게임 프로그래밍

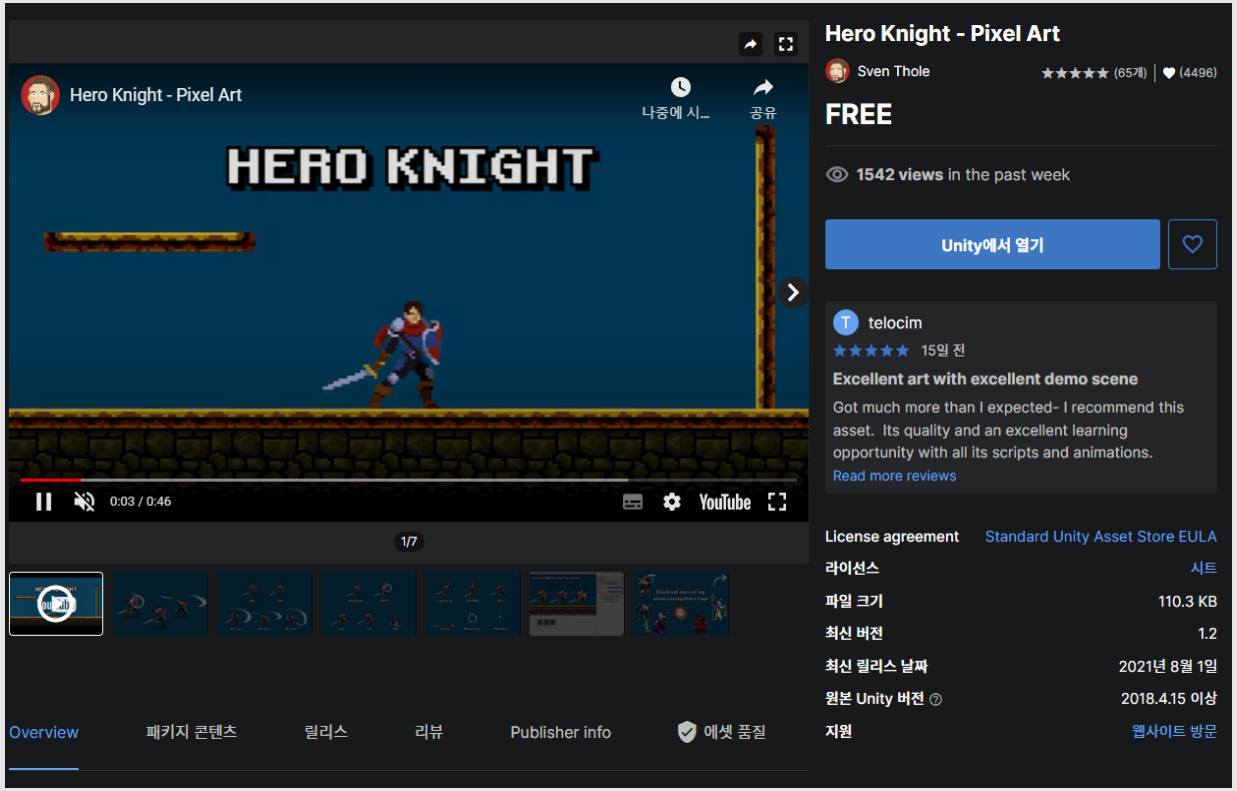
Unity 예제 업그레이드

2023564026
박진성

목차

원본 코드 분석

https://hoil2.tistory.com/5

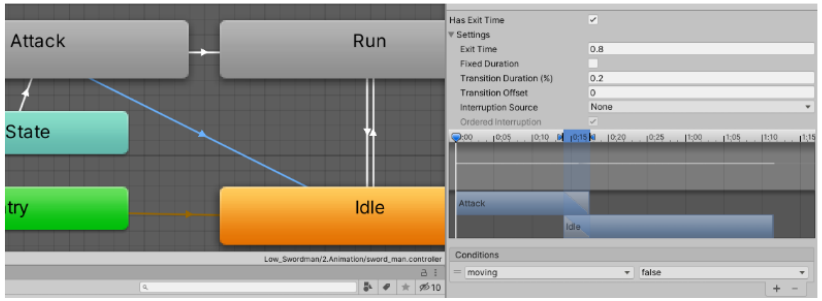


https://assetstore.unity.com/packages/2d/characters/hero-knight-pixel-art-165188

두 곳의 예제를 참고했음
하나는 블로그 예제

나머지 하나는 유니티 무료 Asset

원본 코드 분석



Attack -> Idle

Conditions에 moving false 추가

마찬가지로 Exit Time에 0.8을 입력했습니다.

