



제작자 : 한찬응
Game Portfolio

1 Notice about Halloween Defense

2 About Player

Player Shooting
Player Health

3 About Enemy

Enemy Animation & Attack
Enemy Health

4 Radar System

5 Game Manager

Enemy Manager
Heal Manager

6 Canvas System

Health Slider & Fire Button & Virtual Joy Stick
Fading Image & Damage Image

7 ETC

FOG
Zombie





Notice about Halloween Defense

- “Halloween Defense” 는 한동대학교 캡스톤 프로젝트 “완화치료를 위한 게임 개발”의 일환으로 만들어진 미니 게임입니다.
- Android 용이며, 삼성 갤럭시 화면 1600x960에 맞춰져 있습니다.
- 게임 APK는 아래 링크를 통해 다운 받으실 수 있습니다.

<https://drive.google.com/open?id=1tzdCTl6sYzN15G-GopUmVYc9SN3EFLbK>





Notice about Halloween Defense



본 게임은 Unity 자습서 “Tanks Tutorial”에서 영감을 받아서 만든 게임으로 Tutorial에서 사용되었던 Asset 들이 사용했다는 것을 알려드립니다.

Tanks Tutorial Link: <https://unity3d.com/kr/learn/tutorials/s/tanks-tutorial>



Designed by ccjhde



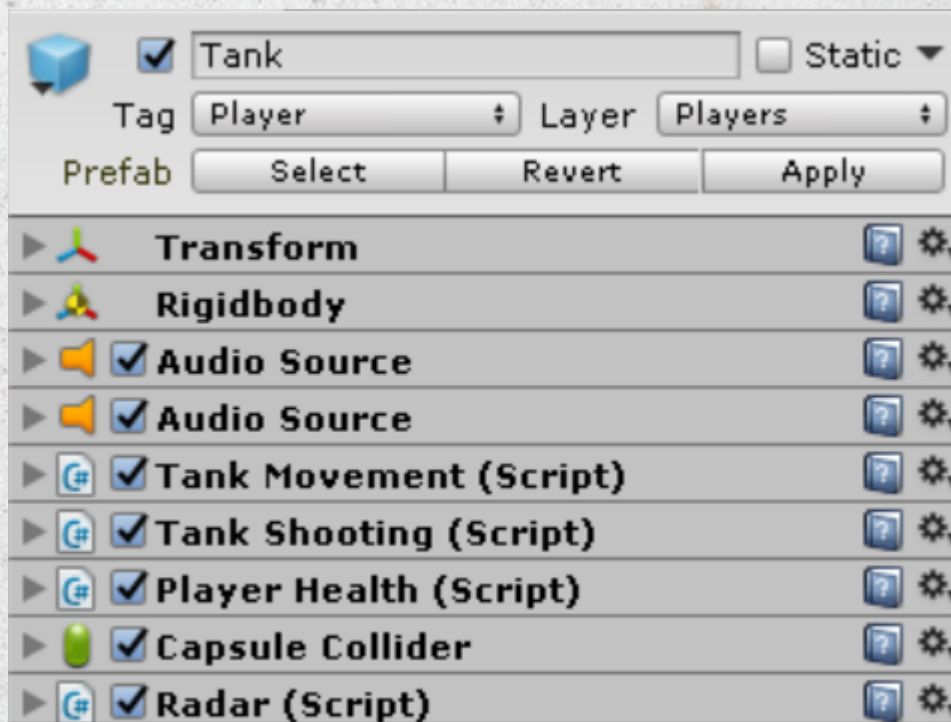
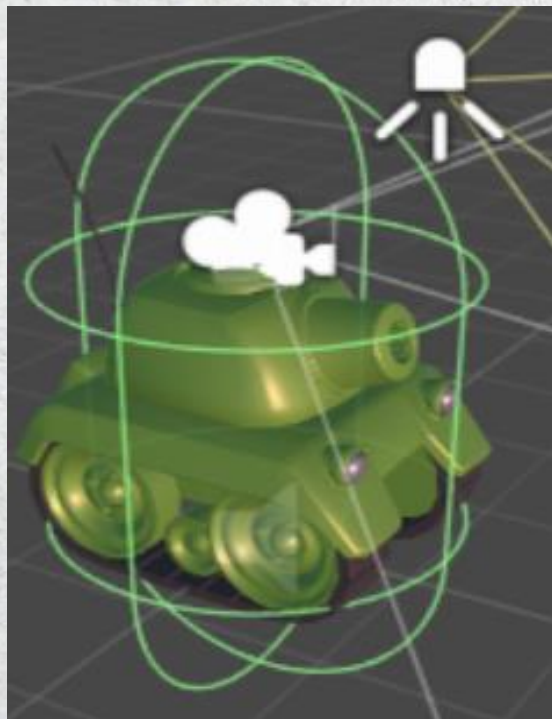
Game Video

