ESCape

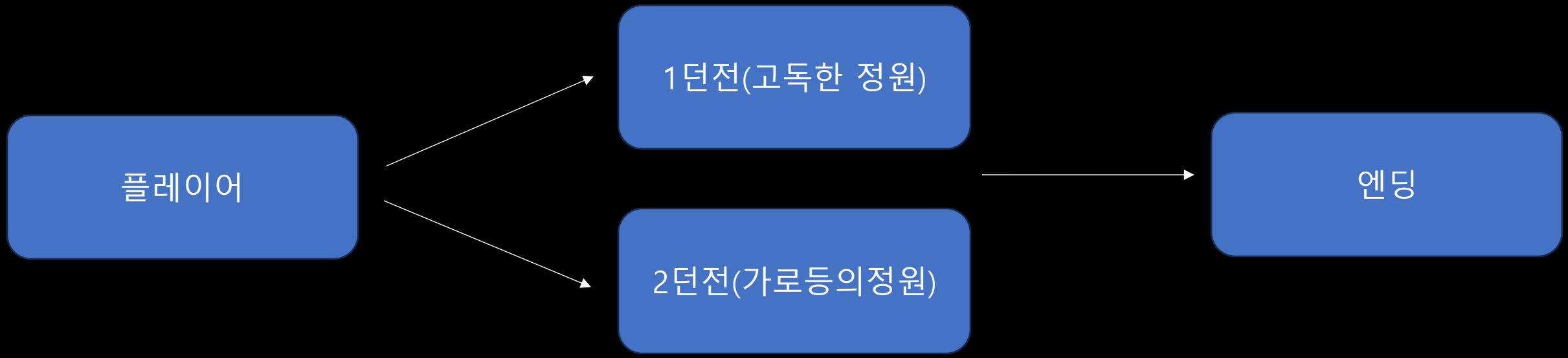
배재영
최정훈
이상혁
이보람

게임의 장르

2D 도트

액션 게임

게임의 목표



역할 분담

배재영	최정훈	이상혁	이보람
기획 보스 메인메뉴 & UI 수집요소 & 엔딩 데이터 매니저	플레이어 무기시스템 오브젝트 풀링 사운드 매니저 미니맵 UI	맵 서버	몬스터

Player부분

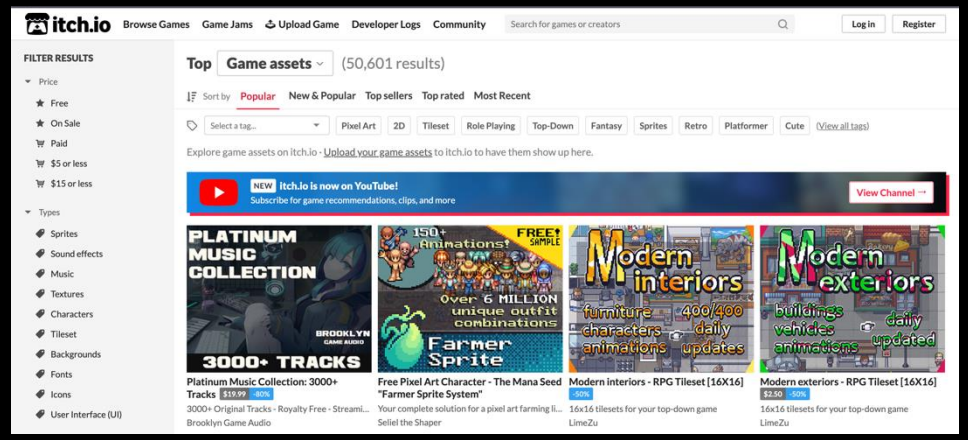
20191681 최정훈

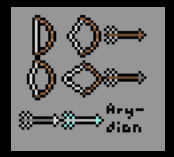
목차

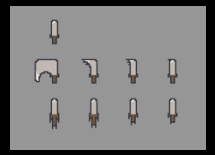
- -Player
- -Object Pooling
- -Sound Manager

- -MiniMap (UI부분으로 인계)

사용한 애셋







- itch.io 사이트 활용

유니티 애셋스토어 이외에도 애셋들을 구할 수 있는 사이트가 많음.

