GAME DEVELOPER PORTFOLIO

박재영 / PARK JAE YOUNG
연락처 : 010-7257-6010
Email : pjyjy610@gmail.com

플레이어 경험을 설계하는 게임 개발자 입니다.

아이콘 클릭 시 이동

GITHUB NOTION BLOG

학력

2024.03 - 2025.02 동양미래대학교 컴퓨터소프트웨어공학과(학사) 졸업 (GPA 3.43 / 4.5)
2019.03 - 2024.02 동양미래대학교 컴퓨터소프트웨어공학과(전문학사) 졸업 (GPA 3.75 / 4.5)
2016.03 - 2019.02 범박 고등학교 졸업

대외 활동

2025 - 스마일게이트 데브 커뮤니티 해커톤 Infinithon 2025 (3일)
2024 - 스마일게이트 UNSEEN (4개월)
2022 - 넥슨 MapleStory Worlds X SUPER HACKATHON 2022 (4개월)

수상

2024 - 동양미래대학교 2024 스마트 프로젝트 경진대회 장려상
2023 - 동양미래대학교 2023 스마트 SW 개발 경진대회 장려상
2022 - 넥슨 MapleStory Worlds X SUPER HACKATHON 2022 최다질문상

목차

INDEX

기술 인사이트

PROJECT : P

UNSEEN 2기 프로젝트

장르 : 쿼터뷰 멀티 액션 게임
기간 : 2024.03 ~ 2024.06 (4개월)
엔진 : Unreal Engine 5.2.1
개발 인원 : 1인

주요 개발 내용

- GAS를 활용한 플레이어, 몬스터 & 보스 AI
- 데디케이티드 서버 멀티플레이 게임 환경 구축
- 프로파일링 기반의 병목 분석 및 성능 최적화
- 대규모 몬스터 군집 환경 최적화

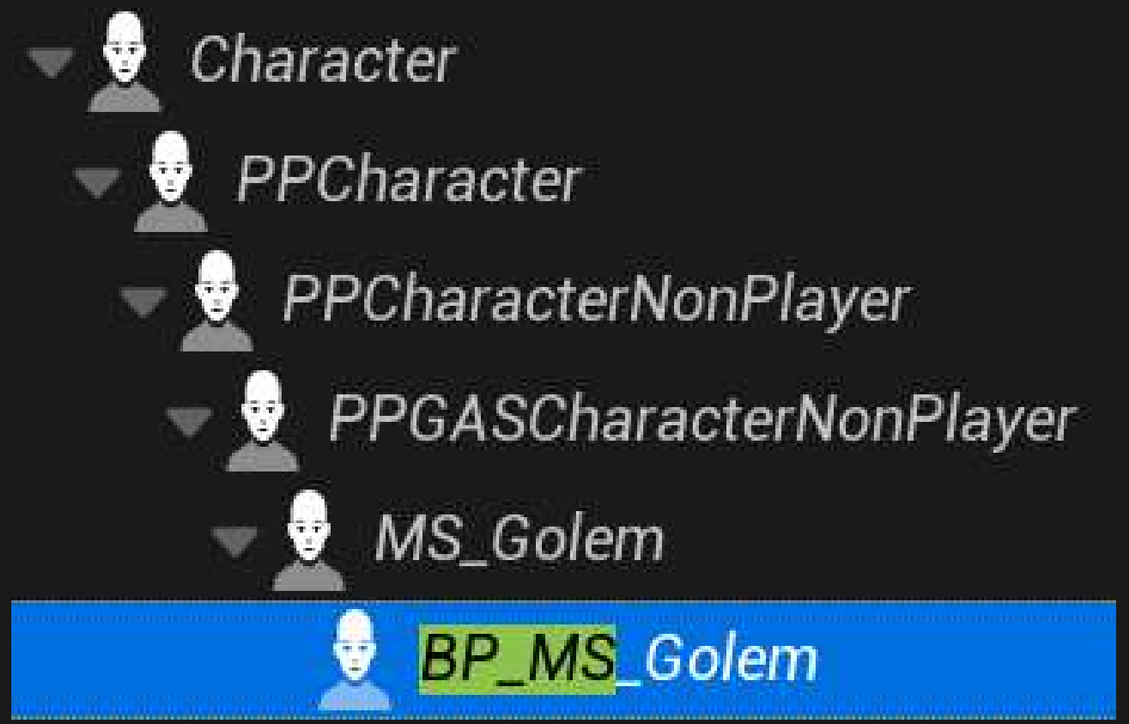
※ GAS : Gameplay Ability System

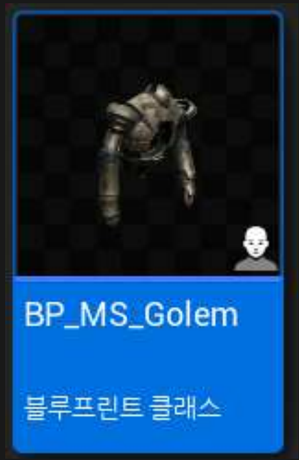
GAS를 활용한 플레이어, 몬스터 & 보스 AI

GAS 상속 구조

RPG 멀티플레이 환경에서 효과적인 시스템을 구축하기 위해 GAS를 사용
플레이어 : PPCharacter를 상속 받는 PPGASCharacter
몬스터 : PPGASCharacterNonPlayer를 상속 받아 구현되는 MS_Golem

