2023년 뉴콘텐츠아카데미 1기 프로젝트 교육
프로젝트 세부 기획안

■ **프로젝트명:** Dugi XR (두기 XR)

기획의도	 Dugi XR 프로젝트는 **현대 기술과 한국의 전통을 융합하여 새로운 형태의 엔터테인먼트 경험을 창출**하고자 합니다. 메타 퀘스트 3의 **공간 컴퓨팅, 패스스루, 핸드트래킹** 기술을 활용하여 한국 전통 놀이를 현대적으로 재해석하고 가상 현실에서 새롭게 경험할 수 있게 함으로써, 전 세계 사용자들에게 한국 문화의 매력을 전달하고자 합니다. 이 프로젝트는 전통 놀이의 본질을 유지하면서도 현대적인 게임 메커니즘과 몰입감 있는 가상 현실 경험을 결합하여, 다양한 세대의 사용자들이 쉽게 즐길 수 있는 게임 플랫폼을 제공하는 것을 목표로 합니다. 이를 통해, Dugi XR은 전통과 현대 기술의 균형을 이루며, 한국 문화의 새로운 매력을 전 세계에 소개할 수 있습니다.
세부 추진내용	 이 프로젝트는 전통놀이를 XR(Extended Reality) 기술을 통해 현대적으로 재해석하고 변형하는데 주력하고 있습니다. 특히, 공간 컴퓨팅과 패스쓰루 기술을 적극적으로 활용하여, 전통놀이가 가지고 있는 시간과 공간에 따른 제약을 극복하고 물리적 제한을 넘어선 새로운 경험을 사용자들에게 제공하고자 합니다. 이 기술적 접근은 사용자들에게 현실과 가상 세계 사이의 경계를 넘나드는 몰입감 있는 경험을 제공함으로써, 전통과 현대 기술이 융합된 새로운 차원의 경험을 가능하게 할 것입니다. 이런 방식으로, 우리 프로젝트는 사용자들에게 전통놀이를 현대적이고 혁신적으로 다시 경험하게 하며, XR 기술을 통해 새로운 현실과 가상의 경계를 효과적으로 탐구할 수 있는 기회를 제공합니다.

1. 투호놀이

- **선정 배경:** 투호놀이는 전통적인 게임이지만, 현대적인 기술과의 결합을 통해 더욱 역동적이고 상호작용적인 게임으로 변모시킬 수 있는 잠재력이 크다고 판단되었다.
- **개요:** 가상의 괴물들과의 대결을 통한 투호놀이는 XR 기술을 이용해 사용자가 실제 공간에서 가상의 요소와 상호작용하며 새로운 형태의 게임 경험을 할 수 있도록 설계되었다. 이는 기존 투호놀이의 경쟁적인 요소를 강화하며, 동시에 현대적인 시각적, 청각적 요소를 추가한다.

2. 제기차기

- **선정 배경:** 제기차기는 그 단순한 플레이 방식 속에 숨겨진 다양한 가능성을 가진 놀이로, XR 기술을 통해 이를 더욱 확장시킬 수 있다고 판단되었다.
- **개요:** XR 환경에서의 제기차기는 전통적인 플레이 방식을 유지하면서도, 가상의 별자리 생성과 같은 시각적 요소를 추가함으로써 사용자에게 새로운 감각적 경험을 제공한다. 이는 전통적인 제기차기에 현대적인 감각을 더하고, 더욱 인터랙티브한 경험을 가능하게 한다.

3. 쥐불놀이 & 달집태우기

- **선정 배경:** 쥐불놀이와 달집태우기는 전통적인 불놀이이지만, 실제로 수행하기에는 안전상의 문제가 있어, XR을 통해 이를 안전하고 현대적으로 재현할 수 있다는 점에서 선정되었다.
- **개요**: XR 환경에서 구현된 쥐불놀이와 달집태우기는 가상의 쥐불과 달집을 사용하여 전통적인 불놀이를 현대적이고 안전한 방식으로 재현한다. 이를 통해 사용자는 전통적인 불놀이를 실제와 같은 경험으로 체험할 수 있다.

4. 낙화놀이

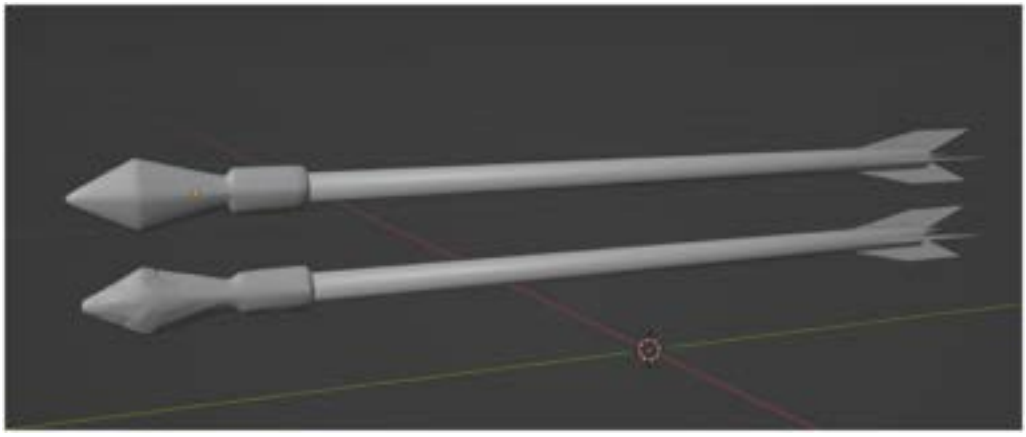
- **선정 배경:** 낙화놀이는 특별한 축제나 행사에서만 경험할 수 있는 전통놀이로, XR 기술을 통해 개인적으로도 이를 체험할 수 있도록 하고자 하는 목적에서 선정되었다.
- **개요:** XR을 통해 구현된 낙화놀이는 사용자가 소원을 적은 종이를 풍등에 올리고 가상의 낙화를 경험할 수 있게 한다. 이는 사용자에게 실제와 같은 시각적 충격과 감동을 제공한다.

2. 콘텐츠 창작내용

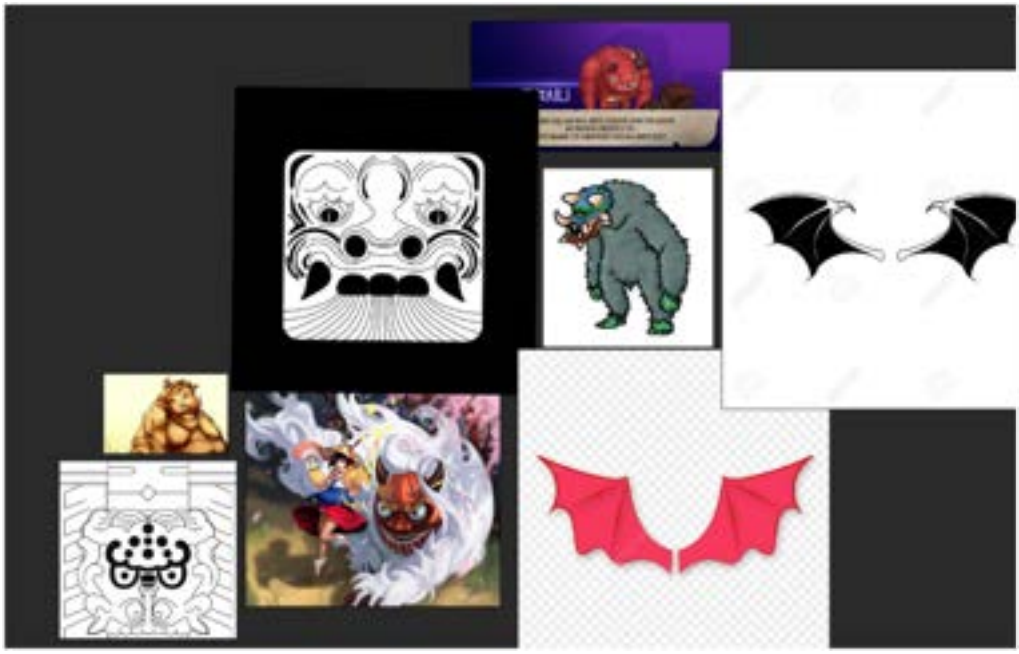
1. 컨셉 디자인 및 아트 리소스 제작

리얼리스틱한 풍보단 캐쥬얼한 스타일라이즈 풍으로 제작하고자 함.