본인이 맡은 부분에서 사용한 애셋들은 모두 해당 사이트에서 제공하는 무료 애셋이다.

플레이어 및 무기 이미지 (https://analogstudios.itch.io/playersprites)

피격 이펙트(https://nyknck.itch.io/fx062)

Player

- 상태패턴으로 구현 – (IDLE, MOVE, STUNNED, DASH, DEAD)
- Player – IDamageAble 인터페이스 적용
- 3가지의 무기(칼, 활, 총) 사용 가능

Player - 상태패턴

- 게임 내 캐릭터들은 다수의 상태(State)를 보유하고 있다 – ex) 달리기, 점프, 공격
- 이러한 상태들을 열거형으로 관리하면 코드의 가독성 증가, 각 상태가 어떤 동작을 하는지 쉽게 파악이 가능하다

IDEL

MOVE

STUNNED

DASH

DEAD

Player - 상태패턴

```
void Update()
{
    Move();
    Dash();
    Idle();
    Dead();
}
```

- 기존의 방식

상태들의 전환점을 if문으로 구분하여 각각의 매서드를 생성하고 이를 Update()문에서 한꺼번에 호출하는 방식

개발이 진행됨에 있어서 상태들이 많이 추가된다면, 조건분기점을 잘못 설정할 가능성도 있고, Update문의 길이가 길어져서 가독성이 떨어질 가능성 존재

상태패턴을 적용해본다면?

Player - 상태패턴

```
void Update()
{
    if (!isPlaying)
    {
        return;
    }

    _stateMachine.Action();

    mousePos = mainCam.ScreenToWorldPoint(Input.mousePosition);
    CurserUpdate();
    MarkCheck();
}
```

- 사용하는 방식

모든 상태의 전환을 _stateMachine.Action()이라는 코드 한 줄로 끝 마침

또한 추후에 상태가 추가된다고 하더라도, Update문의 길이자체는 변함이 없기 때문에 가독성도 뛰어날 것이다.

Player - 상태패턴

```
public class Move : PlayerState
{
    public Move() {}

    Frequently called  1 usage
    public Move(PlayerStateType stateType)
    {
        stateType = PlayerStateType.Move;
    }

    Frequently called  0+1 usages
    public override void Enter(Player player)
    {
        player.anim.SetBool(name: "isMoving", value: true);
    }

    Frequently called  0+1 usages
    public override void Update(Player player)
    {
        player.Move();
        player.SetDash();
        player.PotionHeal();
    }

    Frequently called  0+1 usages
    public override void Exit(Player player)
    {
        player.anim.SetBool(name: "isMoving", value: false);
    }
}
```

- Move상태

- 각각 Move상태에 진입(Enter), 진행(Update), 탈출, (Exit)하였을 때 변화

- Move상태가 진행되는 동안 어떤 행동을 할 수 있는지

Player – IDamageAble 인터페이스 적용

```
public interface IDamageAble
{
    3 usages  9 implementations
    public void TakeDamage(int damage, Transform attacker);

}
```

- 플레이어, 몬스터, 보스 모두 데미지를 받는다 => 인터페이스를 적용해서 간편하게 로직 통합

어떤 물체든지 데미지를 받게하려면, TakeDamage()를 호출하면 된다.

```
public class Player : MonoBehaviour, IDamageAble
```

```
public void TakeDamage(int damage, Transform other)
{
    if (_stateType == PlayerStateType.Dead)
    {
        return;
    }

    if (playerHP > damage)
    {
        SoundManager.instance.SFXPlay(sfxName: "DamagedSFX", damageSFX);
        hitPS.Play();
        playerHP -= damage;
        anim.SetTrigger(name: "isHurt");
        StartCoroutine(routine: Knockback(other));
        StartCoroutine(methodName: "OnDamage");
        StartCoroutine(methodName: "AlphaBlink");
    }
    else if(playerHP <= damage)
    {
        SoundManager.instance.SFXPlay(sfxName: "deadSFX", deadSFX);
        playerHP = 0;
        Dead();
    }
}
```

```
public class Boss1 : MonoBehaviour, IDamageAble
```

