Sakuya's Adventure
졸업 작품 발표

1688021 이강민

목차

게임 소개

게임 소개

제목: Sakuya's Adventure

장르 – 2D 플랫포머
플랫폼 – Unity
타겟층 – 10~20대 이상

주어진 스테이지에서 클리어에 필요한 요소를 찾고 장애물을 잘 파악하고
피하면서 돌파하는 형식의 게임으로 숨겨져 있는 아이템이나 루트를 찾는 재미를 더 했습니다.

게임 특징

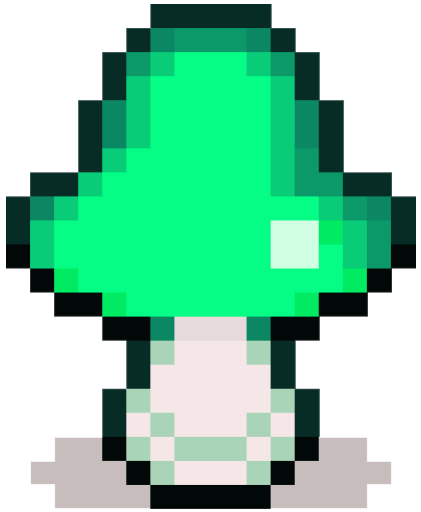
간단한 조작방식

2D 플랫폼인 만큼 간단하지만 점프의 높낮이를 조절하거나 점프거리를 계산해야 하거나 적의 패턴을 파훼해야하는 등 정교한 컨트롤을 통해 게임을 클리어 할 수 있게 구성 되어있다.

숨겨진 요소

사각지대인 부분에 장치하거나, 특정 기술을 통해서 혹은 조건을 만족했을 시 발견할 수 있다.

스토리 소개



무엇이든 고쳐준다는 만병통치약으로 불리는 버섯을 구하기 위해 모험을 떠나는 스토리.
주인공은 버섯에 대한 단서를 얻기 위해 모험을 떠난다.
하지만 그의 여행길은 그렇게 호락호락하지 않았다...

캐릭터 구성도

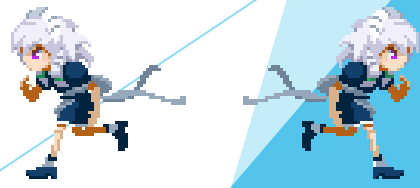
숙이기

점프

공격

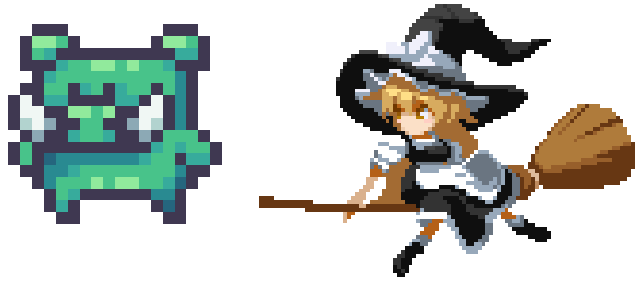
<Command>
이동키: 방향키
점프: 스페이스바
공격: Z
상호작용: E
달리기: Shift