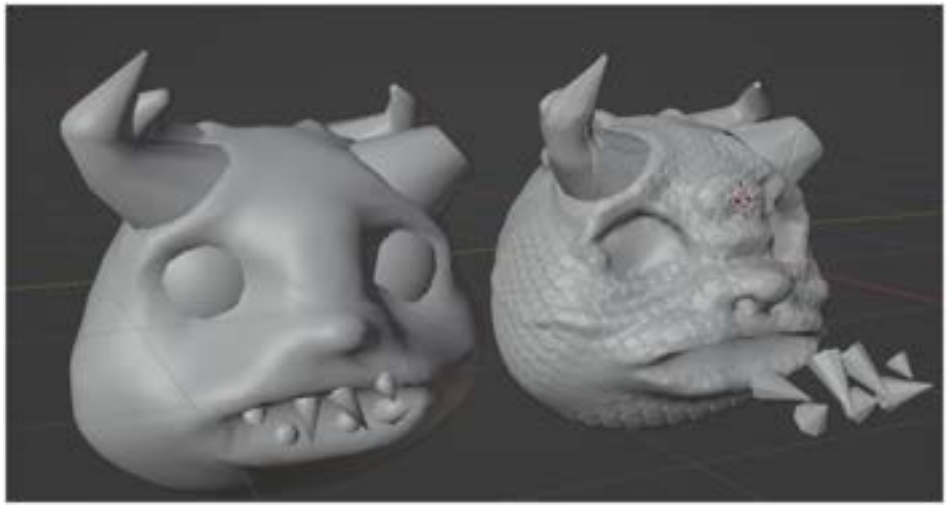
로우폴리 작업 후 하이폴리 스컵팅 작업이후 서브스탠스 페인트 이용한 베이킹 및 텍스쳐 처리



