홍민성

PORTFOLIO

게임 프로그래밍

Contact

Email: timet7429@gmail.com

Phone: 010-8811-1756

PROFILE

홍민성

Hong Minseong

학력 및 수상 경력

[학력]

계원예술대학교 게임미디어학과 2023.03 ~ 2025.02 졸업

수지고등학교 2021.02 졸업

[수상 경력]

제 1회 AI활용 게임 개발 경진대회 최우수상 2024.11

보유 능력

Word
게임 시스템 규칙 설계

Excel
함수 사용, 데이터 테이블 작성

PowerPoint
데이터 스키마 작성 및 함수 사용

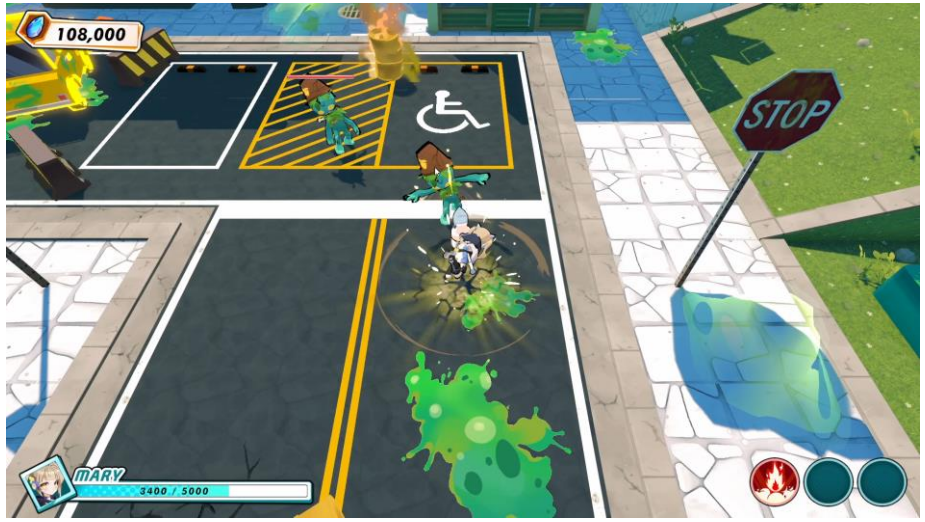
Adobe Photoshop
이미지 편집

Unity 3D
간단한 게임 제작

Git Hub
협업 프로젝트에서 형상관리에 활용

01. IMMACULATE

2024년 계원예술대학교 “털 날리는 팀”의 졸업작품입니다.



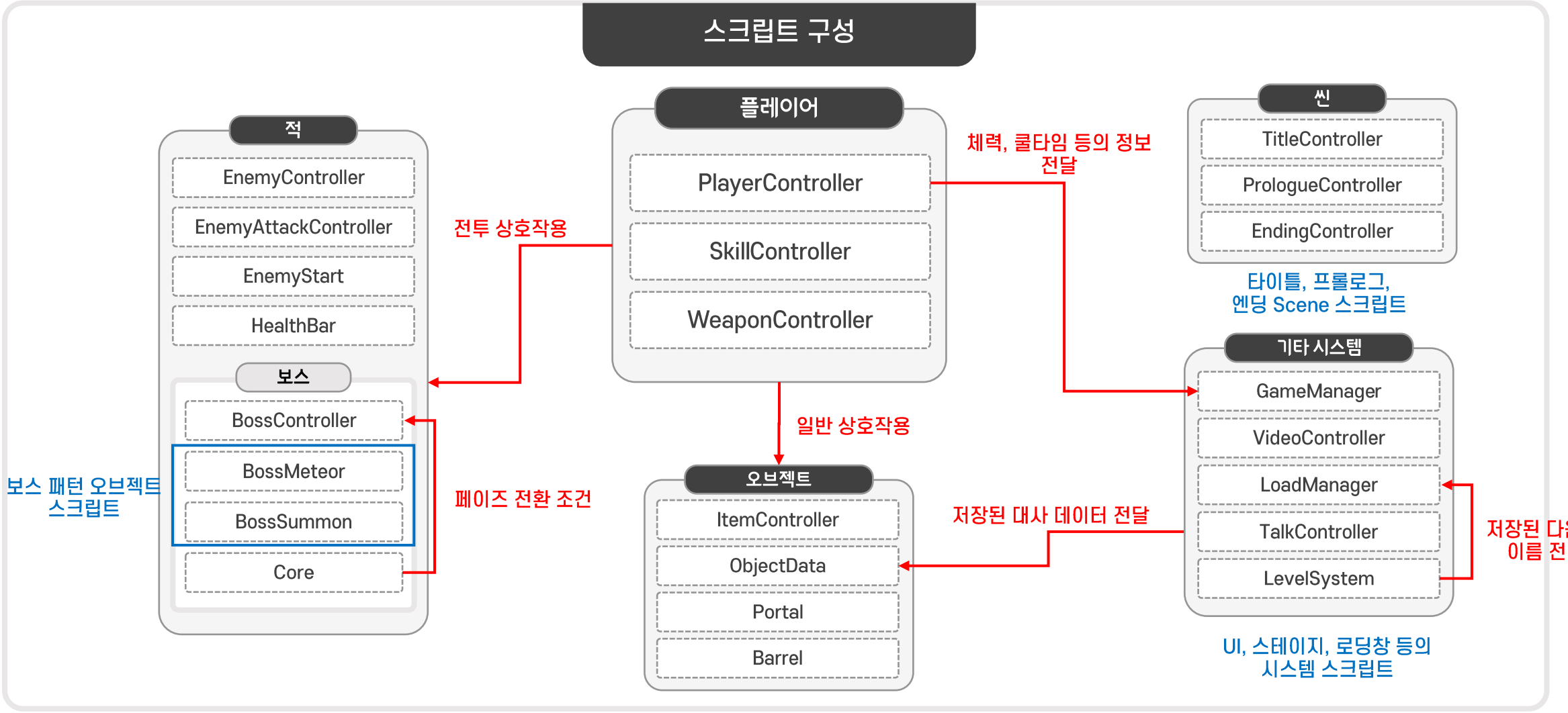
- 우주해결사인 주인공은 의뢰를 받아 트래쉬 몬스터들을 제거하며 목적을 달성해 나가는 3D 액션 어드벤쳐 RPG 게임 입니다.
- 필드를 돌아다니며 스킬을 획득하고 적과 전투하며 경험치를 얻어 플레이어를 성장시킬 수 있습니다.