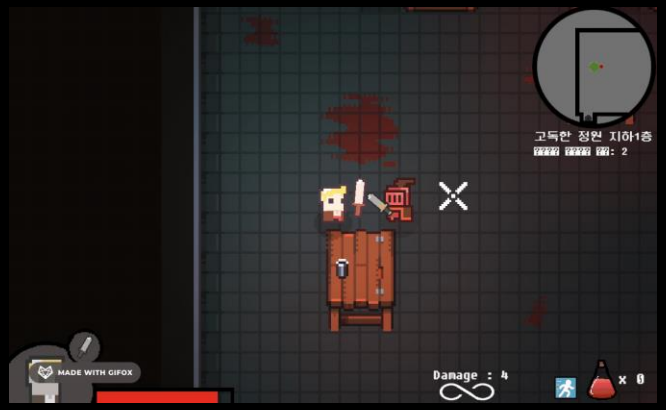
```
public void TakeDamage(int damage, Transform other)
{

        bossHP -= damage;
        StartCoroutine(methodName: "AlphaBlink");
        hpui();

    if (bossHP <= 0)
    {
        bossHP = 0;
        statemachine.SetState(states[Boss1state.Dead]);
    }
}
```

```
public class Enemy : MonoBehaviour, IDamageAble
```

```
0+3 usages  ryu31847 +1
public void TakeDamage(int damage, Transform other)
{
    // Health -= (int)damage;

    if (!isDead)
    {

        Health -= (int)damage;
        SoundManager.instance.SFXPlay(sfxName: "hit_sfx", hit_SFX);
        animator.SetTrigger(name: "Hit");

        if (Health <= 0)
        {
            Destroy(Collider);
            SoundManager.instance.SFXPlay(sfxName: "dead_sfx", dead_SFX);
            animator.SetBool(name: "IsDead", value: true);
            //Dead();
        }
    }

}
```

Player - 무기사용

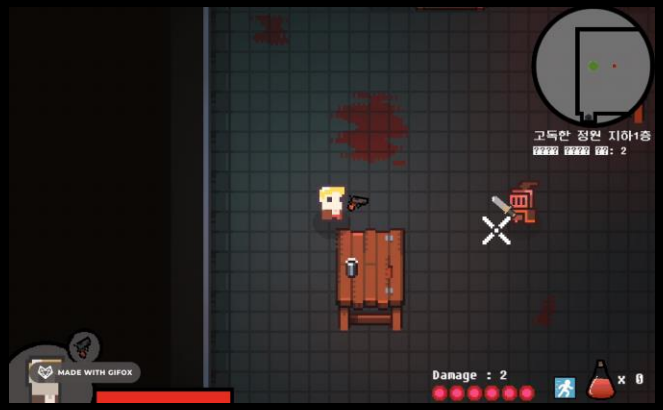
이펙트가 있으면 좋을 것 같다는 교수님의 피드백 반영!