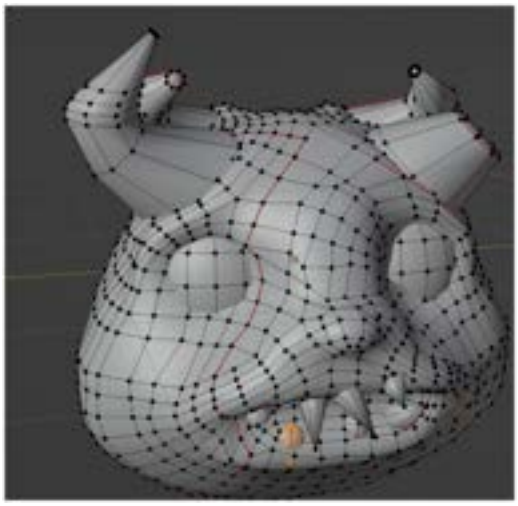
괴물 제작



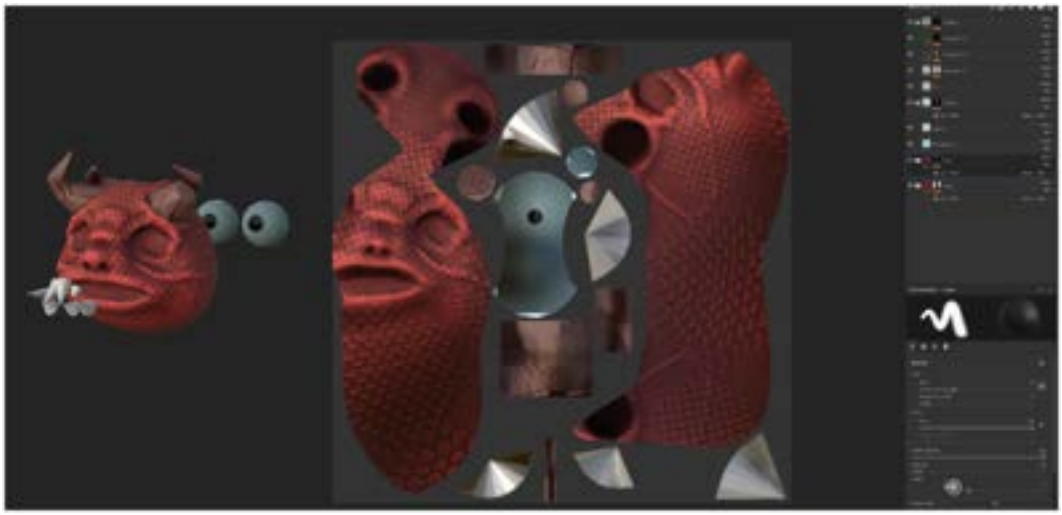
괴물 레퍼런스

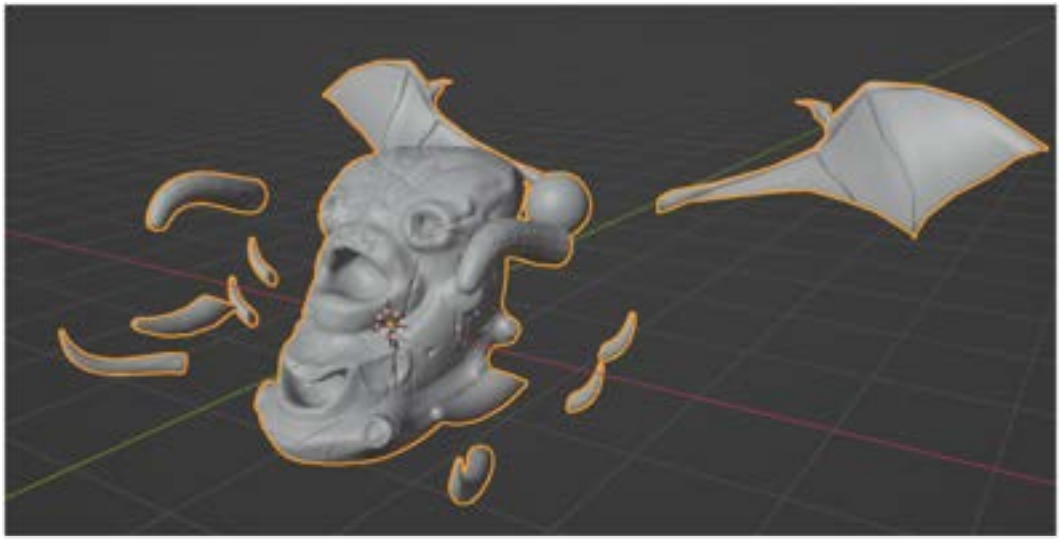


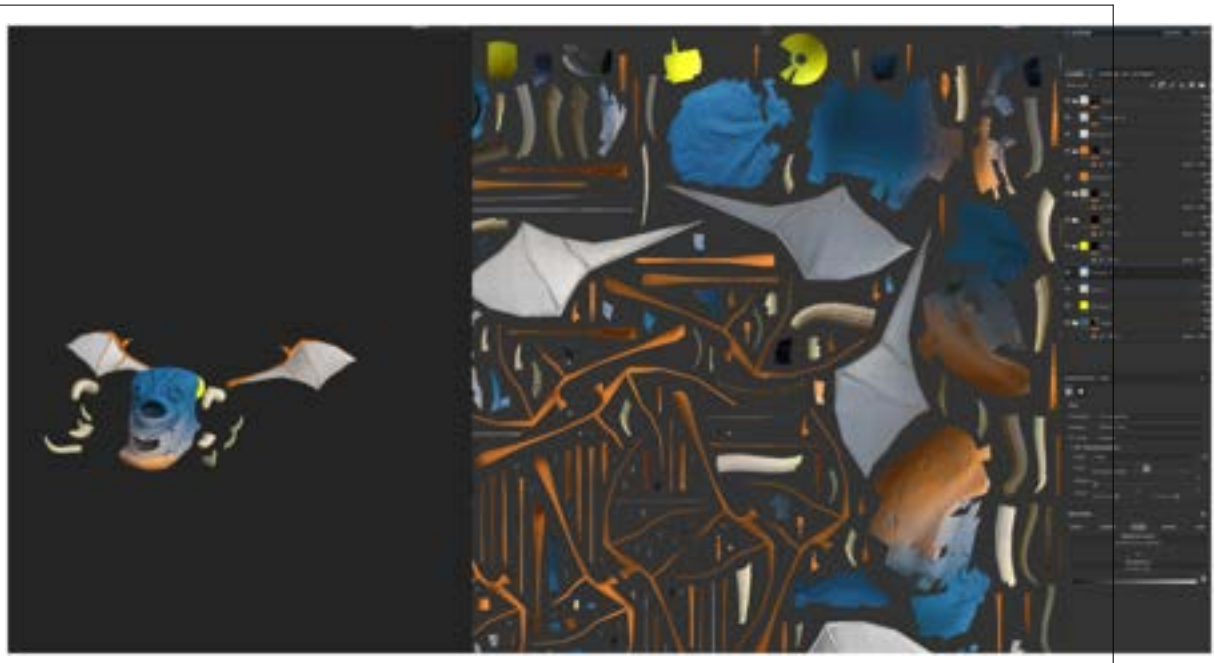
블렌더를 이용해 하이메쉬 스컵팅하여로우 메쉬로 리토폴로지 작업



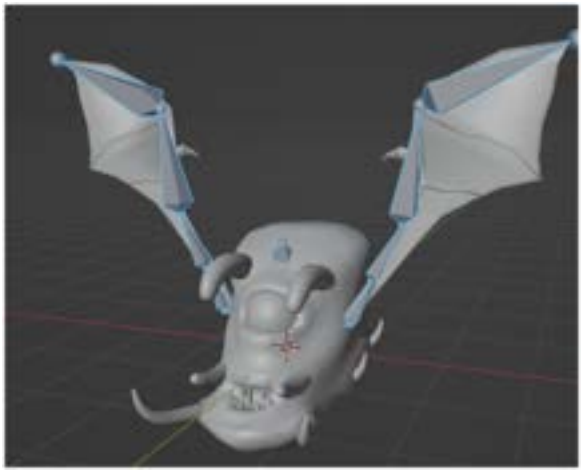
서브 스턴스 페인터를 이용한 텍스쳐 작업

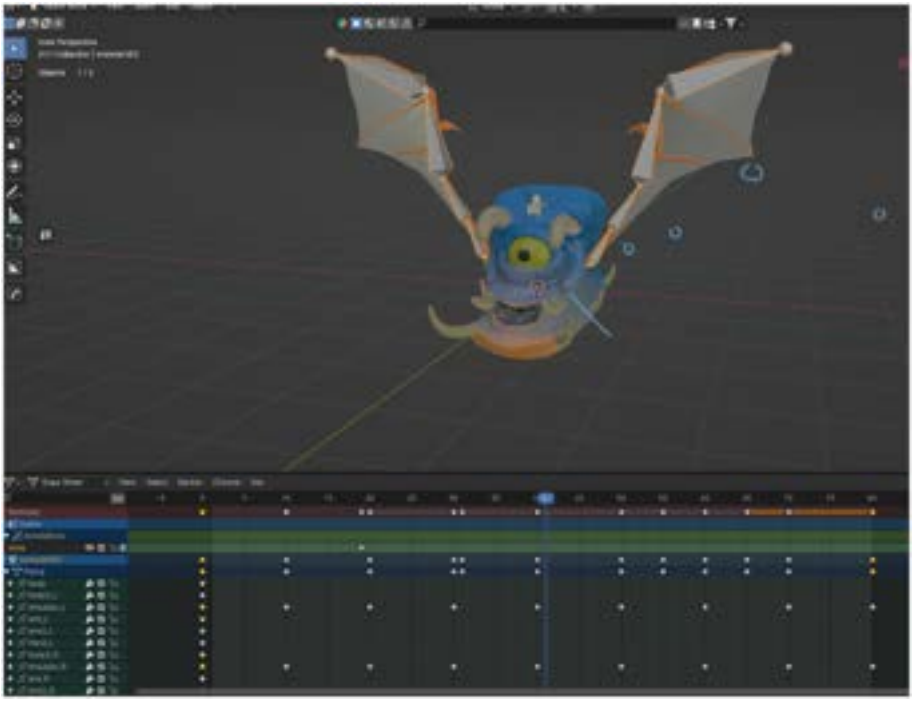




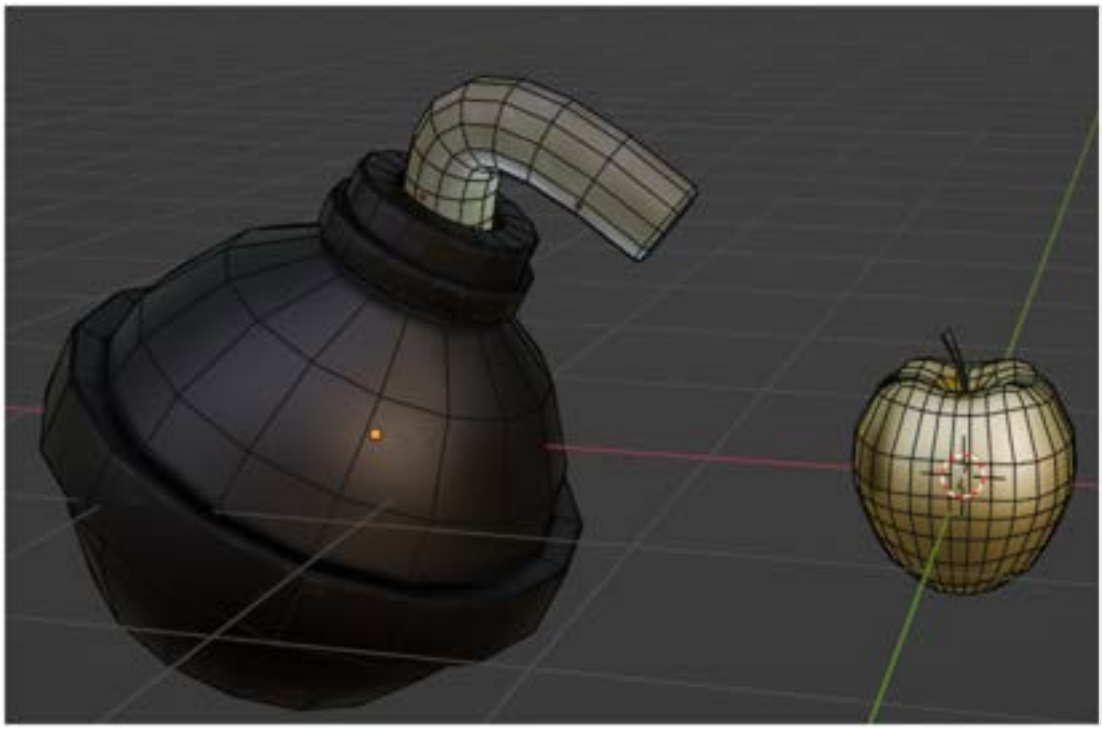


생명체임을 강조하고 공중 몬스터임을 어필하기 위한 애니메이션 제작과 리깅





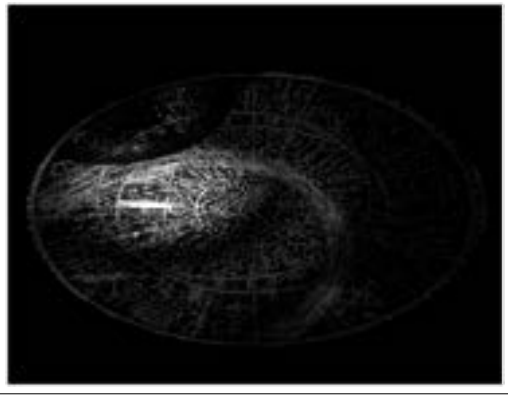
기타 리소스 요청 작업물

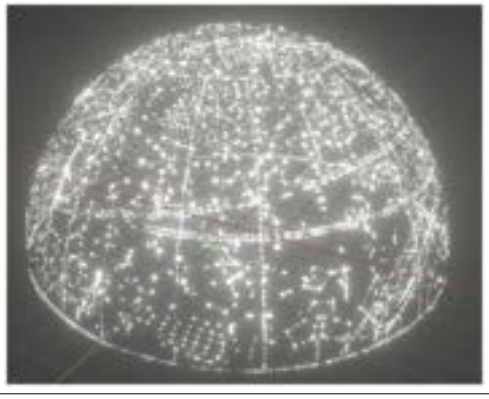


제기차기 리소스 제작

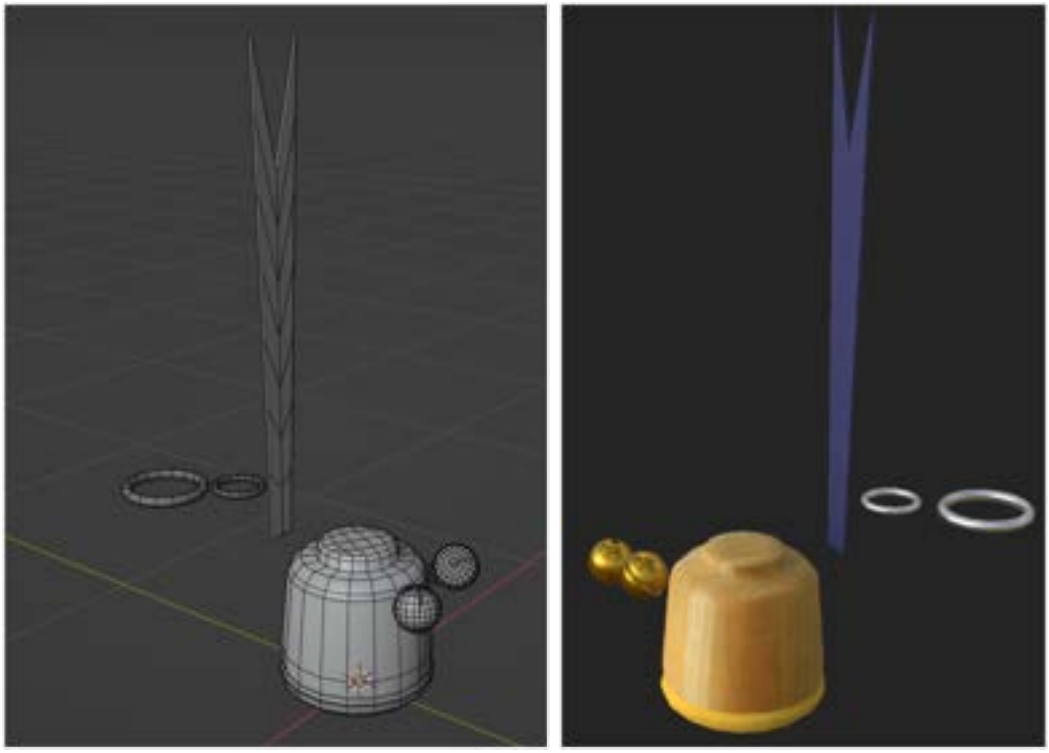
블렌더 쉐이딩 노드를 통한 천상열차 분야지도 리소스 제작 및 베이킹

쉐이더를 오브젝트에 베이킹하여 제작된 알파 및 이미션 맵과 적용 모습



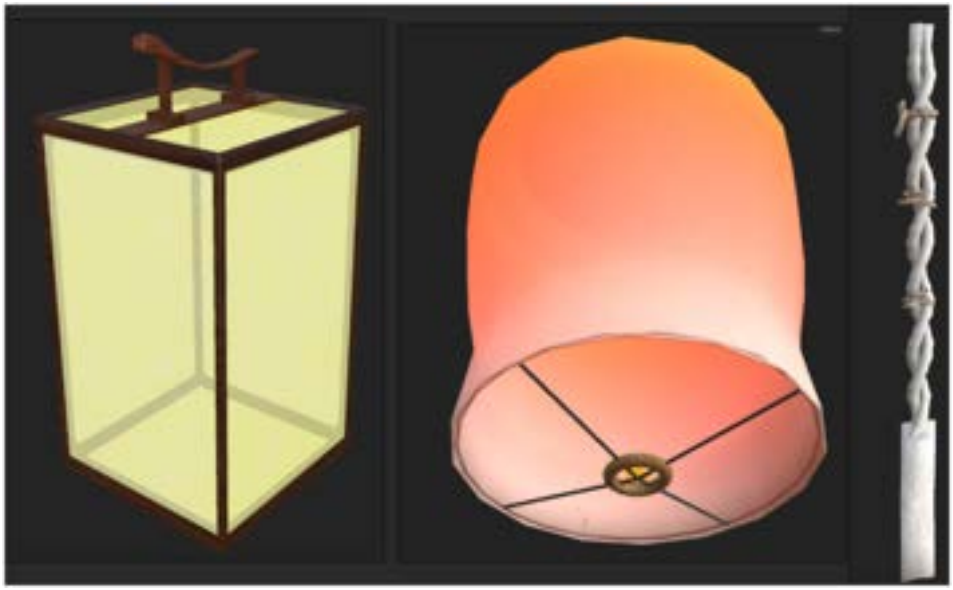


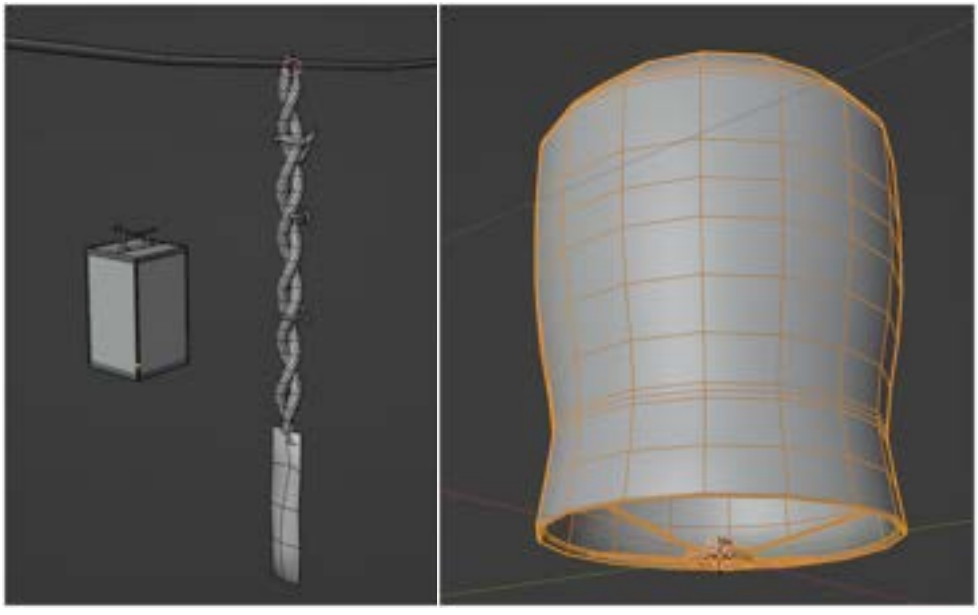
제기 레퍼런스



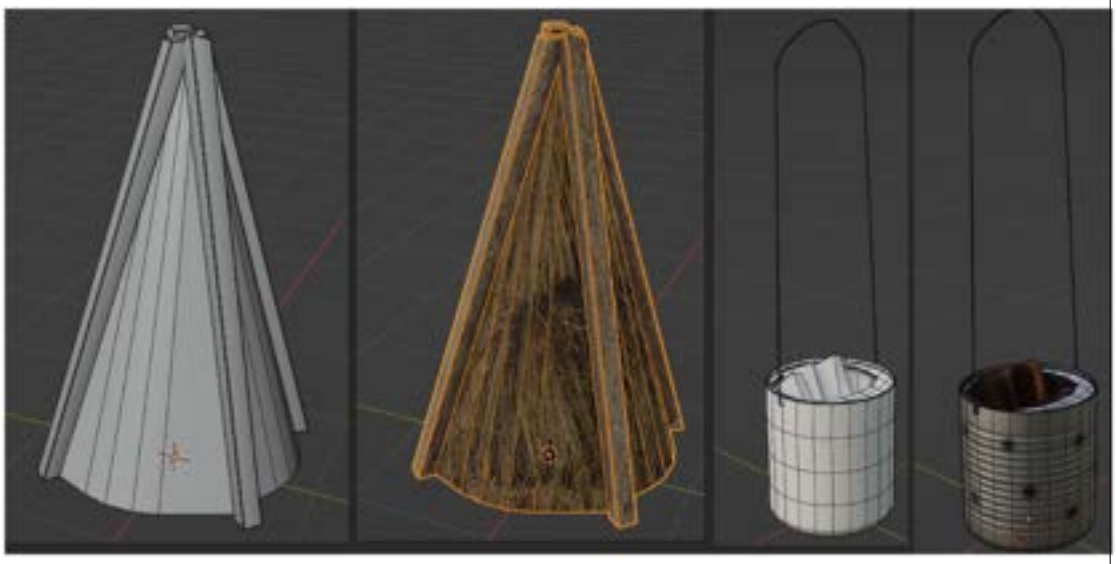
블렌더 모델링 및 서브스턴스 페인터 텍스쳐링

낙화놀이 리소스 제작 레퍼런스





모델링 및 텍스쳐링



2. 게임 별 주요 기능 개발 세부 내용

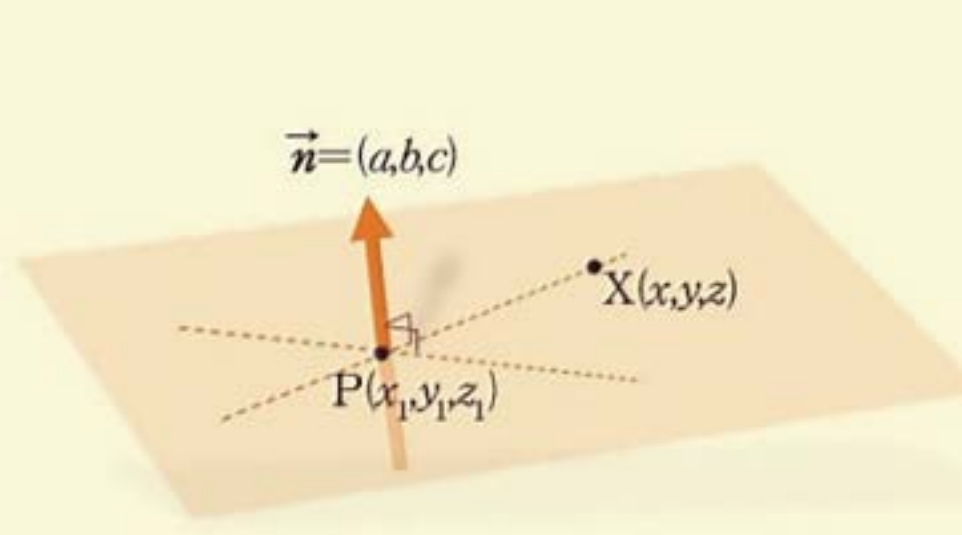
1. 제기 게임

- 제기 모델의 애니메이션 구현