- 화려한 이펙트와 모션, 조작감과 적 피드백 등을 통해 액션과 타격감을 살리고자 하였습니다.
- NPC와의 대화, 2D 컷신, 3D 애니메이션 컷신 등을 통해 더욱 스토리에 몰입할 수 있도록 만들었습니다.

	설명
개발 기간	2024.03.01~2024.09.25
개발 엔진	Unity
플랫폼	PC
장르	3D 액션 어드벤처 RPG

01-1. 스크립트 소개

전체적인 스크립트 구성 및 대략적인 상호작용



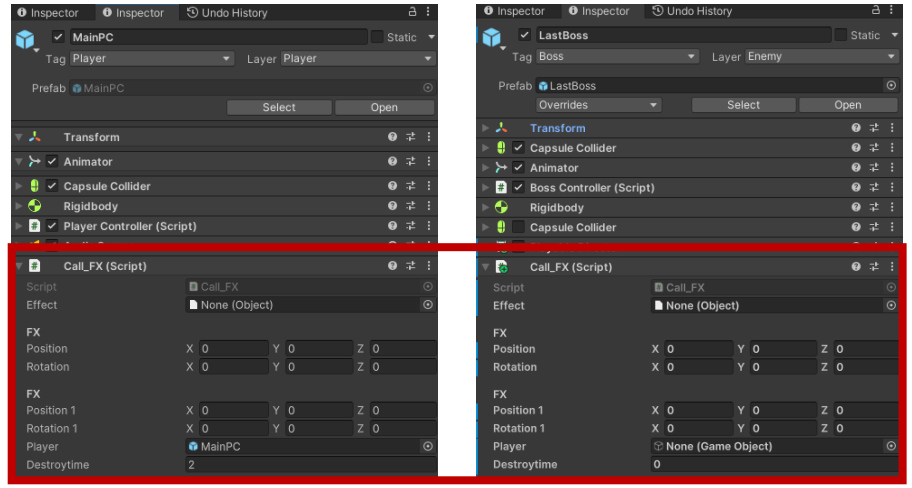
01-2. 대표 스크립트 (Call_FX)

이펙트를 유니티 이벤트로 호출하는 스크립트입니다.
깃포크로 여러 사람이 작업하는 환경 특성상 재활용과 사용의 용이함을 중점적으로 제작하였습니다.

```
참조 0개
private void FX(Object pref)
{
    //지정된 effect 게임 오브젝트를 childFX라는 이름으로 생성
    GameObject childFX = Instantiate(pref) as GameObject;

    childFX.transform.position = Player.transform.position + Position;
    childFX.transform.rotation = Player.transform.rotation;
    //이펙트가 끝나는 시간 뒤 삭제
    Destroy(childFX, Destroytime);
}

참조 0개
private void SkillFX(Object pref)
{
    //지정된 effect 게임 오브젝트를 childFX라는 이름으로 생성
    GameObject childFX = Instantiate(pref) as GameObject;
    //부모 설정
    childFX.transform.SetParent(Player.transform, false);
    childFX.transform.position = Player.transform.position + Position;
    childFX.transform.rotation = Player.transform.rotation;
    //이펙트가 끝나는 시간 뒤 삭제
    Destroy(childFX, 5f);
}
```