Game Video Link : <https://youtu.be/Z79Y9v1v43w>





About Player



Player Script 설명

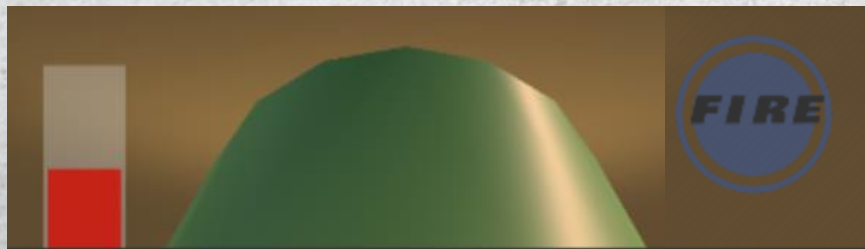
- Player Shooting
- Player Health





About Player Shooting

```
private void Update()
{
    if (!buttonDown && !buttonUp)
    {
        m_CurrentLaunchForce = m_MinLaunchForce;
    }
    else if (buttonDown)
    {
        // Change the clip to the charging clip and start it playing.
        m_ShootingAudio.clip = m_ChargingClip;
        m_ShootingAudio.Play ();
        if (m_CurrentLaunchForce <= m_MaxLaunchForce)
        {
            // Increment the launch force and update the slider.
            m_CurrentLaunchForce += m_ChargeSpeed * Time.deltaTime;
        }
        else
        {
            m_CurrentLaunchForce = m_MaxLaunchForce;
        }
        FireGauge(m_CurrentLaunchForce);
    }
    else if (buttonUp)
    {
        Button_Reset ();
        Fire ();
    }
}
```



```
public void Button_Down()
{
    buttonDown = true;
    buttonUp = false;
}
public void Button_Up()
{
    buttonDown=false;
    buttonUp=true;
}
public void Button_Reset()
{
    buttonDown=false;
    buttonUp=false;
}
```



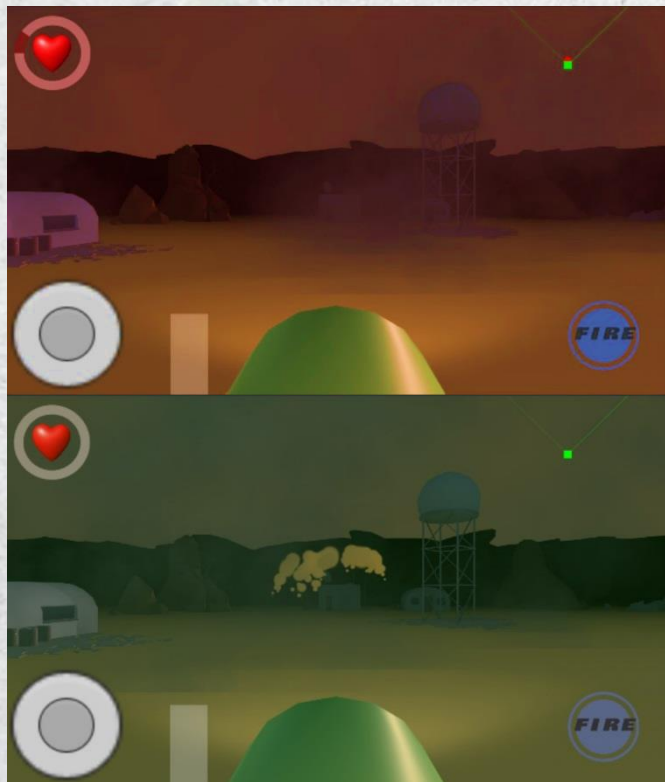
Screen에 Fire 버튼을 누르는 동안
값을 차오르게 만들었고
버튼을 때는 순간 차오른 값에 따
라 발사를 하도록 만들었습니다.