- 칼

4 데미지

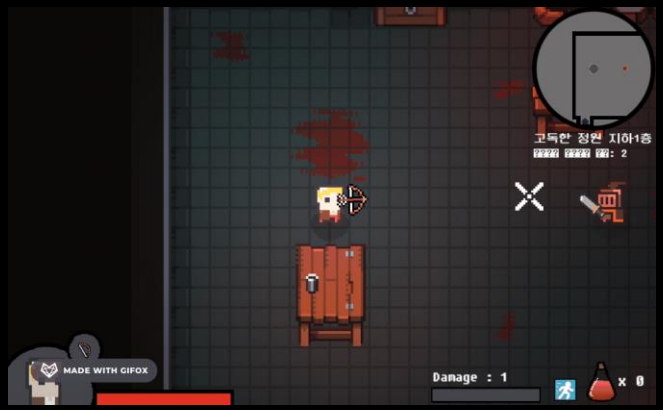
공격시 Trail Lender속성을 활성화 해서 궤적을 표현



- 총

2 데미지

Particle System을 통해서 총구 이펙트 표현



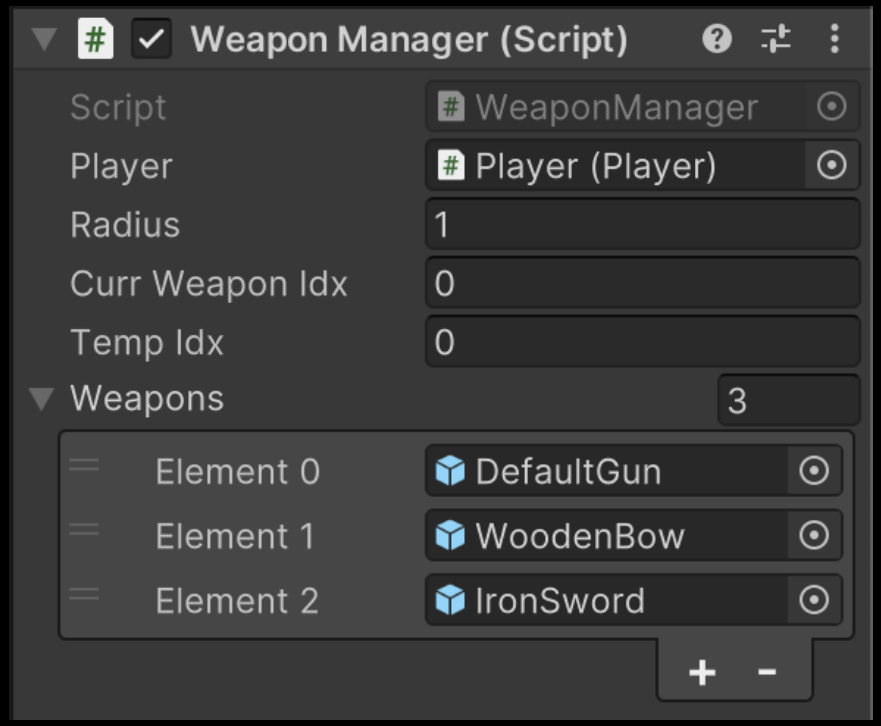
- 활

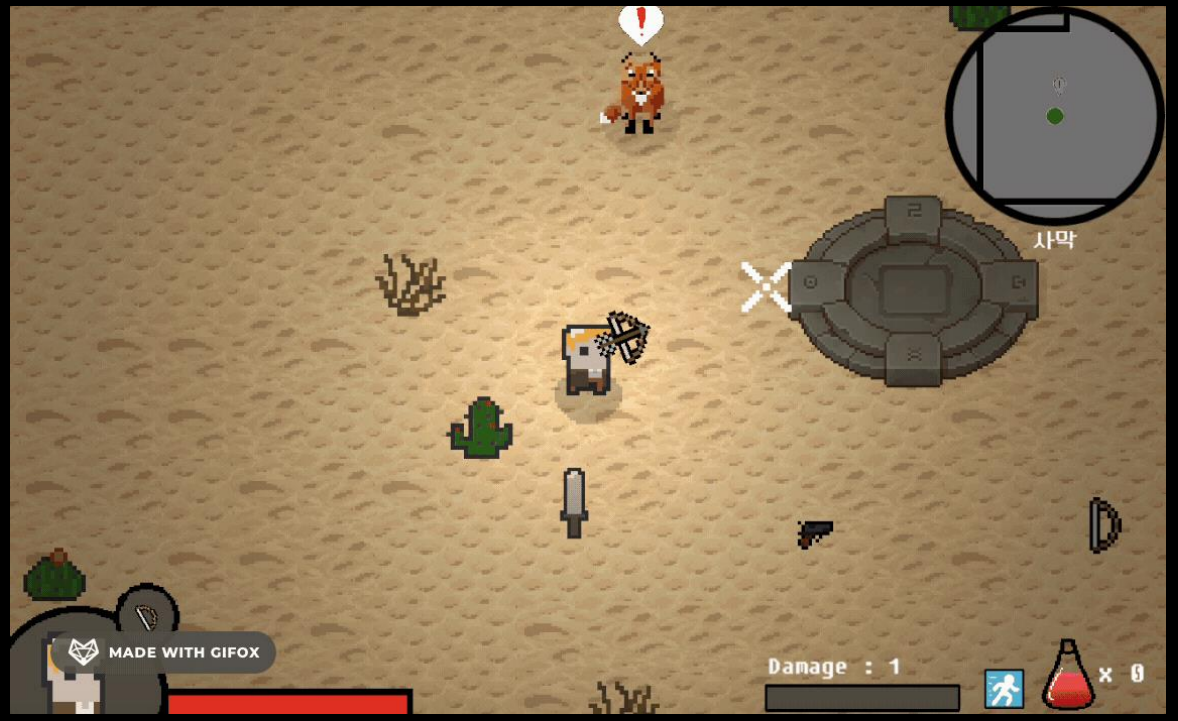
1 ~ 5 데미지 (충전시간에 따라)

