2D Unity Project
점프스퀘어 게임 만들기

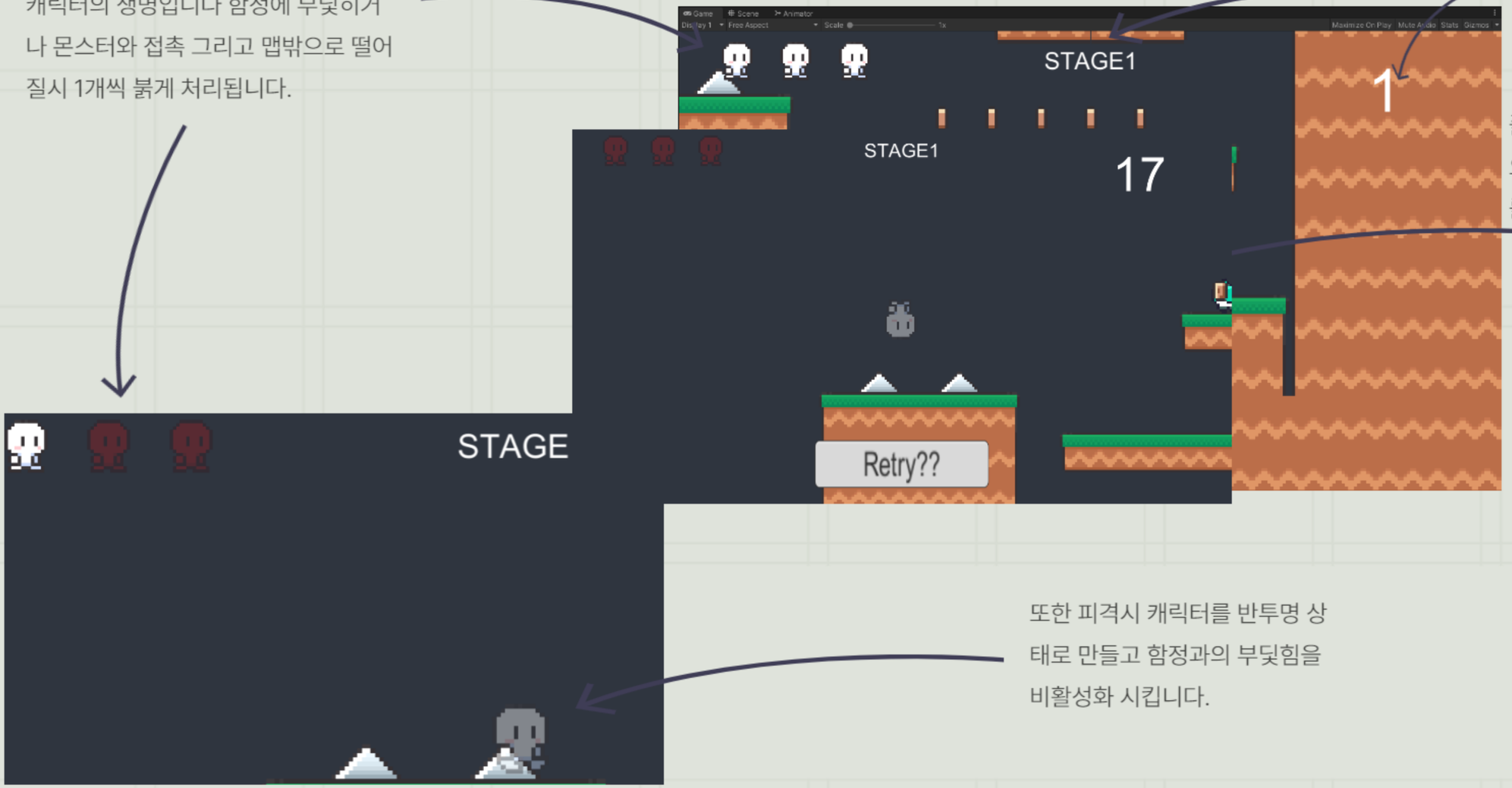
PHONE 010.5026.3204
E-Mail kwon6772@naver.com

목차 소개.

1. 전체 게임 소개

COMPETITION 01

캐릭터의 생명입니다 함정에 부딪히거나 몬스터와 접촉 그리고 맵밖으로 떨어질시 1개씩 붉게 처리됩니다.