스크립트에 공격키를 추가합시다.

```
void Update()
{

    if (Input.GetKey(KeyCode.RightArrow))
    {
        transform.localScale = new Vector3(-1, 1, 1);
        animator.SetBool("moving", true);
        transform.Translate(Vector3.right * Time.deltaTime);
    }
    else if (Input.GetKey(KeyCode.LeftArrow))
    {
        transform.localScale = new Vector3(1, 1, 1);
        animator.SetBool("moving", true);
        transform.Translate(Vector3.left * Time.deltaTime);
    }
    else animator.SetBool("moving", false);

    if (Input.GetKeyDown(KeyCode.A) &&
        !animator.GetCurrentAnimatorStateInfo(0).IsName("Attack"))
    {
        animator.SetTrigger("attack");
    }
}
```

- 블로그 예제를 기초로 삼았음

- 이 예제는 player유닛이 적들을 공격해 쓰러뜨리며 나아가는 진행방식을 가짐

원본 코드 분석

카메라가 이동할 수 있는 영역을 지정하기 위해선 먼저 카메라의 x축 길이, y축 길이를 알아야 합니다.

```
cameraHalfWidth = Camera.main.aspect * Camera.main.orthographicSize;
cameraHalfHeight = Camera.main.orthographicSize;
```

코드로 간단하게 구할 수 있습니다.

Camera.main.aspect는 해상도 width/height를 계산한 비율을 나타내는 속성입니다.

Camera.main.orthographicSize는 카메라의 Size입니다.

실제로 16/9*5를 하면 카메라 x축의 절반 값이 나옵니다.

```
public float limitMinX, limitMaxX, limitMinY, limitMaxY;
```

맵의 최소, 최대 값을 받아서 영역을 지정합니다. 귀찮지만 하나하나 적어야 합니다.

```
Vector3 desiredPosition = new Vector3(
    Mathf.Clamp(target.position.x + offset.x, limitMinX + cameraHalfWidth, limitMaxX - cameraHa
    Mathf.Clamp(target.position.y + offset.y, limitMinY + cameraHalfHeight, limitMaxY - cameraH
    -10); // Z
```

Mathf.Clamp(값, 최솟값, 최댓값)으로 최솟값, 최댓값을 넘지 않게 지정할 수 있습니다.

Mathf.Clamp의 간단한 예제

float num = Mathf.Clamp(150, 100, 200);

->num의 값은 150

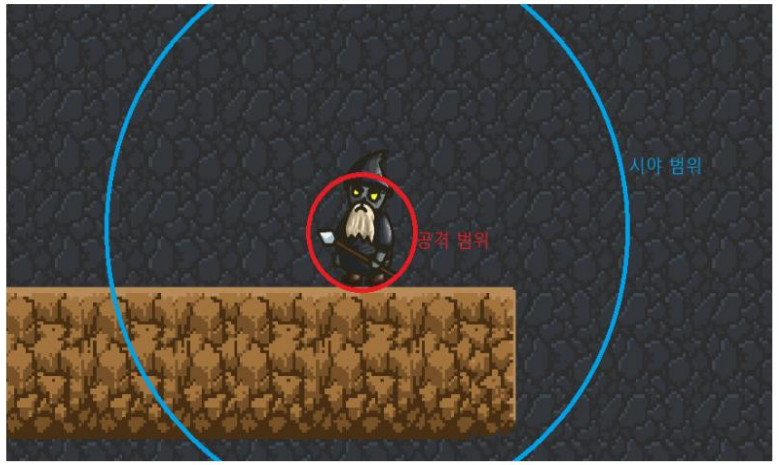
float num2 = Mathf.Clamp(50, 100, 200);

->num2의 값은 100

float num3 = Mathf.Clamp(250, 100, 200);