화살이 남아있음

화살에 Trail Lender속성을 추가해서 날아가는 이펙트 추가

Player – 무기교체



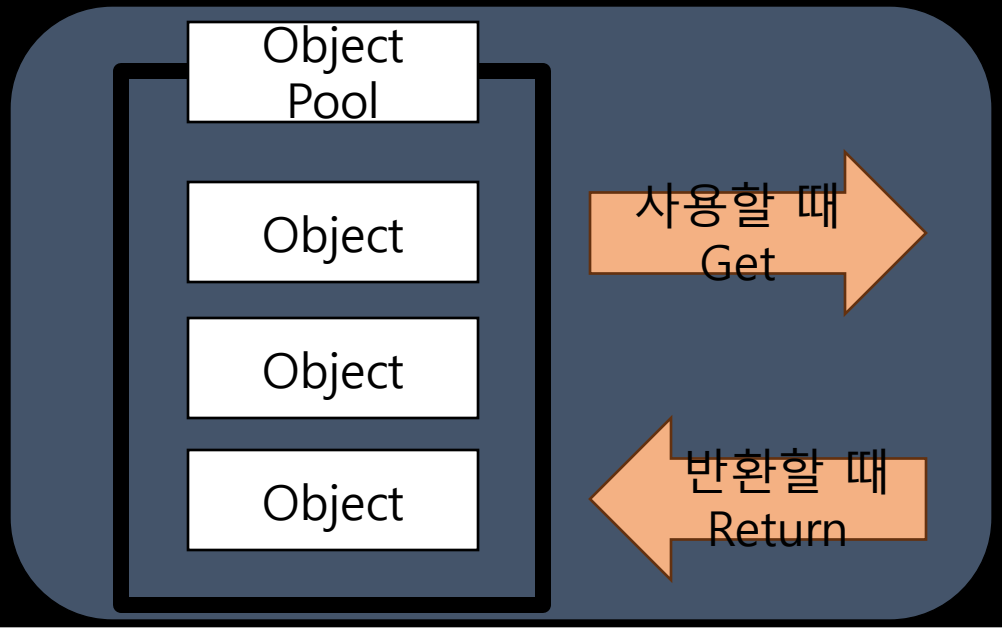


- Weapons라는 list에 전체 무기들을 넣어주고 idx 번호로 SetActive ON/OFF하는 방식

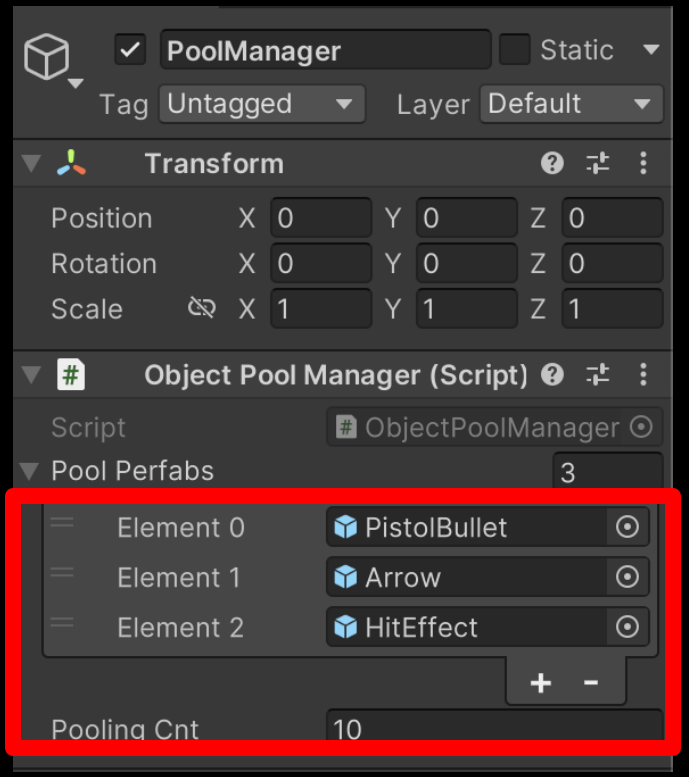
Object Pooling

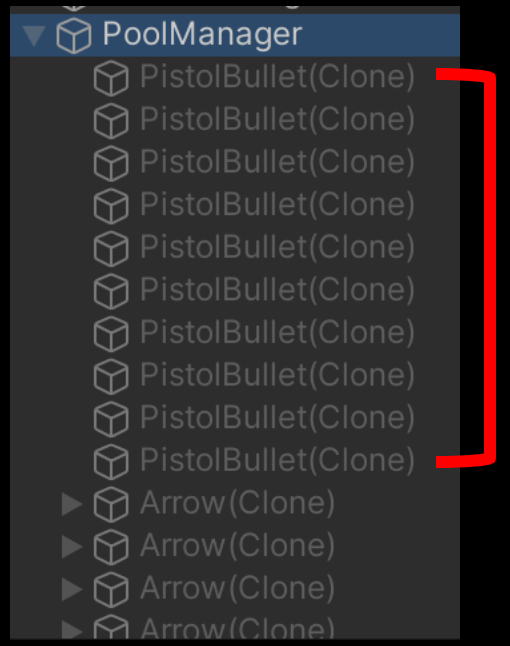
