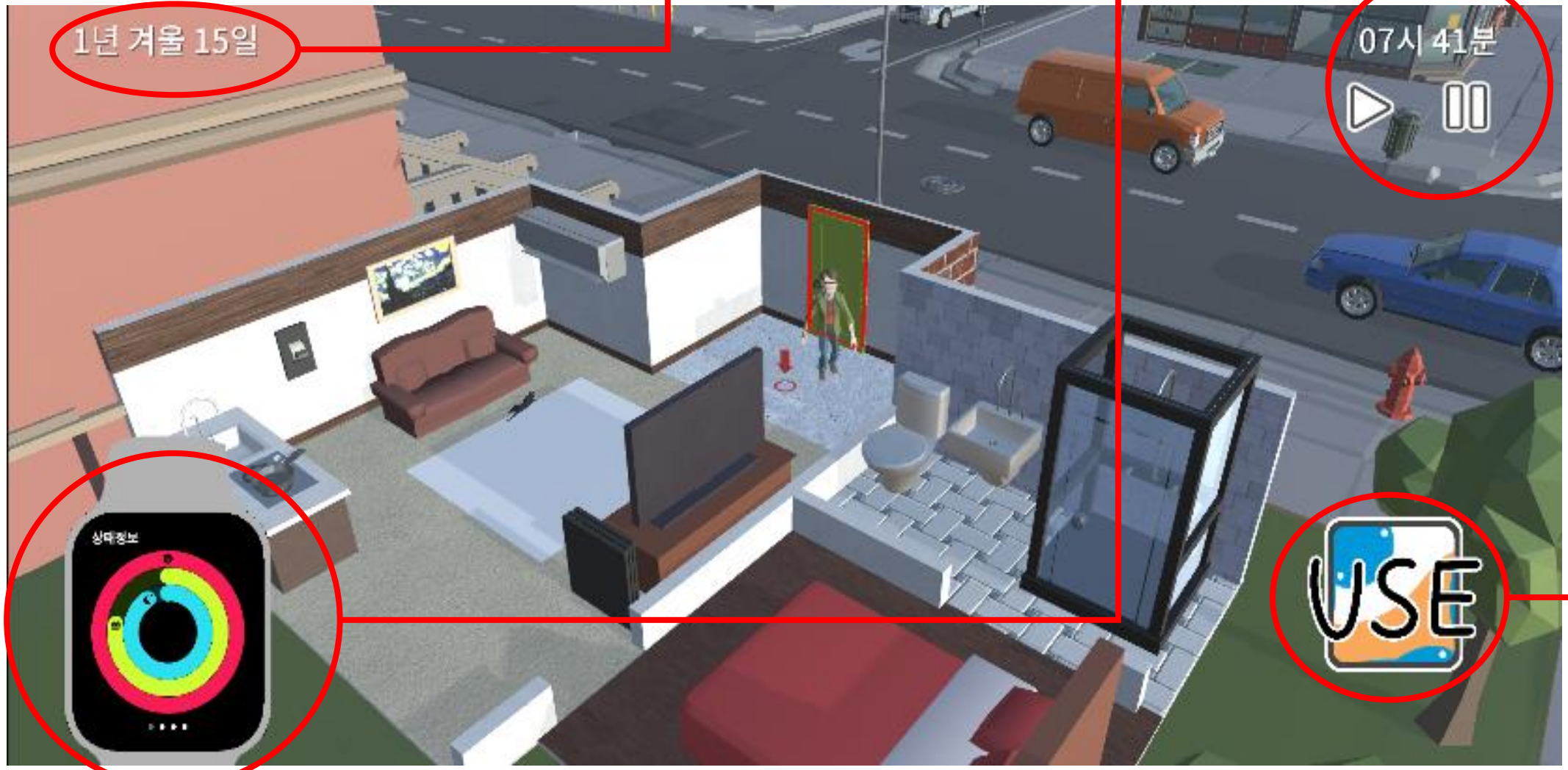
- UserSelect
- MainGame
- PenguinRun
- WaterDoggy
- ShotBear

# 메인게임

게임속의 계절과 날짜를 표시.
시간의 흐름을 알려줌.