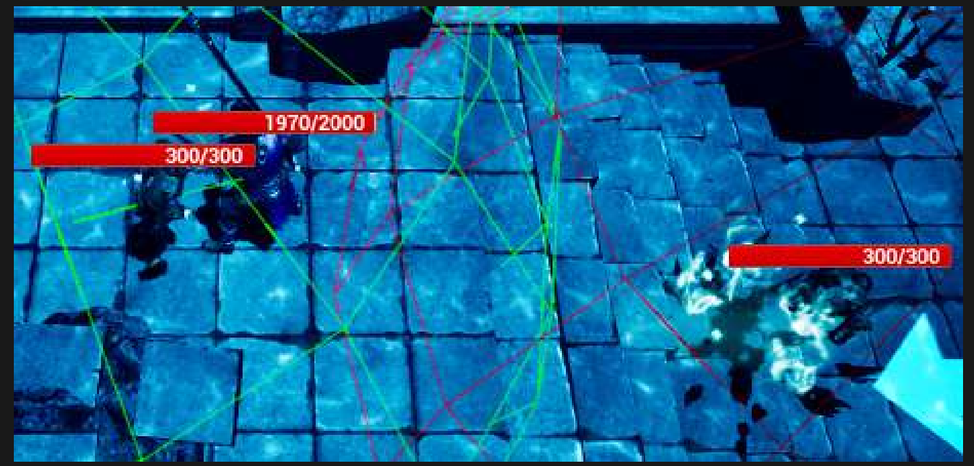
역할 분리 상속으로 안정성과 확장성을 확보한 GAS 구조를 설계





공격 범위 내 플레이어 공격

범위 내 플레이어 추적



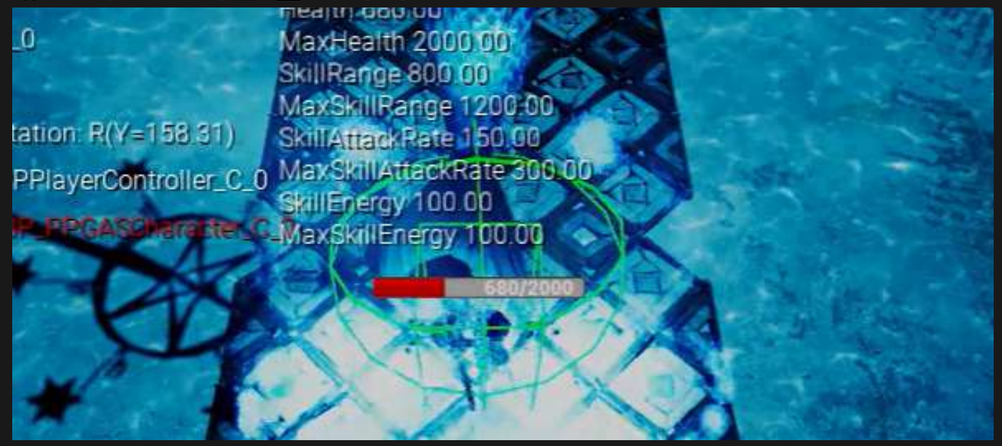
GAS를 활용한 플레이어, 몬스터 & 보스 AI

보스 AI

보스 AI를 구성하기 위한 고민 -> GAS 활용 아이디어 구상 -> GA를 태스크로 활용
GAS 기반 Task를 통해 Behavior Tree 구성 및 패턴 실행