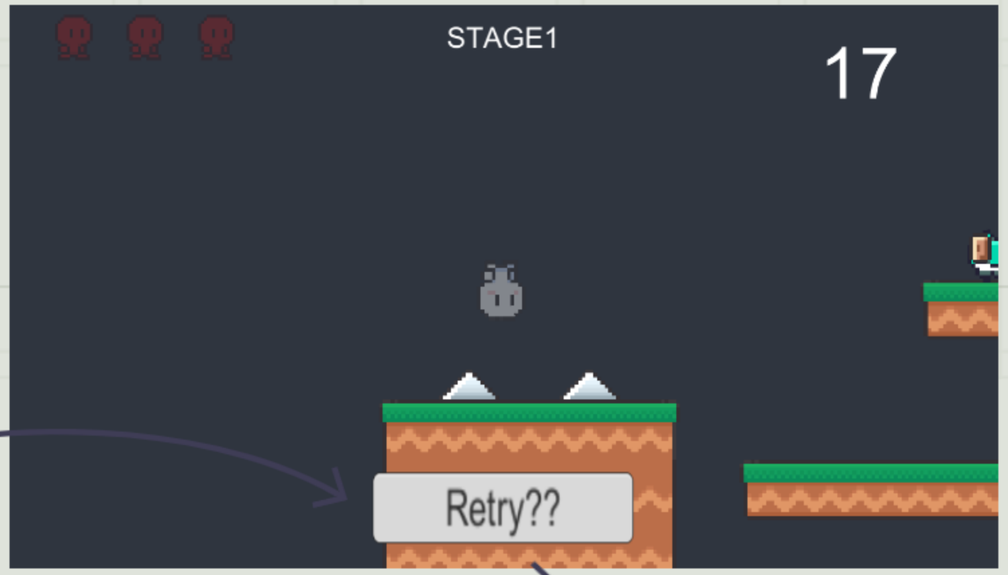
UI에 있는 스테이지 및 점수 입니다. 코인을 먹거나 몬스터를 밟아서 처치시 점수가 오릅니다.

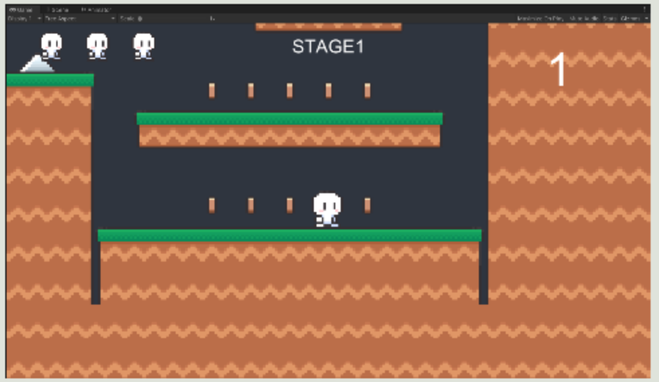
플레이어 입니다. 캐릭터 소스는 오픈 에셋 스토어에서 오픈 소스를 가져와서 활용했습니다.

또한 피격시 캐릭터를 반투명 상태로 만들고 함정과의 부딪힘을 비활성화 시킵니다.

모든 생명을 소진할시 캐릭터를 영구적으로 반투명하게 만들고 맵아래로 떨어지게 만들었습니다.

그리고 사망시 재시작 버튼을 활성화 시키고 클릭시 처음부터 다시 시작하게끔 만들었습니다.





맵에 존재하는 몬스터를 점프하여 위에서 밟을시 뒤집히며 처치됩니다.

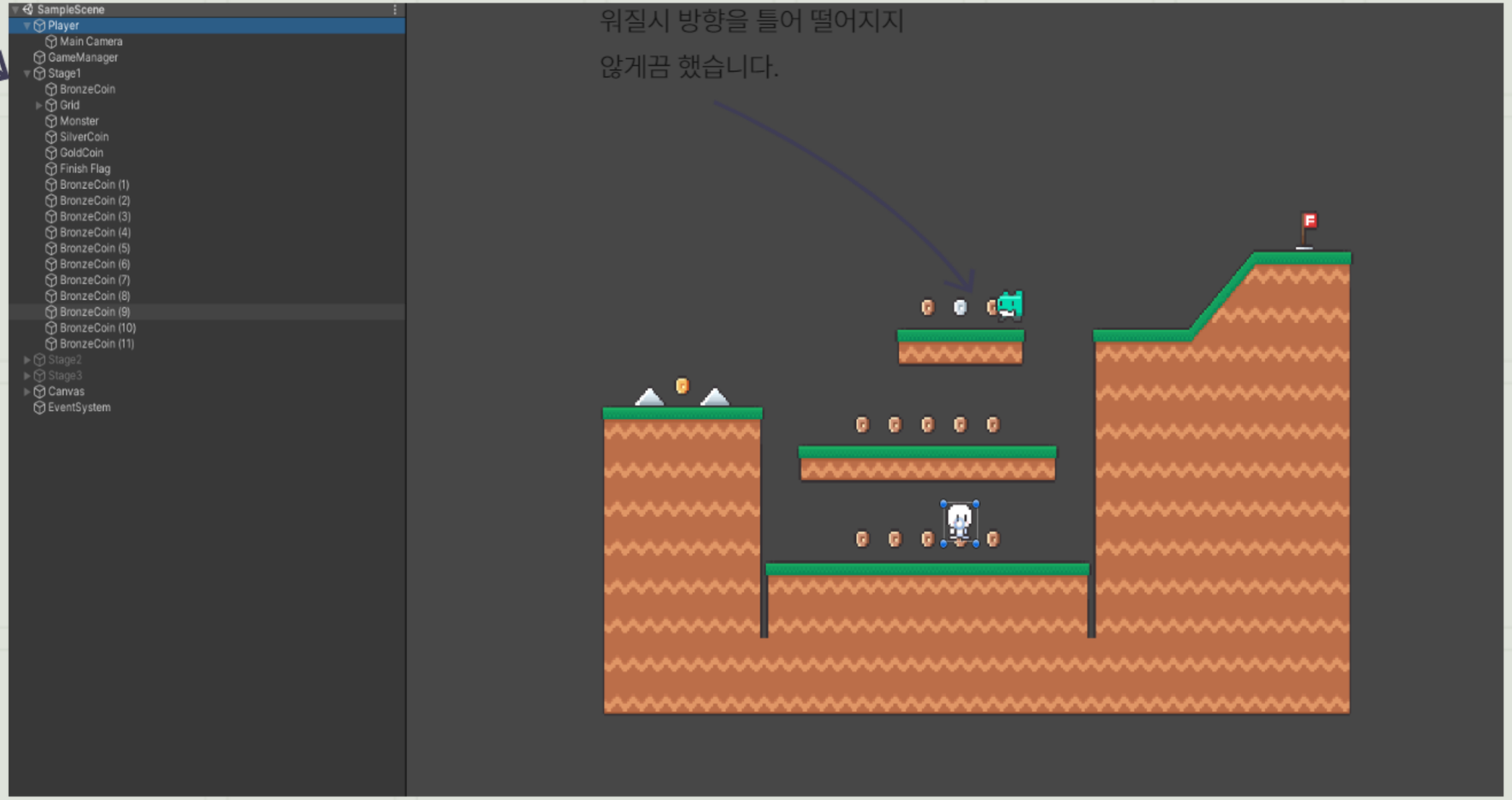
맵에 끝에 위치한 깃발에 닿으면 다음 맵으로 이동되게끔 만들었습니다.