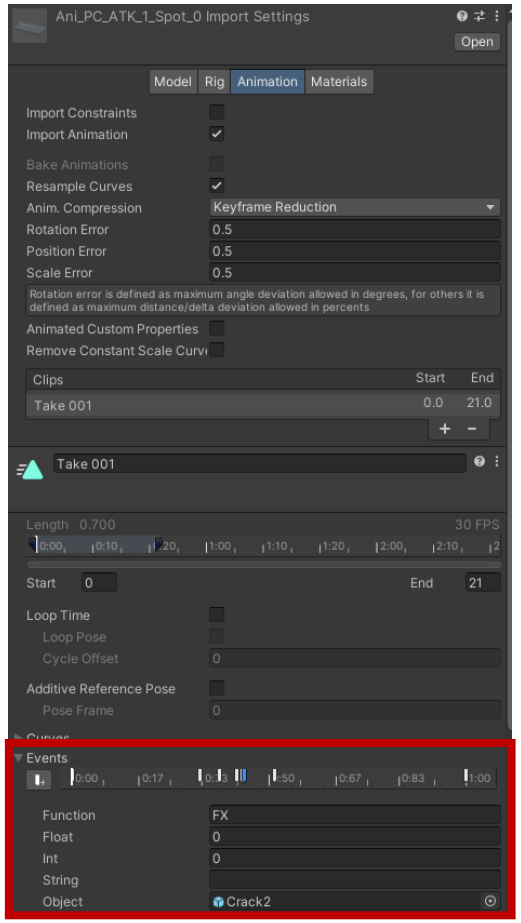
월드에 소환되어야 하는
일반 이펙트인 FX와

오브젝트와 함께 움직이는
스킬 이펙트인 SkillFX의 코드를 분리



<플레이어> <보스>

다른 오브젝트에서도 재활용이 가능하도록 캡슐화



유니티 애니메이션의 이벤트를 활용하여
프로그래머가 아닌 디자이너도 이펙트를 쉽게 호출
가능하게 제작

01-2. 대표 스크립트 (대화창)

대화 UI에 필요한 스크립트에 관한 설명입니다.
엑셀에 정리된 데이터를 바탕으로 제작되었습니다. 추후에 엑셀과 연동해서 자동으로 데이터를 불러오는 시스템을 추가한다면 더욱 관리가 용이하겠다고 느꼈습니다.

```
public class TalkController : MonoBehaviour
{
    Dictionary<int, string[]> talkData;
    Dictionary<int, string[]> charactorData;
    Dictionary<int, Sprite> ImgData;
    [SerializeField] Sprite[] Arr;
    Unity 메시지 | 참조 0개
    void Awake()
    {
        talkData = new Dictionary<int, string[]>();
        charactorData = new Dictionary<int, string[]>();
        ImgData = new Dictionary<int, Sprite>();
        GenerateData();
    }

    // Update is called once per frame
    참조 1개
    void GenerateData()
    {
        //이름 : 이미지 : 대사
        talkData.Add(120, new string[] {
            "말이:1001:카~ 쏨쏨하고 탑탑한 공기~ 이곳이 나의 무대인 건가~!",
            "탐이:1013:으윽 속이 울렁거려요 냥.. 어쩜 이리 운전을 난폭하게 하십니까앙...",
            "말이:1001:하핫 내가 장롱면허라.. 안전하게 잘 도착하면 된 거지~! 잘 부탁드려요~! 의뢰냥 님!",
            "탐이:1013:하아.. 정말 믿을만한 건지.. 큰 맘먹고 거장 우주 해결사께 의뢰한 거란 말입니다! 진지하게 임해주세용.!!"});

        talkData.Add(130, new string[] {
            "말이:1002:아앗? 저기 무엇이 있는데.;.?",
            "말이:1000:문명의 파편..이 별에서의 물건들과 접촉하면 주요 파편을 얻을 수 있는 건가 학원에서 배운 대로네 공부한 보람이 있어",
            "탐이:1012:냐옹~역시 해결사님! 미법도 부리시는군요~",
            "말이:1001:이 파편은 해당 파편의 원소와 관련된 문명의 기록이 담겨있어.이 기록들을 토대로 해당 파편의 능력을 쓸 수 있다구 해",
            "탐이:1011:냐아옹~!이 파편들의 기억과 힘을 이용해 지구를 되돌릴 수 있다는 거군요 에스코트 특특히 해드릴께요.!그럼 바로 가볼까요~!"
            });

        talkData.Add(160, new string[] {
```

```
    talkData.Add(260, new string[] {
        "탐이:1011:앗 지금 번뜩이 생겼어요!",
        "탐이:1011:맵 가장자리에 있는 가스등을 터뜨려 몬스터를 기절시켜보세요 냥..!",
        "말이:1001:오케이~ 지금 바로 간다!!",
        "탐이:1012:냐냐아!! 조금만 더 힘내세요!!"
    });
    //주인공
    //0 기본, 1 기쁨, 2 놀람, 3 걱정
    ImgData.Add(1000 + 0, Arr[0]);
    ImgData.Add(1000 + 1, Arr[1]);
    ImgData.Add(1000 + 2, Arr[2]);
    ImgData.Add(1000 + 3, Arr[3]);

    //애옹이
    //0 기본, 1 기쁨, 2 놀람, 3 걱정
    ImgData.Add(1010 + 0, Arr[4]);
    ImgData.Add(1010 + 1, Arr[5]);
    ImgData.Add(1010 + 2, Arr[6]);
    ImgData.Add(1010 + 3, Arr[7]);
}

참조 1개
public string GetTalk(int id, int talkIndex)
{
    if (talkIndex == talkData[id].Length)
        return null;
    else
        return talkData[id][talkIndex];
}

참조 2개
public Sprite GetPortrait(int id, int PortraitIndex)
{
    return ImgData[id + PortraitIndex];
}
```