- Instansiate, Destroy를 통한 생성/파괴는 생각보다 부담이 크다, '메모리 단편화'의 가능성
  => 이를 방지하기 위해서 도입

- 총알, 화살과 같이 자주 생성하는 Object들을 보관함(Pool)에 보관했다가 필요하면 꺼내서 사용하는 방식



Object Pooling

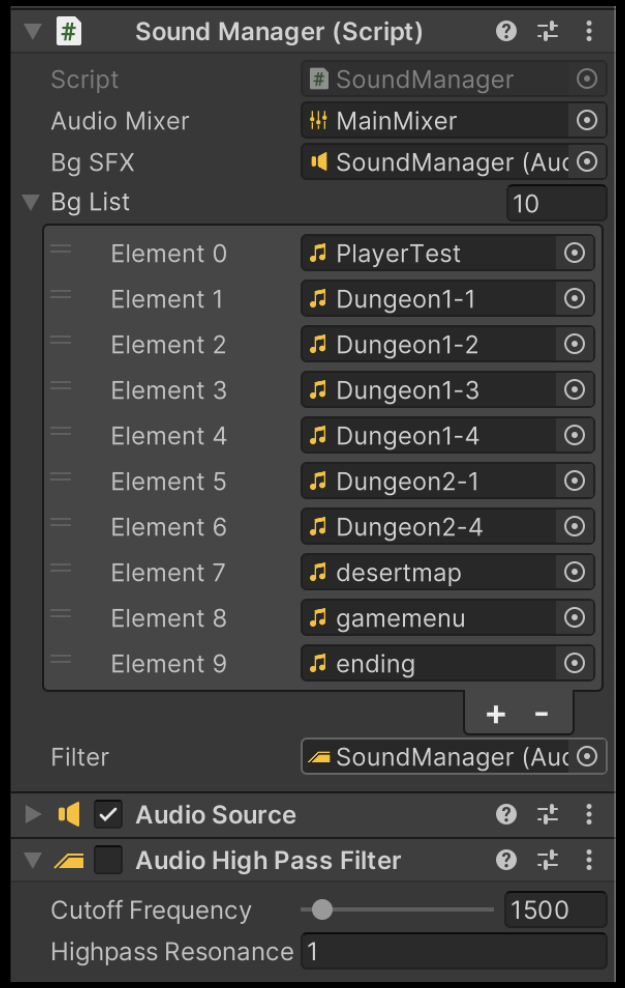




- 게임시작시 Pooling Cnt만큼 비활성화 된 상태로 생성.