+Canvas에 FireGauge()를
추가해서 탄의 강도를
Visual하게 나타냄





About Player Health



```
void Update ()
{
    // If the player has just been damaged...
    if(damaged)
    {
        // ... set the colour of the damageImage to the flash colour.
        damageImage.color = flashColour;
    }
    if (healed)
    {
        damageImage.color = flashColourPlus;
    }
    // Otherwise...
    else
    {
        // ... transition the colour back to clear.
        damageImage.color = Color.Lerp (damageImage.color, Color.clear, flashSpeed * Time.deltaTime);
    }

    // Reset the damaged flag.
    damaged = false;
    healed = false;
}
```

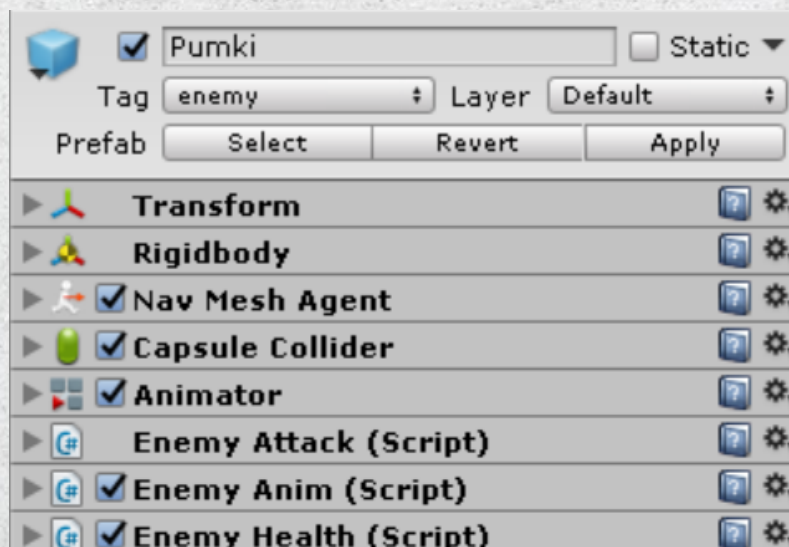
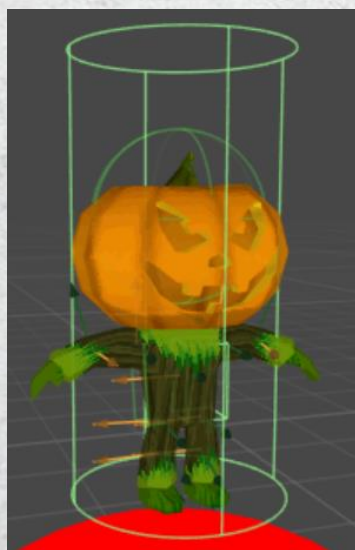
Player 가 Enemy에게 공격을 받을 때와 에너지를 탄으로 쏘아서 먹었을 때를 분리하여 공격을 받았을 때는 (-)에너지와 붉은 화면 힐을 했을 때는 (+)에너지와 초록색 화면이 되도록 만들었습니다.

```
public void TakeDamage (int amount)
{
    // Set the damaged flag so the screen will flash.
    damaged = true;
    // Reduce the current health by the damage amount.
    currentHealth -= amount;
    // Set the health bar's value to the current health.
    healthSlider.value = currentHealth;
    // If the player has lost all it's health and the dea
    if(currentHealth <= 0 && !isDead)
    {
        StartCoroutine (Fading("GameOver"));
    }
}
```

```
public void HealPoint(int amount)
{
    // Set the healed flag so the screen will flash.
    healed = true;
    if (currentHealth != startingHealth) {
        currentHealth += amount;
        // Set the health bar's value to the current heal
        healthSlider.value = currentHealth;
    } else {
        Debug.Log("The Player health is full");
    }
}
```