AI의 결정과 스킬의 실행을 분리(디커플링)한 아키텍처를 구성

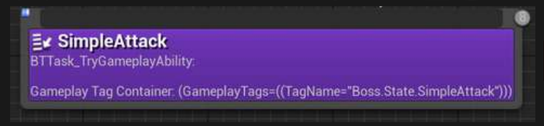
공격에 피격되는 플레이어



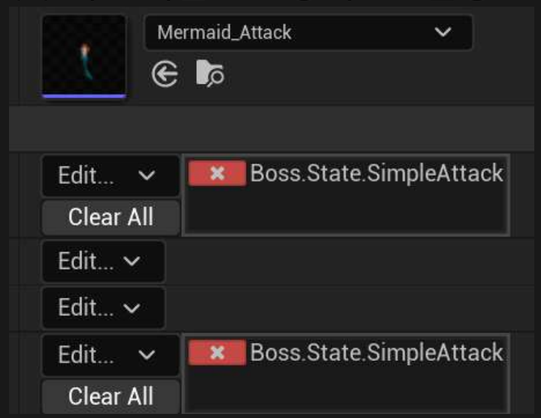
보스 패턴 실행 과정

디커플링된 각 단계는 의존성이 낮고, 다른 패턴으로 파생 가능

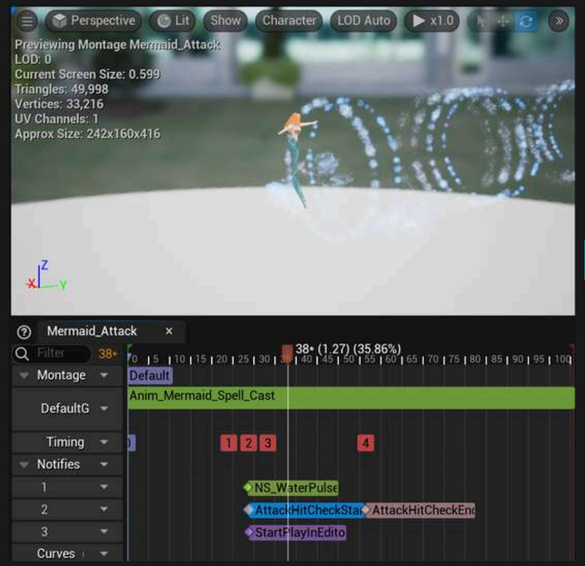
(1) Behavior Tree 태스크 실행



(2) 태그 기반으로 공격 GA 실행

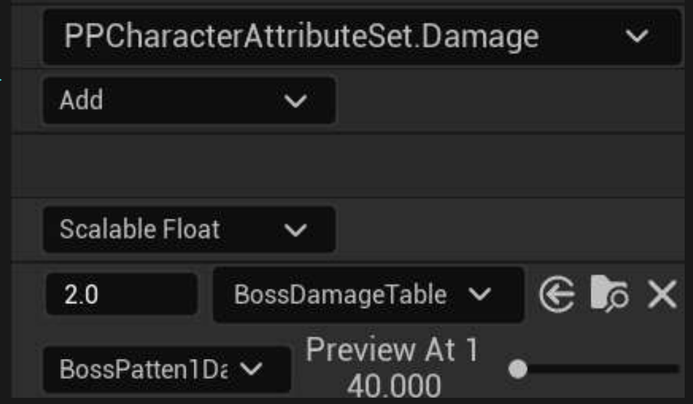


(3) 애니메이션 실행



(4) 공격 판정 GA 실행

(5) 데미지 적용 GE 실행



※ GA : GameplayAbility
※ GE : GameplayEffect

데디케이티드 서버 멀티플레이 게임 환경 구축

AWS EC2 인스턴스 -> 리눅스 서버 사용

데디케이티드 서버 멀티플레이 환경 구축을 위해 AWS를 통해 리눅스 서버 사용
멀티플레이 시스템 동기화를 위해 Replication, RPC(Remote Procedure Call) 활용

서버에 연결된 두 클라이언트가 실행된 모습

사망 애니메이션 RPC 처리

서버는 애니메이션과 같은 시각적 표현을 실행하지 않기 때문에 RPC를 통해 클라이언트에게 전파
사망 애니메이션 MulticastRPC로 전송 -> HasAuthority 체크하여 클라이언트에서만 실행 -> 클라이언트와 서버의 부하 분산 및 동기화

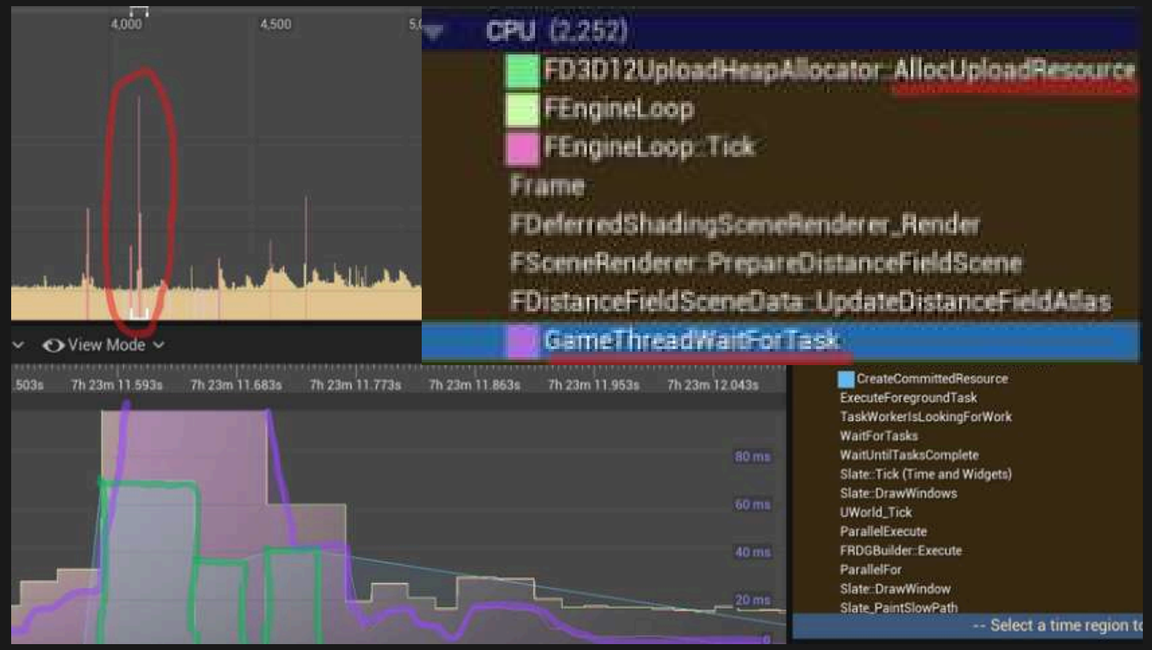
```
void APPCharacter::DeadMulticastRPC_Implementation()
{
    if (!HasAuthority())
    {
        PlayDeadAnimation();
    }
}

void APPCharacter::PlayDeadAnimation()
{
    UAnimInstance* AnimInstance = GetMesh()->GetAnimInstance();
    if (IsValid(AnimInstance))
    {
        AnimInstance->StopAllMontages(0.0f);
        AnimInstance->Montage_Play(DeadMontage);
    }
```