사용해야 하는 순간이 오면 활성화
사용이 끝나면 다시 비활성화

Sound Manager

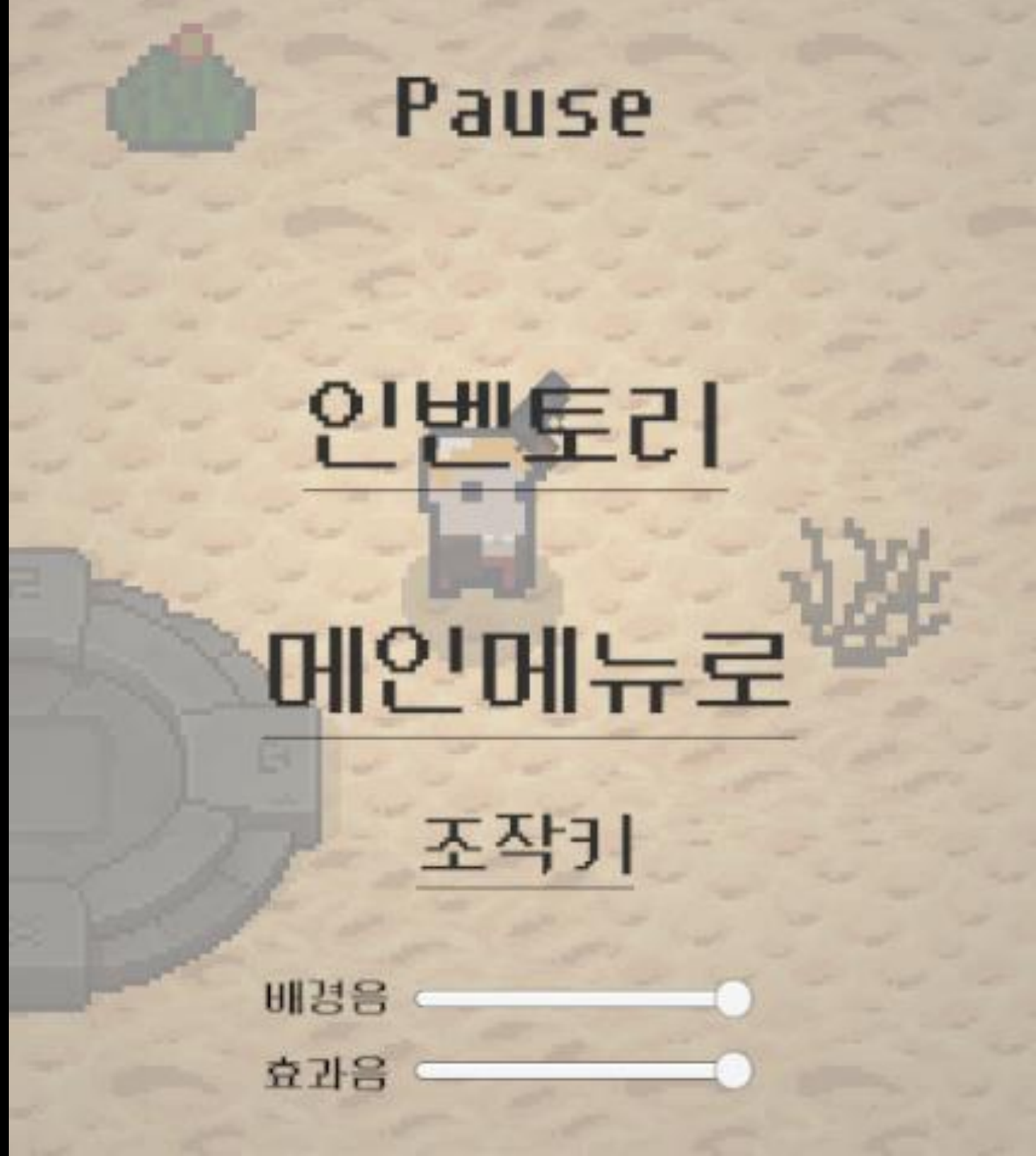


- 4가지의 기능

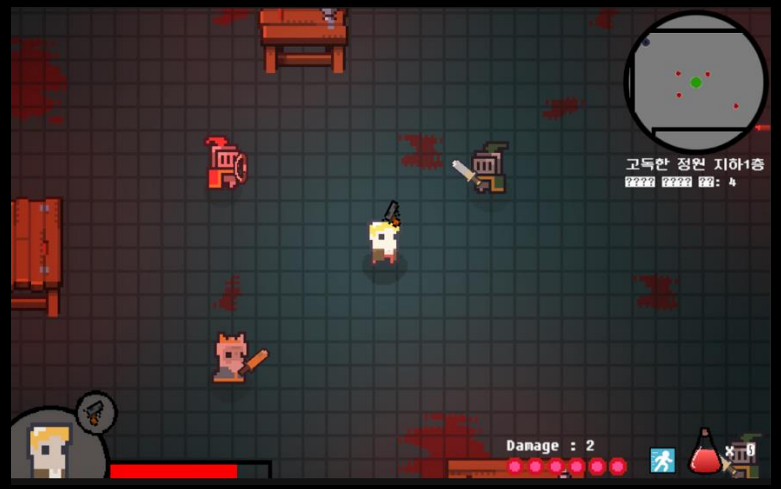
1. 각 씬에 맞는 배경음악 재생
2. 효과음 재생 ex) 재장전 소리, 피격소리 등등
3. 볼륨 조절
4. 필터링 기능을 통해서 배경음악에 변화를 줌 (ESC를 눌러서 게임을 정지했을 때)
   => 큰 체감은 아니지만 몰입도에 차이를 준다고 생각

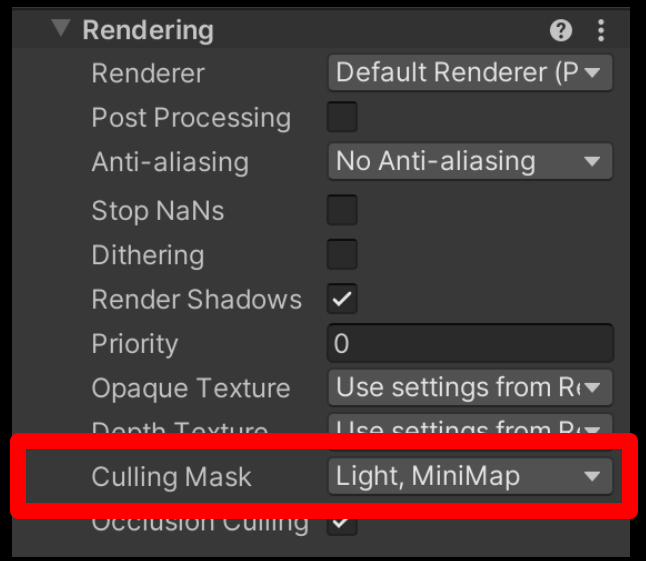
Sound Manager

- 2개의 슬라이더를 통해서 볼륨조절이 가능하다
- 배경음 : 각 씬마다 있는 고유의 배경음악
- 효과음 : 배경음을 제외한 나머지 음향



MiniMap (UI부분으로 인계)





- Camera의 Culling Mask기능을 활용

- MiniMap만 보여주는 카메라를 비치하여 UI에 보여줌

- 맵에서 플레이어의 위치 및 몬스터, 보스의 위치를 파악가능

- GameObject의 Layer중에서 Light, Minimap Layer를 가지는 Object들만 보여줌

- Light(광원)를 따로 두어야 하나?
=> Light를 포함하지 않는 경우, 미니맵 카메라가 온통 검은색이다. 3D에서와 달리 2D에서는 빛도 하나의 Layer를 차지한다