About Enemy



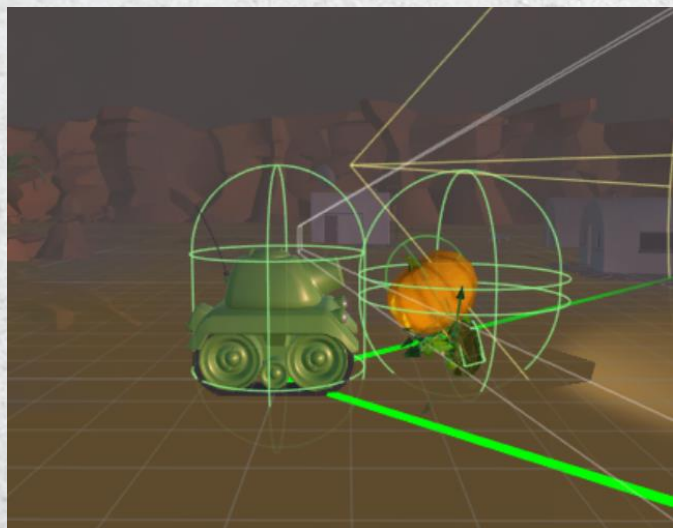
Enemy Script 설명

- Enemy Animation & Attack
- Enemy Health





About Enemy Animation & Attack



Enemy가 Player 앞에서 계속 공격을 하는 것 보다 몸통 박치기로 공격과 동시에 사라지도록 만들었습니다.



```
void OnTriggerEnter (Collider other)
{
    // If the entering collider is the player...
    if(other.gameObject == player)
    {
        // ... the player is in range.
        anim.SetBool("isAttacking",true);
        enemyHealth.currentHealth = -1;
        StartSinking ();
    }
}
```

Player의 collider와 충돌을 하게 되면 “isAttacking” 애니메이션을 실행하게 하고 StartSinking()으로 Enemy가 없어지게 만들었습니다.

```
public void StartSinking ()
{
    // Find and disable the Nav Mesh Agent.
    GetComponent <UnityEngine.AI.NavMeshAgent> ().enabled = false;

    // Find the rigidbody component and make it kinematic (since we
    GetComponent <Rigidbody> ().isKinematic = true;
    // The enemy should no sink.
    isSinking = true;
    // After 2 seconds destroy the enemy.
    Destroy (gameObject, 0.5f);
}
```





