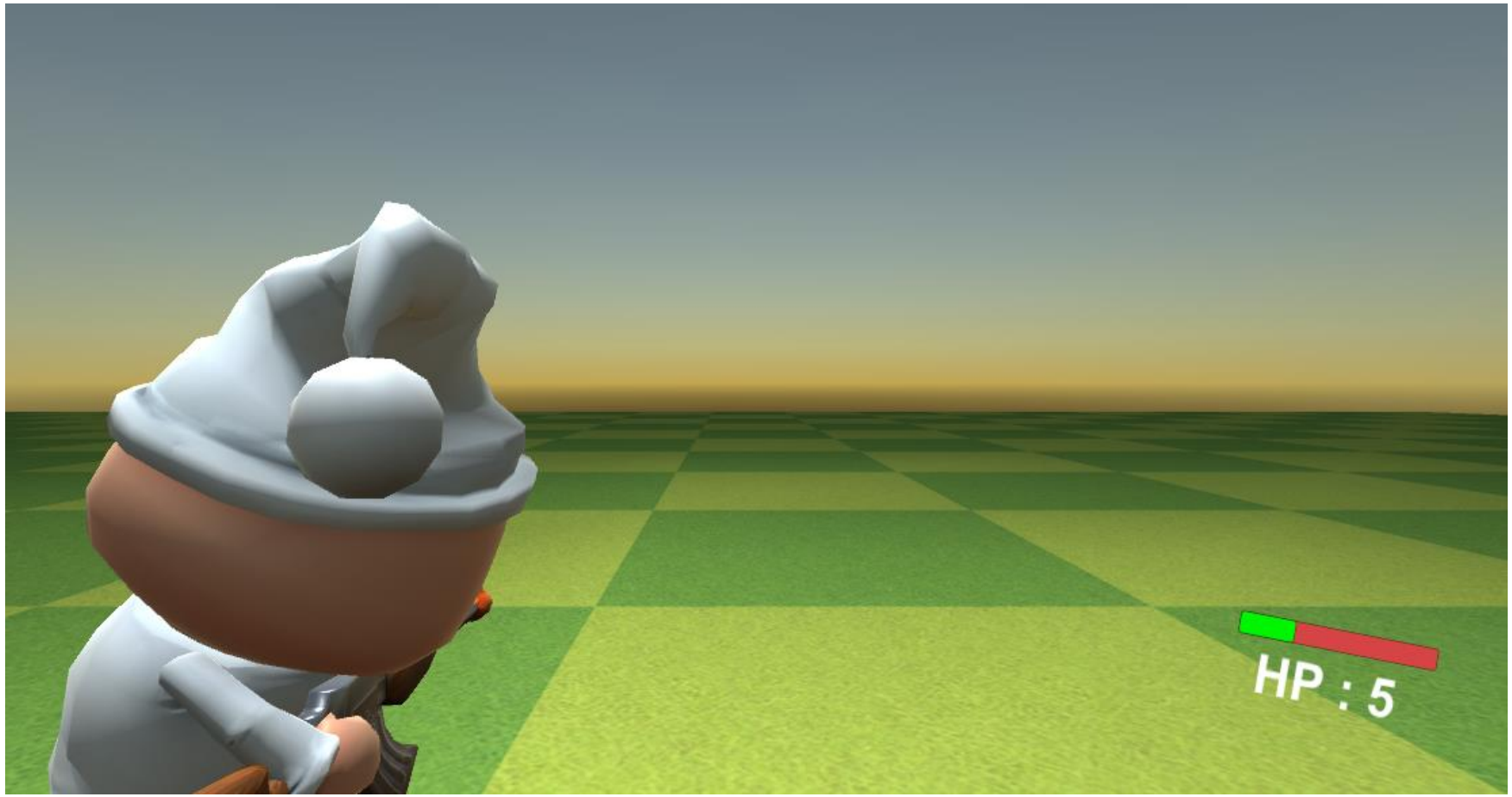
Unity – Damage Process

NHN NEXT
서형석

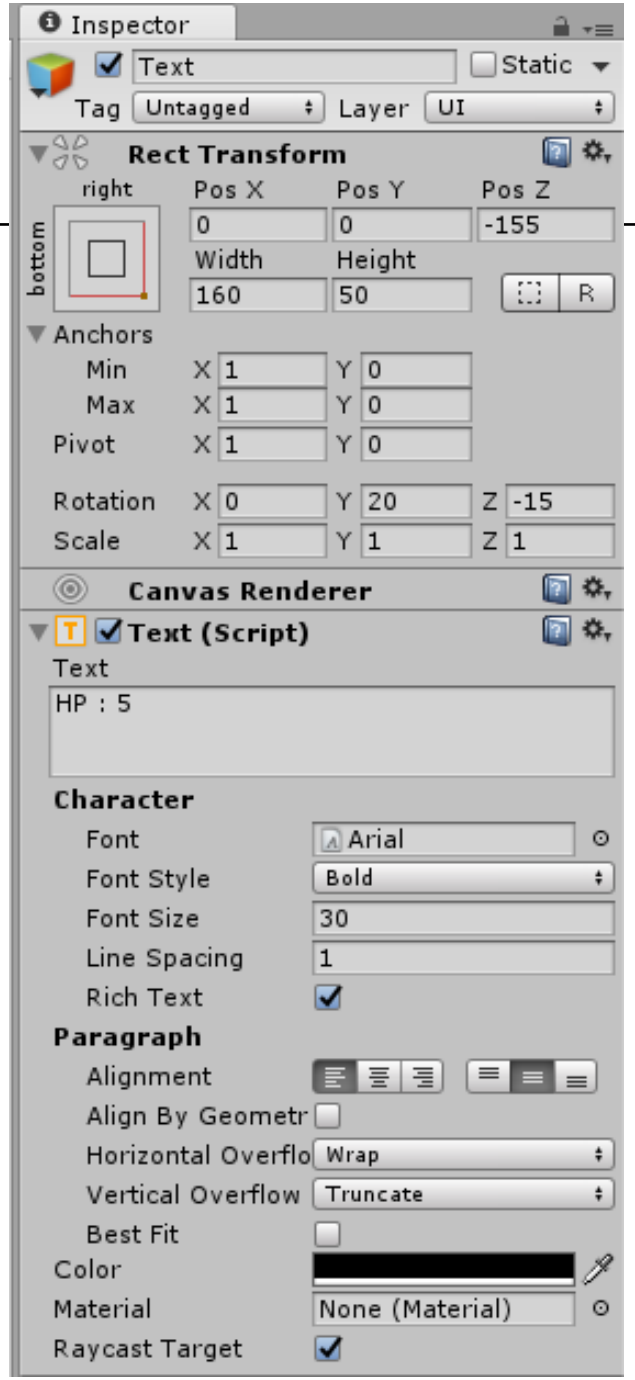
Unity

- 피격 처리를 위한 체력 UI 구성



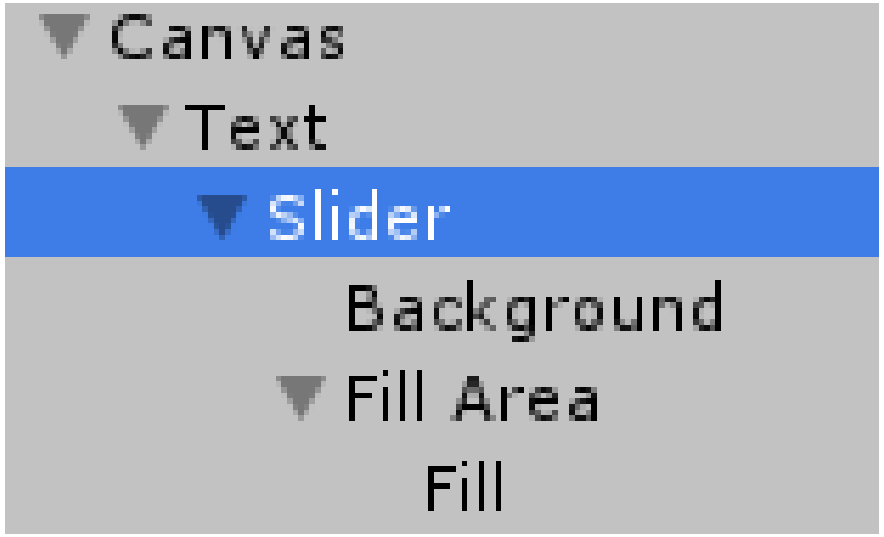
Unity

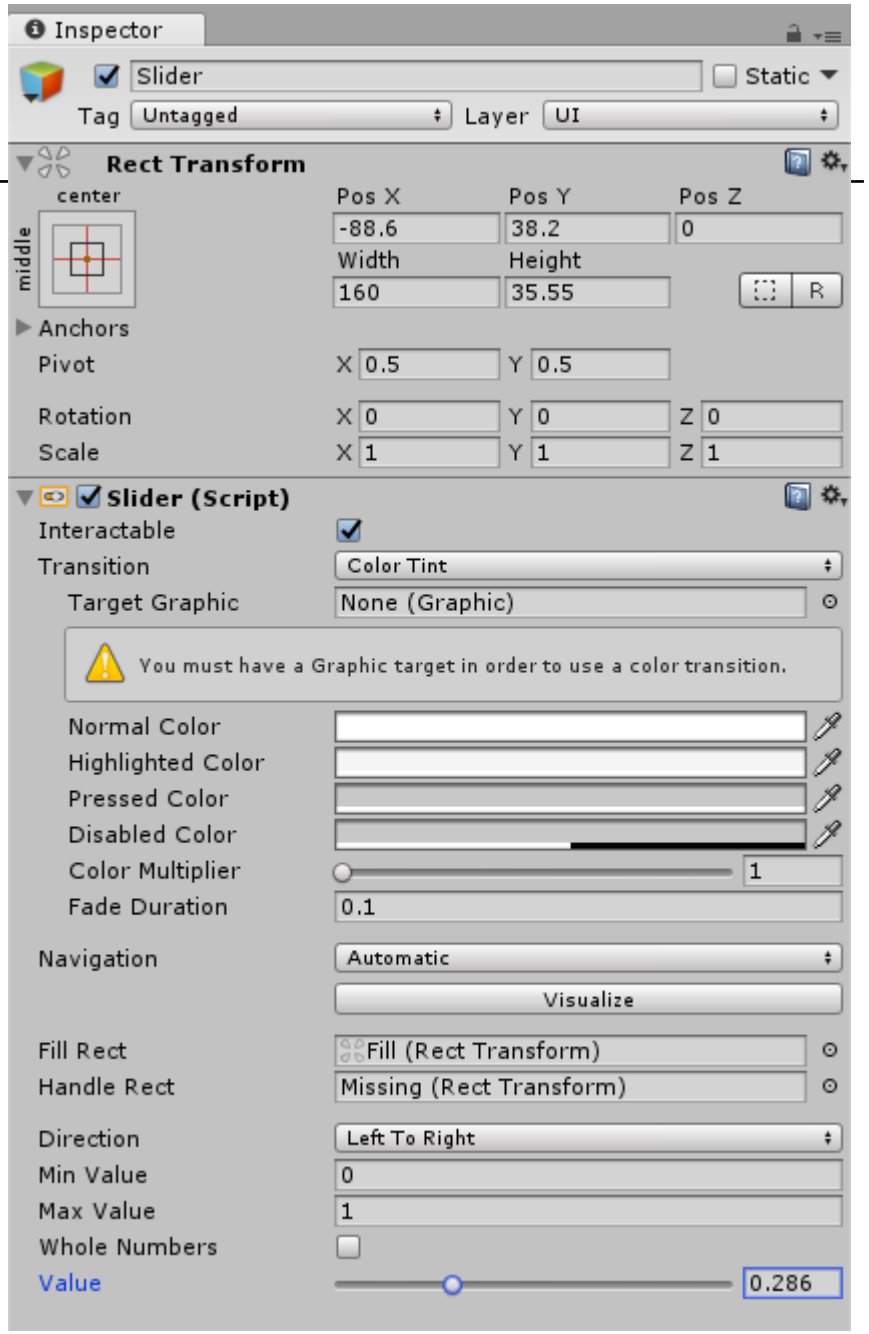
- 피격 처리를 위한 체력 UI : Text 추가



Unity

- 피격 처리를 위한 체력 UI : Slider 추가





Unity

- PlayerState : 체력 처리 스크립트

```
public class PlayerState : MonoBehaviour
{
    public   UnityEngine.UI.Text    hpText;
    public   UnityEngine.UI.Slider  hpSlider;

    public   int   healthPoint = 5;
    public   int   maxHealthPoint = 5;
}
```

Unity

- PlayerState : 체력 감소용 함수 구현

```
public class PlayerState : MonoBehaviour
{
    ……

    public  void  DamageByEnemy()
    {
        --healthPoint;
        hpText.text = "HP : " + healthPoint;
        hpSlider.value = (float)healthPoint / maxHealthPoint;
    }
}
```

Unity

- 피격 함수 호출

```
public class Zombie : MonoBehaviour
{
    ……
    Transform    target = null;
    PlayerState   playerState = null;

    void    Start()
    {
        target   =   GameObject.Find( “Player” ).transform;
        playerState = target.GetComponent< PlayerState >();
    }
}
```