2.상세 설명

COMPETITION 02

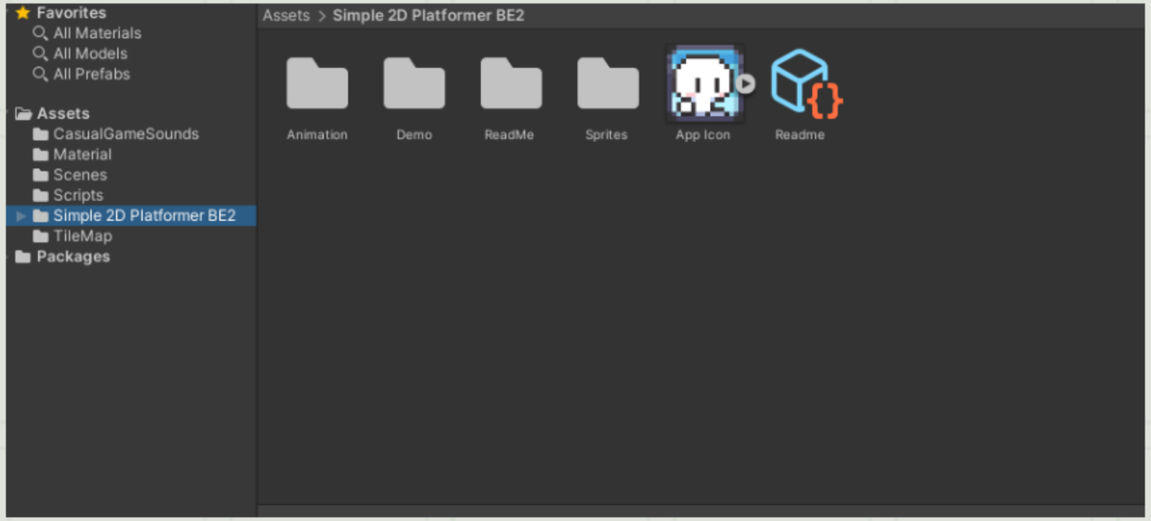
오브젝트들 카메라의 경우 플레이어에게 귀속시켜서 플레이어를 따라다닌다.

비활성화된 스테이지, 플레이어가 깃발에 닿을경우 순차적으로 활성화 되고 그전 스테이지를 비활성화 시킴

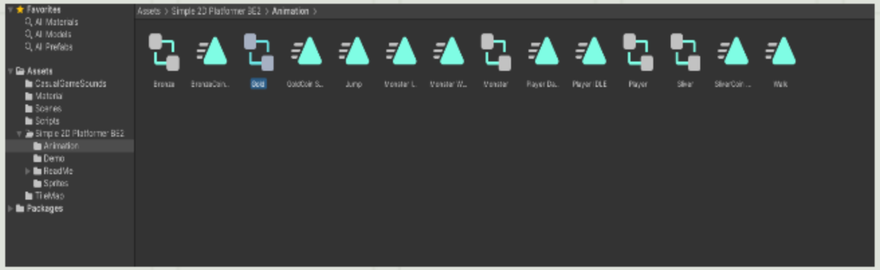


동전은 색깔별로 동,은,금이 존재합니다. 또한 몬스터가 일부 필드에 존재하는데 절벽에 가까워질시 방향을 틀어 떨어지지 않게끔 했습니다.