->num3의 값은 200

카메라의 범위지정과 player 와의 거리에 따른 가속



실행 흐름 의사 코드

while True

  if 시야 범위 안에 들어왔을 때 then

    타깃 방향 쳐다보기

    if 공격 범위 안에 들어왔을 때 then

      공격

    else

      추적

    end if

  endif

endwhile

Player와 적 유닛의 거리를 계산하여 Distance로 상시로 저장

눈여겨 볼 만한 코드는

- 카메라 이동방식 코드

- Player 감지 방식

두가지가 있었음

원본 코드 분석

```
//Check if character just landed on the ground
if (!m_grounded && m_groundSensor.State())
{
    m_grounded = true;
    m_animator.SetBool("Grounded", m_grounded);
}

//Check if character just started falling
if (m_grounded && !m_groundSensor.State())
{
    m_grounded = false;
    m_animator.SetBool("Grounded", m_grounded);
}

// -- Handle input and movement --
float inputX = Input.GetAxis("Horizontal");

// Swap direction of sprite depending on walk direction
if (inputX > 0)
{
    GetComponent<SpriteRenderer>().flipX = false;
    m_facingDirection = 1;
}

else if (inputX < 0)
{
    GetComponent<SpriteRenderer>().flipX = true;
    m_facingDirection = -1;
}

// Move
if (!m_rolling && m_timeSinceAttack > 0.3f && !blocking && stun <= 0)
    m_body2d.linearVelocity = new Vector2(inputX * m_speed, m_body2d.linearVelocity.y);

//Set AirSpeed in animator
m_animator.SetFloat("AirSpeedY", m_body2d.linearVelocity.y);

// -- Handle Animations --
//Wall Slide
m_isWallSliding = (m_wallSensorR1.State() && m_wallSensorR2.State()) || (m_wallSensorL1.S
m_animator.SetBool("WallSlide", m_isWallSliding);
```