About Enemy Health



```
void OnTriggerEnter (Collider other)
{
    // If the entering collider is the player...
    if(other.gameObject == GameObject.FindGameObjectWithTag ("shell"))
    {
        Debug.Log("enemy got hit");
        TakeDamage (shellDamage);
    }
}

void Update ()
{
    // If the enemy should be sinking...
    if(isSinking)
    {
        // ... move the enemy down by the sinkSpeed per second.
        transform.Translate (-Vector3.up * sinkSpeed * Time.deltaTime);
    }
}
```

```
public void TakeDamage (int amount)
{
    // If the enemy is dead...
    if(isDead)
        // ... no need to take damage so exit the function.
        return;
    // Reduce the current health by the amount of damage sustained.
    currentHealth -= amount;

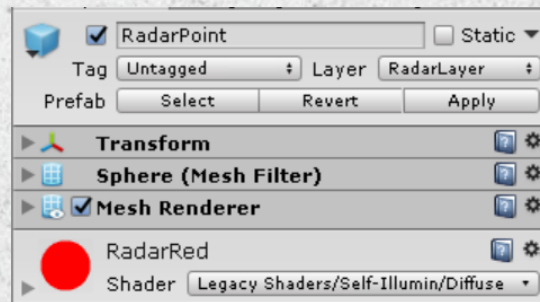
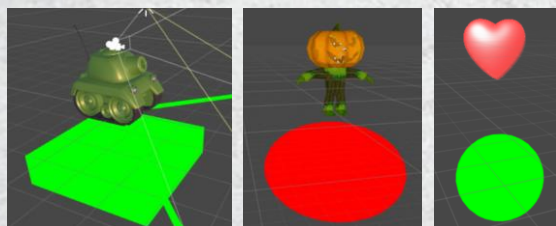
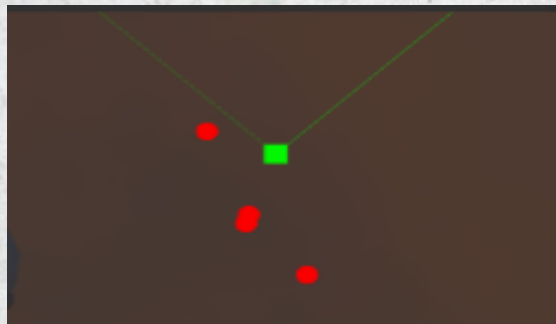
    // If the current health is less than or equal to zero...
    if(currentHealth <= 0)
    {
        // ... the enemy is dead.
        Death ();
    }
}
```

```
void Death ()
{
    // The enemy is dead.
    isDead = true;
    // Tell the animator that the enemy is dead.
    anim.SetBool("isDead",true);
    StartSinking ();
}
```

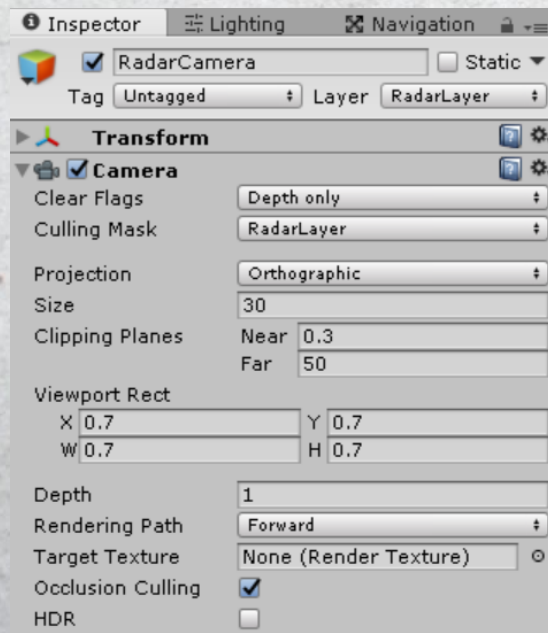
탄환("shell")이 Enemy를 타격하게 되면
Object가 shell 인지 확인을 한 후, 맞으면
Takedamage를 실행하여 죽음 애니메이션인
isDead를 실행하게 되고
Attack 때와 같은 StartSinking 메소드를 사용하여
Enemy가 사라지도록 만들었습니다.



Radar System



화면 우측 상단에 레이더 시스템을 만들어서
Player의 위치, Enemy의 위치, Heart의 위치를
바로 확인 할 수 있게 만들었습니다.



RadarLayer를 지정해 주었고


Radar Camera를 따로 만들어서 게임을 하는 중
카메라로 볼 수 있도록 만들었습니다.