유니티의 Dictionary 자료구조를 이용하여 대사, 캐릭터 일러스트 인덱스 저장

```
참조 1개
private void Talk(int id)
{
    string talkData = talkManager.GetTalk(id, talkIndex);

    if (talkData == null)
    {
        isAction = false;
        talkIndex = 0;
        CloseUI(5);
        if (player.nearObject.tag == "ForceTalking")
        {
            Destroy(player.nearObject);
        }
        return;
    }

    //이름 : 이미지 : 대사
    talkName.text = talkData.Split(':')[0];
    talktext.text = talkData.Split(':')[2];


    //주인공
    if (int.Parse(talkData.Split(':')[1]) < 1010)
    {
        MainPCImg.sprite = talkManager.GetPortrait(id, int.Parse(talkData.Split(':')[1]));
        MainPCImg.color = new Color(1, 1, 1, 1);
        RamyImg.color = new Color(1, 1, 1, 0);
    } //고양이
    else if (int.Parse(talkData.Split(':')[1]) < 1020)
    {
        RamyImg.sprite = talkManager.GetPortrait(id, int.Parse(talkData.Split(':')[1]));
        AudioSource.PlayClipAtPoint(catSound, transform.position, soundVolume);
        RamyImg.color = new Color(1, 1, 1, 1);
        MainPCImg.color = new Color(1, 1, 1, 0);
    }


    isAction = true;
    talkIndex++;
}
```

저장된 string 데이터를
“ : “를 기준으로 화자 이름,
이미지 인덱스, 대사로 분리

다음 저장된 대사 데이터가 없을 시
대화창을 닫고 대사 출력을 중지

```
참조 4개
public void TalkAction(GameObject scanObj)
{
    OpenUI(5);

    scanObject = scanObj;
    ObjectData objData = scanObject.GetComponent<ObjectData>();
    Talk(objData.id);

    talkPanel.SetActive(isAction);
}
```

<대사 상호작용 스크립트>

<출력된 대사 창>

01-2. 대표 스크립트 (타격감 구현 1 – 타게팅 시스템)

타격감을 살리기 위해 타게팅, 카메라 진동 아이디어를 구현했습니다.
타게팅 시스템은 조작감 개선, 카메라 진동은 타격 연출을 보충하기 위해 기획했습니다.

```
void FindAndAttackClosestEnemy()
{
    // 원하는 탐지 반경 설정
    //A. 마우스 안보이고 플레이어 가장 근처 적만 타격
    //B. 마우스 방향으로만
    //C. 마우스 방향으로 타게팅
    //D. 하댕스처럼
    float fanAngle = targettingAngle;
    closestEnemy = null;
    float maxDistance = 100.0f; // SphereCast의 최대 거리 설정

    if (controllType == ControllType.A || controllType == ControllType.C)
    {
        RaycastHit[] hits = Physics.SphereCastAll(transform.position, targettingRadius, Vector3.up, maxDistance);
        float closestDistance = Mathf.Infinity;

        foreach (RaycastHit hit in hits)
        {
            if (hit.collider.CompareTag("Enemy") || hit.collider.CompareTag("Barrel") || hit.collider.CompareTag("Boss"))
            {

                float distance = Vector3.Distance(transform.position, hit.transform.position);
                Vector3 toTarget = hit.transform.position - transform.position;

                float angleToTarget = Vector3.Angle(transform.forward, toTarget);

                switch (controllType)
                {
                    case ControllType.A:
                        if (distance < closestDistance)
                        {
                            closestDistance = distance;
                            closestEnemy = hit.transform;
                        }
                        break;
                    case ControllType.C:
                        if (distance < closestDistance && angleToTarget < fanAngle / 2)
                        {
                            closestDistance = distance;
                            closestEnemy = hit.transform;
                            if (closestEnemy.gameObject.CompareTag("Boss"))
                            {
                                targetBoss = true;
                            }
                        }
                        break;
                }
            }
        }
    }
    else if (controllType == ControllType.D)
    {
        RaycastHit[] hits = Physics.SphereCastAll(mousePosition, targettingRadius, Vector3.up, maxDistance);
        float closestDistance = Mathf.Infinity;

        foreach (RaycastHit hit in hits)
        {
            if (hit.collider.CompareTag("Enemy") || hit.collider.CompareTag("Barrel"))
            {
                float distance = Vector3.Distance(mousePosition, hit.transform.position);
                Vector3 toTarget = hit.transform.position - transform.position;
                if (distance < closestDistance)
                {
                    closestDistance = distance;
                    closestEnemy = hit.transform;
                }
            }
        }
    }
    if (closestEnemy != null)
    {
        closestEnemy.gameObject.layer = 9;
        DashAndAttack(closestEnemy);
    }
}
```