유니티 예제는 간결한 코드로 적절한 모션 제어를 한 것이 인상 깊었음

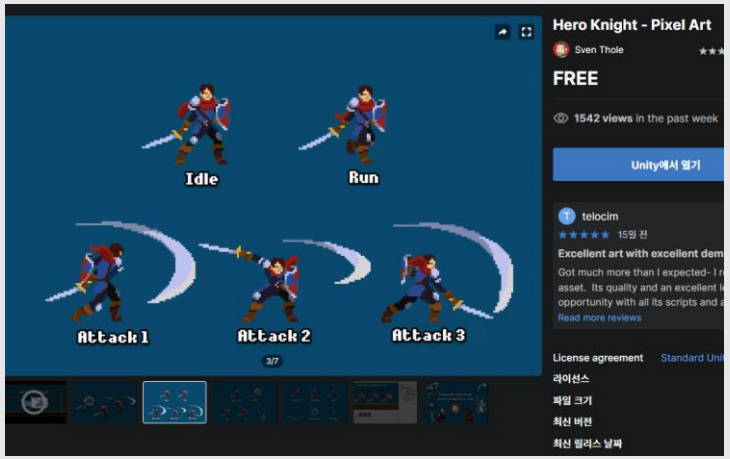
추가 요소

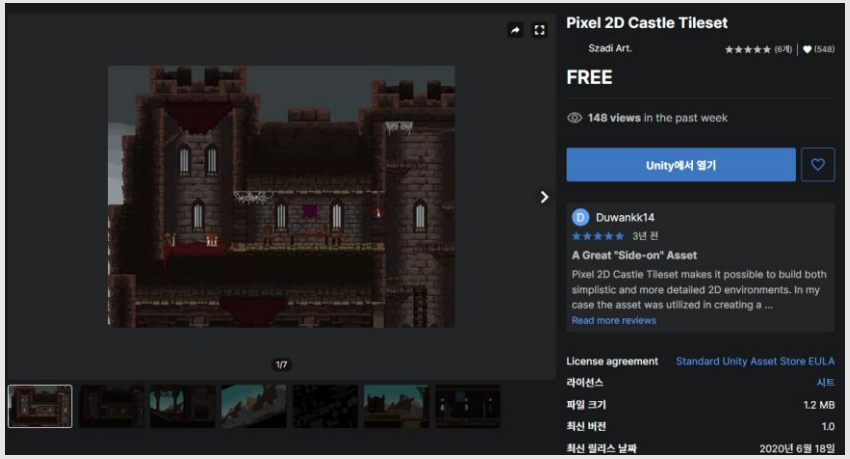
더 많은 엑티브 요소

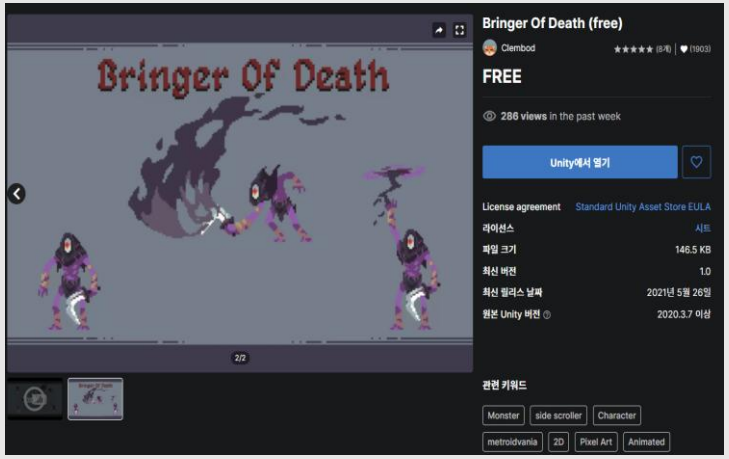
———————————— + 이 둘을 목표로 게임을 개선했음

예외 처리

추가1 – 에셋 재선별

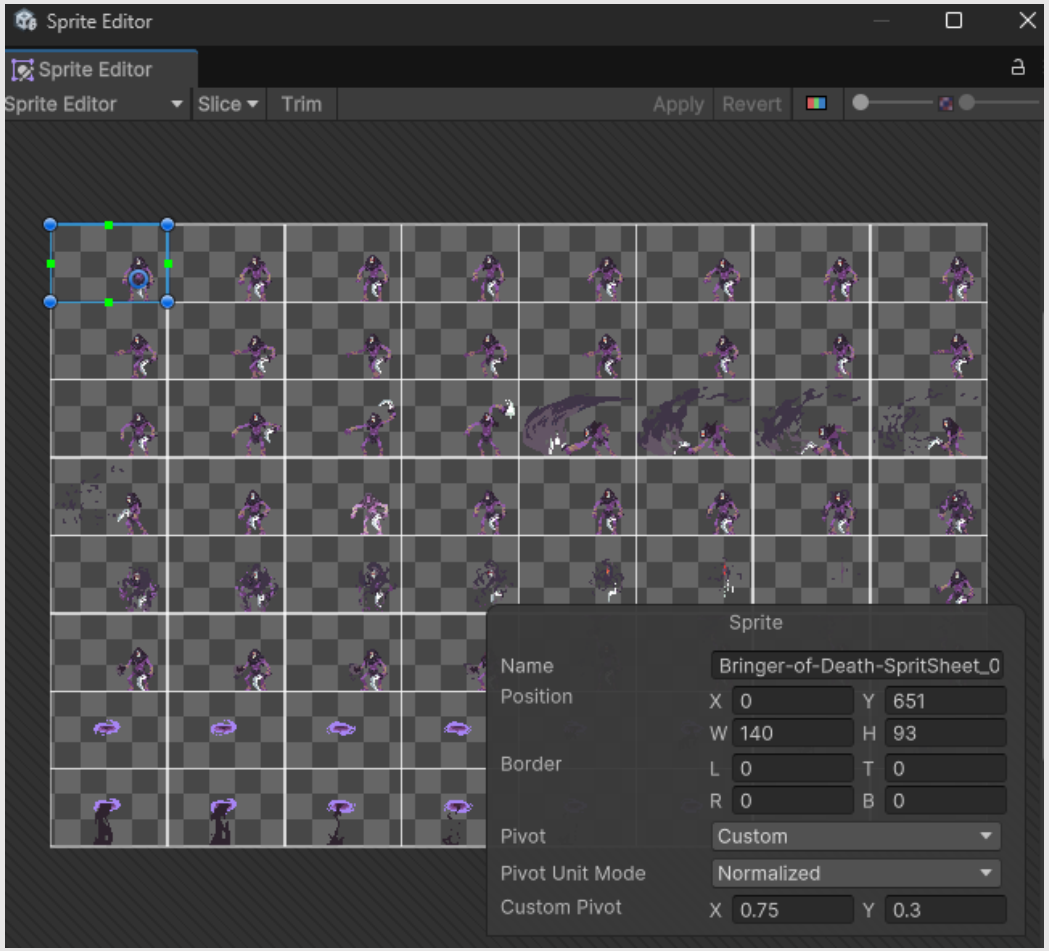






총 3개의 무료에셋을 선별, 최적화하여 사용했음

추가1 – 에셋 재선별



Pivot의 위치가 애매한 에셋들도 있어서 따로 설정해줌

추가2 – 공격 기능 구현

모션 껍데기만 존재했던 공격에 공격판정과
적절한 딜레이를 줌

추가2 – 공격기능 구현

```
else if(Input.GetKeyDown("j") && m_timeSinceAttack > 0.3f && !m_rolling && !blocking && stun <= 0)
{
    m_currentAttack++;
    m_timeSinceAttack = 0.0f;
    Invoke("AttackTrue", 0.015f);

    // Loop back to one after third attack
    if (m_currentAttack > 3) {
        m_currentAttack = 1;
        m_timeSinceAttack = -0.6f;
    }

    // Reset Attack combo if time since last attack is too large
    if (m_timeSinceAttack > 1.0f)
        m_currentAttack = 1;

    // Call one of three attack animations "Attack1", "Attack2", "Attack3"
    m_animator.SetTrigger("Attack" + m_currentAttack);
}
```