PPCharacter - 플레이어 사망 애니메이션 처리

양쪽 클라이언트에서 보이는 사망 애니메이션 처리

프로파일링 기반의 병목 분석 및 성능 최적화

언리얼 인사이트를 활용한 CPU 부하 분석

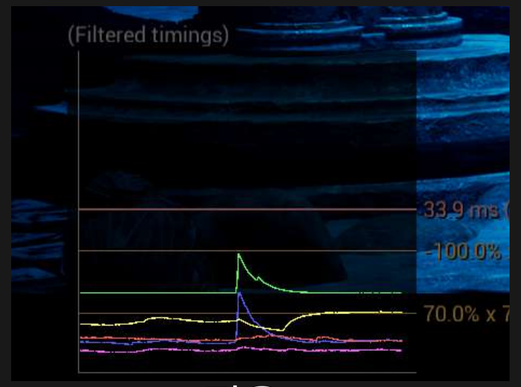


[문제 상황]
보스 컷 씬 재생 타이밍 (빨강색) : 게임 중 가장 큰 부하를 나타내는 타이밍

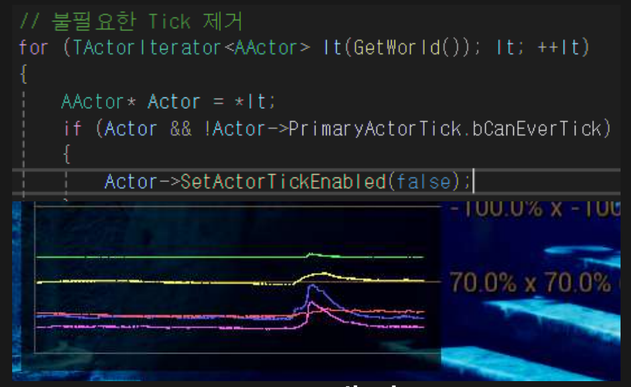
[문제 분석]
AllocUpdateResource : GPU 리소스의 업데이트로 인해 발생
GameThreadWaitForTask : 태스크 종료를 기다리는 지연 시간
-> **빠른 줌 인으로 인한 보스와 맵의 급격한 렌더링으로 순간적인 부하가 발생**

[해결 방안]
- 컷 씬 전에 보스, 맵을 미리 로드(레벨 스트리밍, 비동기 에셋 로드)
- Tick 최적화(불필요한 액터의 Tick 제거 및 비활성화)

해결 방안 적용 => 부하가 높은 환경에서도 눈에 띄게 성능이 개선되어 안정적인 프레임 달성



적용 X



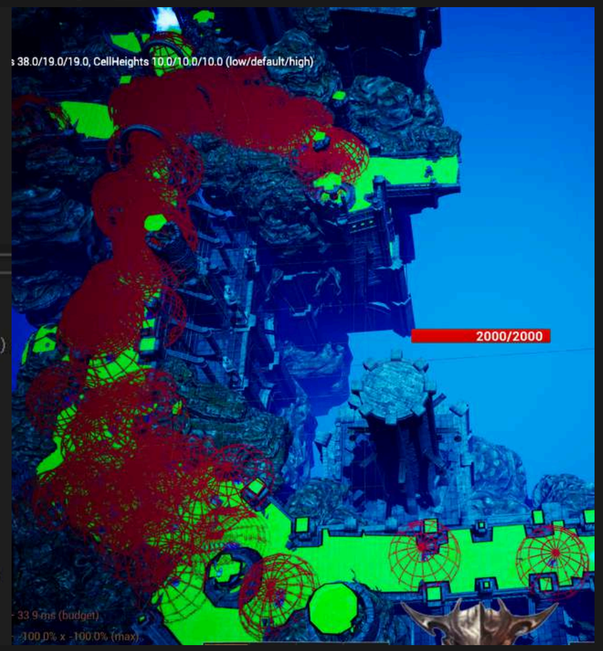
Tick 제거

Tick 제거 + 레벨 스트리밍

대규모 몬스터 군집 환경 최적화

몬스터 AI 옥트리(OCTREE) 탐색 구현

```
void FGameOctree::Insert(const FGameOctreeElement& Element)
{
    if (!Bounds.IsInside(Element.Bounds.Origin)) return;

    if (bIsLeafNode && Elements.Num() < MaxElements)
    {
        Elements.Add(Element);
        return;
    }

    if (bIsLeafNode) Subdivide();

    for (const auto& Elem : Elements)
    {
        for (int i = 0; i < 8; ++i)
        {
            if (Children[i]->Bounds.IsInside(Elem.Bounds.Origin))
            {
                Children[i]->Insert(Elem);
                break;
            }
        }
    }
    Elements.Empty();

    for (int i = 0; i < 8; ++i)
    {
        if (Children[i]->Bounds.IsInside(Element.Bounds.Origin))
        {
            Children[i]->Insert(Element);
            return;
```



[목표]
다수의 몬스터 객체가 등장하는 고밀도 환경에서도
끊김 없는 플레이를 제공하는 것을 목표로 최적화를 시도

[문제 상황 시나리오]
몬스터 105마리가 0.05초마다 Detect를 실행하는 극심한 부하 상황 가정

[문제 분석 및 해결 시도]
기존 방식 : OverlapMultiByChannel -> 월드에 있는 모든 액터를 확인
옥트리 방식 : 3차원 공간을 8개의 구역으로 재귀 분할해 구역 별 검색

옥트리 탐색 적용

기존 방식(78.8ms)

옥트리 적용(77.4ms)

Stat AI : 평균 3.54ms -> 2.37ms로 약 33% 감소

Cycle counters (flat)	CallCount	InclusiveAvg	InclusiveMax	ExclusiveAvg
Overall AI Time	105	3.54 ms	3.97 ms	0.01 ms
MoveTo	1	0.06 ms	0.17 ms	0.02 ms
Perception System	1	0.00 ms	0.00 ms	0.00 ms
Perception System - Process Stim	1	0.00 ms	0.00 ms	0.00 ms