이동



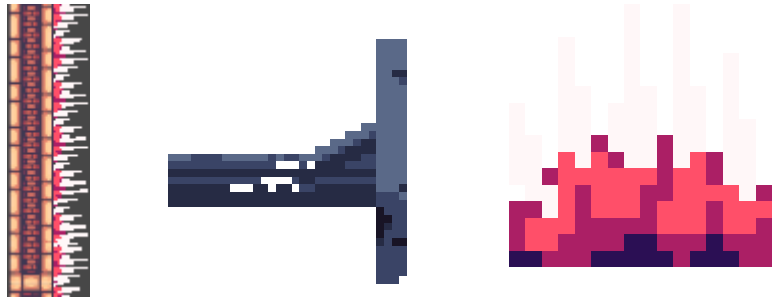
주요 오브젝트

Enemy

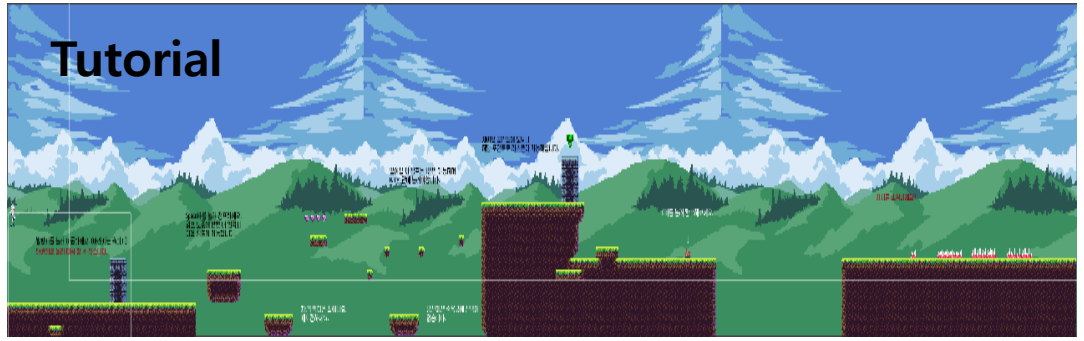


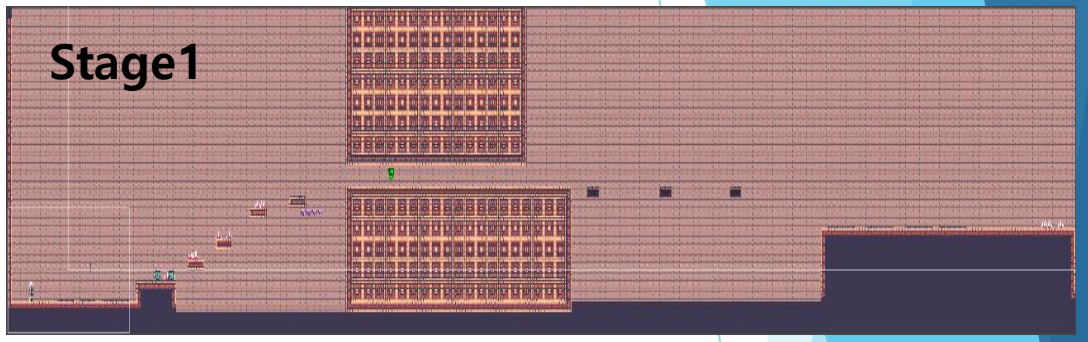
상호작용 객체

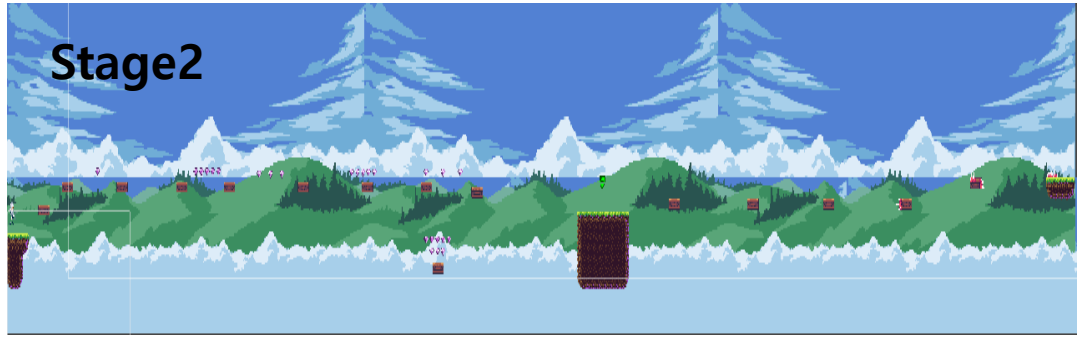
Trap

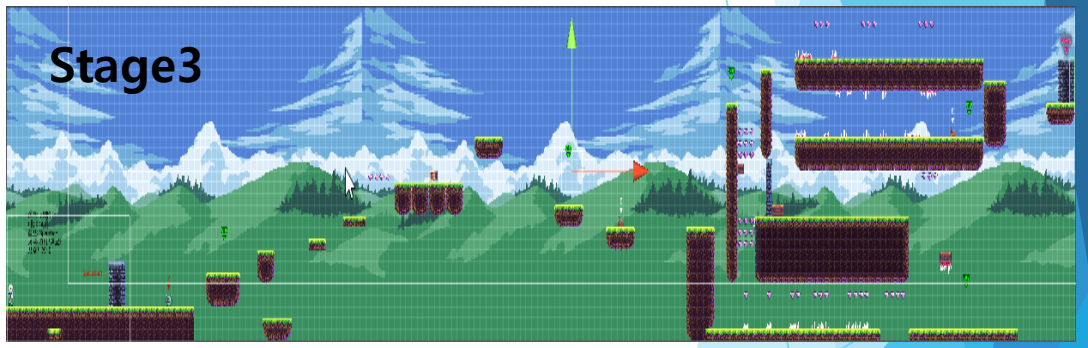


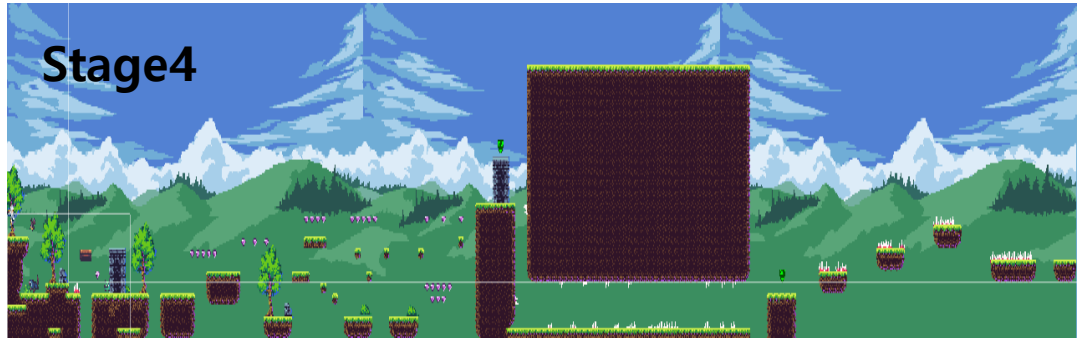
스테이지 구성

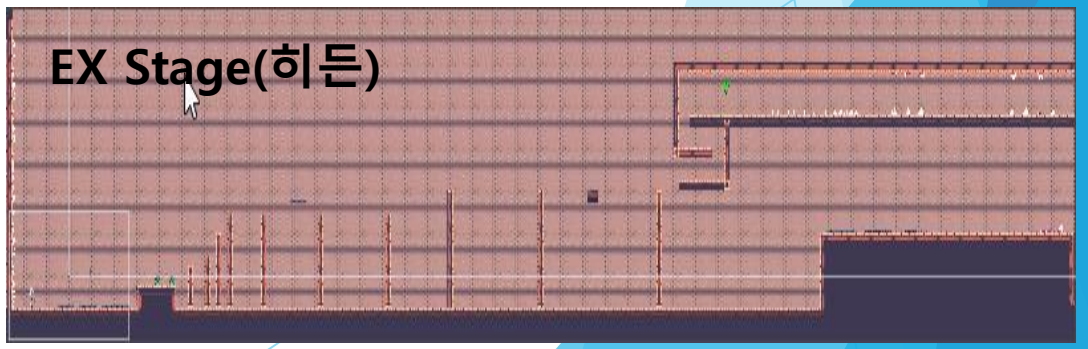












게임 시스템

Enemy

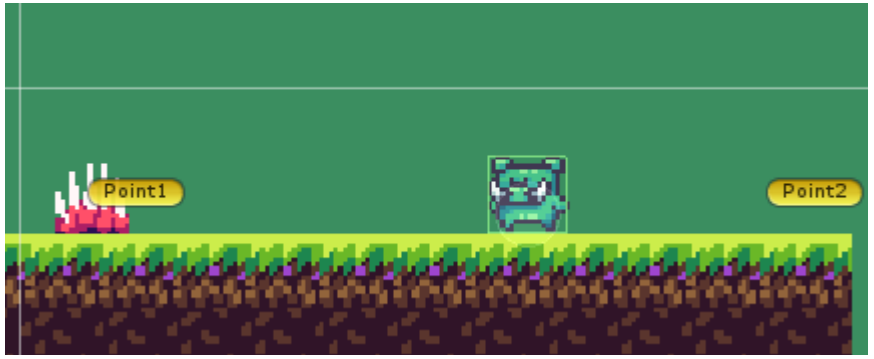
이름: Green Pig

<대기 상태>

<사망>

<이동>

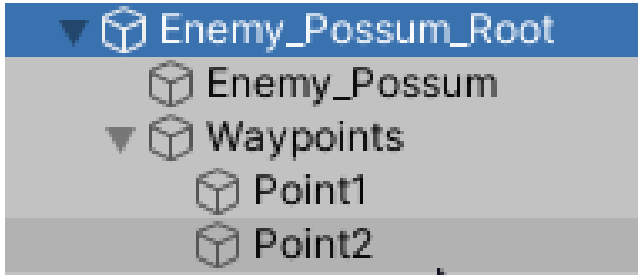
Enemy AI 설정



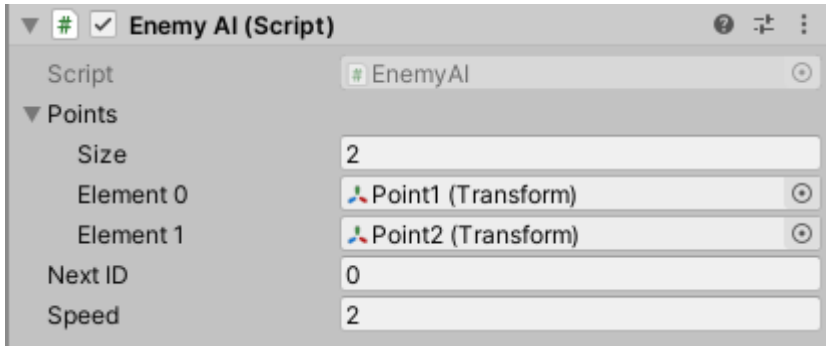
Point에 찍힌 곳을
순회하는 방식의 AI

```
참조 1개
void Init()
{
    // Make box collider trigger
    GetComponent<BoxCollider2D>().isTrigger = true;

    // Create Root object
    GameObject root = new GameObject(name + "_Root");
    // Reset Position of Root to this enemy object
    root.transform.position = transform.position;
    // Set enemy object as child of root
    transform.SetParent(root.transform);
    // Create Waypoints object
    GameObject waypoints = new GameObject("Waypoints");
    // Reset Waypoints position to root
    // Make Waypoints object child of root
    waypoints.transform.SetParent(root.transform);
    waypoints.transform.position = root.transform.position;
    // Create two points (gameobject) and reset their position to waypoints objects
    // Make the points children of waypoint object
    GameObject p1 = new GameObject("Point1"); p1.transform.SetParent(waypoints.transform); p1.transform.position = root.transform.position;
    GameObject p2 = new GameObject("Point2"); p2.transform.SetParent(waypoints.transform); p2.transform.position = root.transform.position;

    // Init points list then add the points to it
    points = new List<Transform>();
    points.Add(p1.transform);
    points.Add(p2.transform);
}
```