캐릭터의 상태,
전날대비 오염정도,
랭킹, 로그아웃 기능.

시간의 흐름을 조절.
X1, x2, x4배가 있다.
일시정지 기능도 구현.




사물과 상호작용
하는 버튼

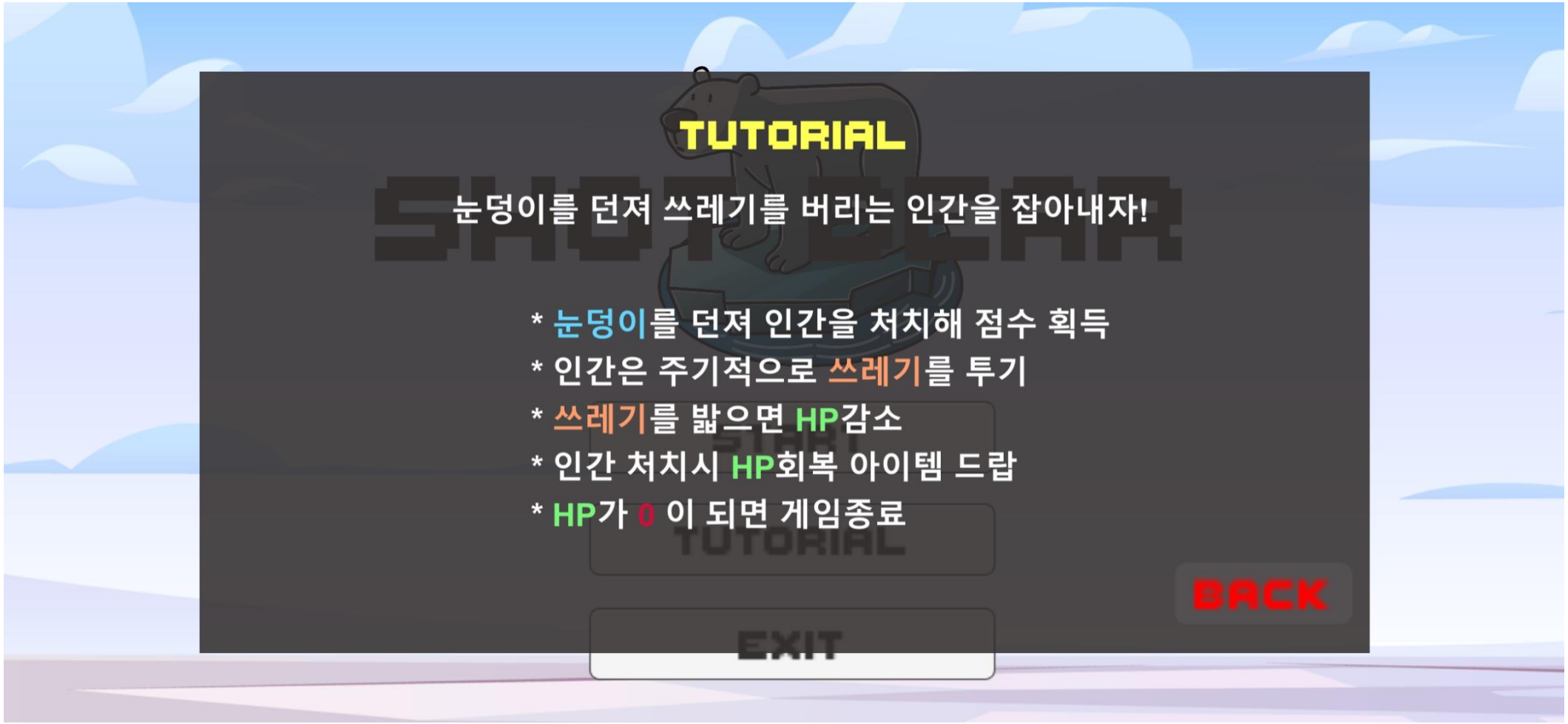
# 메인게임

TV를 선택했을 경우 나오는 미니게임 선택창

# 미니게임


게임 시작 화면

# 미니게임




게임 튜토리얼 화면

# 미니게임

적을 맞추거나 처지 시 증가
쓰레기 발판 밟으면 감소

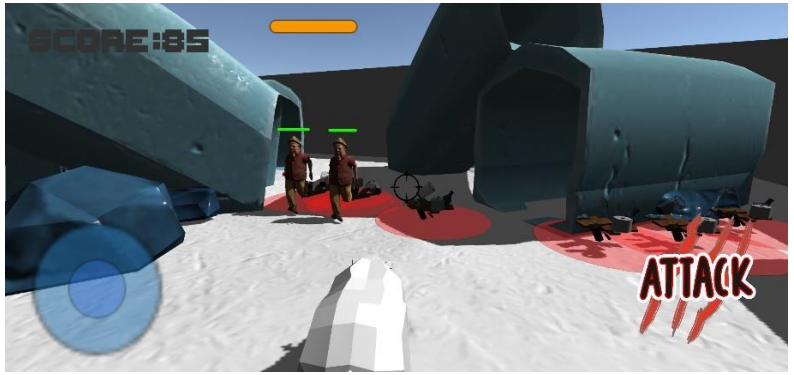
지속적으로 일정량 감소
쓰레기 발판 밟으면 감소
회복 발판 밟으면 증가.

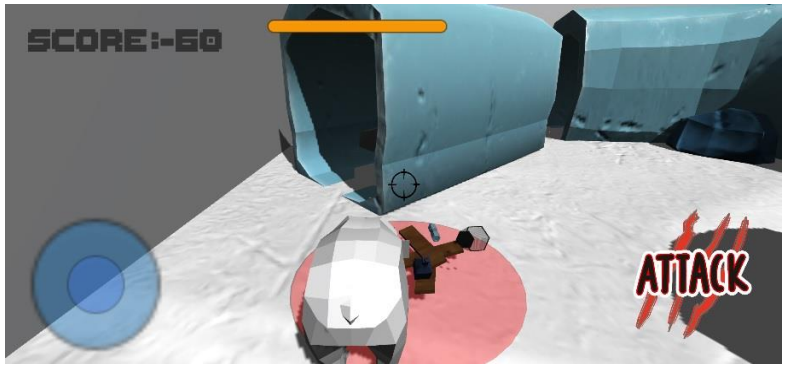

곰을 이동할 수 있는 버튼

맵을 무작위로 이동
일정시간마다 쓰레기 투척
죽으면 회복 발판 생성.

가운데 조준점을 향해 눈덩이 투척

# 미니게임


## 환경 파괴적 선택이 많을 경우

적 개체 수 증가
쓰레기 장판 출현 빈도 증가

쓰레기 장판 데미지 증가
스코어 감소량 증가

# 코드 설명

```
public class GameDirector : MonoBehaviour
{
    public GameObject player;
    public EnemySpawnGenerator enemySpawnGenerator;
    private float timeLeft = 1.0f;
    private float nextTime = 0.0f;
    private bool enemyState = false;
    private bool playerState = true;

    Unity 메시지|참조 0개
    private void Start()...

    Unity 메시지|참조 0개
    private void Update()
    {
        //게임 시작과 동시에 플레이어는 매초 1의 피해를 받습니다.
        if (Time.time>nextTime && enemyState==true && playerState ==true)
        {
            nextTime = Time.time + timeLeft;
            TakeDamageRegulary();
        }

        //Enemy가 맵에 살아있는지 확인합니다. 죽었을 경우, 새로운 장소에 리스폰시킵니다.
        if(enemyState == false)
        {
            enemyState = true;
            enemySpawnGenerator.RespawnEnemySpot();
        }
    }

    참조 1개
    void TakeDamageRegulary()
    {
        //게임 시작과 동시에 플레이어는 매초 1의 피해를 받습니다.
        player.GetComponent<Player>().TakeHit(1);
        UIManager.Instance.UpdateHp(player.GetComponent<Player>().health);
        if(player.GetComponent<Player>().health<=0)
        {
            playerState = false;
            player.GetComponent<Player>().DeadState();
        }
    }
```

## GameDirector

- 게임 시작 시, 플레이어에게 일정시간마다 일정량 데미지를 입힙니다.
- Enemy의 상태를 체크합니다.
- Enemy 사망 시, 랜덤한 장소에 리스폰 시킵니다.