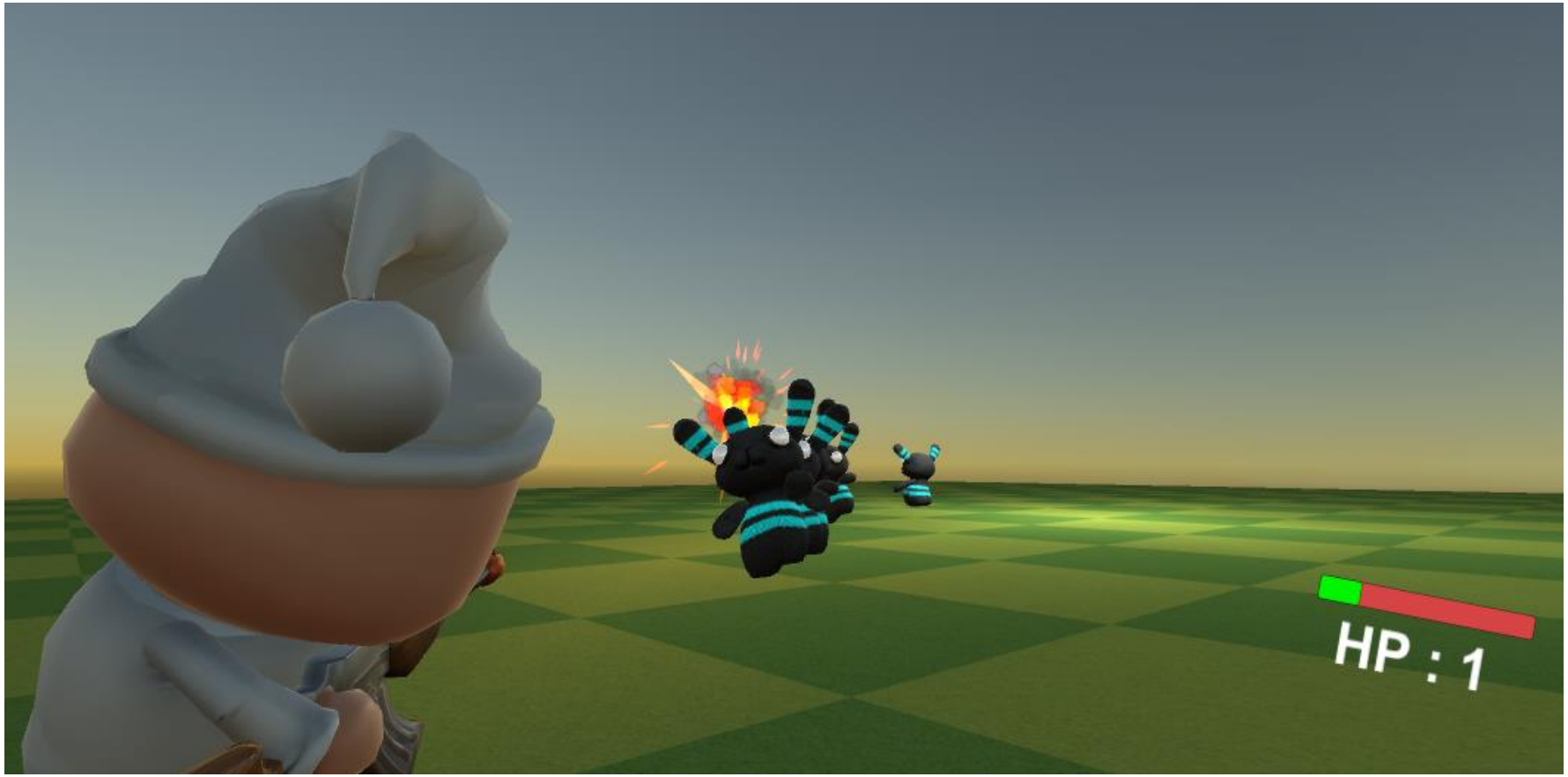
Unity

- 피격 함수 호출

```
public class Zombie : MonoBehaviour
{
    ……
    case ENEMYSTATE.ATTACK:
    {
        stateTime += Time.deltaTime;
        if( stateTime > attackStateMaxTime )
        {
            ……
            playerState.DamageByEnemy();
        }
    }
    break;
}
```

Unity

- 체력 수치 확인

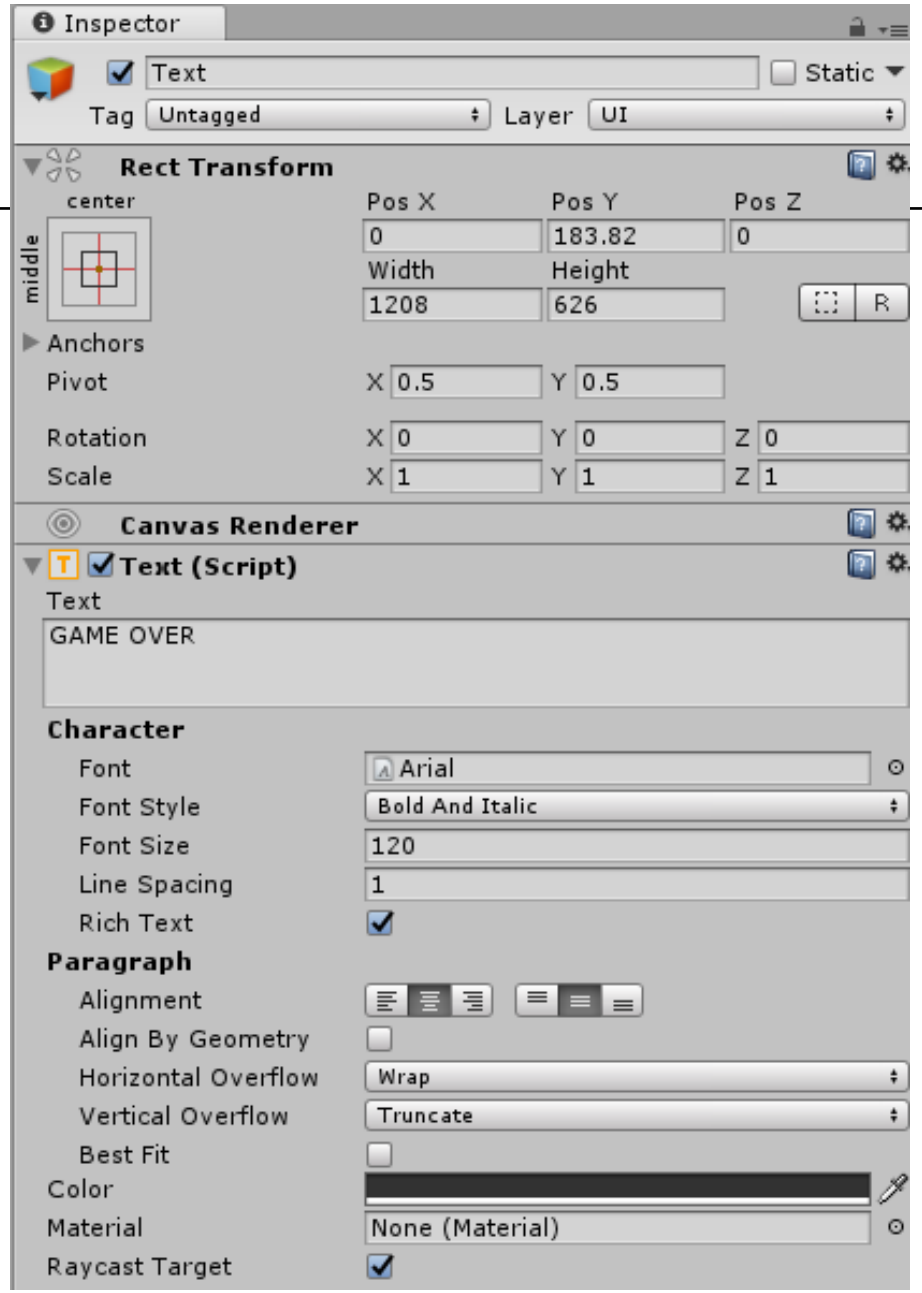


Unity

- 게임 오버용 UI 추가

Unity

- 게임 오버용 Text UI 추가



```
public class PlayerState : MonoBehaviour
{
    ......
    public  bool  isDead = false;
    public GameObject  gameOverText;

   void Awake()
   {
      gameOverText.SetActive(false);
   }

   public   void   DamageByEnemy()
   {
      if (isDead)
         return;

      --healthPoint;
      hpText.text = "HP : " + healthPoint;
      hpSlider.value = (float)healthPoint / maxHealthPoint;

      if (healthPoint <= 0)
      {
         isDead = true;
         gameOverText.SetActive(true);
      }
   }
}
```

상태별 처리 구조 구현

Unity

- 동작 확인

Unity

- 생각해 볼 점 :
  죽었을 때 캐릭터 이동 및 카메라 회전, 포탄 발사는
  적절치 않을 수 있다.
  죽었을 때에 대한 구현 방안을 고민해 볼 것.