초기화 설정 코드



Enemy 전체를 담당하는뿌리
|
Enemy 객체
|
움직이는 루트



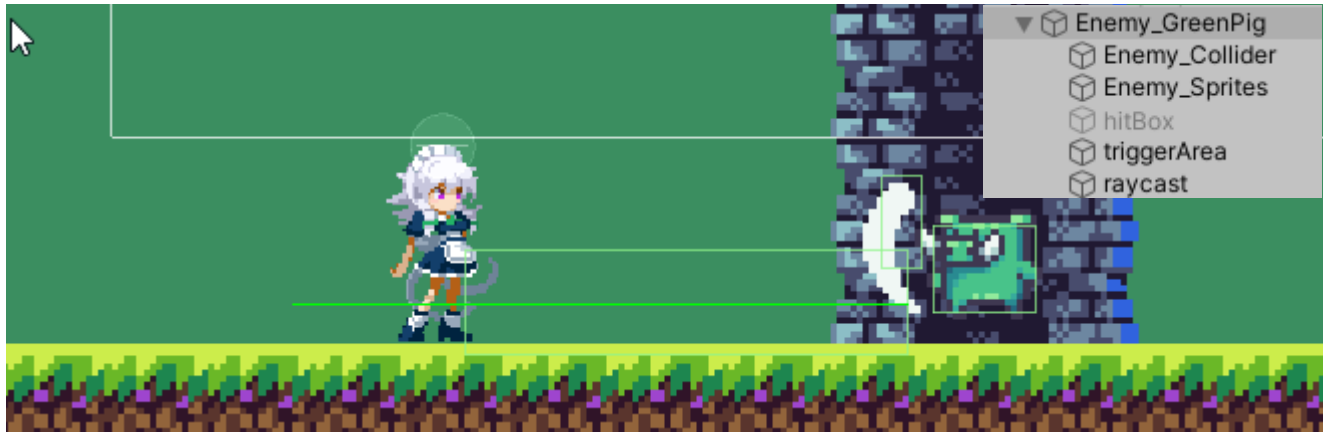
Waypoint의 Size를 설정
size에 따라 포인트지점 배치
이를 ID 값으로 받아 순회하게끔
알고리즘을 구현함.

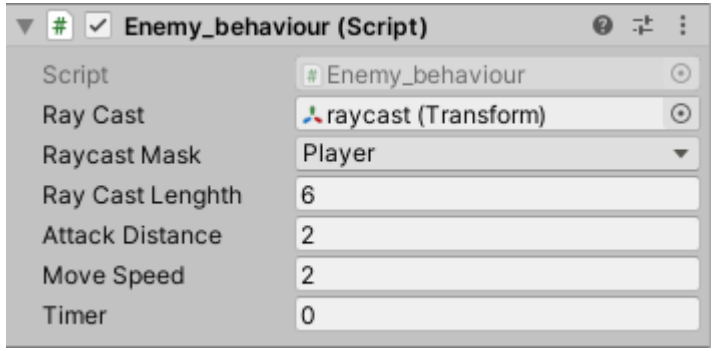
```
private void Update()
{
    MoveToNextPoint();
}

참조 1개
void MoveToNextPoint()
{
    // Get the next Point transform
    Transform goalPoint = points[nextID];
    // Flip the enemy transform to look into the point's direction
    if (goalPoint.transform.position.x > transform.position.x)
        transform.localScale = new Vector3(-1, 1, 1);
    else
        transform.localScale = new Vector3(1, 1, 1);
    // Move the enemy towards the goal point
    transform.position = Vector2.MoveTowards(transform.position, goalPoint.position, speed * Time.deltaTime);
    // Check the distance between enemy and goal point to trigger next point
    if(Vector2.Distance(transform.position, goalPoint.position)<0.2f)
    {
        // Check if we are at the end of the line (make the change -1)
        if (nextID == points.Count - 1)
            idChangeValue = -1;
        // Check if we are at the start of the line (make the change +1)
        if (nextID == 0)
            idChangeValue = 1;
        // Apply the change on the nextID
        nextID += idChangeValue;
    }
}
```

이동
메커니즘
코드

Enemy AI 설정 심화

Triggerarea에 플레이어 감지 – raycast
플레이어 충돌감지 – EnemyLogic() 실행





Raycast의 길이 설정,
공격 거리 설정,
이동속도 설정,
공격딜레이 설정.