- 히트박스식 공격방식을 채용했음

- 적이 히트박스에 들어오면 적의 데미지 스크립트가 실행

- 모션과 함께 데미지를 입음

```
public void AttackTrue()
{
    Collider2D[] AtackRan = Physics2D.OverlapBoxAll(pos.position, boxSize, 0);

    foreach(Collider2D col in AtackRan){
        if(col.CompareTag("Enemy")){
            col.GetComponent<Mob1>().TakeDamage(atkDmg + m_currentAttack * 3);
        }
    }
}
```

```
public void TakeDamage(int dmg){
    HP -= dmg;
    freeze = 0.2f;
    if(!death){
        CancelInvoke();
        animator.SetTrigger("hurt");
    }
}
```

추가3 – 행동 분리

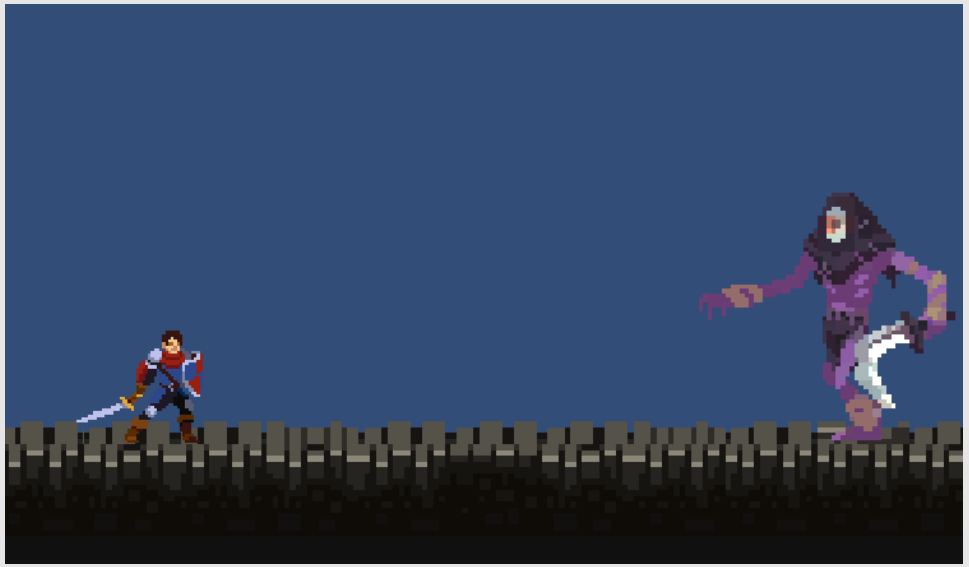
```
//Attack
else if(Input.GetKeyDown("j") && m_timeSinceAttack > 0.3f && !m_rolling && !blocking && stun <= 0)
{
    m_currentAttack++;
    m_timeSinceAttack = 0.0f;
    Invoke("AttackTrue", 0.015f);

    // Loop back to one after third attack
    if (m_currentAttack > 3) {
        m_currentAttack = 1;
        m_timeSinceAttack = -0.6f;
    }

    // Reset Attack combo if time since last attack is too large
    if (m_timeSinceAttack > 1.0f)
        m_currentAttack = 1;

    // Call one of three attack animations "Attack1", "Attack2", "Attack3"
    m_animator.SetTrigger("Attack" + m_currentAttack);
}

// Block
else if (Input.GetKeyDown("k") && !m_rolling && stun <= 0)
{
    blocking = true;
    if(guardc <= 0){
        guardc = 0.3f;
        justTime = 0.15f;
    }
    m_animator.SetTrigger("Block");
    m_animator.SetBool("IdleBlock", true);
}
```