Unity

- 죽을때의 상태 처리

```
public class PlayerMove : MonoBehaviour
{
    ......
    PlayerState    playerState = null;

    void Start()
    {
        characterController = GetComponent< CharacterController >();
        playerState  =  GetComponent< PlayerState >();
    }


    ......
    void Update()
    {
         if( playerState.isDead )
              return;
         ......
    }
}
```

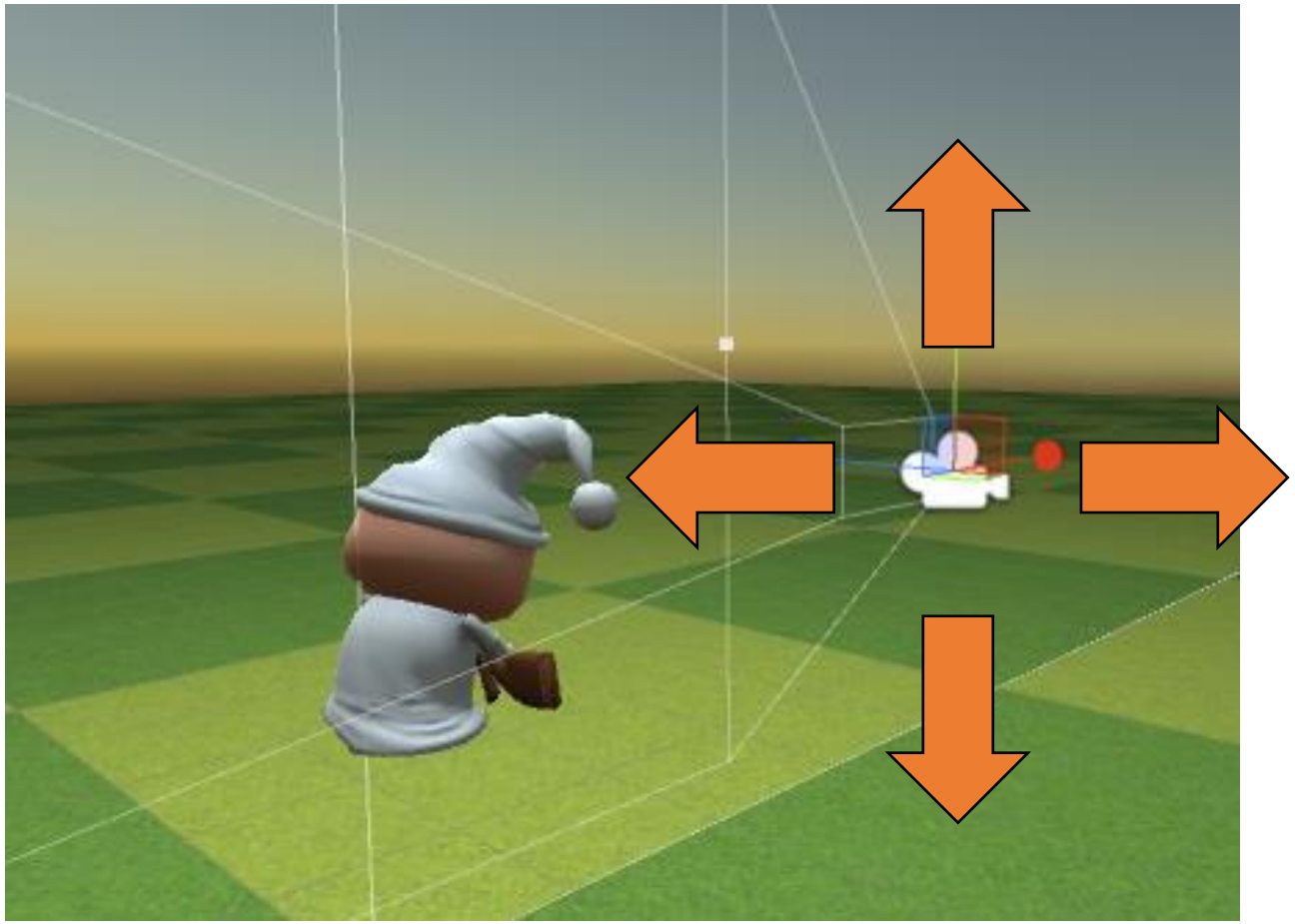
Unity

- 죽을때의 상태 처리

```
public class PlayerState : MonoBehaviour
{
    ......
    public    FireBall    fireBall;
    public    CameraControl     cameraControl;

   public   void   DamageByEnemy()
   {
      ......

      if (healthPoint <= 0)
      {
         isDead = true;
         gameOverText.SetActive(true);

         fireBall.enabled = false;
         cameraControl.enabled = false;
      }
   }
}
```