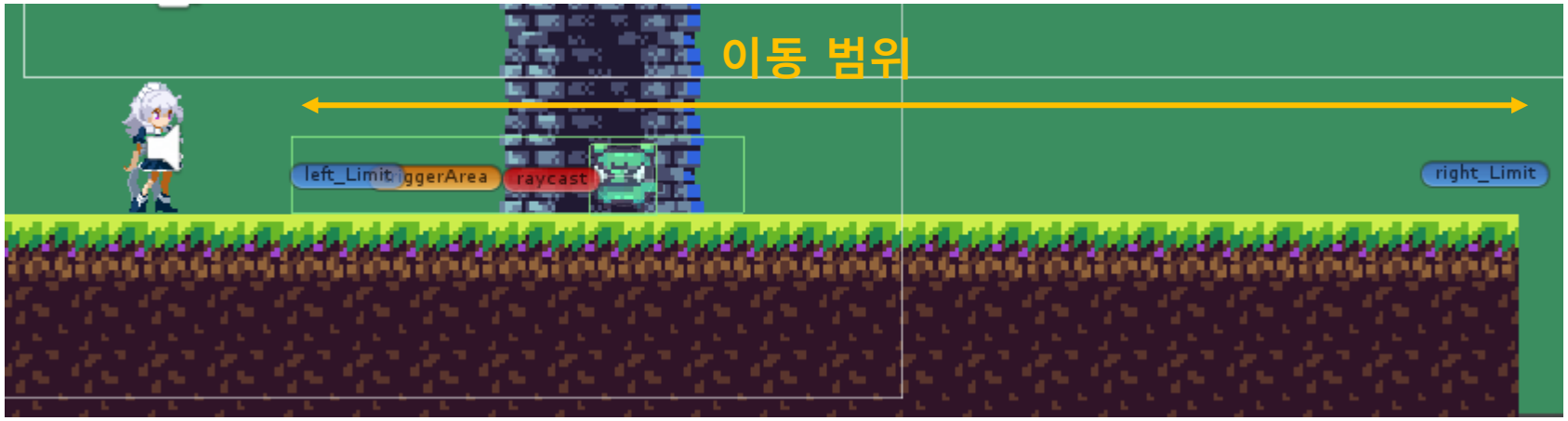
플레이어 충돌체크

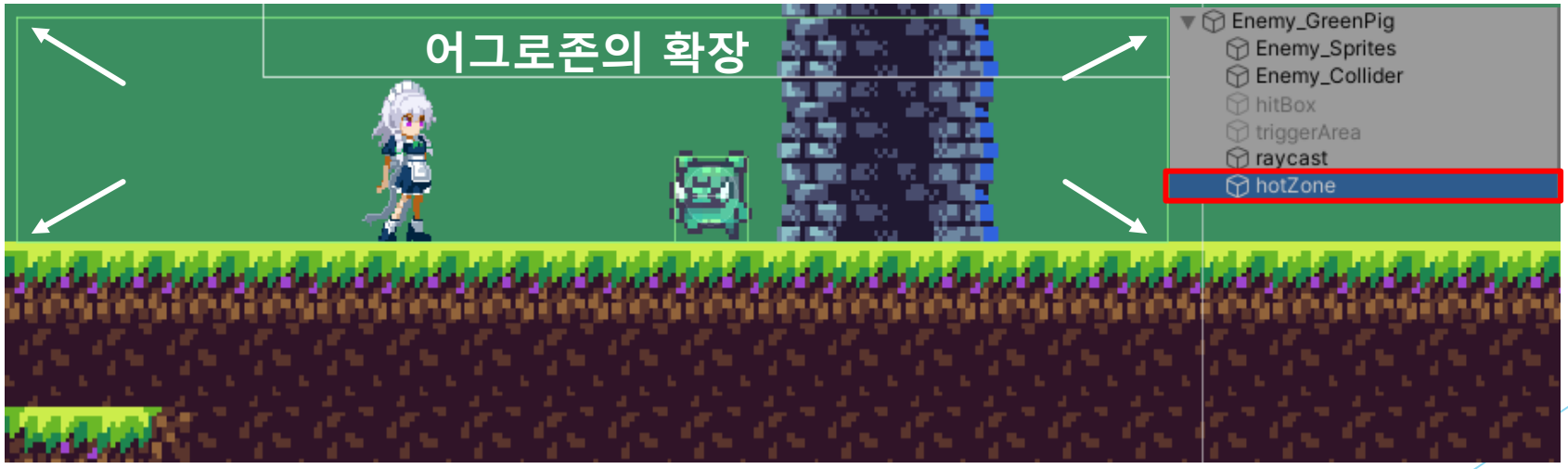
```
void Update()
{
    if (inRange)
    {
        hit = Physics2D.Raycast(rayCast.position, Vector2.left, rayCastLenghth, raycastMask);
        RaycastDebugger();
    }

    // When Player is detected
    if(hit.collider != null)
    {
        EnemyLogic();
    }

    else if(hit.collider == null)
    {
        inRange = false;
    }

    if(inRange == false)
    {
        anim.SetBool("canWalk", false);
        StopAttack();
    }
}
```

공격 메커니즘 코드

```
void EnemyLogic()
{
    distance = Vector2.Distance(transform.position, target.transform.position);

    if(distance > attackDistance)
    {
        Move();
        StopAttack();
    }

    else if(attackDistance >= distance && cooling == false)
    {
        Attack();
    }

    if (cooling)
    {
        anim.SetBool("Attack", false);
    }
}
```

Enemy AI 설정 심화





트리거에리어 체크

```
public class TriggerAreaCheck : MonoBehaviour
{
    private Enemy_behaviour3 enemyParent;

    private void Awake()
    {
        enemyParent = GetComponentInParent<Enemy_behaviour3>();
    }

    private void OnTriggerEnter2D(Collider2D collider)
    {
        if (collider.gameObject.CompareTag("Player"))
        {
            gameObject.SetActive(false);
            enemyParent.target = collider.transform;
            enemyParent.inRange = true;
            enemyParent.hotZone.SetActive(true);
        }

    }
}
```

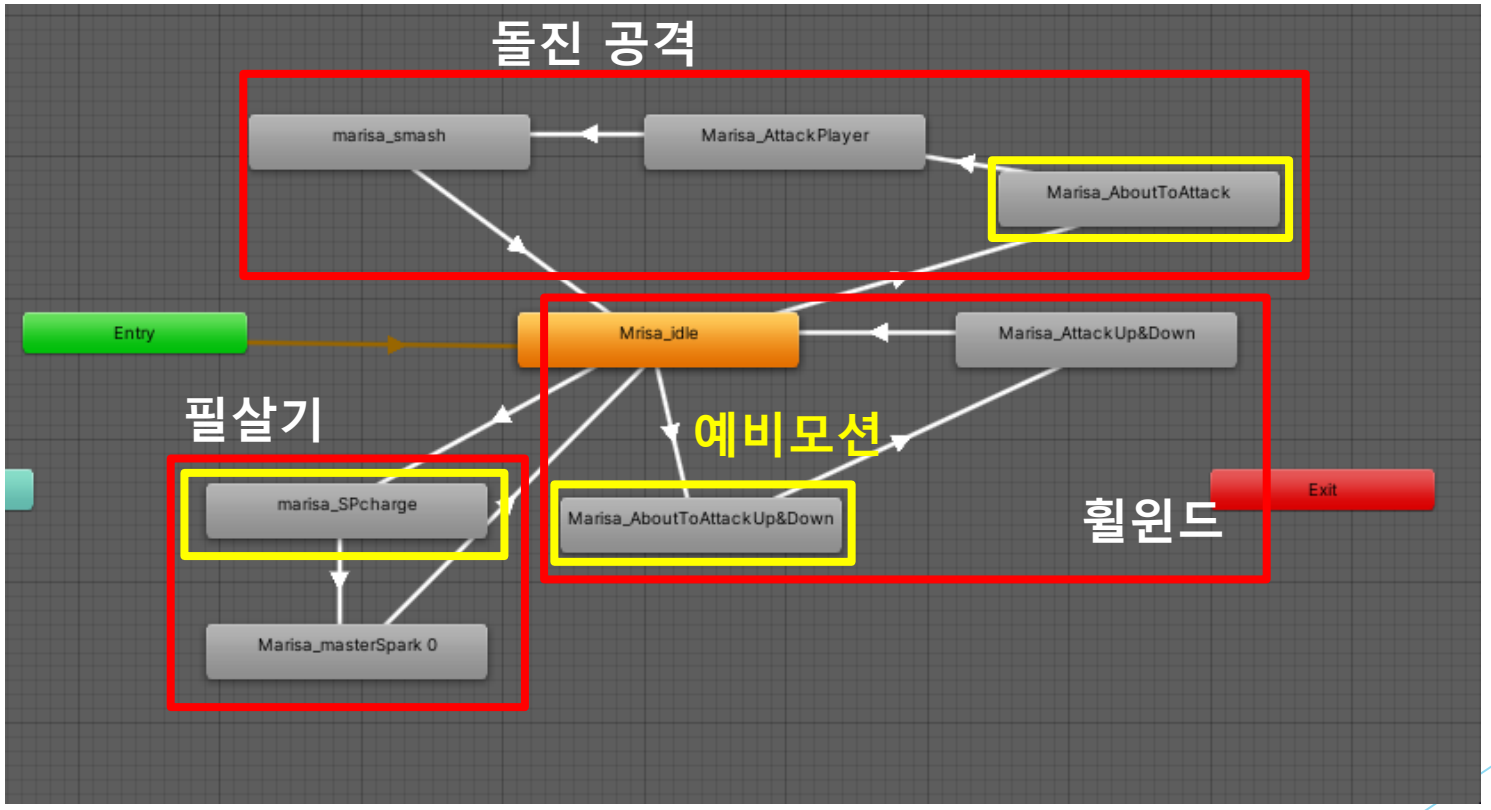
어그로존 확장 코드

```
    private void Awake()
    {
        enemyParent = GetComponentInParent<Enemy_behaviour3>();
        anim = GetComponentInParent<Animator>();
    }

    private void Update()
    {
        if(inRange && !anim.GetCurrentAnimatorStateInfo(0).IsName("Attack"))
        {
            enemyParent.Flip();
        }
    }

    private void OnTriggerEnter2D(Collider2D collider)
    {
        if (collider.gameObject.CompareTag("Player"))
        {
            inRange = true;
        }
    }

    private void OnTriggerExit2D(Collider2D collider)
    {
        if(collider.gameObject.CompareTag("Player"))
        {
            inRange = false;
            gameObject.SetActive(false);
            enemyParent.triggerArea.SetActive(true);
            enemyParent.inRange = false;
            enemyParent.SelectTarget();
        }
    }
```