제기 오브젝트를 던지고 받는 과정에서 오브젝트가 사실적으로 보일 수 있도록 하기 위한 애니메이션 제작

- 제기에 작용되는 물리력 구현

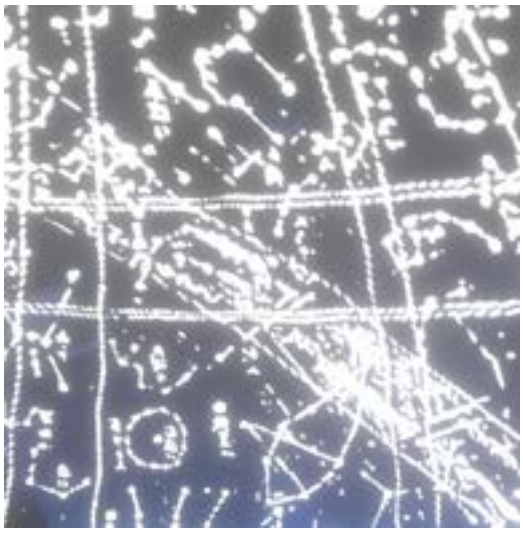
손을 활용하여 제기를 치는 과정에서 손의 방향에 따라 제기에 가해지는 힘을 조절할 수 있도록 설정 (Prototype에서는 간단하게 확인하기 위해서 점수 초기화 정지 및 제기 오브젝트 y축으로만 이동하도록 설정)

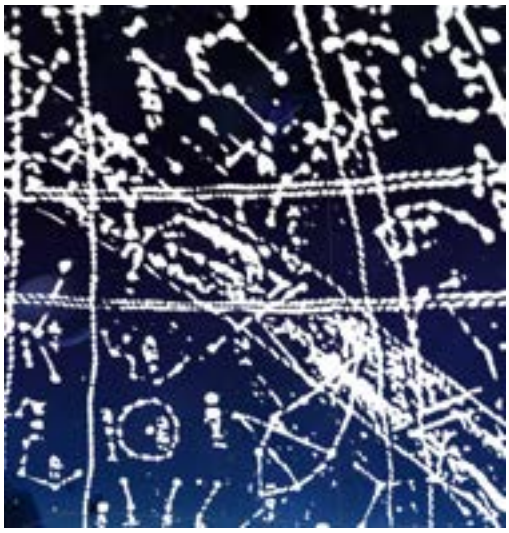



```
    if (segint.isStart)
    {
        isAttached = true;
        if (collision.transform.tag == "Hand")
        {
            segint.ShowStars();
            Vector3 groundNormal = collision.ClosestPoint(transform.position) - transform.position;
            Vector3 bounceDirection = Vector3.Reflect(Vector3.up, groundNormal).normalized;

            // 실제 튕길 방향
            //Vector3 finalBounceDirection = new Vector3(-bounceDirection.x, bounceDirection.y, -bounceDirection.z);
            // 테스트용
            Vector3 finalBounceDirection = new Vector3(0, bounceDirection.y, 0);

            GetComponent<Rigidbody>().AddForce(finalBounceDirection * forceMagnitude, ForceMode.Impulse);
            StartCoroutine(EffectAndSound());
        }
        if (collision.transform.tag == "Destroy")
        {
            GameSet();
        }
    }
}

// Unity 메시지 참조 0개 | Changed by kirito070604@gmail.com on 2023년 12월 4일 월요일
public void OnTriggerExit(Collider collision)
{
    isAttached = false;
}

참조 2개 | Changed by kirito070604@gmail.com on 2023년 12월 4일 월요일
public void GameSet()
{
    isAttached = false;
    segint.isStart = false;
    segint.ResetStars();
    rb.constraints = RigidbodyConstraints.FreezePosition;
    forceMagnitude = 10f;
    Instantiate(selfPrefabs, new Vector3(-0.4f, 0.3f, -0.15f), Quaternion.identity);
    Destroy(this.gameObject);
}

참조 0개 | Added by kirito070604@gmail.com on 2023년 12월 4일 월요일
public void isGrab(bool check)
{
    isAttached = check;
}

참조 1개 | Added by kirito070604@gmail.com on 2023년 12월 4일 월요일
IEnumerator EffectAndSound()
{
    triggerAudio.Play();
    yield return new WaitForSeconds(0.5f);
```