Game Manager

- Enemy Manager

 ☒ EnemyManager ☐ Static ▾




Tag Layer

▼  **Transform**  

Position X Y Z

Rotation X Y Z

Scale X Y Z

▼  ☒ **Wave Spawner (Script)**  

Script

Instruction

▼ Waves

Size

▼ wave1

Name

▼ Enemy

Size

Element 0

Count

Rate

► wave2

► wave3

► wave4

► wave5


► **Spawn Points**

Time Between Wave

- Heal Manager

 ☒ HealManager ☐ Static ▾

Tag Layer

▼  **Transform**  

Position X Y Z

Rotation X Y Z

Scale X Y Z

▼  ☒ **Heal Manager (Script)**  

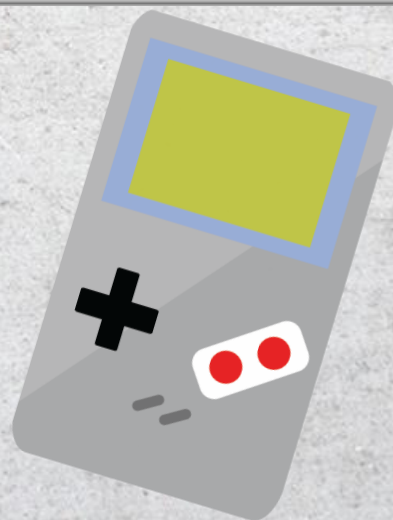
Script

Heart

Min

Max

► Heal Spawn Points





Enemy Manager



Player 주변에 Spawn point를 여러 개 만들어 놓았고 Random 함수를 사용하여 무작위로 Enemy를 생성하여 Player에게 다가가도록 만들었습니다.

```
IEnumerator SpawnWave(Wave _wave)
{
    Debug.Log ("Spawning Wave: " + _wave.name);
    state = SpawnState.SPAWNING;

    //spawn
    for (int i = 0; i < _wave.count; i++)
    {
        for (int j = 0; j < _wave.enemy.Length; j++)
        {
            instruction.text = "";
            SpawnEnemy (_wave.enemy[j] );
        }
        //SpawnEnemy (_wave.enemy2 );
        yield return new WaitForSeconds (1f / _wave.rate);
    }
    state = SpawnState.WAITING;

    yield break;
}
```

```
void SpawnEnemy(Transform _enemy)
{
    // Spawn enemy
    Debug.Log("Spawning enemy: " + _enemy.name);
    Transform _sp = spawnPoints [Random.Range (0, spawnPoints.Length)];
    Instantiate(_enemy, _sp.position, _sp.rotation);
}
```

Enemy Wave 제도로 구상을 했기 때문에 적들 사이에 생성시간을 IEnumerator로 조정하는 rate value를 추가시켰습니다.





Enemy Manager

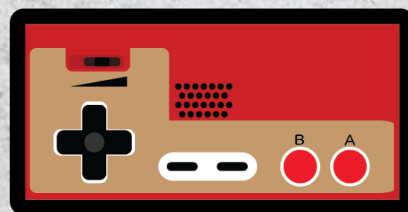
```
void WaveCompleted ()
{
    //begin a new round
    Debug.Log("Wave Completed");
    state = SpawnState.COUNTING;
    waveCountdown = timeBetweenWaves;
    //check if this is the last wave / if it is
    if (nextWave + 1 > waves.Length - 1) {
        nextWave = 0;
    } else
    {
        nextWave++;
    }
}
```

Wave를 1~5까지 나누었고, Player가 모든 enemy를 사살하게 되면 다음 wave로 넘어가게 만들었습니다.



```
//checking when enemy is alive
bool EnemyIsAlive()
{
    searchCountdown -= Time.deltaTime;
    if (searchCountdown <= 0f)
    {
        searchCountdown = 1f;
        if (GameObject.FindGameObjectsWithTag ("enemy").Length == 0)
        {
            Debug.Log ("all enemy are dead");
            return false;
        }
    }
    return true;
}
```

적들이 살아있는지를 확인하기 위해 매 frame마다 확인을 하는 것이 아닌 1초에 한번씩 확인을 하도록 만들었습니다.