- 총 4종류의 조작 시스템 구상

1. 마우스 커서를 없애고
   플레이어로부터 가장 가까운
   대상으로 이동하며 공격
2. 마우스 방향으로 고정된 거리만큼
   이동하며 공격
3. 마우스 방향으로 일정 각도 내에서
   가장 가까운 적에게 이동하며 공격
4. 마우스로 클릭한 위치에서 가장
   가까운 적에게 이동하며 공격

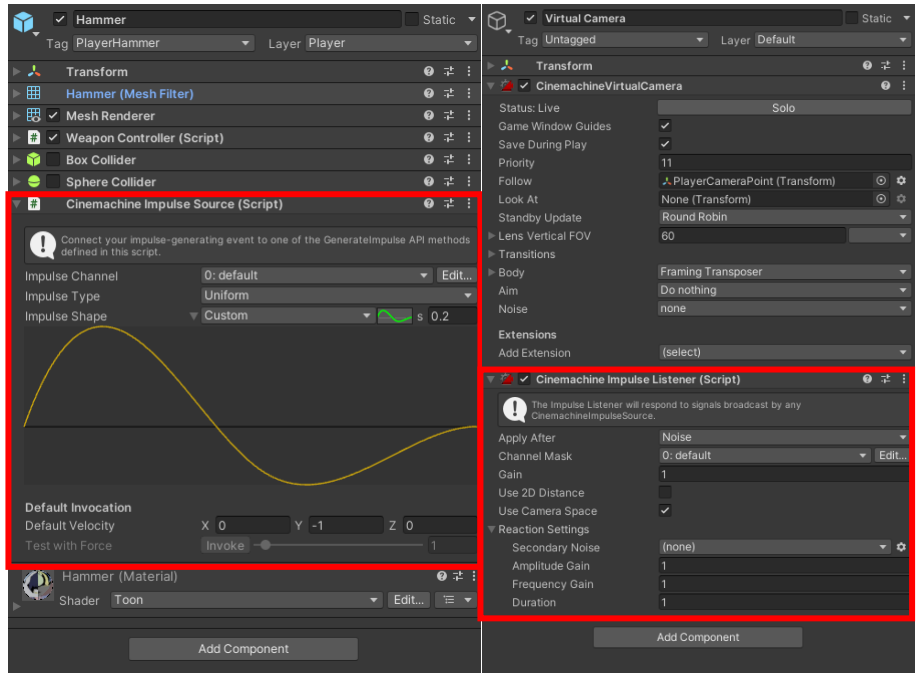
```
참조 1개
IEnumerator DashTowardsEnemy(Transform enemy)
{
    Debug.Log("대쉬 얍");
    targetMove = true;

    Vector3 directionToEnemy = (enemy.position - transform.position).normalized;

    Vector3 startPosition = transform.position;
    Vector3 endPosition = Vector3.zero;
    if (targetBoss)
    {
        endPosition = enemy.position - directionToEnemy * 5f;
        targetBoss = false;
    }
    else
    {
        endPosition = enemy.position - directionToEnemy * 1f;
    }
    float elapsedTime = 0;

    transform.LookAt(enemy.position);

    while (elapsedTime < dashTime[comboStep])
    {
        Debug.Log("이동합니다");

        rigid.MovePosition(Vector3.Lerp(startPosition, endPosition, (elapsedTime / dashTime[comboStep])));

        elapsedTime += Time.deltaTime;
        yield return new WaitForEndOfFrame();  //한 프레임 대기
    }
    yield return new WaitForSeconds(dashTime[comboStep]);
    targetMove = false;
}
```

<플레이어 대쉬 공격>

플레이어의 위치와 지정된 적의 위치를 기반으로
Vector3.Lerp를 사용해 보간된 값을 계산하고
이를 코루틴과 반복문을 활용해
프레임 단위로 플레이어를 적의 위치까지 부드럽게 이동시킵니다.

01-2. 대표 스크립트 (타격감 구현 2 – 카메라 진동)

타격감을 살리기 위해 타게팅, 카메라 진동 아이디어를 구현했습니다.
타게팅 시스템은 조작감 개선, 카메라 진동은 타격 연출을 보충하기 위해 기획했습니다.



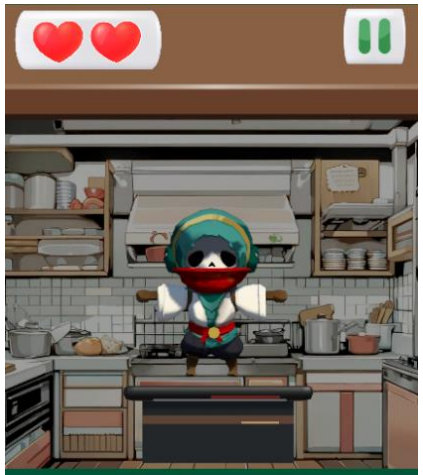
유니티의 Cinemachine Impulse 기능을 활용

```
Unity 메시지 | 참조 0개
private void OnTriggerEnter(Collider other)
{

    if (other.tag == "Barrel")
    {
        Barrel barrel = other.GetComponent<Barrel>();
        barrel.explode = true;
    }
    if (other.tag == "Core")
    {
        Core core = other.GetComponent<Core>();
        DamageCaculate();

        core.Damaged(damage, transform.position);
        Debug.Log("코어가 아픕니다. ");

    }
    if (other.tag == "Enemy")
    {
        CameraShake();

        Debug.Log("적과 망치가 닿음!");
    }
}

참조 1개
public void CameraShake()
{
    ImpulseSource.GenerateImpulse();
    Debug.Log("카메라가 흔들립니다");
}
```

충돌 시 카메라 진동 실행

02. Food in Poison

2024년 계원예술대학교 “제 1회 AI활용 게임 개발 경진대회” 출품작입니다.