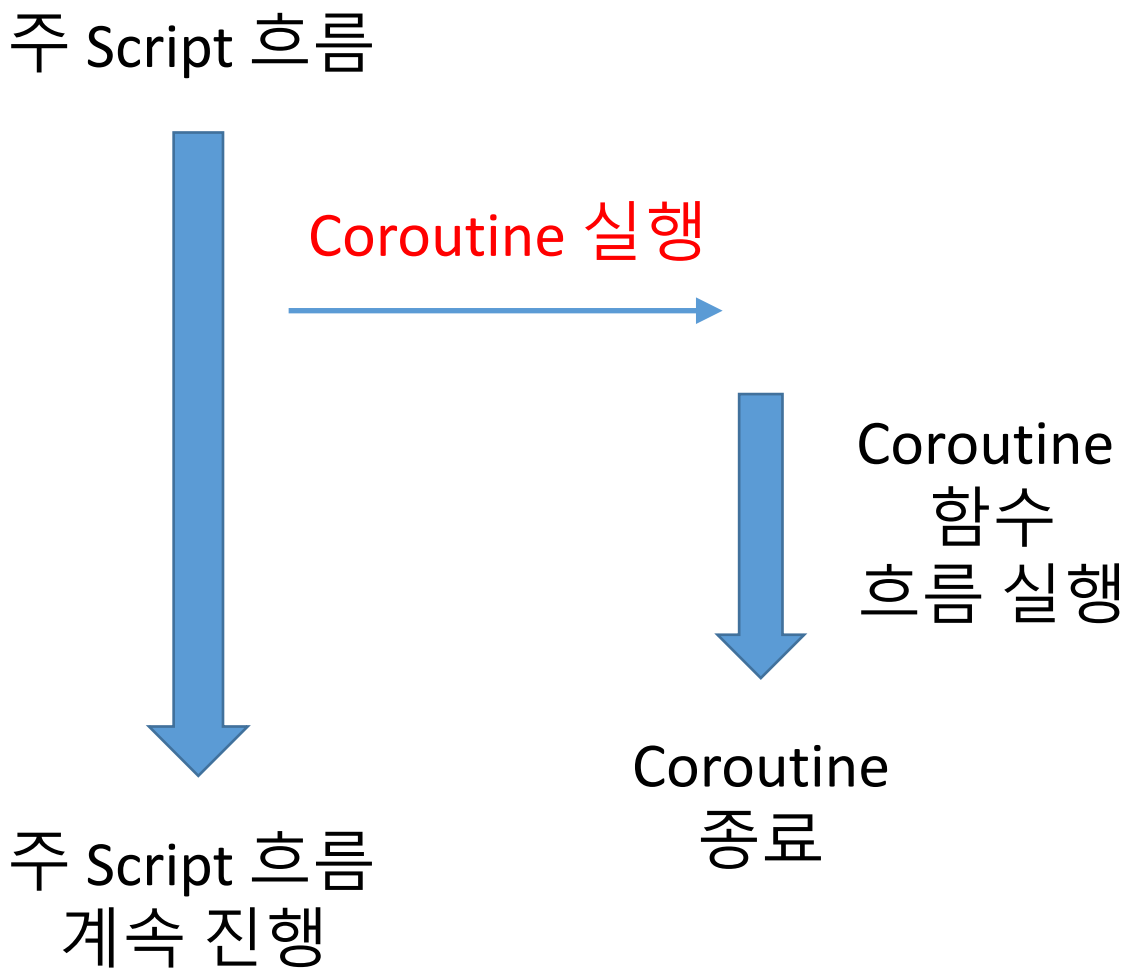
Unity

- 카메라 흔들기

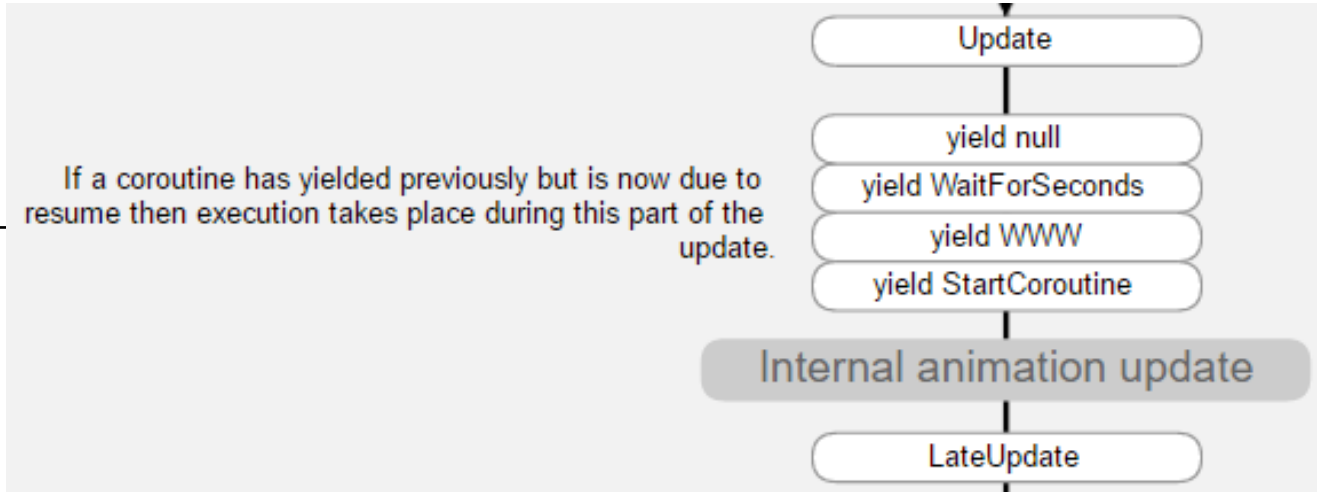


Unity

- 코루틴


주 Script 흐름
Coroutine 실행
Coroutine
함수
흐름 실행
Coroutine
종료
주 Script 흐름
계속 진행

Unity



- 코루틴

  일시적으로 생성한 별도의 흐름

  https://docs.unity3d.com/Manual/ExecutionOrder.html

  이해하기 쉬운 개념 : 쓰레드 처럼 생각해도 무방
  실제 동작 : LateUpdate 단에서 호출되는 흐름

Assets/Script/CameraShake.cs(19,17): error CS0161: `CameraShake.CameraShakeProcess(float, float)': not all code paths return a value