Heal Manager

```
void Update()
{
    healCountdown -= Time.deltaTime;
    if (healCountdown <= 0)
    {
        SpawnHeal ();
        setRandomHealTime ();
    }
}

void setRandomHealTime()
{
    timeBetweenHeals = Random.Range (min, max);
    healCountdown = timeBetweenHeals;
}

void SpawnHeal()
{
    Transform _sp = healSpawnPoints [Random.Range (0, healSpawnPoints.Length)];
    Instantiate(heart, _sp.position, _sp.rotation);
}
```

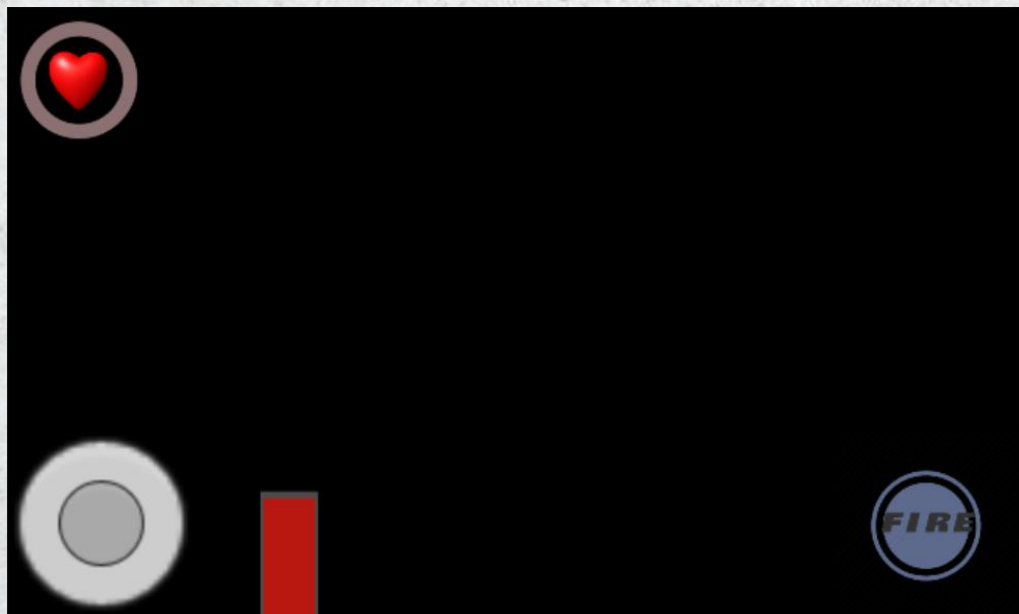


Player의 에너지를 채워줄 Heart를 위와 같이 지정된 포지션 중에서 랜덤하게 나올 수 있도록 만들었습니다.
또 heart가 나오는 시간 또한 랜덤으로 나오도록 만들었습니다.





Canvas System



▼ Canvas

- fadingImage
- ▶ HealthSlider
- DamageImage
- ▶ Background
- WaveIndexText
- ▶ VirtualJoystick
- ▶ FireButton

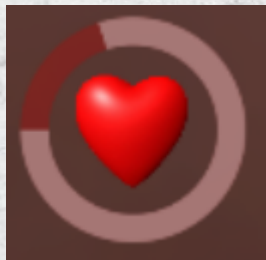
Canvas 설명

- Health Slider & Fire Button & Virtual Joy Stick
- Fading Image & Damage Image





Health Slider & Fire Button & Virtual Joy Stick



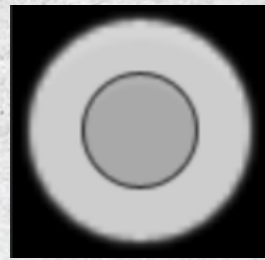
Health Slider

Enemy와 Player의 script가 amount value를 조정하여 총 100%의 slider을 (+,-)10씩 조정하도록 만들었습니다.