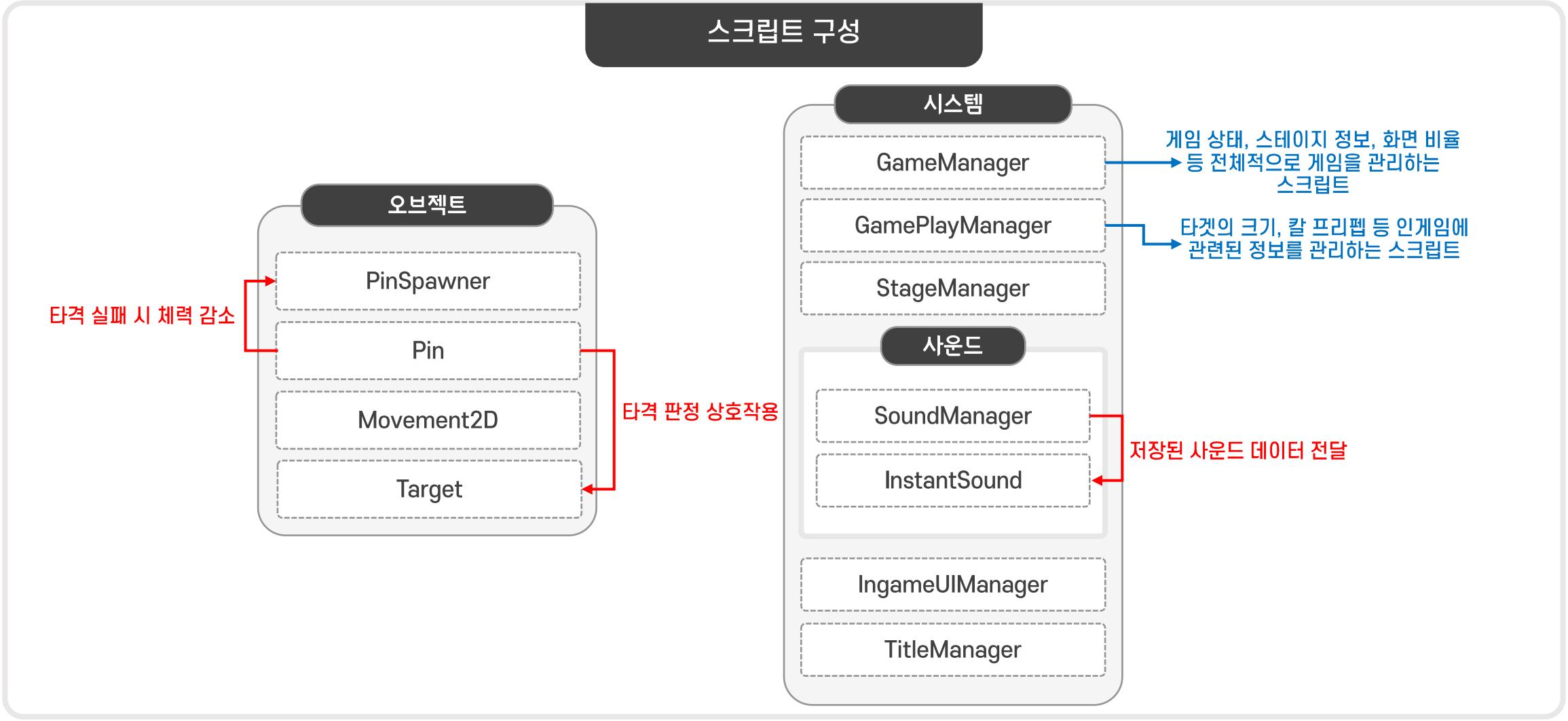
- 회전하는 복어에 칼을 꽂아넣는 2D 캐주얼 아케이드 게임입니다.
- 던진 칼이 이미 꽂힌 칼이나 가시와 충돌 시 플레이어의 체력이 감소합니다.
- 체력이 모두 닳기 전에 지정된 개수만큼 칼을 꽂는다면 클리어 됩니다.

- AI로 생성한 리소스를 활용하여 만든 게임입니다.
- 대회 목표인 AI의 활용에 초점을 맞춰 대량의 리소스를 간편하게 적용시킬 수 있도록 설계하였습니다.