▼

Cycle counters (flat)	CallCount	InclusiveAvg	InclusiveMax	ExclusiveAvg
Overall AI Time	105	2.37 ms	2.72 ms	0.01 ms
MoveTo	1	0.06 ms	0.16 ms	0.02 ms
Perception System	1	0.00 ms	0.01 ms	0.00 ms
Perception System - Process Stim	1	0.00 ms	0.00 ms	0.00 ms

=> 규모가 커질수록 높은 개선 폭을 보이는 것을 감안해도 미미한 개선 폭

대규모 몬스터 군집 환경 최적화

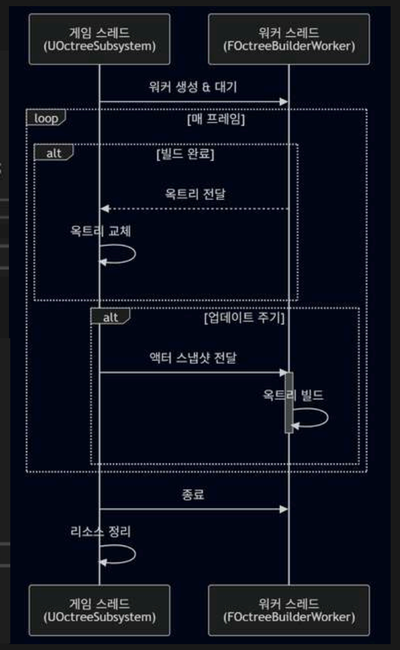
멀티스레딩(더블버퍼링) 옥트리 탐색 구현

OctreeSubsystem : CustomTick()

```
if (Worker.IsValid() && Worker->bIsWorkDone)
{
    TUniquePtr<FGameOctree> NewOctreeResult = Worker->GetResult();
    if (NewOctreeResult)
    {
        TSharedPtr<FGameOctree, ESPMode::ThreadSafe> NewActiveTree(NewOctreeResult.Release());
        FScopeLock Lock(&OctreeSwapSection);
        ActiveOctree = NewActiveTree;
    }
}
if (ActorsSnapshot.Num() > 0) Worker->StartWork(MoveTemp(ActorsSnapshot));
```

OctreeBuilderWorker : Run()

```
// 항상 새 옥트리를 처음부터 생성합니다.
BuiltOctree = MakeUnique<FGameOctree>(Bounds, MaxElements);

// 게임 스레드로부터 전달받은 (최신) 액터 목록을 순회하며 옥트리에 삽입
if (BuiltOctree.IsValid())
{
    for (AActor* Actor : ActorsToProcess)
    {
        if (IsValid(Actor)) BuiltOctree->Insert(FGameOctreeElement(Actor));
    }
}
bIsWorkDone = true;
```



멀티스레딩(더블버퍼링)을 통해 추가 개선 시도

=> 무거운 옥트리 재구축 작업을 워커 스레드로 넘기고, 게임 스레드는 이전 옥트리를 사용하며, 워커 스레드에서 새 옥트리가 완성되면 교체

[이점]

- 게임 스레드의 부담을 덜어 부드러운 게임 제공
- 항상 최신 상태의 공간 검색 구조 유지

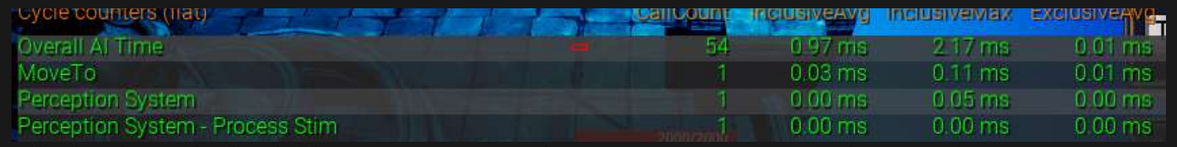
=> 쿼터뷰에서는 평면적인 처리가 많이 이뤄지기 때문에 쿼드트리 탐색도 유용하지만, 3D 범용성을 위해 옥트리 사용, 77.4ms -> 38.9ms (약 50% 개선)

=> 원활한 플레이를 위해서는 최소 60FPS(16.67ms)의 Frame 달성 필요

멀티스레딩(더블버퍼링) 적용

멀티스레딩 옥트리 적용(38.9ms)

Stat AI : 평균 2.37ms -> 0.97ms로 약 59% 감소



엔진 시스템에 의해 CallCount가 105 -> 54로 줄어든 모습

최소 FPS를 위해 약57%의 추가 개선 요구됨

-> 추가 최적화 필요

=> 관심영역 설정 및 몬스터 상태 변경 구현

대규모 몬스터 군집 환경 최적화

관심 영역을 통한 플레이어 주변 몬스터 상태 관리

InterestManager : UpdateMonstersState()

```
// 모든 플레이어와의 거리를 체크하여 몬스터의 최종 상태를 결정
for (const FVector& PlayerLocation : AllPlayerLocations)
{
    const float HorizontalDistanceSq = FVector::DistSquared2D(PlayerLocation, MonsterLocation);
    const float VerticalDistance = FMath::Abs(PlayerLocation.Z - MonsterLocation.Z);

    if (VerticalDistance <= VerticalTolerance)
    {
        if (HorizontalDistanceSq < FMath::Square(ActiveRadius))
        {
            DesiredState = EAIState::Active;
            break; // 이미 Active이므로 더 이상 다른 플레이어와 비교할 필요 없음
        }
        else if (HorizontalDistanceSq < FMath::Square(RelevantRadius))
        {
            // Active가 아닌 경우에만 Relevant로 설정 (이미 Active이면 유지)
            if (DesiredState != EAIState::Active)
            {
                DesiredState = EAIState::Relevant;
```