이동을 하면서 공격을 처리하는 등
부자연스러운 움직임이 많아

모션 대부분의 예외처리를 진행함

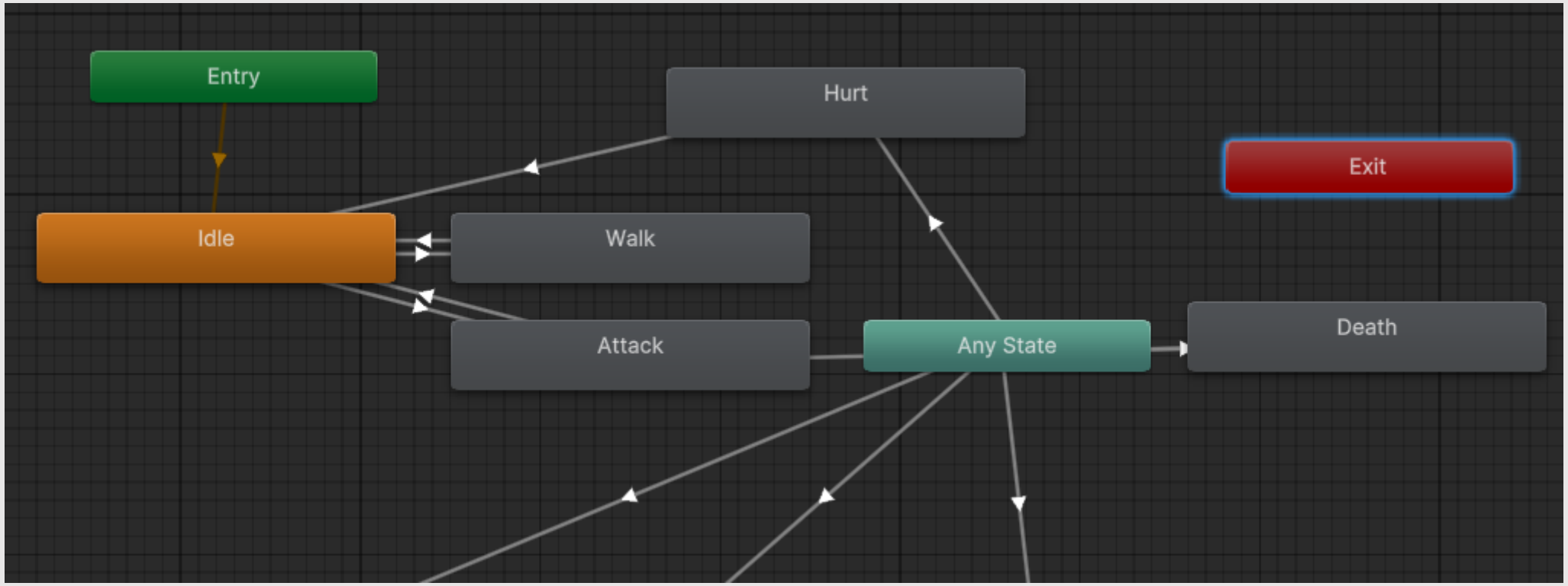
추가4 – 몬스터 AI 개선

```
float distance = Vector3.Distance(transform.position, target.position);

if(!playerDetect){
    if(distance <= 10){
        playerDetect = true;
    }
}

if(attackDelay > 0){
    attackDelay -= Time.deltaTime;
}

if(playerDetect && freeze <= 0){
    if(distance <= atkRange){
        animator.SetBool("walking",false);
        if(attackDelay <= 0){
            attackDelay = 4f;
            Invoke("AttackTrue", 0.4f);
            animator.SetTrigger("attack");
        }
    }
    else{
        transform.Translate(new Vector2(dir, 0) * 1 * Time.deltaTime);
        animator.SetBool("walking", true);
    }
}
```



Player 인식 유무에 따라 행동 방식이 트리거식으로 변경됨

추가4 – 몬스터 AI 개선



몬스터로 사용한 에셋은 모션만 있는 에셋이였기에
모션 노드와 코드를 추가로 작성해줌

추가5 – 피격 스턴

```
public void TakeDamage(int dmg){
    HP -= dmg;
    freeze = 0.2f;
    if(!death){
        CancelInvoke();
        animator.SetTrigger("hurt");
    }
}
```