	설명
개발 기간	2024.10.01~2024.10.31
개발 엔진	Unity
플랫폼	PC (추후 모바일로 변경 예정)
장르	2D 캐주얼 아케이드 게임

02-1. 스크립트 소개

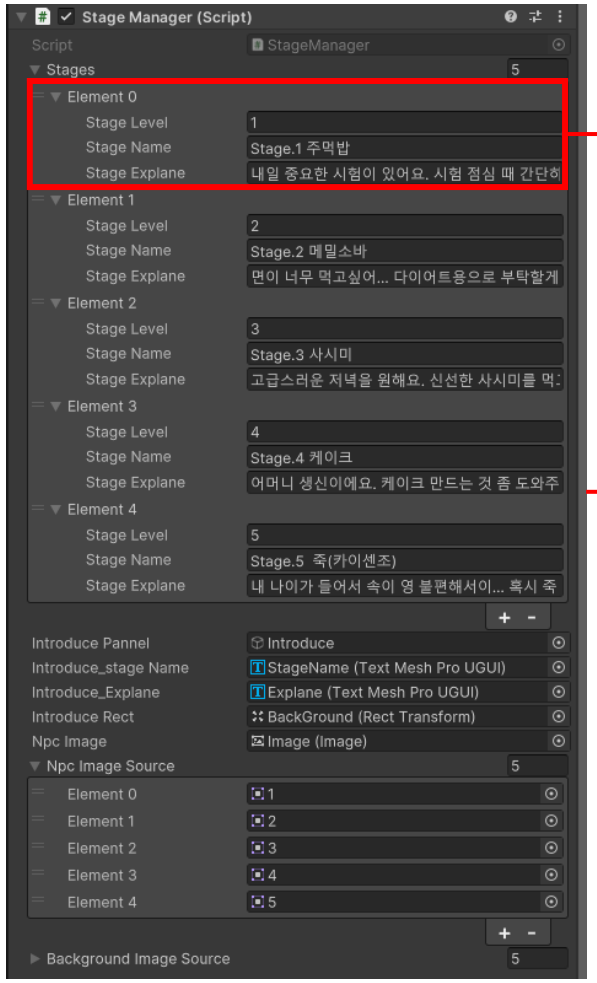
전체적인 스크립트 구성 및 상호작용



02-2. 대표 스크립트 (스테이지 설정 구조화)

스테이지 별 정보를 쉽게 추가하고 데이터를 입력할 수 있도록 구조를 만들었습니다.
이미지는 대량의 리소스를 한 번에 입력할 수 있도록 Stage 클래스에 추가하지 않았습니다.

유니티 Inspector 창



스크립트

```
[Serializable]
참조 4개
public class Stage
{
    public int stageLevel;
    public string stageName;
    public string stageExplane;
}
```

Stage 내부의 정보를 캡슐화하고
List와 SerializeField를 활용하여 유니티
Inspector 창에서도
간편하게 관리할 수 있도록 설계

```
[SerializeField] private List<Stage> stages = new List<Stage>(); // 여러 스테이지를 리스트로 관리

[SerializeField] GameObject introducePannel;
[SerializeField] TextMeshProUGUI introduce_stageName;
[SerializeField] TextMeshProUGUI introduce_Explane;
Image introduceImage;
Image introduce_background;
[SerializeField] RectTransform introduceRect;

[SerializeField] Image npcImage;
[SerializeField] Sprite[] npcImageSource;
[SerializeField] Sprite[] backgroundImageSource;
```

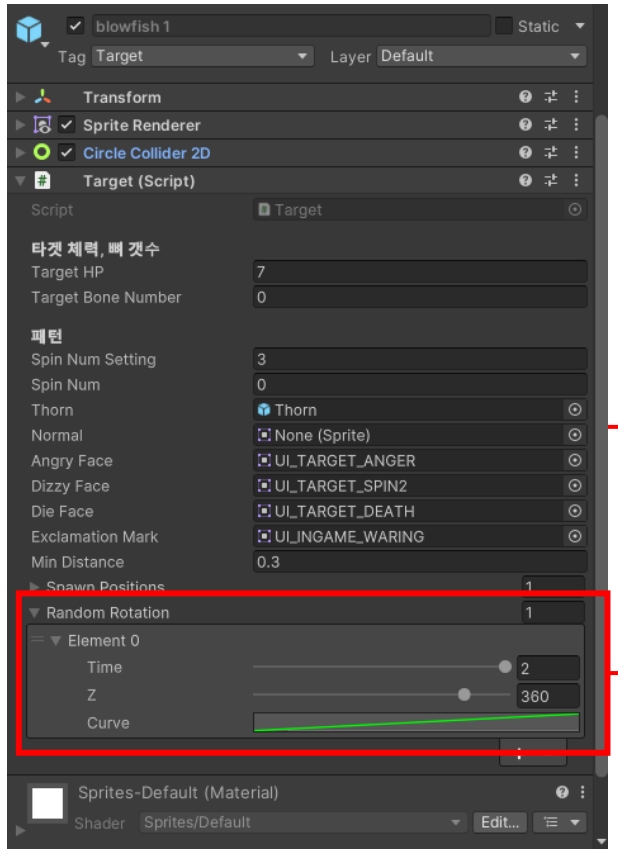
```
public void StageIntroduceOpen(int stageIndex)
{

    GameManager manager = GameObject.FindWithTag("GameManager").GetComponent<GameManager>();
    // 스테이지 이름으로 스테이지 데이터를 찾음
    Stage selectedStage = stages.Find(stage => stage.stageLevel == stageIndex);

    manager.StageInfo(selectedStage);

    introducePannel.SetActive(true);

    introduceRect.DOMoveY(463, 1f);
    introduceImage.DOFade(0.76f, 1f);
    introduce_Explane.text = selectedStage.stageExplane;
    introduce_stageName.text = selectedStage.stageName;
    npcImage.sprite = npcImageSource[selectedStage.stageLevel - 1];
    introduce_background.sprite = backgroundImageSource[selectedStage.stageLevel - 1];
}
```