플레이어 주변의 관심영역을 통해 거리에 따라 몬스터의 상태를
Dormant(휴면) / Relevant(영향권) / Active(활성) 상태로 구분해 관리

범위 밖 -> Dormant(휴면) : 몬스터 비활성화
최대 시야 범위 -> Relevant(영향권) : 몬스터 활성화 / AI 비활성화
근접 범위 -> Active(활성) : 몬스터 활성화

[성능 개선 비교]
멀티스레딩 옥트리와 동일 조건 측정 값 :16.67ms로 60프레임 유지
가장 부하가 높은 위치 측정 값 : 19.12ms로 52.3프레임 (약 51% 개선)
=> 극심한 스트레스 상황에서도 큰 폭으로 개선된 모습을 확인

관심 영역 적용

Active, Relevant 영역

영역에 따라 관리되는 몬스터의 상태

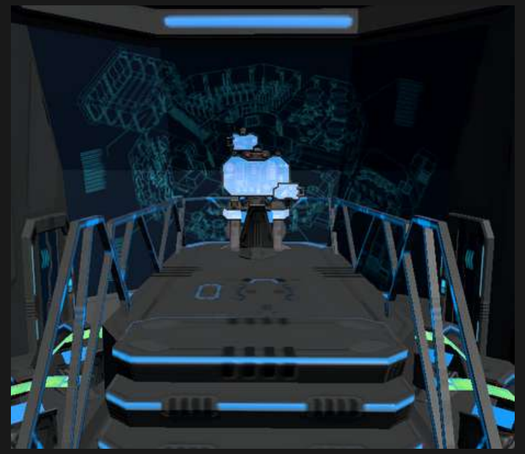
18~19ms 유지

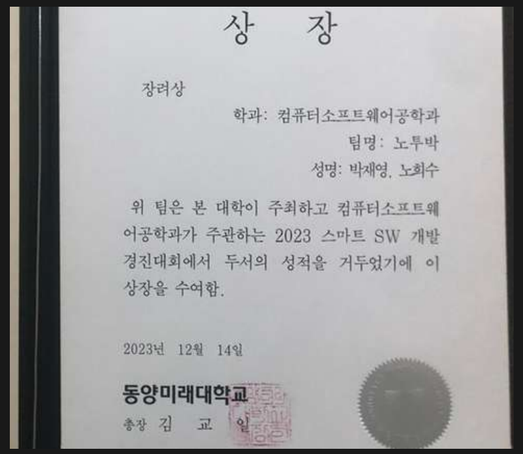
<결론>
[문제 상황]
105마리의 몬스터 AI가 0.05초 마다 Detect(탐색) 실행

[적용 기술]
멀티스레딩 옥트리 + 관심 영역

[결과]
78.8 -> 19.12(ms)
12.69 -> 52.3(FPS)
총 75.74%의 성능 개선

IKARIA : TAINTED SKIES





상 장

장려상

학과: 컴퓨터소프트웨어공학과
팀명: 노두박
성명: 박재영, 노희수

위 팀은 본 대학이 주최하고 컴퓨터소프트웨어공학과가 주관하는 2023 스마트 SW 개발 경진대회에서 두서의 성적을 거두었기에 이 상장을 수여함.

2023년 12월 14일

동양미래대학교
총장 김 교 일

3학년 졸업 작품

장르 : 3D 어드벤처 게임
기간 : 2023.03 ~ 2023.11 (총 8개월)
엔진 : Unity Engine 2021.3.16f1 LTS
개발 인원 : 2인