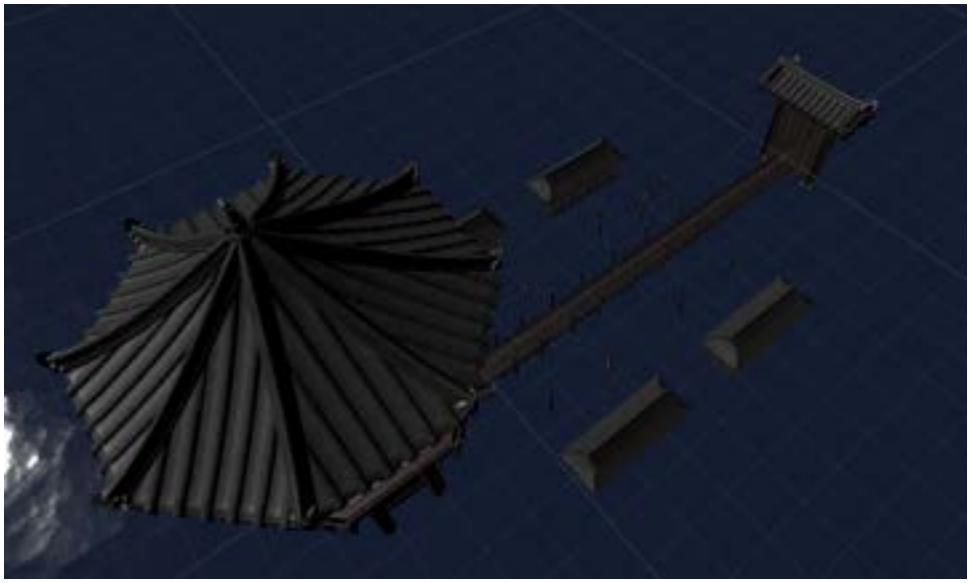
- Post Processing을 활용한 광원 및 Bloom 구현

유니티 내부의 Post Processing을 활용하여 Material을 광원 느낌이 나도록 처리



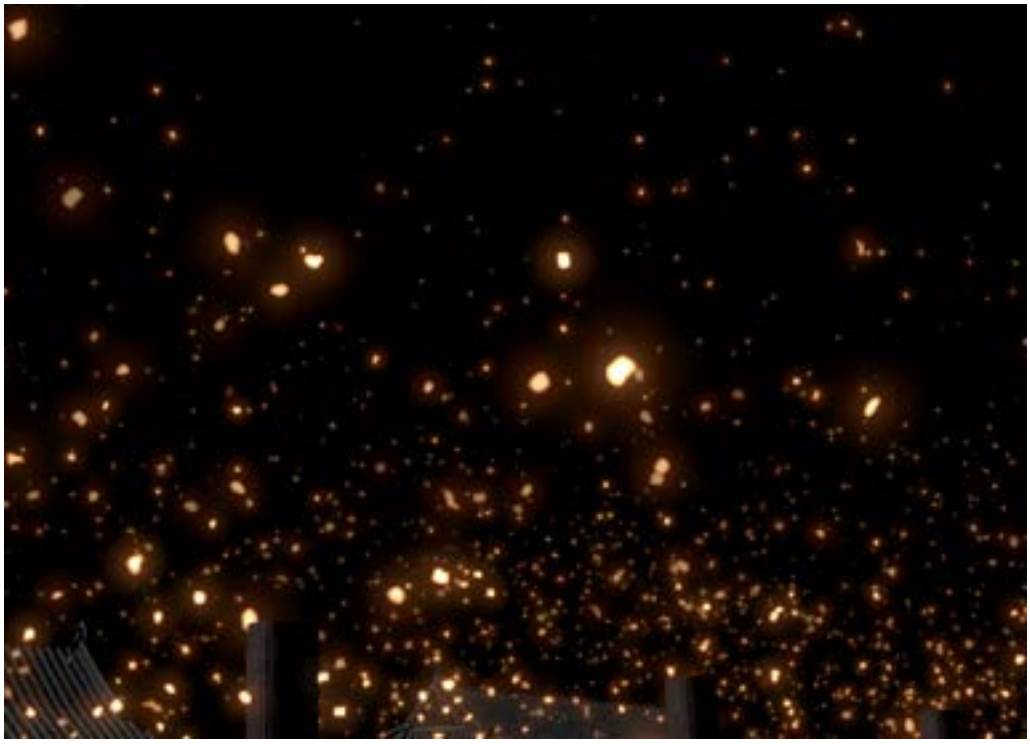


- 점수 계산, 플레이 공간 설정 등 게임 구현



ii. 낙화놀이

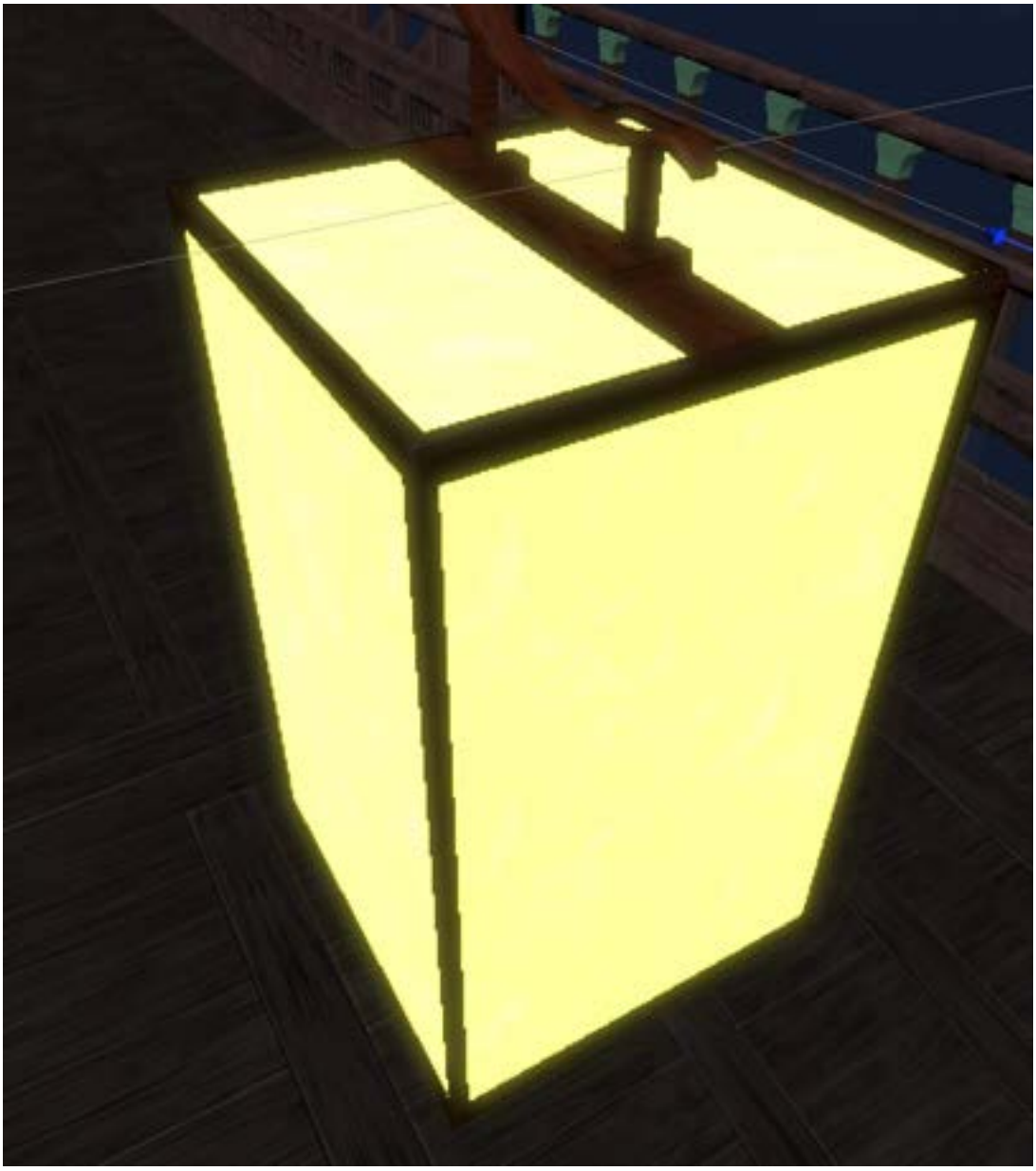
• 파티클 시스템을 활용한 불꽃 표현파티클 시스템을 통해서 불꽃 모양의 파티클이 생성되도록 설정 및 바람을 활용해 실제 낙화놀이와 최대한 비슷한 효과 구현



- 이동 기능 구현

등을 활용하여 플레이어가 이동할 수 있게 하여 원하는 위치서 관람할 수 있도록 설정

```
※Unity 스크립트(자산 참조 1개)|참조 0개|Added by kirito970904@gmail.com on 2023년 12월 4일 월요일
public class HandLightManager : MonoBehaviour
{
    public bool isWalking;
    public Transform Player;

    private Rigidbody rb;

    ※Unity 메시지|참조 0개|Added by kirito970904@gmail.com on 2023년 12월 4일 월요일
    void Start()
    {
        rb = GetComponent<Rigidbody>();
    }

    ※Unity 메시지|참조 0개|Added by kirito970904@gmail.com on 2023년 12월 4일 월요일
    void Update()
    {
        if (isWalking)
        {
            Player.position += new Vector3(0, 0, 1) * Time.deltaTime;
        }
    }

    참조 0개|Added by kirito970904@gmail.com on 2023년 12월 4일 월요일
    public void MoveForword()
    {
        isWalking = true;
    }
    참조 0개|Added by kirito970904@gmail.com on 2023년 12월 4일 월요일
    public void StopMoving()
    {
        isWalking = false;
        rb.rotation = Quaternion.Euler(-90, 0, 0);
    }
}
```