보스 AI - 메커니즘

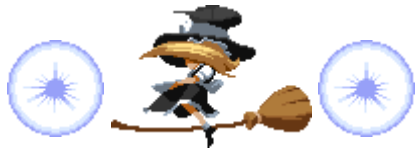
Marisa

1. Idle 상태에서 패턴 랜덤설정 (플레이어 추격)
2. 공격 패턴 진행 시 SP포인트 1씩 증가
3. SP포인트가 5이상 쌓일 시 필살기 발동.



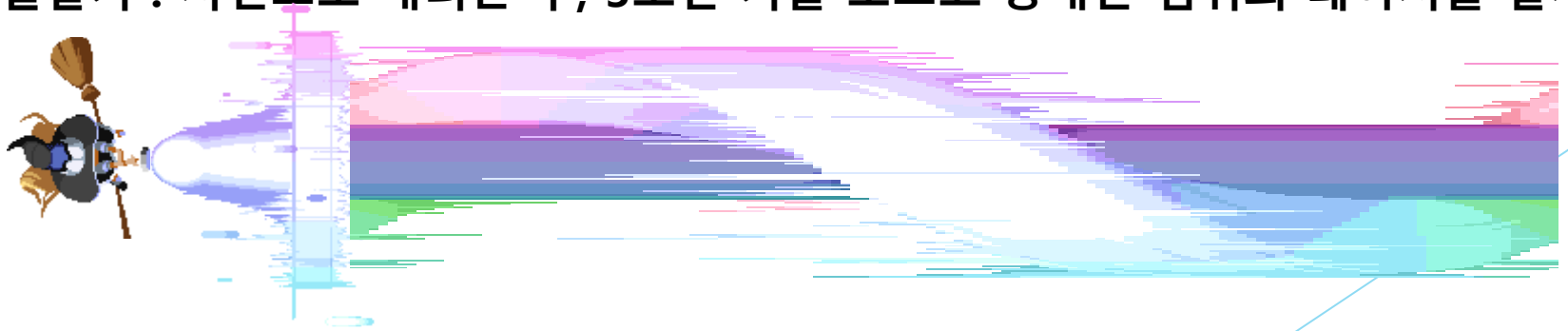
보스 AI - 기술

휠윈드 : 위 아래로 회전 하면서 움직이는 기술, 지형에 닿을 시 양쪽에 탄막을 발사한다.



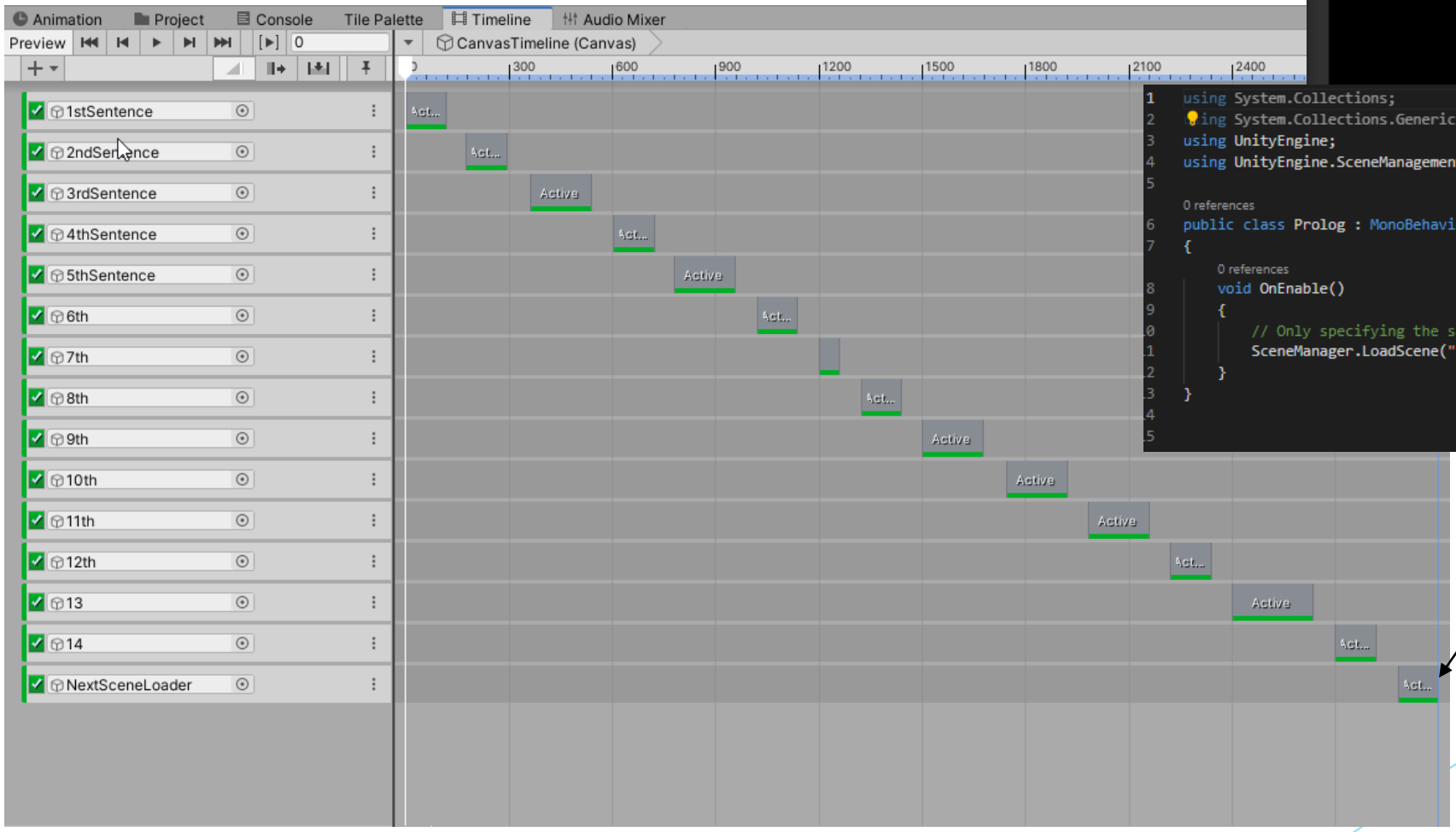
돌진 공격 : 플레이어를 향하여 돌진하는 기술.

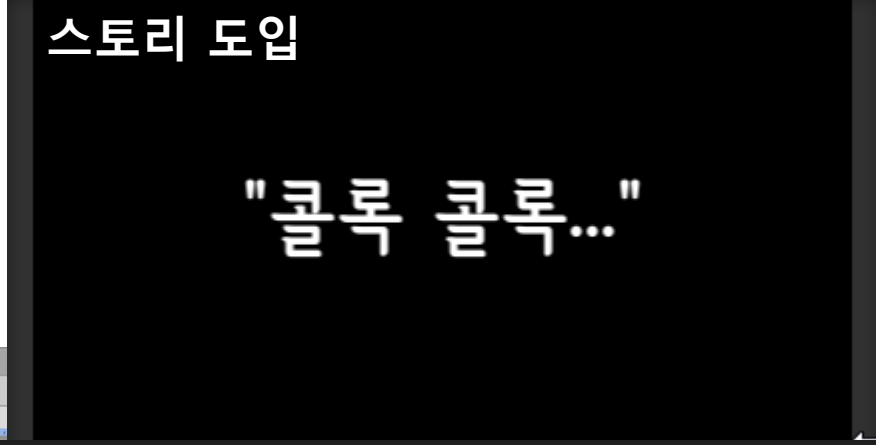
필살기 : 지면으로 내려온 후, 3초간 기를 모으고 광대한 범위의 레이저를 발사하는 기술.