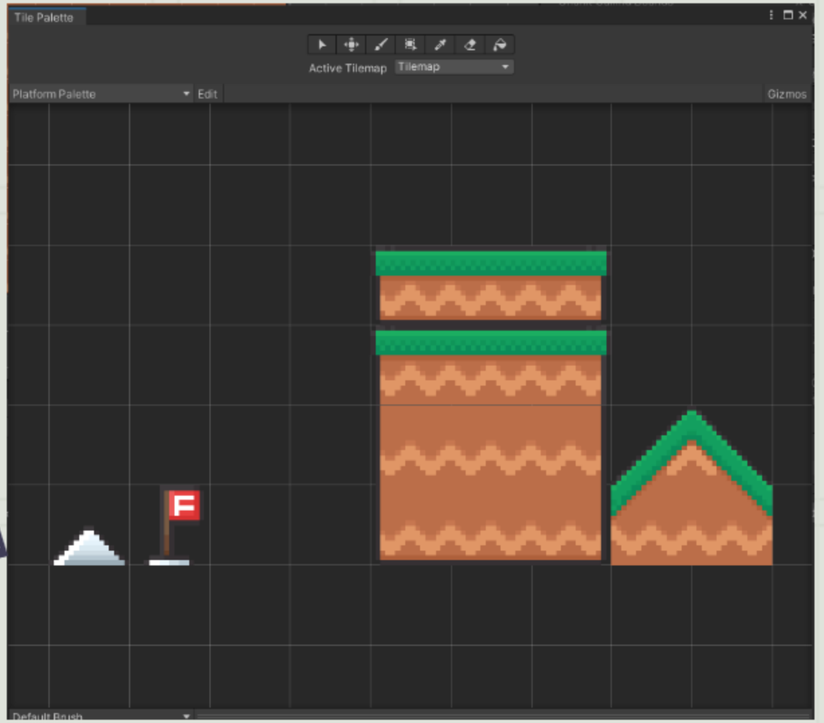
에셋 스토어에서 다운받은 무료 에셋이다 안에는 캐릭터,몬스터,코인 등의 이미지가 있다



애니메이션이다 캐릭터의 움직임을 자연스럽게 하기 위해 애니메이션에서 제작하고 애니메이터에서 경로를 설정해 다른 모션의 발생처리를 해주었다.

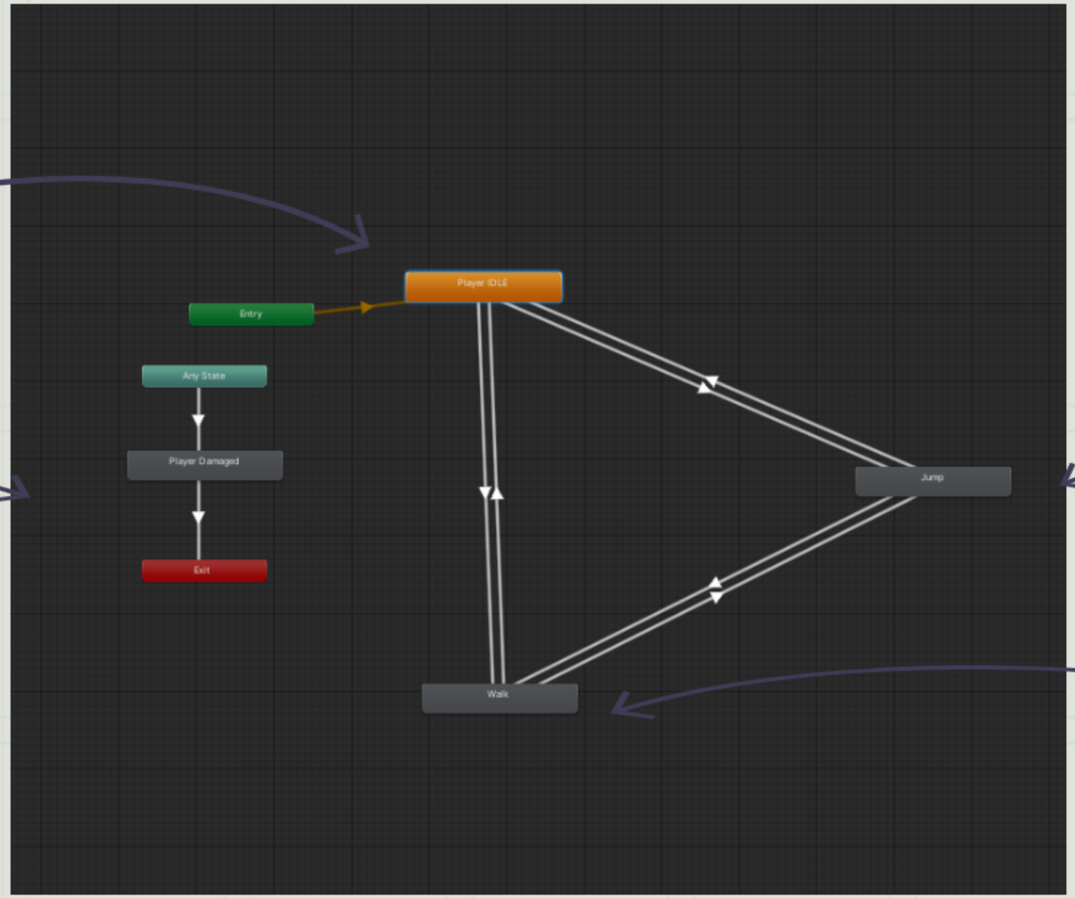


타일 팔레트를 활용하여 지정한후 맵을 그리듯이 만들수 있으며 그림과 다르게 실제 오브젝트의 물리범위를 조절할수 있다.