Unity

- Camera Shake

```
public class CameraShake : MonoBehaviour
{
    Vector3  localPosition  =  Vector3.zero;

    void  Start()
    {
        localPosition  =  transform.localPosition;
    }

   public  void  PlayCameraShake()
   {
      StopAllCoroutines();
      StartCoroutine(  CameraShakeProcess(1.0f, 0.2f) );
   }
```

```
    IEnumerator CameraShakeProcess( float shakeTime, float shakeSense )
    {
        float deltaTime = 0.0f;

        while( deltaTime < shakeTime )
        {
            deltaTime  +=  Time.deltaTime;

            transform.localPosition  =  localPosition;
            Vector3  pos  =  Vector3.zero;
            pos.x  =  Random.Range( -shakeSense, shakeSense );
            pos.y  =  Random.Range( -shakeSense, shakeSense );
            pos.z  =  Random.Range( -shakeSense, shakeSense );
            transform.localPosition  +=  pos;

            yield  return  new  WaitForEndOfFrame();
        }

        transform.localPosition  =  localPosition;

        yield  return  null;
    }
}
```

Unity

- 코루틴 반환값

코루틴용 데이터	엔진이 수행하는 기능
yield return null	다음 프레임까지 대기
yield return new WaitForSeconds(float)	지정된 초 만큼 대기
yield return new WaitForEndOfFrame()	모든 렌더링작업이 끝날 때까지 대기
yield return new WaitForFixedUpdate()	다음 물리 프레임까지 대기까지 대기
yield return StartCoRoutine(string)	다른 코루틴이 끝날 때까지 대기
yield return new WWW(string)	웹 통신 작업이 끝날 때까지 대기
yield return new AsyncOperation	비동기 작업이 끝날 때까지 대기 (씬로딩)

Unity

- 카메라 쉐이크 호출

```
public class PlayerState : MonoBehaviour
{
    .....
    CameraShake    cameraShake   =   null;

    void Start()
    {
       cameraShake =
            GetComponentInChildren<CameraShake>();
    }
```

```
    public void DamageByEnemy()
    {
        if( isDead )
            return;

        --healthPoint;

        cameraShake.PlayCameraShake();

        if (healthPoint <= 0)
        {
            isDead = true;
        }
    }
}
```

Unity

- 코루틴 복습 : 죽은 흔적 구현

```
public class Zombie : MonoBehaviour
{
    enum   ENEMYSTATE
    {
        NONE = -1,
        IDLE = 0,
        MOVE,
        ATTACK,
        DAMAGE,
        DEAD
    }

    ENEMYSTATE    enemyState    =    ENEMYSTATE.IDLE;
}
```

Unity

- 죽음 처리

```
case ENEMYSTATE.DEAD:
{
    // Destroy( gameObject );
    StartCoroutine( "DeadProcess" );
    enemyState = ENEMYSTATE.NONE;
}
break;
```

Unity

- 죽음 처리시 포탄 처리가 진행되지 않도록 수정

```
void OnCollisionEnter( Collision   collision   )
{
    if(  enemyState   ==  ENEMYSTATE.NONE   ||
         enemyState   ==  ENEMYSTATE.DEAD )
         return;


    ……
}
```

Unity

- 죽음 처리 진행 1

```
    IEnumerator DeadProcess()
    {
        CancelInvoke();
        characterController.enabled = false;    // 죽은 후 충돌처리 여부

        anim[ “Death" ].speed = 2.0f;
        anim.Play( "Death" );

//        yield return new WaitForSeconds(   anim[ “Death" ].Length / 2.0f    );
        while(  anim.isPlaying  )
        {
            yield return new WaitForEndOfFrame();
        }

        Destroy( gameObject );
    }
```