스토리 컷씬 시스템 – Timeline

Timeline을 통해 인트로 스토리 시스템 구현



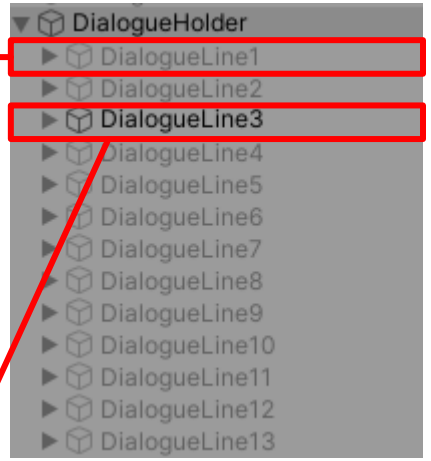


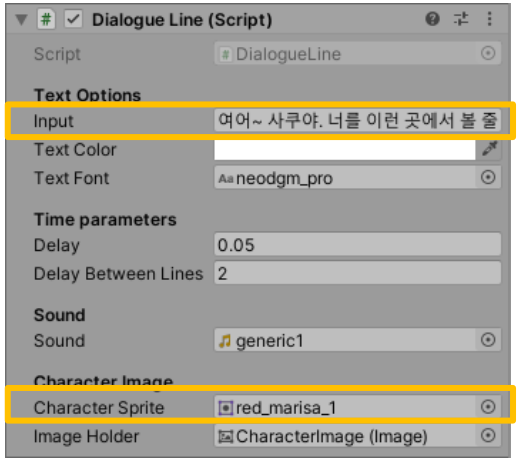
Prolog.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

0 references
public class Prolog : MonoBehaviour
{
    0 references
    void OnEnable()
    {
        // Only specifying the sceneName or sceneBuildIndex will load the Scene with the single mode
        SceneManager.LoadScene("Proto", LoadSceneMode.Single);
    }
}
```

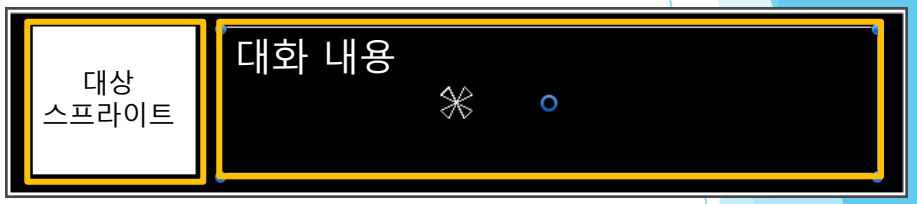
스토리 컷씬 시스템 – Dialogue System

대화창 시스템





커맨드 입력을 받을 시
해당 라인을 활성화/비활성화
하는 방식으로 구현



스토리 컷씬 시스템 – Dialogue System

DialogueSystem 코드

DialogueBaseClass.cs

```
namespace DialogueSystem
{
    public class DialogueBaseClass : MonoBehaviour
    {
        public bool finished { get; private set; }

        protected IEnumerator WriteText(string input, Text textHolder, Color textColor, Font textFont, float delay, AudioClip sound, float delayBetweenLines)
        {
            textHolder.color = textColor;
            textHolder.font = textFont;

            for (int i = 0; i < input.Length; i++)
            {
                textHolder.text += input[i];
                SoundManager.instance.PlaySound(sound);
                yield return new WaitForSeconds(delay);
            }

            //yield return new WaitForSeconds(delayBetweenLines);
            yield return new WaitUntil(() => Input.GetMouseButton(0));
            finished = true;
        }
    }
}
```

DialogueHolder.cs

```
namespace DialogueSystem
{
    public class DialogueHolder : MonoBehaviour
    {
        private void Awake()
        {
            StartCoroutine(dialogueSequence());
        }

        private IEnumerator dialogueSequence()
        {
            for (int i = 0; i < transform.childCount; i++)
            {
                Deactivate();
                transform.GetChild(i).gameObject.SetActive(true);
                yield return new WaitUntil(() => transform.GetChild(i).GetComponent<DialogueLine>().finished);
            }
            gameObject.SetActive(false);
        }

        private void Deactivate()
        {
            for (int i = 0; i < transform.childCount; i++)
            {
                transform.GetChild(i).gameObject.SetActive(false);
            }
        }
    }
}
```

DialogueLine.cs

```
namespace DialogueSystem
{
    public class DialogueLine : DialogueBaseClass
    {
        private Text textHolder;

        [Header ("Text Options")]
        [SerializeField] private string input;
        [SerializeField] private Color textColor;
        [SerializeField] private Font textFont;

        [Header("Time parameters")]
        [SerializeField] private float delay;
        [SerializeField] private float delayBetweenLines;

        [Header("Sound")]
        [SerializeField] private AudioClip sound;

        [Header("Character Image")]
        [SerializeField] private Sprite characterSprite;
        [SerializeField] private Image imageHolder;

        private void Awake()
        {
            textHolder = GetComponent<Text>();
            textHolder.text = "";

            imageHolder.sprite = characterSprite;
            imageHolder.preserveAspect = true;
        }

        private void Start()
        {
            StartCoroutine(WriteText(input, textHolder, textColor, textFont, delay, sound, delayBetweenLines));
        }
    }
}
```

리스폰&체크포인트

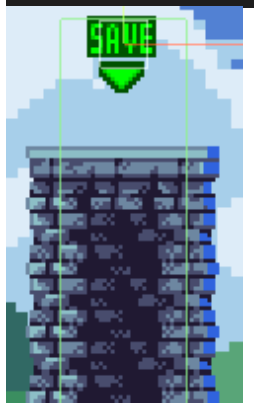
사망 시 해당 포인트 지점으로 리스폰 되는 세이브 포인트 구현.

Level Manager 코드

```
using UnityEngine;
using UnityEngine.SceneManagement;

참조 3개
public class LevelManager : MonoBehaviour
{
    public Vector2 playerInitPosition;

    참조 0개
    void Start()
    {
        playerInitPosition = FindObjectOfType<playerController2>().transform.position;
    }
    참조 2개
    public void Restart()
    {
        // 1- Restart the scene
        // SceneManager.LoadScene(SceneManager.GetActiveScene().name);
        // 2- Reset the player's position
        // Save the player's initial position when game starts
        // When respawning simply reposition the player to that init position
        // Reset the player's movement speed
        FindObjectOfType<playerController2>().ResetPlayer();
        FindObjectOfType<playerController2>().transform.position = playerInitPosition;
        // Reset the life count
    }
}
```

Check Point 코드

```
using System.Collections.Generic;
using UnityEngine;

참조 0개
public class CheckPoint : MonoBehaviour
{
    // Start is called before the first frame update
    참조 0개
    void Start()
    {

    }

    // Update is called once per frame
    참조 0개
    void Update()
    {

    }

    참조 0개
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if(collision.gameObject.CompareTag("Player"))
        {
            FindObjectOfType<LevelManager>().playerInitPosition = transform.position;
        }
    }
}
```



해당영역에 닿을시 저장.

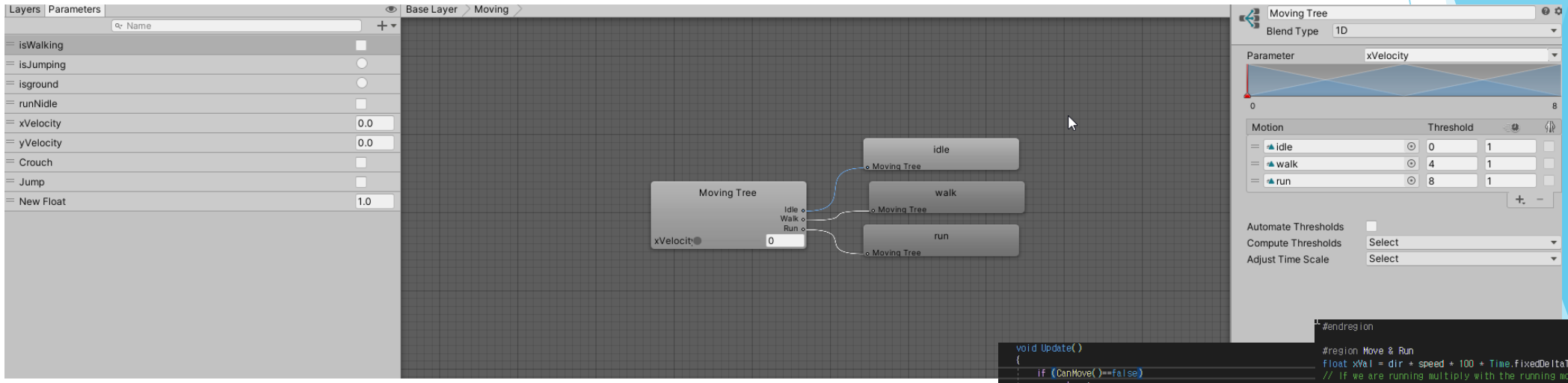
Player Controller 코드