# 코드 설명

```
public class LivingEntity : MonoBehaviour
{
    public UnityEvent DieEvent;

    public int startingHealth;
    public int health;
    public bool dead;
    public Animator anim;

    //체력을 초기화합니다.
    Unity 메시지 | 참조 4개
    protected virtual void Start()
    {
        health = startingHealth;
    }

    //공격을 받은 경우.
    참조 3개
    public void TakeHit(int damage)
    {
        health -= damage;
        Debug.Log("Hit");
        if(health <=0&&!dead)
        {
            health = 0;
            Die();
            return;
        }
    }
```

```
    //체력을 회복한 경우.
    참조 1개
    public void TakeHeal(int heal)
    {
        if(health>=100)
        {
            return;
        }
        else if(health<100)
        {
            if(health>=100)
            {
                health = 100;
            }
            Debug.Log("Heal");
            health += heal;
            return;
        }
    }

    //객체가 죽은 경우.
    참조 1개
    protected void Die()
    {
        Debug.Log("죽었습니다..");
        dead = true;
        //유니티 이벤트만들기
        DieEvent.Invoke();
    }
```

## LivingEntity

- Player와 Enemy가 상속받습니다.
- 데미지를 받은 경우, TakeHit() 메소드를 실행합니다.
- 체력을 회복하는 경우, TakeHeal() 메소드를 실행합니다.
- 죽은 경우, Die() 메소드를 실행합니다.

# 코드 설명

```
public enum State
{
    idle, hitPack, healPack, inArea
};
public State playerState  = State.idle;

Unity 메시지|참조 4개
protected override void Start()
{
    base.Start();
    StartCoroutine("CheckState");
}

Unity 메시지|참조 0개
private void OnTriggerStay(Collider other)
{
    //해당 Area에 있으면 피해를 받거나, 회복합니다.

    if (other.tag =="HealPack")
    {
        Debug.Log("[힐팩] 경계안에서 초당 3씩 회복합니다!");
        playerState = State.healPack;
    }
    if(other.tag =="AntiPack")
    {
        Debug.Log("[안티팩] 경계안에서 초당 5씩 공격받습니다!");
        playerState = State.hitPack;
    }

}
Unity 메시지|참조 0개
private void OnTriggerExit(Collider other)
{
    playerState = State.idle;
}
```

```
IEnumerator CheckState()
{
    while(true)
    {
        yield return new WaitForSeconds(0.5f);
        //초당 10의 데미지를받고, 20의 스코어가 감소합니다.
        if (playerState == State.hitPack)
        {
            UIManager.Instance.UpdateHp(health);
            aud.PlayOneShot(antiPackSound);
            base.TakeHit(antipackDamage);
            UIManager.Instance.UpdateScoreText(-antipackDamage);
            playerState = State.idle;
            if (dead == true)
            {
                break;
            }
        }
        //초당 10의 체력을 회복합니다.
        else if (playerState == State.healPack)
        {
            UIManager.Instance.UpdateHp(health);
            aud.volume = 1f;
            aud.PlayOneShot(healPackSound);
            base.TakeHeal(healpackDamage);
            UIManager.Instance.UpdateScoreText(healpackDamage);
            playerState = State.idle;
        }
    }
    DeadState();
}

참조 2개
public void DeadState()
{
    gameObject.GetComponent<PlayerInput>().enabled = false;
    gameObject.GetComponent<PlayerMovement>().enabled = false;
    anim.SetTrigger("IsDead2");
    aud.PlayOneShot(deadSound);
    Invoke("DestroyObject", 5.0f);
}
```

## Player

- 플레이어의 데미지 및 체력회복 상태체크를 위해 코루틴을 사용합니다.
- Player의 상태 변화를 UIManager 변수에 적용합니다.