Fire Button

버튼을 누르면 Player shooting 스크립트에 서 버튼이 down, up을 확인하여 탄환(shell)을 발사하게 됩니다.



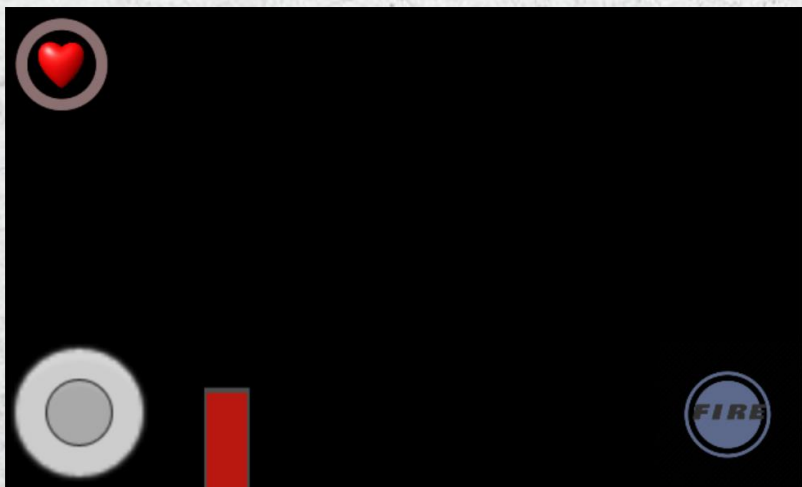
Joy Stick

좌우로만 움직이도록 만들어 놓았고 감도를 조정하여 중심과 멀어질 수록 player의 시선을 더 빨리 움직이게 만들었습니다.





Fading Image & Damage Image



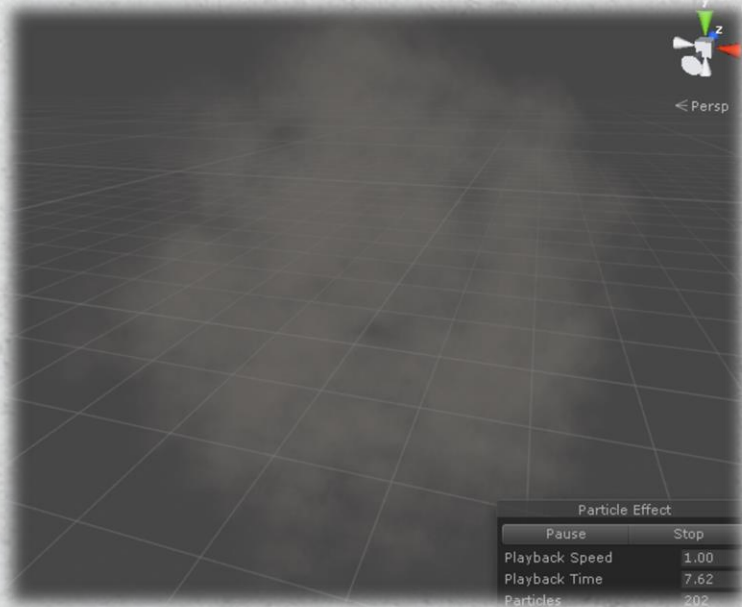
Fading Image

시작 또는 game over 시에 검정색으로 fade in, fade out을 하도록 만들었습니다.

Damage Image

Player가 공격을 당했을 때나 힐을 했을 때에 색이 붉은색, 초록색으로 바뀌도록 만들었습니다.





FOG

유니티의 Lighting 시스템과 Particle 시스템을 사용하여 안개를 만들어서 게임의 분위기를 조금 더 흥미롭게 만들었습니다.



Zombie

재미 요소를 위하여 가장 어려운 난이도인 Wave 5 에 가장 어려운 enemy가 나오도록 만들었습니다.





THE END
THANK U

**INSERT COINS
TO CONTINUE**