- 적, 또는 Player 본인이 공격에 피격 당했을 시 기존 행동이 취소되고 약간의 스턴이 적용됨

- 스턴 시간동안 행동을 하지 못하게 예외처리함

추가6 – 가드, 저스트 가드

피해를 줄여주는 가드와
완벽한 타이밍에 가드하면 피해를 받지 않도록 해주는 저스트 가드를 만듦

왼쪽이 기본, 오른쪽이 저스트

추가6 – 가드, 저스트 가드

```
// Block
else if (Input.GetKeyDown("k") && !m_rolling && stun <= 0)
{
    blocking = true;
    if(guardc <= 0){
        guardc = 0.3f;
        justTime = 0.15f;
    }
    m_animator.SetTrigger("Block");
    m_animator.SetBool("IdleBlock", true);
}

else if (Input.GetKeyUp("k"))
{
    blocking = false;
    m_animator.SetBool("IdleBlock", false);
}
```

```
public void TakeDmg(int dmg){
    if(justTime >= 0 ){
        AttackTrue();
    }
    else if(blocking){
        PlayerHP -= dmg / 5;
        stun = 0.1f;
    }
    else{
        PlayerHP -= dmg;
        stun = 0.4f;
        if(!death)
            m_animator.SetTrigger("Hurt");
    }
}
```

기존에 제작, 최적화 해놓았던 몬스터 코드에 시너지를 내어 간결하게 제작함

가드 비율은 80%

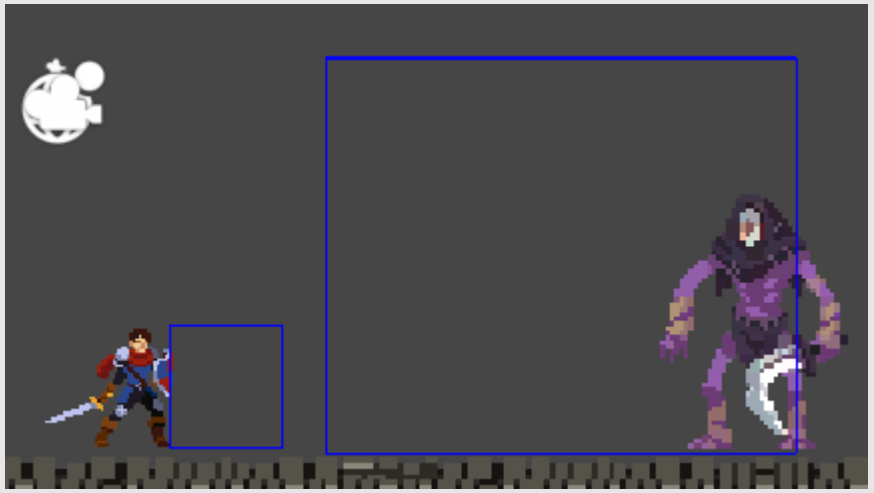
세부 추가1 – 변수 정리

```
1 reference
[SerializeField] float      m_speed = 4.0f;
1 reference
[SerializeField] float      m_jumpForce = 8f;
1 reference
[SerializeField] float      m_rollForce = 10.0f;
1 reference
[SerializeField] bool       m_noBlood = false;
2 references
[SerializeField] GameObject m_slideDust;

17 references
private Animator            m_animator;
8 references
private Rigidbody2D         m_body2d;
4 references
private Sensor_HeroKnight   m_groundSensor;
2 references
private Sensor_HeroKnight   m_wallSensorR1;
3 references
private Sensor_HeroKnight   m_wallSensorR2;
2 references
private Sensor_HeroKnight   m_wallSensorL1;
3 references
private Sensor_HeroKnight   m_wallSensorL2;
3 references
private bool                m_isWallSliding = false;
9 references
private bool                m_grounded = false;
9 references
private bool                m_rolling = false;
1 reference
private bool                death = false;
8 references
private bool                blocking = false;
5 references
private int                 m_facingDirection = 1;
6 references
private int                 m_currentAttack = 0;
7 references
private float               m_timeSinceAttack = 0.0f;
3 references
private float               m_delayToIdle = 0.0f;
1 reference
private float               m_rollDuration = 8.0f / 14.0f;
2 references
private float               m_rollCurrentTime;
9 references
private float               stun;
4 references
private float               justTime = 0f;
4 references
private float               guardc = 0f;
```

```
private Animator animator;
0 references
private Rigidbody2D rig2d;
5 references
private float freeze = 0f;
2 references
public Transform target;


7 references
private float attackDelay;
1 reference
public int MaxHP;
3 references
public int HP;
2 references
public int atkDmg;
4 references
public bool death;
4 references
public bool playerDetect;
2 references
public float atkRange;
4 references
public int dir;


2 references
public Transform pos;
2 references
public Vector2 boxSize;
```