# 코드 설명

```
IEnumerator CheckState()
{
    // Enemy가 살아있다면, 플레이어와의 거리에 따라 trace모드,
    // idle 모드로 전환합니다.
    while(!isDead)
    {
        yield return new WaitForSeconds(0.2f);
        float dist = Vector3.Distance(playerTransform.position, _transform.position);

        if(dist<=traceDist)
        {
            curState = CurrentState.trace;
        }

        else
        {
            curState = CurrentState.idle;
        }
    }
    // Enemy가 죽었을 경우, 상태체크를 stop하고 IsDead 애니메이션 재생,
    // 이후 오브젝트 destroy 합니다.
    if (isDead)
    {
        Invoke("DestroyObject", 4.3f);
        anim.SetTrigger("IsDead");
        StopCoroutine(CheckState());
    }
}
```

## EnemyController

- Enemy의 상태를 일정시간 마다 체크합니다.
- Enemy와 Player의 거리에 따라 Enemy의 상태를 Idle모드, Trace모드로 변경합니다.
- Trace모드 시, Player의 추격을 회피합니다.

# 코드 설명

```
// 상태에 따라 애니메이션을 변경합니다.
참조 1개
IEnumerator CheckStateForAction()
{
    while(!isDead)
    {
        switch(curState)
        {
            case CurrentState.idle:
                nvAgent.Stop();
                anim.SetBool("isMoving", false);

                break;
            case CurrentState.trace:
                nvAgent.destination = playerTransform.position;
                nvAgent.Resume();
                anim.SetBool("isMoving", true);
                break;
            case CurrentState.attack:
                //어택옵션 추가 시, 애니메이션 재생
                break;
        }
        yield return null;
    }
    if(isDead)
    {
        nvAgent.Stop();
    }
}
```

## EnemyController

- UnityEngine.AI 의 NavMeshAgent를 활용하여 Enemy가 상태에 따라 맵을 이동하게 합니다.
- Enemy 상태에 따른 애니메이션 및 명령을 실행합니다.
- Enemy 사망 시, NavMeshAgent를 종료합니다.

# 코드 설명

```
public class CubeMove : MonoBehaviour
{
    private float x;
    private float z;
    private Vector3 movingPoint;
    public Transform cube;

    Unity 메시지 | 참조 0개
    private void Start()
    {
        StartCoroutine(RandomMove());
    }

    //3.5초마다 Enemy가 무작위로 움직일 수 있도록, 맵의 무작위 좌표에 이동할 곳 생성.
    참조 1개
    IEnumerator RandomMove()
    {
        while(true)
        {
            yield return new WaitForSeconds(3.5f);
            x = Random.Range(-30.0f, 30.0f);
            z = Random.Range(-30.0f, 30.0f);
            movingPoint = new Vector3(x, 1, z);
            cube.position = movingPoint;
        }
    }
}
```

## CubeMove

- Enemy가 맵을 무작위로 돌아다닐 수 있도록,
  랜덤하게 좌표를 찍어주는 Cube를 생성합니다.

# 코드 설명

```
private void Start()
{
    StartCoroutine(InstantiateAnitPack());
}
Unity 메시지 | 참조 0개
private void Update()
{
    if(dead)
    {
        StopCoroutine(InstantiateAnitPack());
    }
}

//AntiPack을 인스턴트 시킵니다.
참조 3개
IEnumerator InstantiateAnitPack()
{
    while(!dead)
    {
        if(dead==true)
        {
            StopCoroutine(InstantiateAnitPack());
        }
        Instantiate(antiPackPrefab,
            new Vector3(spawner.position.x, spawner.position.y, spawner.position.z), spawner.rotation);
        yield return new WaitForSeconds(instantTime);
    }
}

//HealPack을 인스턴트 시킵니다.
참조 0개
public void InstantiateHealPack()
{
    Instantiate(healPackPrefab,
        new Vector3(spawner.position.x, spawner.position.y+1, spawner.position.z+2), spawner.rotation);
}

참조 0개
public void ChangeStateDead()
{
    dead = true;
}
```

## PackGenerator

- AntiPack 혹은 HealPack을 인스턴트 합니다.
- 코루틴 사용으로 설정한 시간 마다 인스턴트 되도록 조절합니다.

# 감사합니다.