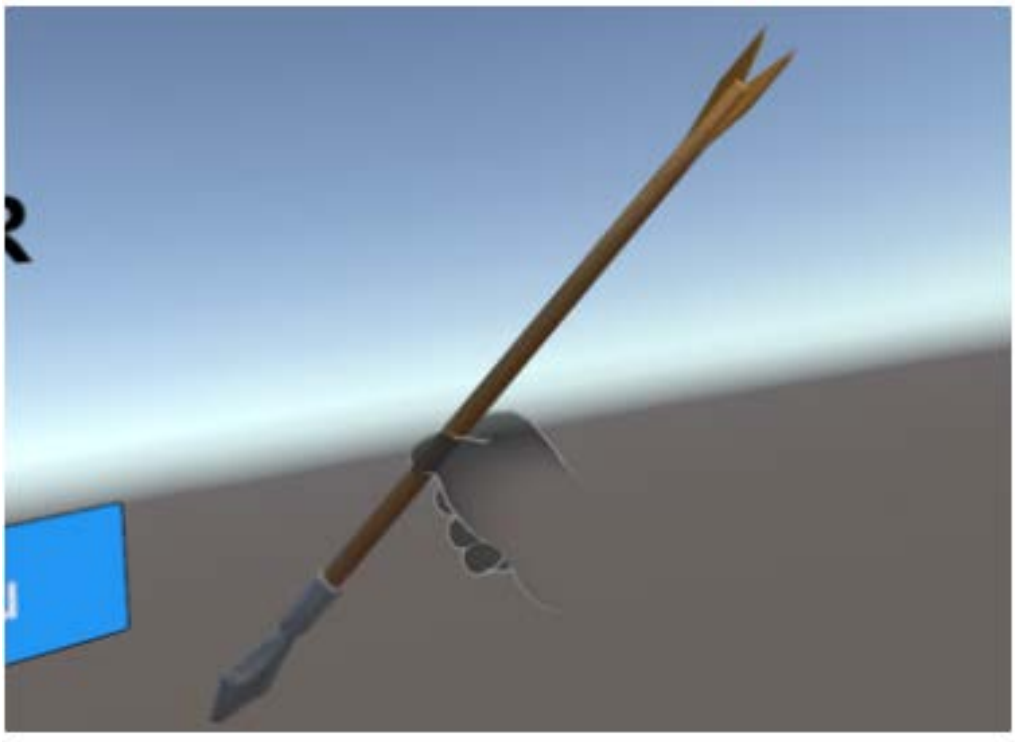
iii. 로딩 화면

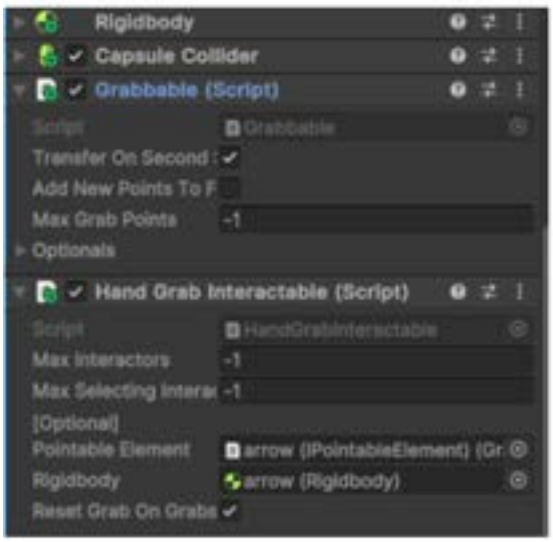
- 로딩 딜레이 기능 구현
파티클이나 Post Process등 여러가지 효과에 의해 딜레이가 걸리는 문제를 해결하기 위해 딜레이를 걸기위한 로딩 화면 제작

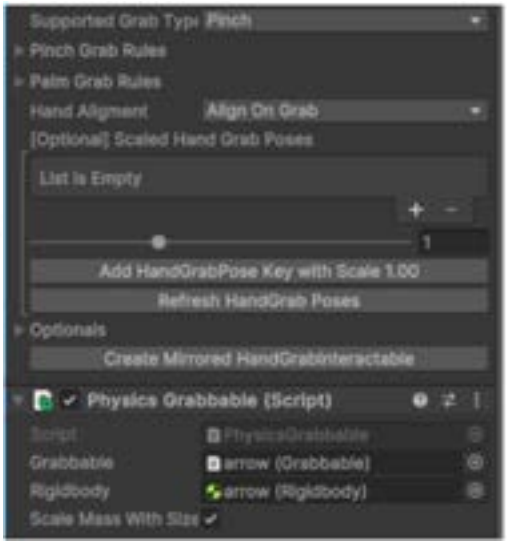
테스트, 시연영상

https://youtu.be/DoZBhQhffvk

3. 투호 게임
- 핸드트래킹을 이용해 투호 집기
투호를 집기 위한 Oculus Integration에서 제공하는 컴포넌트 이용
투호를 집고 던지는 작용을 위한 Physics Grabbable 추가







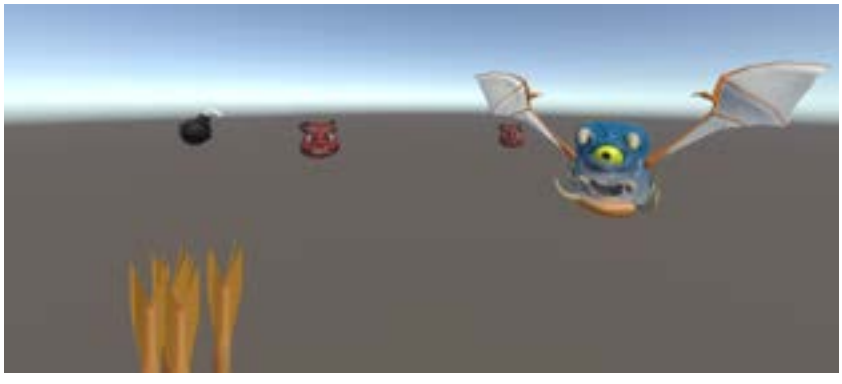
- 몬스터 생성 (SpawnMonster.cs)

카메라가 바라보는 방향에서 어느정도 떨어진 거리에서 랜덤으로 몬스터들이 생성
각각 다른 위치에 몬스터들이 생성되므로 각 위치에서 카메라를 바라보도록 회전

```
void Start()
{
    audioSource = gameObject.AddComponent<AudioSource>();

    // 초기 위치를 -0.1f에서 0.1f 사이의 값으로 설정
    float randomYOffset = Random.Range(-0.1f, 0.1f);
    initialPosition = new Vector3(transform.position.x, randomYOffset, transform.position.z);
    startTime = Time.time;

    mainCamera = Camera.main.transform;
}

void Update()
{

    if (mainCamera != null)
    {
        float yOffset = amplitude * Mathf.Sin(Time.time - startTime * speed);
        float step = speed * Time.deltaTime;

        // 카메라 쪽으로 이동하면서 y 방향으로 피그닌 움직임 추가
        transform.position = Vector3.MoveTowards(transform.position, mainCamera.position, step);
        transform.position = new Vector3(transform.position.x, initialPosition.y + yOffset, transform.position.z);
    }

    // 카메라로부터 1 이내의 거리에 들어오면 파괴
    if (Vector3.Distance(transform.position, mainCamera.position) <= 0.3f)
    {
        if (HurtSound != null)
        {
            AudioSource.PlayClipAtPoint(HurtSound, transform.position);
        }
        DestroyMonsterAndDecreaseLives();
    }
}
```

- 몬스터 이동 및 기능 구현 (Monster.cs)

몬스터는 위 아래로 조금씩 움직이면서 카메라 방향으로 점점 가까이 다가옴
카메라로부터 어느 정도 거리에 들어오면 몬스터 파괴



몬스터가 죽어서 파괴될 때는 몬스터 폭발 이펙트, 플레이어 아파하는 소리, 목숨 감소

몬스터가 플레이어 피격시 플레이어 목숨 감소 함수 호출

```
IEnumerator ShowExplosionAndDestroy()
{

    // 폭발 효과 생성
    GameObject explosion = Instantiate(explosionPrefab, transform.position, Quaternion.identity);

    //폭발 효과를 1초 동안 보이게
    yield return new WaitForSeconds(1f);

    // 폭발 효과 제거
    Destroy(explosion);

    if (explosionSound != null)
    {
        AudioSource.PlayClipAtPoint(explosionSound, transform.position);
    }

    // 몬스터 제거
    Destroy(gameObject);

    // 점수 증가
    ScoreManager scoreManager = FindObjectOfType<ScoreManager>();
    if (scoreManager != null)
    {
        scoreManager.IncreaseScore(scoreValue);
    }
}

void DestroyMonsterAndDecreaseLives()
{
    Destroy(gameObject);

    // 플레이어의 목숨 감소
    Player player = FindObjectOfType<Player>();
    if (player != null)
    {
        player.DecreaseLives();
    }
}
```