주요 개발 내용

- 오클루전 컬링 등을 활용한 그래픽스 최적화
- CSV 파일을 활용한 대화 스크립트 구현

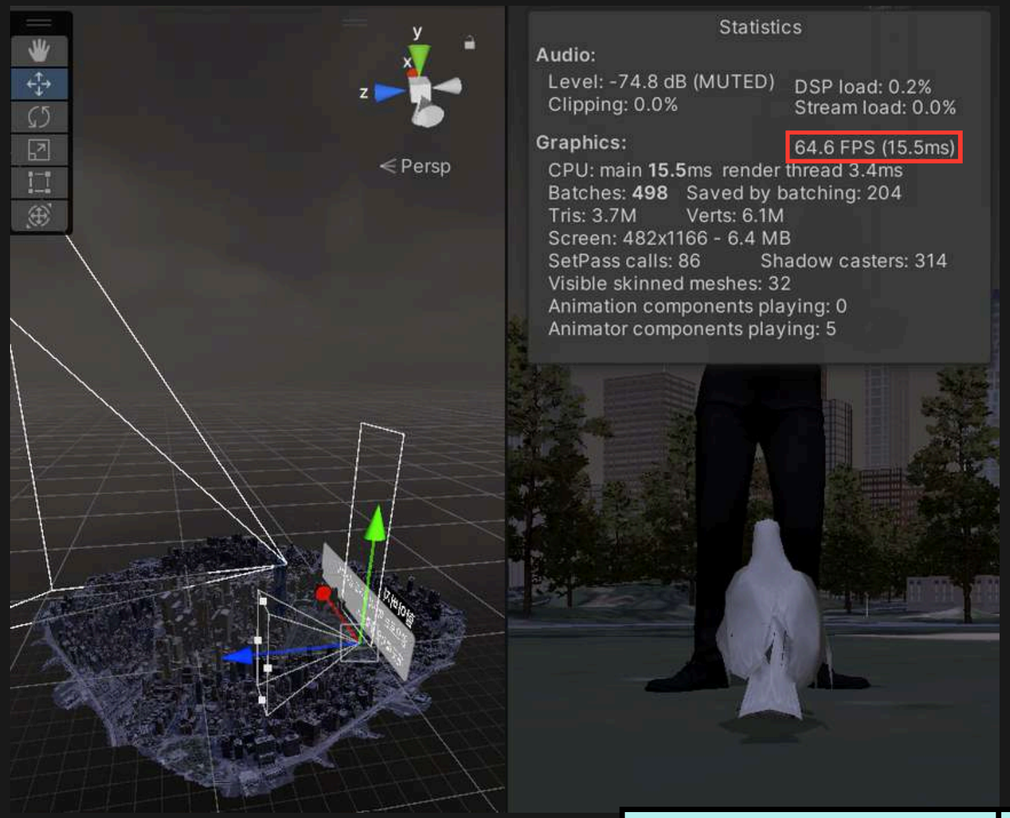
오클루전 컬링 등을 활용한 그래픽스 최적화

오클루전 컬링(OCCLUSION CULLING)

대량의 오브젝트가 존재하는 환경에서 성능 저하를 막기 위해 사용

규모가 큰 도시 맵에서 큰 성능 개선 효과를 볼 수 있었음

저사양 환경에서도 문제 없이 동작하도록 하기 위한 시도



Occlusion Culling	OFF	ON	Optimization
FPS	64.6	86.1	33% 증가
Batches	498	433	13% 감소
Saved by bacthing	204	416	104% 증가

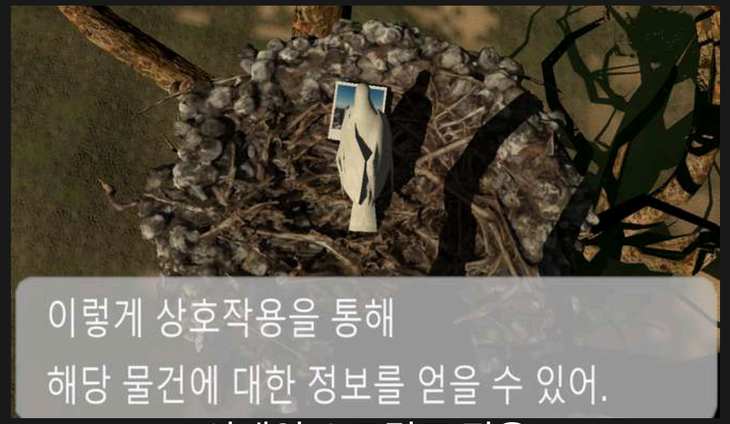
CSV 파일을 활용한 대화 스크립트 구현

인게임 대화 스크립트

다량의 스크립트를 효율적으로 관리 및 유지보수하기 위해
CSV파일을 통해 스토리 진행에 사용되는 대화 스크립트를 구성

대사
게임을 시작하기 전에 간단하게 \n내가 움직이는 방법을 설명해줄게.
먼저 기본적인 움직임이야.
ⓡW`S`A`Dⓑ를 이용하여 앞`뒤`양옆으로 움직일 수 있어.
다음은 날기야.
ⓡSpaceBarⓑ를 이용하여 고도를 상승시키고` \n공중에서 왼쪽 ⓡShift(
다음은 게임 진행에 있어서 필수적인 상호작용이야.
상호작용은 ⓡFⓑ키를 이용하여 할 수 있어. \n옆에 보이는 ⓨ사진ⓑ에 상
나도 언젠간… \n이렇게 깨끗한 도시에서 자유롭게 날 수 있겠지?
이렇게 상호작용을 통해 \n해당 물건에 대한 정보를 얻을 수 있어.
이제 나무 아래에서 기다리고 있는\n 고양이에게 가볼까?