<리소스 적용 스크립트>

02-2. 대표 스크립트 (적 스테이터스, 움직임 설정)

적 오브젝트의 스테이터스와 패턴을 쉽게 수정하고 테스트할 수 있도록 프리팹화하여 구조를 설계했습니다.

유니티 Inspector 창



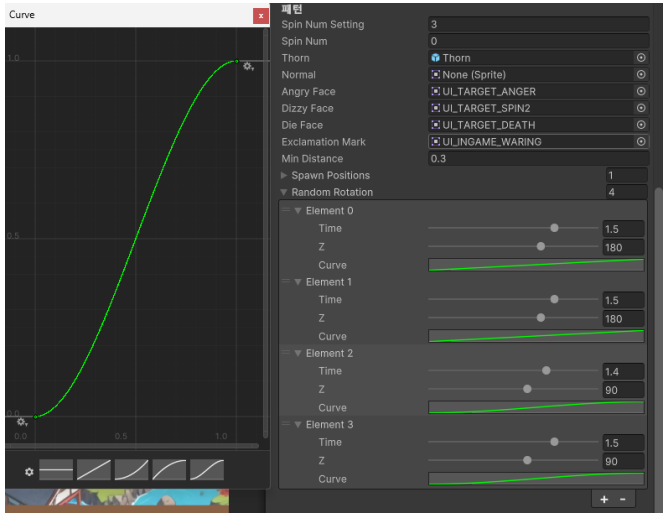
스크립트

```
[Header("타겟 체력, 뼈 갯수")]
public int targetHP;
[SerializeField] private int targetBoneNumber;

[Header("패턴")]
[SerializeField] private int spinNumSetting = 3;
[SerializeField] private int spinNum;

[SerializeField] GameObject thorn;

[SerializeField] Sprite normal;
[SerializeField] Sprite angryFace;
[SerializeField] Sprite dizzyFace;
[SerializeField] Sprite dieFace;
//[SerializeField] GameObject sprit;
[SerializeField] Sprite exclamationMark;

[SerializeField] float minDistance = 0.3f;

private float radius = 0.8f;

[SerializeField] private List<Vector2> spawnPositions = new List<Vector2>(); // 저장된 좌표 리스트
public List<RotationVariation> RandomRotation = new List<RotationVariation>();  //패턴 리스트
```

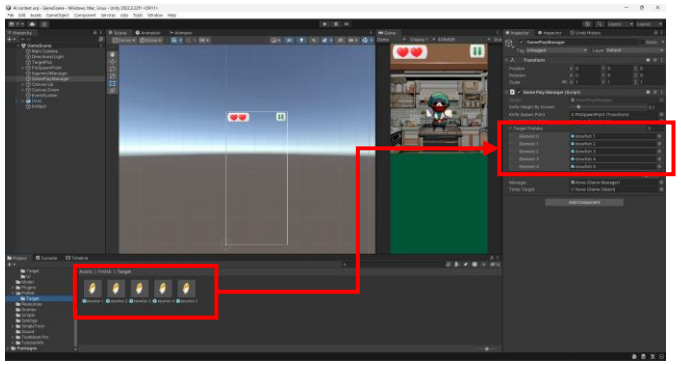
```
[System.Serializable]
참조 2개
public class RotationVariation
{
    [Range(0f, 2f)]
    public float time = 0f;
    [Range(-540, 540)]
    public float z = 0f;
    public AnimationCurve curve;
}
```

몬스터의 움직임을 스테이지와 동일하게 구조화하여 움직이는 시간과 각도, 속도 등을 조절하고 추가할 수 있다

```
void Moving()
{

    currentRoationIndex = (currentRoationIndex + 1) % RandomRotation.Count;

    spinNum--;
    if (targetBoneNumber > 0 && spinNum <= 0)
    {
        Invoke("ActivateThornPattern", 1f);
        spriteRenderer.sprite = angryFace;
        spinNum = spinNumSetting;
    }

    transform.DORotate(new Vector3(0f, 0f, transform.localRotation.eulerAngles.z + RandomRotation[currentRoationIndex].z),
        RandomRotation[currentRoationIndex].time, RotateMode.FastBeyond360)
        .SetEase(RandomRotation[0].curve)
        .OnComplete(() => Moving());
}
```



(다양한 움직임 패턴을 추가 및 수정할 수 있다)



(적 오브젝트의 프리팹화 및 적용)

Thank you!

Contact

Email: timet7429@gmail.com
Phone: 010-8811-1756