평소에 가만히 있을때 유지되는 모션

어떤 상태에서도 발생할수 있는 모션, 모션을 취한후 Exit로 해당 상태를 탈출시켜준다



점프하는 모션

걷는 모션

3.코드 설명

COMPETITION 03

```
public class Player_Move : MonoBehaviour
{

    public GameManager gamemanager;
    public float Maxspd;
    public float jumpPower;
    Rigidbody2D rigid2d;
    SpriteRenderer spriteRenderer;
    Animator animator;
    Collider2D col;
    public AudioClip audioJump;
    public AudioClip audioAttack;
    public AudioClip audioDamaged;
    public AudioClip audioCoin;
    public AudioClip audioDie;
    public AudioClip audioFinish;

    AudioSource audiosource;
    void Awake(){
        rigid2d = GetComponent<Rigidbody2D>();
        spriteRenderer = GetComponent<SpriteRenderer>();
        animator = GetComponent<Animator>();
        col = GetComponent<CapsuleCollider2D>();
        audiosource = GetComponent<AudioSource>();
    }
```

전역 변수 선언 및 Unity 에서 지원하는 기능 선언

Awake() 매서드로 GetComponent해서 기능을 참조하여 적용

스위치 문을 이용해서 상황에 따른 사운드를 출력하는 함수

```
void PlaySound(string action){
    switch (action)
    {
        case "Jump":
            audiosource.clip = audioJump;
            audiosource.Play();
            break;
        case "Attack":
            audiosource.clip = audioAttack;
            audiosource.Play();
            break;
        case "Damaged":
            audiosource.clip = audioDamaged;
            audiosource.Play();
            break;
        case "Coin":
            audiosource.clip = audioCoin;
            audiosource.Play();
            break;
        case "Die":
            audiosource.clip = audioDie;
            audiosource.Play();
            break;
        case "Finish":
            audiosource.clip = audioFinish;
            audiosource.Play();
            break;
    }
}
```

각각 점프,공격(밟기),피격,코인 습득, 쓰러짐,골인에 대항하는 소리발생 을 준비했다.

Update() 메소드에 점프기능을 구현, 점프 버튼 을 받고, 애니메이터에서 IsJumping에 대한 참 거짓 값을 따지는 조건문을 사용했다.
조건문이 돌면 캐릭터가 위로 점프하게 되고 SetBool로 점프하는 애니메이션을 출력, playsoung 함수에 해당 문자열을 담아서 스위치문을 돌게한다.

```
void Update()
{
    //Jump
    if(Input.GetButtonDown("Jump") && !animator.GetBool("isJumping")){
        rigid2d.AddForce(Vector2.up * jumpPower ,ForceMode2D.Impulse);
        animator.SetBool("isJumping", true);
        PlaySound("Jump");
    }



    if(Input.GetButtonUp("Horizontal")){
        //normalized벡터 크기를 1로 만든 상태
        rigid2d.velocity = new Vector2(rigid2d.velocity.normalized.x * 0.5f, rigid2d.velocity.y);
    }

    if(Input.GetButton("Horizontal"))
        spriteRenderer.flipX = Input.GetAxisRaw("Horizontal") == -1;

    if(Mathf.Abs(rigid2d.velocity.x) < 0.3)
        animator.SetBool("isWalking", false);
    else
        animator.SetBool("isWalking", true);


}
```

아래에는 각각 캐릭터를 움직이게끔 하고 버튼을 입력받을때의 방향을 맞춰서 캐릭터 이미지가 좌우 반전을 하게 만든다 입력이 끝날땐 빠르게 속도를 줄여서 캐릭터가 미끄러지지 않고 딱 멈추게 만들었다. 그리고 캐릭터 움직이는 속도에 맞춰서 걷는 애니메이션을 출력한다.

캐릭터가 움직일때 설정한 최고속도보다 낮을경우를 조건으로 주고 캐릭터가 움직일때 속도를 주게끔 만들었다

```
0 references
void FixedUpdate() {
    //Move
    float h = Input.GetAxisRaw("Horizontal");

    rigid2d.AddForce(Vector2.right * h * 2, ForceMode2D.Impulse);

    if(rigid2d.velocity.x > Maxspd)//Right Max Speed
        rigid2d.velocity = new Vector2(Maxspd, rigid2d.velocity.y);
    else if(rigid2d.velocity.x < Maxspd*(-1))//left max speed
        rigid2d.velocity = new Vector2(Maxspd*(-1), rigid2d.velocity.y);

    //Landing Platform
    if(rigid2d.velocity.y < 0){
        Debug.DrawRay(rigid2d.position, Vector3.down, new Color(0,1,0));
        RaycastHit2D rayhit = Physics2D.Raycast(rigid2d.position, Vector3.down, 1, LayerMask.GetMask("Platform"));
        if(rayhit.collider != null){
            if(rayhit.distance < 0.5f)
                animator.SetBool("isJumping", false);
        }
    }
}
```

```
0 references
void FixedUpdate() {
    //Move
    float h = Input.GetAxisRaw("Horizontal");

    rigid2d.AddForce(Vector2.right * h * 2, ForceMode2D.Impulse);

    if(rigid2d.velocity.x > Maxspd)//Right Max Speed
        rigid2d.velocity = new Vector2(Maxspd, rigid2d.velocity.y);
    else if(rigid2d.velocity.x < Maxspd*(-1))//left max speed
        rigid2d.velocity = new Vector2(Maxspd*(-1), rigid2d.velocity.y);

    //Landing Platform
    if(rigid2d.velocity.y < 0){
        Debug.DrawRay(rigid2d.position, Vector3.down, new Color(0,1,0));
        RaycastHit2D rayhit = Physics2D.Raycast(rigid2d.position, Vector3.down, 1, LayerMask.GetMask("Platform"));
        if(rayhit.collider != null){
            if(rayhit.distance < 0.5f)
                animator.SetBool("isJumping", false);
        }
    }
}
```