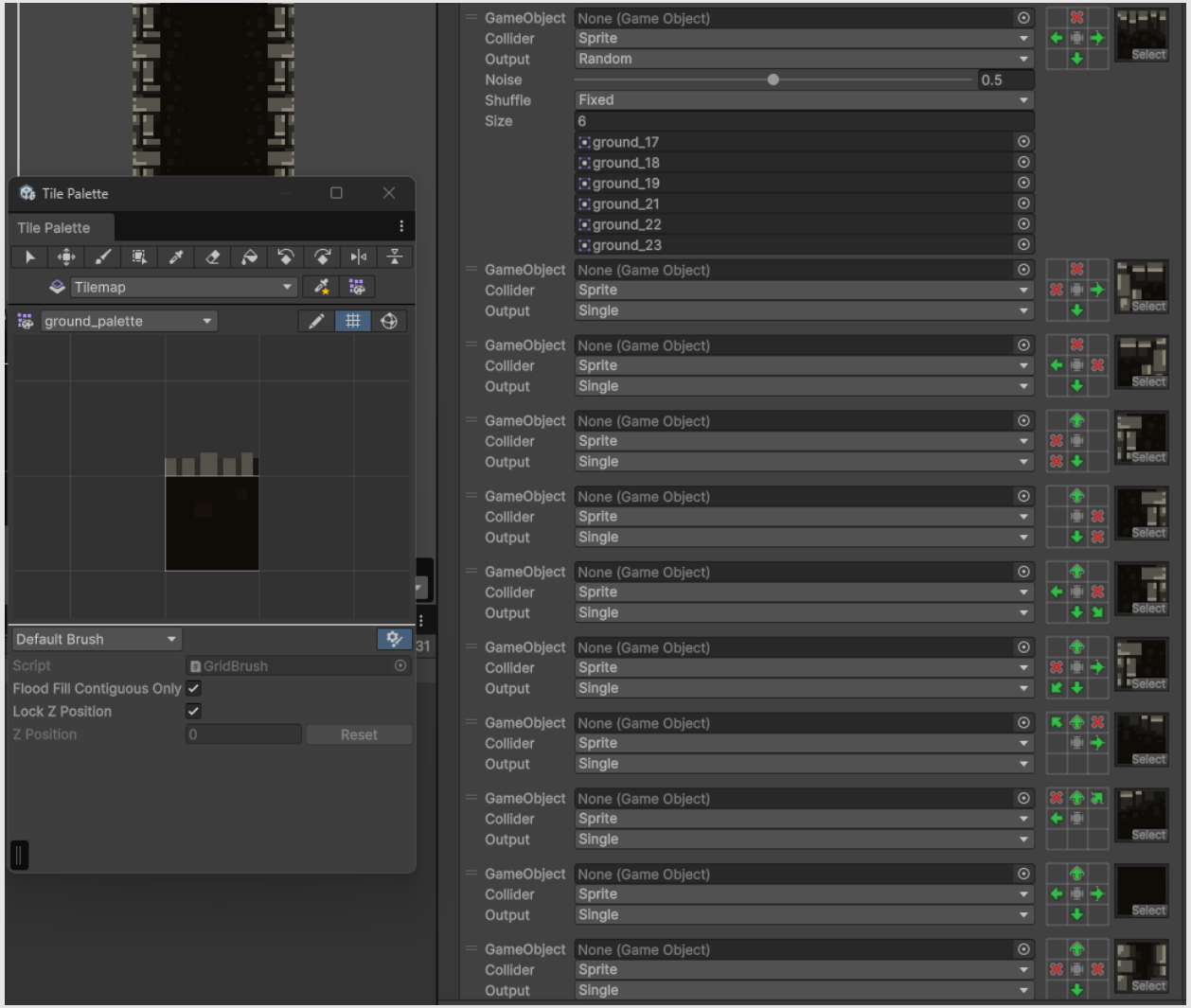
제작 과정에서 필요 이상의 변수를 제작하지 않기 위해 변수 정리 과정을 거침

세부 추가2 – 히트박스



상황에 따라 변화할 수 있는 공격 히트박스를 제작, 활용함

세부 추가3 – 룰타일



참고자료 어느 쪽에서도 룰타일의 언급이 없어

환경에 따라 변화하는 룰타일을 제작, 활용함

Reference

- https://assetstore.unity.com/packages/2d/characters/bringer-of-death-free-195719
- https://assetstore.unity.com/packages/2d/textures-materials/tiles/pixel-2d-castle-tileset-135397
- https://assetstore.unity.com/packages/2d/characters/hero-knight-pixel-art-165188
- https://hoil2.tistory.com/32?category=857730