```
7 references
public void Die()
{
    AudioManager.instance.PlaySFX("die");
    isDead = true;
    //StartCoroutine(Respawn());



    FindObjectOfType<LevelManager>().Restart();
}

1 reference
public void ResetPlayer()
{
    horizontalValue = 0;
    isDead = false;
    canAtk = false;
    isAttacking = false;
    combo = 0;
}

0 references
IEnumerator Respawn()
{
    animator.SetTrigger("Death");
    yield return new WaitForSeconds(0.9f);
    FindObjectOfType<LevelManager>().Restart();

}
```

애니메이션 설정

이동, 대쉬

블렌드 트리 - Moving



각각의 변수값에 Threshold값을 적용하여

대기모션, 걷는 모션, 대쉬모션의 전환을 부드럽게 만듬.

구동원리)
x값과 y값의 수치가 일정 범위(0~8)를 넘을 경우
각각 Walk, RUN 애니메이션을 호출

키입력

```
void Update()
{
    if (CanMove()==false)
        return;
    if (isDead)
        return;

    // Store the horizontal value
    horizontalValue = Input.GetAxisRaw("Horizontal");

    // If LShift is clicked enable isRunning
    if (Input.GetKeyDown(KeyCode.LeftShift))
        isRunning = true;
    // If LShift is clicked disable isRunning
    if (Input.GetKeyUp(KeyCode.LeftShift))
        isRunning = false;

    // If we press jump button enable jump
    if (Input.GetButtonDown("Jump"))
        Jump();

    // If we press Crouch button enable crouch
    if (Input.GetButtonDown("Crouch") && isGrounded)
        crouchPressed = true;
    // Otherwise disable it
    else if (Input.GetButtonUp("Crouch"))
        crouchPressed = false;

    // Set the yVelocity in the animator
    animator.SetFloat("yVelocity", rb.velocity.y);
}
```

이동&대쉬

```
#endregion

#region Move & Run
float xVal = dir * speed * 100 * Time.fixedDeltaTime; // Set value of x using dir and speed
// If we are running multiply with the running modifier
if (isRunning)
    xVal *= runSpeedModifier;
// If we are running multiply with the running modifier
if (crouchFlag)
    xVal *= crouchSpeedModifier;
Vector2 targetVelocity = new Vector2(xVal, rb.velocity.y); //Create Vec2 for the velocity
rb.velocity = targetVelocity; //Set the player's velocity

//If looking right and clicked left (flip to the left)
if (facingRight && dir < 0)
{
    transform.localScale = new Vector3(-1, 1, 1);
    facingRight = false;
}

//If looking left and clicked right (flip to the left)
else if (!facingRight && dir > 0)
{
    transform.localScale = new Vector3(1, 1, 1);
    facingRight = true;
}

// 0 idle, xVal walking, 2xVal running ( 0, 1, 2 )
// Set the float XVelocity according to the x value
// of the RigidBody2D velocity
animator.SetFloat("xVelocity", Mathf.Abs(rb.velocity.x));
#endregion
//Debug.Log(rb.velocity.x);
```

숙이기(Crouch)

```
58
59          // If we press Crouch button enable crouch
60          if (Input.GetButtonDown("Crouch"))
61              crouchPressed = true;
62          // Otherwise disable it
63          else if (Input.GetButtonUp("Crouch"))
64              crouchPressed = false;
65
```

```
109         if (isGrounded)
110         {
111             standingCollider.enabled = !crouchFlag;
112
113
114             // If the player is grounded and pressed space Jump
115             if (jumpFlag)
116             {
117                 jumpFlag = false;
118                 //isGrounded = false;
119                 // Add jump force
120                 rb.AddForce(new Vector2(0f, jumpPower));
121             }
122         }
123
124         animator.SetBool("Crouch", crouchFlag);
125
```

```
133         // If we are running multiply with the running modifier
134         if (crouchFlag)
135             xVal *= crouchSpeedModifier;
136         Vector2 targetVelocity = new Vector2(xVal, rb.velocity.y); //Create Vec2 for the velocity
137         rb.velocity = targetVelocity; //Set the player's velocity
138
```

구동원리) 숙이기에 할당된 키를 입력할 시 스탠딩 충돌범위를 off하고 숙이기 충돌범위를 ON시킴.

길쭉한 부분이 standing, 반쪽부분이 crouch범위

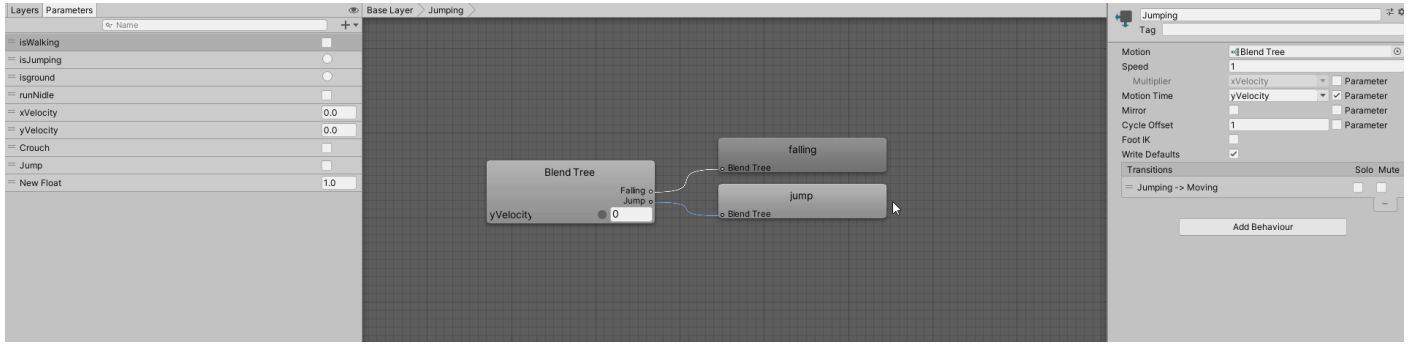
점프, 이단점프

```
void  Jump()
{
    if (isGrounded)
    {
        multipleJump = true;
        availableJumps--;

        rb.velocity = Vector2.up * jumpPower;
        animator.SetBool("Jump", true);
        AudioManager.instance.PlaySFX("jump");
    }
```