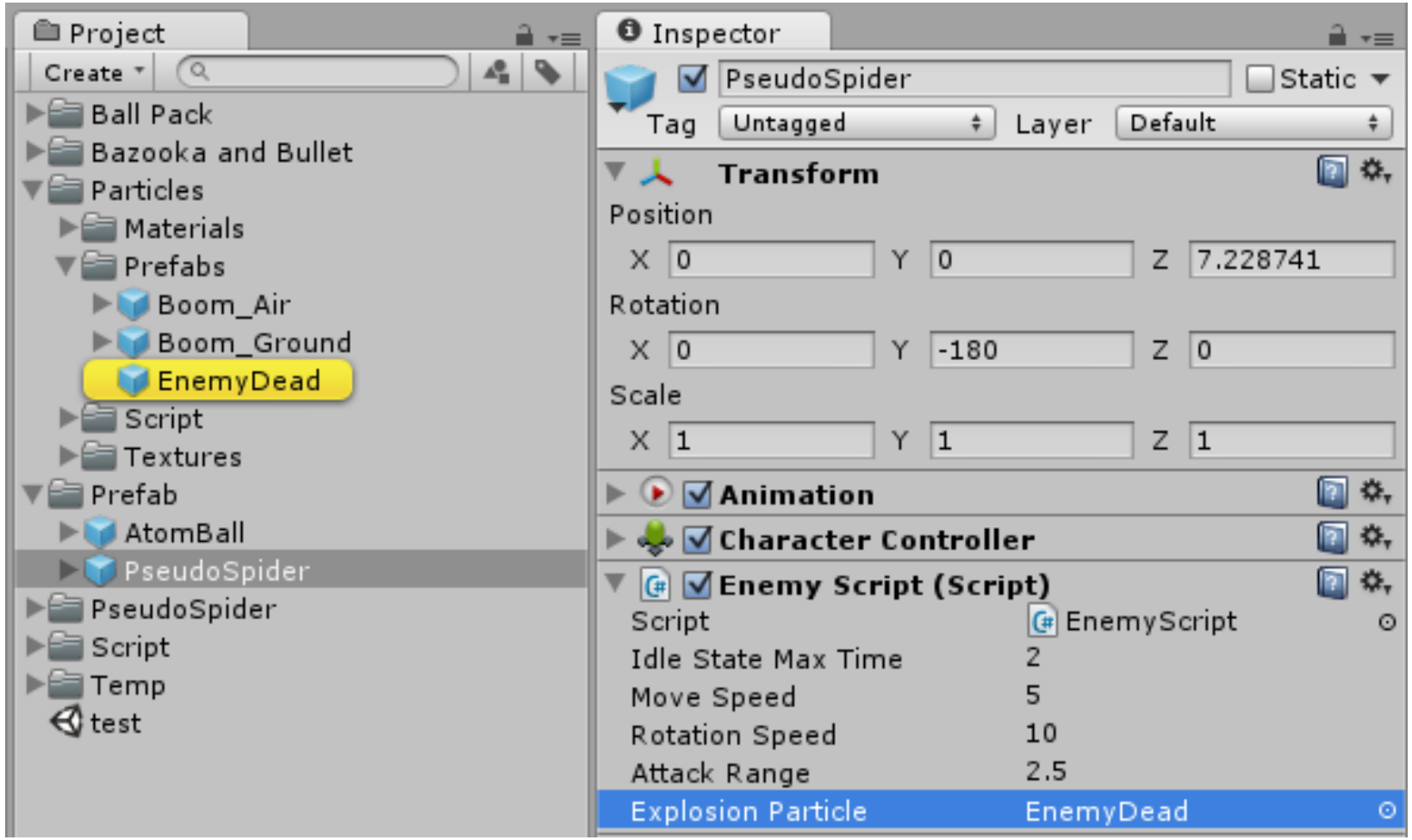
Unity

- 죽음 처리 진행 2

```
public  GameObject   explosionParticle = null;

IEnumerator DeadProcess()
{
    ......
    yield return new WaitForSeconds(1.0f);

    GameObject  explosionObj = Instantiate( explosionParticle ) as GameObject;
    Vector3 explosionObjPos = transform.position;
    explosionObjPos.y = 1.0f;
    explosionObj.transform.position = explosionObjPos;

    Destroy( gameObject );
}
```

Unity

- 죽음 처리 진행 – 파티클 연결



Unity

- DeadObject package import

Unity

- 죽음 처리 진행 3

```
public GameObject deadObject = null;

IEnumerator DeadProcess()
{
  ……
  yield return new WaitForSeconds(0.5f);

  GameObject deadObj = Instantiate(deadObject) as GameObject;
  Vector3 deadObjPos = transform.position;
  deadObjPos.y = 2.0f;
  deadObj.transform.position = deadObjPos;

  deadObj.transform.rotation = Random.rotation;
  Destroy( gameObject );
}
```

Unity

- 죽음 처리 진행 3

```
IEnumerator DeadProcess()
{
  ……
  GameObject  deadObj = Instantiate(deadObject) as GameObject;
  Vector3  deadObjPos  =  transform.position;
  deadObjPos.y  =  2.0f;
  deadObj.transform.position  =  deadObjPos;

  deadObj.transform.rotation = Random.rotation;

  Rigidbody  rb          = deadObj.GetComponent<Rigidbody>();
  rb.velocity            = new Vector3(0.0f, Random.Range(2, 5), 0.0f);
  rb.angularVelocity     = Vector3.one * Random.Range(1.0f, 10.0f);

  Destroy( gameObject );
}
```

Unity

- [Quiz] 남겨진 뼈가 땅으로 사라지는 효과를 직접 구현해 보기

  [조건]
  1. 일정 시간 뒤 지면 아래로 사라지고 파괴할 것.
     예) 2초뒤 아래로 사라지고 파괴할 것.

  2. Coroutine을 활용할 것.

```
public class DeadZombie : MonoBehaviour
{
    public   Rigidbody   rb;
    public   MeshCollider   meshCollider;
    public   float  downSpeed = 0.5f;
```

```
IEnumerator    Start ()
{
    while (rb.velocity != Vector3.zero)
    {
        yield return new WaitForEndOfFrame();
    }
    rb.isKinematic = true;
    meshCollider.isTrigger = true;

    while (transform.position.y > -2.0f)
    {
        Vector3   temp = transform.position;
        temp.y -= downSpeed * Time.deltaTime;
        transform.position = temp;

        yield return new WaitForEndOfFrame();
    }

    Destroy( gameObject );
}
```