DrawRay : 캐릭터 아래로 시작지점,길이,색이 지정된 선을 하나 그린다.
RaycastHit2D : Ray에 대한 상호작용에 대한 기능(저기선 충돌 여부를 보고있다)
Platform이란 명칭의 레이어에 따른 충돌 여부에 따라 점프 모션을 유지하거나 유지하지 않는다.
(점프후 착지전까지 점프모션을 유지하거나 유지하지 않는다)

```
void OnCollisionEnter2D(Collision2D collision) {
    if(collision.gameObject.tag == "Enemy"){
        if(rigid2d.velocity.y < 0 && transform.position.y > collision.transform.position.y){
            OnAttack(collision.transform);
            gamemanager.stagePoint += 5;
        }
        else{
        OnDamaged(collision.transform.position);
        }
    }
    else if(collision.gameObject.tag == "Trap")
        OnDamaged(collision.transform.position);
}
```

충돌이벤트
충돌 대상의 태그가 Enemy일 경우 Y 위치값의 비교 값여부에 따라 공격이벤트가 발생되거나 데미지 이벤트가 발생한다.
하지만 대상의 태그가 Trap일 경우 데미지 이벤트만 발생한다.

```
private void OnTriggerEnter2D(Collider2D collision) {
    if(collision.gameObject.tag == "Coin"){
        bool isBronze = collision.gameObject.name.Contains("Bronze");
        bool isSilver = collision.gameObject.name.Contains("Silver");
        bool isGold = collision.gameObject.name.Contains("Gold");
        PlaySound("Coin");

        if(isBronze){
            gamemanager.stagePoint += 1;
        }
        if(isSilver){
            gamemanager.stagePoint += 5;
        }
        if(isGold){
            gamemanager.stagePoint += 10;
        }

        collision.gameObject.SetActive(false);
        // if(collision.gameObject.layer == 10)
        //     game_manager.stagePoint += 1;
        // else if(collision.gameObject.layer == 11)
        //     game_manager.stagePoint += 5;
        // else
        //     game_manager.stagePoint += 10;
    }
    else if(collision.gameObject.tag == "Finish"){
        gamemanager.NextStage();

        PlaySound("Finish");

    }
}
```

트리거 이벤트

오브젝트 대상의 태그가 Coin 일 경우 bool을 이용해서 해당 오브젝트의 이름에 따른 참 거짓을 판단하면서 Coin 사운드를 낸다.

그후 오브젝트의 따른 점수를 다르게 올라가게끔 만들었다.

닿은 오브젝트는 비활성화 시킨다.

닿은 오브젝트의 태그가 Finish일 경우 다음 스테이지로 넘어가면서 소리가 나게끔 만들었다.

```
1 reference
void OnAttack(Transform enemy){
    PlaySound("Attack");
    Monster_Move monster_Move = enemy.GetComponent<Monster_Move>();
    monster_Move.OnDamaged();
    rigid2d.AddForce(Vector2.up * 15,ForceMode2D.Impulse);
}
2 references
void OnDamaged(Vector2 targetPosition)
{
    PlaySound("Damaged");

    gamemanager.HealthDown();

    gameObject.layer = 30;

    spriteRenderer.color = new Color(1,1,1,0.4f);

    int dirc = transform.position.x-targetPosition.x > 0 ? 1: -1;
    rigid2d.AddForce(new Vector2(dirc, 1)*7 ,ForceMode2D.Impulse);

    animator.SetTrigger("doDamaged");

    if(gamemanager.health > 0){
        Invoke("Normal", 2);
    }
}
```

몬스터에게 공격을 할시

색에 투명도를 준후 위아래로 뒤집어준다. col(capsulCollider2D)를 끄고 밟혔을때 위로 뛰어오른다(타격감) 5초뒤 Deactive 함수 실행(오브젝트를 끈다)