플레이어 피격 구현 (Player.cs)

피격 시 빨간 화면 깜빡임

```
IEnumerator FlashScreen()
{
    if (screenFlashImage != null)
    {
        // 플래시 색상 설정
        screenFlashImage.color = flashColor;

        // 지정된 기간 동안 대기
        yield return new WaitForSeconds(flashDuration);

        // 색상을 투명하게 재설정
        screenFlashImage.color = Color.clear;
    }
}
```

- 다양한 아이템 추가 (Apple.cs, Bomb.cs)

  사과를 맞출 시 목숨 증가

  폭탄을 맞출 시 생성되어있는 모든 몬스터 파괴 후 점수 추가

```
void Start()
{
    float randomYOffset = Random.Range(-0.1f, 0.1f);
    initialPosition = new Vector3(transform.position.x, randomYOffset, transform.position.z);

    mainCamera = Camera.main.transform;
}

void Update()
{
    if (mainCamera != null)
    {
        float step = speed * Time.deltaTime;
        transform.position = Vector3.MoveTowards(transform.position, mainCamera.position, step);
    }

    // 카메라 가까워지면 파괴
    if (Vector3.Distance(transform.position, mainCamera.position) <= 0.3f)
    {
        Destroy(gameObject);
    }
}

void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.CompareTag("Arrow"))
    {
        if (popSound != null)
        {
            AudioSource.PlayClipAtPoint(popSound, transform.position);
        }

        Destroy(gameObject);

        Player player = FindObjectOfType<Player>();
        if (player != null)
        {
            player.IncreaseLives();
        }
    }
}
```

```
void Start()
{
    float randomYOffset = Random.Range(-0.1f, 0.1f);
    initialPosition = new Vector3(transform.position.x, randomYOffset, transform.position.z);

    mainCamera = Camera.main.transform;
}

void Update()
{
    if (mainCamera != null)
    {
        float step = speed * Time.deltaTime;
        transform.position = Vector3.MoveTowards(transform.position, mainCamera.position, step);
    }

    // 카메라 가까워지면 파괴
    if (Vector3.Distance(transform.position, mainCamera.position) <= 0.3f)
    {
        Destroy(gameObject);
    }
}

void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.CompareTag("Arrow"))
    {
        Destroy(gameObject);

        Monster[] monsters = FindObjectsOfType<Monster>();
        foreach (Monster monster in monsters)
        {
            monster.BombExplosion();
        }
    }
}
```

ii. 달집 놀이 (쥐불 놀이)

- 쥐불의 불의 세기를 조절하는 기능 구현

  불의 세기를 직접 조절한 후 쥐불을 달집에 향해 던져 달집의 불길 유지

```
void Start()
{
    rb = GetComponent<Rigidbody>();
}

void Update()
{
    // 불세기 강화
    float intensity = initialIntensity * rb.velocity.magnitude * intensityMultiplier;

    // 불세기 조절
    GetComponent<ParticleSystem>().startLifetime = intensity;

    // 불씨가 너무 강해져 없어질 경우
    if (intensity >= 10.0f)
    {
        DestroyFire();
    }

    // 불의 세기가 너무 약해져서 없어질 경우
    if (intensity <= 0.1f)
    {
        DestroyFire();
    }

    Debug.Log("불의 세기:"+ intensity);
}

void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.CompareTag("Moon"))
    {
        Destroy(gameObject);
    }
}
```

3. 홍보 및 IP 확장

홍보 콘텐츠 제작 - 카드뉴스 : Instagram

스토리 소개:
프로젝트의 기획 의도와 전통 게임의 독특함을 강조한 내러티브를 Instagram 스토리를 통해 소개합니다. 이를 통해 브랜드 인지도를 높이고 일관된 시각적 아이덴티티와 메시지를 유지합니다.

적용 기술 소개 (영문):
게임에 적용된 XR 기술인 Casual Game, Immersive, Multiplayer, 3D Modeling, Game Animation, Virtual Reality, Hand-tracking, Pass-through, HMD 등을 영문으로 설명하여 기술적인 측면을 강조합니다.

DRIFT interactive studio:
프로젝트를 개발하는 DRIFT interactive studio 팀에 대한 소개를 포함하여 팀 아이덴티티를 강조합니다.

b. 홍보 콘텐츠 제작 - AR : TikTok, Instagram

AR 필터 제작: 게임과 관련된 AR 필터를 제작하여 TikTok과 Instagram을 통해 사용자에게 즐거운 콘텐츠를 제공하고 추가적인 이벤트를 진행합니다.
c. 홍보 콘텐츠 제작 - 영상, 릴스, 쇼츠

쇼츠 및 릴스: 짧고 다양한 콘텐츠를 TikTok, Instagram, Youtube에 공유하여 게임 플레이, 트레일러, 브랜드 컨셉 소개, 게임 개발 과정 등을 소개합니다.

영상: Youtube를 통해 초기 데모 버전, 게임 튜토리얼 안내, 홍보용 트레일러 등을 소개하고 게임의 상세한 정보를 제공합니다.

d. 광고 계획 - TikTok, Instagram, Youtube

무료 광고: SNS 이벤트, 설문 조사, Q&A 세션 등을 통해 무료 광고를 효과적으로 활용하여 사용자들과의 상호작용을 높입니다.

유료 광고: Youtube 프리롤 광고, Instagram Stories 및 스폰서 광고, TikTok 인피드 광고 등을 통해 브랜드 인지도를 증가시키고 프로젝트를 홍보합니다.

IP 커머스 및 NFT 전략

IP 커머스 확장: 투호놀이, 제기차기, 쥐불놀이, 낙화놀이 등을 기반으로 한 게임 IP를 활용하여 가상 아이템 서비스를 제공하는 IP 커머스 플랫폼을 구축합니다. 이를 통해 사용자들은 게임에서 획득한 가상 아이템을 실제로 구매하거나 교환할 수 있습니다.

아이템 NFT화: IP를 NFT 플러그인과 연동하여 게임 내 아이템을 NFT로 제작합니다. 이로써 사용자들은 게임에서 얻은 아이템을 디지털 자산으로 소유하고 거래할 수 있게 됩니다.

NFT 거래소 출시: 게임과 관련된 NFT 아이템을 전문적인 NFT 거래소에 출시하여 사용자 간 거래를 촉진하고 아이템의 가치를 높입니다.

유저 혜택 및 이벤트: 게임의 진행도에 따른 NFT 아이템 보상이나 이벤트를 통해 사용자들에게 간접 P2E(Play-to-Earn) 형태로 혜택을 제공합니다.

진행과정:
NFT 아이템 제작
IP 커머스 플랫폼과 홍보 캠페인 구축
NFT 시장과 연계 및 아이템 거래 활동 감시
유저 혜택 및 이벤트 실행 및 관리
효과적인 마케팅 및 홍보 전략 수립과 실행

예상 결과 산출물:
유저들이 전통 놀이에 대한 참여와 관심이 크게 증가할 것으로 예상됩니다.
NFT 아이템 거래의 발생 및 수익화로 인해 IP 커머스의 매출이 상당히 증가할 것입니다.
프로젝트의 수익과 브랜드 인지도가 글로벌 출시로 인해 향상될 것으로 기대됩니다.

향후 확장 방향 및 기대 효과:

NFT 시장 동향을 지속적으로 관찰하여 적절한 타이밍에 거래 활동을 조절하고 최적화할 수 있습니다.
유저 데이터와 피드백을 활용하여 서비스를 지속적으로 개선하고 사용자 경험을 향상시킬 수 있습니다.
글로벌 출시를 통해 다양한 지역의 사용자들에게 접근하고 IP 커머스를 확장할 수 있습니다.
IP 커머스와 NFT 결합 프로모션을 진행하여 브랜드 인지도와 사용자 참여를 증가시킬 것으로 예상됩니다.