```
    if(multipleJump && availableJumps>0)
    {
        availableJumps--;

        rb.velocity = Vector2.up * jumpPower;
        animator.SetBool("Jump", true);
        AudioManager.instance.PlaySFX("jump");
    }
```

발동원리)
점프카운트를 설정(2), 캐릭터와 지면이 닿은 상태인지 확인, 스페이스바를 누를시 점프, 스페이스바를 누를때마다 점프 카운트를 -1씩 감소,

점프카운트가 0일 경우 점프불가능, 캐릭터와 지면이 닿았을 때 다시 점프카운트를 2로 회복.

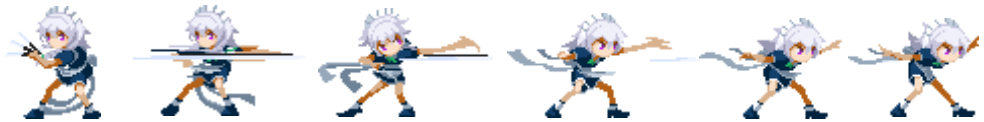


구동원리)

각각의 변수값에 Threshold값을 적용하여 x값과 y값의 수치가 일정 범위를 넘을 경우 각각 Jump, Falling 애니메이션을 호출.

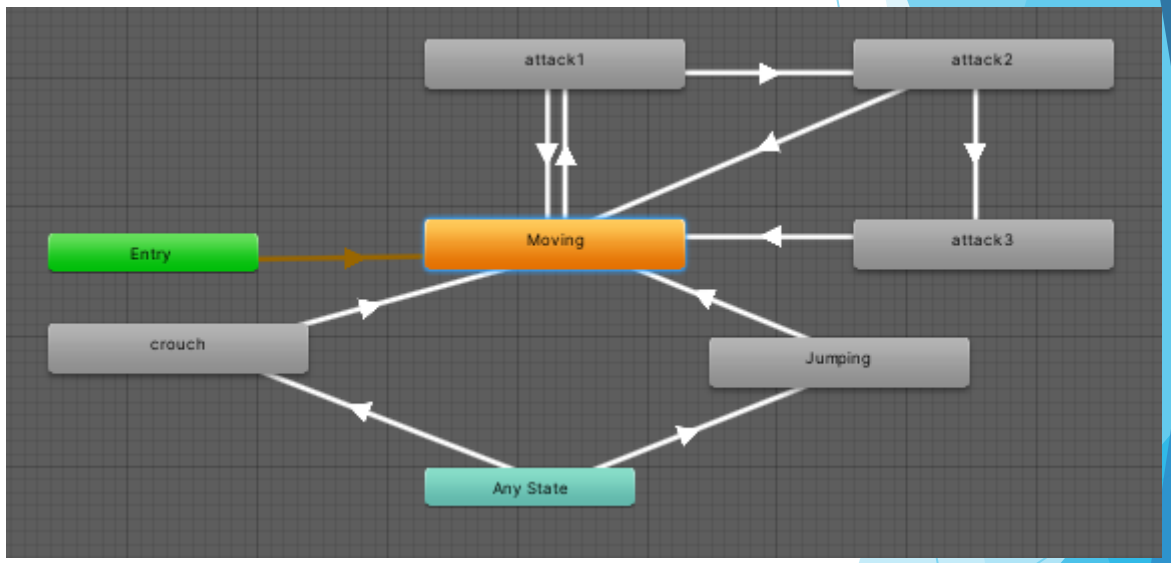
공격

1단 공격



2단 공격

3단 공격

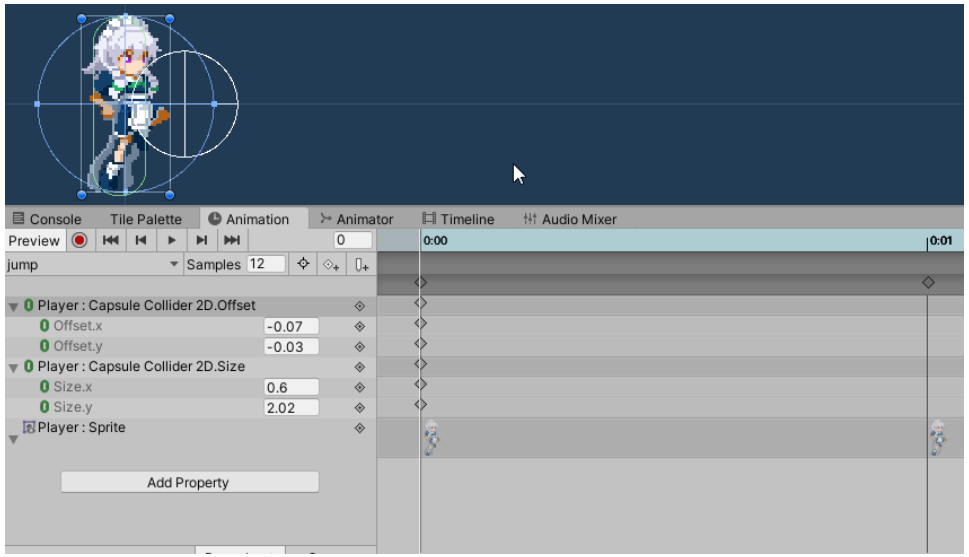


각각의 공격모션에 트리거 값(0,1,2)을 부여하여 콤보공격이 가능하게끔 구현함.

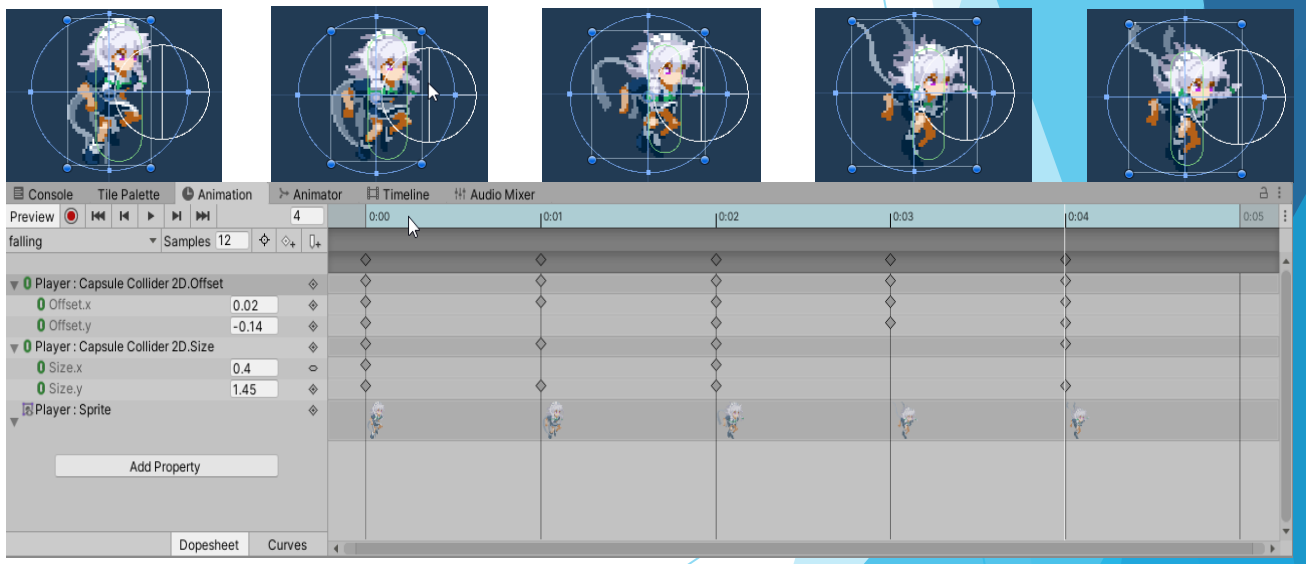
캐릭터 충돌범위 세부설정

캐릭터의 행동에 따라 모션이 변하기 때문에 그에 따른 충돌 영역을 프레임별로 조절.

Jump



Falling



영상 시연

소감 및 아쉬운 점

소감 및 아쉬운 점

소감

머릿속에서 구상했던 것이 이렇게 실제로 제 눈앞에 존재하는 것이 너무 신기한 것 같습니다. 정말 많은 시행착오도 겪고 포기하고 싶었던 순간도 많았지만 매주 교수님이 버그에 대해 같이 고민해주고 피드백을 잘해 주신 덕분에 여기까지 왔다고 생각합니다. 감사합니다.

아쉬운 점

애써 만든 AI기능에 비해 Enemy의 종류가 빈약.
플레이어의 한정적인 공격방식.
Boss의 Flip()기능이 정상적으로 작동하지 않음.
Boss패턴을 좀 더 다양화 하지 못한 점. Ex) 발악기, 무적패턴 등등
보스 SFX의 부재.
UI 디자인, 스토리 컷씬, 엔딩의 퀄리티.

Q & A

감사합니다.