역으로 공격을 받을시 데미지 사운드를 실행, 체력감소 함수를 실행하고 플레이어의 레이어를 30번(데미지 전용 레이어)로 바꾸어준다(무적시간 적용) 그 사이동안 캐릭터를 불투명하게 만들고 맞았을때 캐릭터를 해당 오브젝트와 부딪힌 방향의 반댓방향으로 가볍게 튕겨낸다. 그리고 체력이 1이라도 남아있다면 2초후 캐릭터 를 다시 원래상태인 31번 레이어로 되돌린다(무적시간 삭제, 불투명도 복구)

```
1 reference
public void OnDamaged()
{
    spriteRenderer.color = new Color(1,1,1,0.4f);

    spriteRenderer.flipY = true;

    col.enabled = false;

    rigid2d.AddForce(Vector2.up * 10 , ForceMode2D.Impulse);

    Invoke("DeActive", 5);
}

0 references
void DeActive()
{
    gameObject.SetActive(false);
}
```

```
0 references
void Normal(){

    gameObject.layer = 31;

    spriteRenderer.color = new Color(1,1,1,1);
}
```

```
2 references
public class Monster_Move : MonoBehaviour
{
    7 references
    Rigidbody2D rigid2d;

    9 references
    public int nextMove;
    2 references
    Animator animator;
    5 references
    SpriteRenderer spriteRenderer;

    2 references
    CapsuleCollider2D col;

    0 references
    void Awake(){
        rigid2d = GetComponent<Rigidbody2D>();
        animator = GetComponent<Animator>();
        spriteRenderer = GetComponent<SpriteRenderer>();
        col = GetComponent<CapsuleCollider2D>();

        Think();
    }
```

물리 및 애니메이션 참조

Awake에서 Think함수를 부르면 Think 함수내에서 자기 자신을 부르는 재귀함수로 작동된다.
랜덤 을 사용하여 -1,0,1에 따라 다른 행동을 하고 2~5초의 랜덤한 간격으로 행동을 다르게 한다(좌로 이동, 제자리 대기,우로 이동)

```
void Think()
{
    //Set Next Active
    nextMove = Random.Range(-1,2);

    //Recursive
    float nextThinkTime = Random.Range(2f,5f);
    Invoke("Think", nextThinkTime);

    //Sprite Animation
    animator.SetInteger("WalkSpeed", nextMove);

    //방향전환 Flip Sprite
    if(nextMove != 0)
        spriteRenderer.flipX = nextMove == 1;
}

1 reference
void Turn(){
        nextMove = nextMove * -1;
        spriteRenderer.flipX = nextMove == 1;
        CancelInvoke();
        Invoke("Think", 2);
}
```

```
0 references
void FixedUpdate()
{
    //Move
    rigid2d.velocity = new Vector2(nextMove, rigid2d.velocity.y);
    // if(nextMove == -1){
    //     animator.SetBool("isWalking",true);
    //     spriteRenderer.flipX = false;
    // }
    // else if(nextMove == 0)
    //     animator.SetBool("isWalking",false);
    // else{
    //     animator.SetBool("isWalking",true);
    //     spriteRenderer.flipX = true;
    // }

    //Platform Check
    Vector2 frontVec = new Vector2(rigid2d.position.x + nextMove*0.3f, rigid2d.position.y);
    Debug.DrawRay(frontVec, Vector3.down, new Color(0,1,0));
    RaycastHit2D rayhit = Physics2D.Raycast(frontVec, Vector3.down, 1 , LayerMask.GetMask("Platform"));
    if(rayhit.collider == null){
        Turn();
    }
}
```

인스턴스로 랜덤값과 속력값을 담는다.

```
void Think()
{
    //Set Next Active
    nextMove = Random.Range(-1,2);
```

위에서 생성된 값을 가지고 앞으로 걸을지 뒤로 걸을지 판별,Ray를 만들되 frontVec을 선언해서 몬스터의 중심이 아닌 다른 위치에 생성되게끔 했다.
Platform에 충돌이 null이 될시 몬스터를 강제로 반댓방향으로 돌린다(몬스터 추락 방지)

게임 매니저 클래스에서 게임에 필요한 UI나 Scene 을 바꾸기 위한 SceneManagement 사용.

```
using UnityEngine.UI;
using UnityEngine.SceneManagement;

1 reference
public class GameManager : MonoBehaviour
{
    2 references
    public int totalPoint;
    7 references
    public int stagePoint;
    5 references
    public int StageIndex;
    6 references
    public int health;
    3 references
    public Player_Move player;
    3 references
    public GameObject[] stages;

    3 references
    public Image[] UIhealth;
    1 reference
    public Text UIPoint;
    1 reference
    public Text UIStage;
    3 references
    public GameObject RestartBtn;
```

스테이지 포인트 및 총 포인트를 합산해서 매순간 모은 포인트를 보이게끔 함

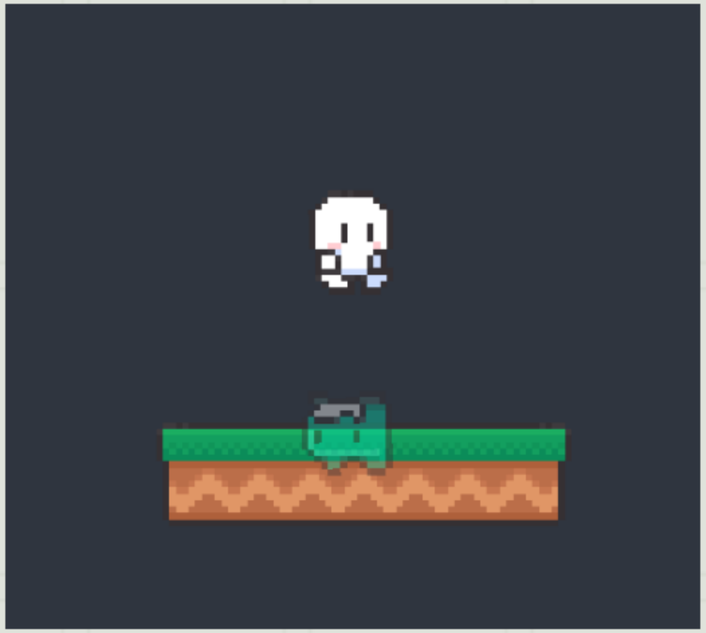
필요한 전역 변수 및 클래스 참조, 스테이지 배열 , 체력 배열

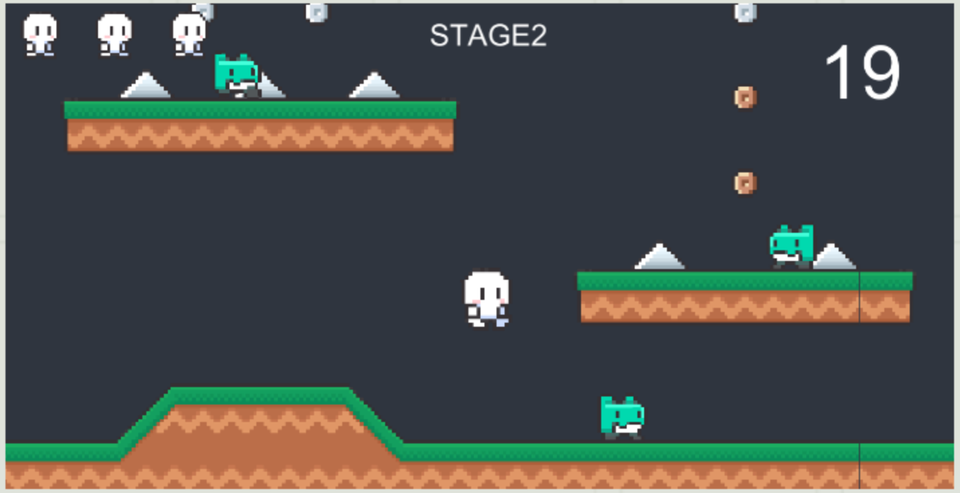
UI텍스트 및 버튼 오브젝트

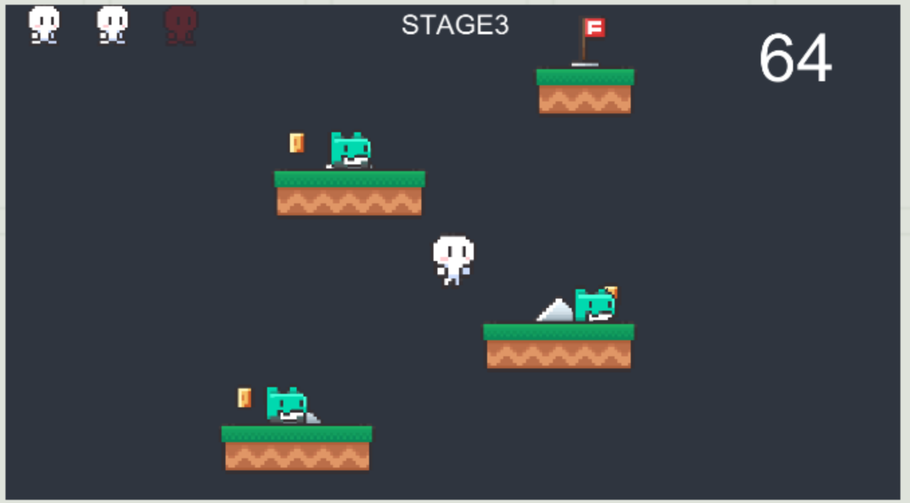
```
private void Update() {
    UIPoint.text = (totalPoint + stagePoint).ToString();
}

1 reference
public void NextStage()
{
    if(StageIndex < stages.Length-1){
        stages[StageIndex].SetActive(false);
        StageIndex++;
        stages[StageIndex].SetActive(true);
        PlayerReposition();

        UIStage.text = "STAGE" + (StageIndex+1);
    }
    else{
        Time.timeScale = 0;
        Debug.Log("게임 끝");
        Text btnText = RestartBtn.GetComponentInChildren<Text>();
        btnText.text = "Game Clear";
        RestartBtn.SetActive(true);
    }

    totalPoint += stagePoint;
    stagePoint = 0;
}
```

stageIndex와 스테이지 배열의 길이를 비교해서 다음으로 넘길지 클리어인지 여부 파악, 클리어할시 timescale = 0 으로 하여 시간을 멈추고 gameClear 가 쓰여진 버튼을 띄운다.

플레이 장면







마치며...

3D 와는 다르게 또다른 느낌을 받았습니다 대부분 컴포넌트가 3D가 아닌 2D로 이름이 붙여져 있었고 RigidBody 나 physics Material 2D로 도 차이가 꽤 분명했다
이번이 유니티로 간단한 게임 만들기 2편이다 에셋은 에셋 스토어에서 무료로 제공되는게 많다보니 접근하기 쉬웠다.
다음번엔 기회가 되면 에셋또한 내가 직접 만들고 만들어진 에셋으로 모든부분을 내가 만든 에셋으로 채워진 게임을 만들어보면 참 재밌을것 같다.

Thank you for enjoying

권용규

PHONE 010 5026 3204
E-Mail kwon6772@naver.com