CSV 파일에 대화 스크립트 작성



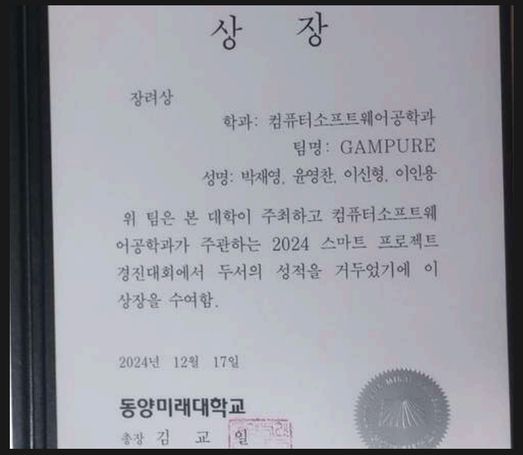
인게임 스크립트 적용

DialogueParser

```
for (int i = 1; i < data.Length;)
{
    string[] row = data[i].Split(new char[] { ',
    Dialogue dialgoue = new Dialogue();
    dialgoue.name = row[1];
    List<string> contextList = new List<string>(
    do
    {
        contextList.Add(row[2]);
        if (++i < data.Length)
            row = data[i].Split(new char[] { ','
        else break;
    }
    while (row[0].ToString() == "");
    dialgoue.contexts = contextList.ToArray();
    dialgoueList.Add(dialgoue);
}
return dialgoueList.ToArray();
```

```
if (isFirst) // 첫번째 대화가 진행됬는지 체크
{
    dialgoue.dialogues = DatabaseManager.instance.GetDialogue((int)dialgoue.line.x, (int)dialgoue.line.y);
    isFirst = false;
    if(transform.GetComponent<MissionWayPoint>() != null) { transform.GetComponent<MissionWayPoint>().markerOn = t

    return dialgoue.dialogues;
}
else // 두번째 이후 대화라면 dialoguesB를 실행
{
    dialgoue.dialoguesB = DatabaseManager.instance.GetDialogue((int)dialgoue.lineB.x, (int)dialgoue.lineB.y);
    isSecond = false;
    return dialgoue.dialoguesB;
}


e //일반 대화오브젝트라면 자체 대화를 실행

return dialgoue.dialogues;
```

MONSTER KILLER



상 장

장려상

학과: 컴퓨터소프트웨어공학과
팀명: GAMPURE
성명: 박재영, 윤영찬, 이신형, 이인용

위 팀은 본 대학이 주최하고 컴퓨터소프트웨어공학과가 주관하는 2024 스마트 프로젝트 경진대회에서 두서의 성적을 거두었기에 이 상장을 수여함.

2024년 12월 17일

동양미래대학교

총장 김 교 일

4학년 졸업 작품

장르 : 캐주얼 웹 게임
기간 : 2024.03 ~ 2024.11 (총 8개월)
엔진 : Unity Engine 2022.3.28f1 LTS
개발 인원 : 4인

주요 개발 내용

- 오브젝트 풀링을 활용한 몬스터 스폰 시스템
- FSM을 적용한 보스 패턴 구현

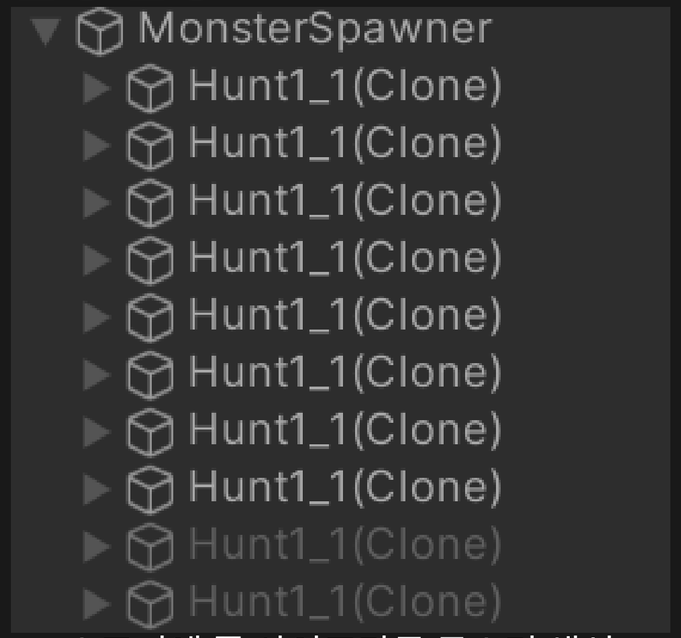
오브젝트 풀링을 활용한 몬스터 스폰 시스템

오브젝트 풀로 구현한 몬스터 스포너

대량 소환되는 몬스터의 효율적인 관리로 부하를 줄이기 위해 구현
풀에 생성된 몬스터를 재사용하여 생성, 파괴 비용을 크게 감축

```
// 유효한 오브젝트가 나올 때까지 큐에서 꺼내기
while (poolQueue.Count > 0)
{
    obj = poolQueue.Dequeue();
    if (obj != null) break;
}

// 파괴되었거나 큐가 비어 있었다면 새로 생성
if (obj == null) obj = GameObject.Instantiate(prefab, parent);
obj.transform.SetPositionAndRotation(position, rotation);
obj.SetActive(true);
IPoolable poolable = obj.GetComponent<IPoolable>();
poolable?.OnSpawned();
return obj;
```

풀링 실행



스포너에 풀 사이즈만큼 몬스터 생성

몬스터를 스포너에서 꺼내 소환 -> 처치 시 풀로 복귀

FSM을 적용한 보스 패턴 구현

몬스터 소환 패턴(FSM)

기존에 애니메이터로 단순 동작하던 공격 패턴을 체계화하기 위해 구현
플레이어를 추적하며 충돌 시 지속 공격을 가하는 몬스터 소환 패턴
보스 체력이 50% 이하로 떨어지면 소환 속도가 3배가 되도록 설정

```
private void Summon()
{
    var boss = controller as SummonBossController;

    // 소환 위치는 플레이어 위치로
    Vector3 spawnPos = boss.Target.position;
    Quaternion rot = boss.SummonPrefab.transform.rotation;

    GameObject.Instantiate(boss.SummonPrefab, spawnPos, rot);
    Debug.Log("플레이어 위치에 몬스터 소환");

    // 소환 애니메이션 트리거 실행
    controller.Animator.SetTrigger("Attack");
```

몬스터 소환 실행

Idle 상태의 보스

플레이어를 공격하는 몬스터 소환

자세한 학습 및 구현 과정 - NOTION

DREAM SCAPE 사이드 프로젝트

장르 : 3인칭 멀티 액션 어드벤처 게임
엔진 : Unreal Engine 5.4.4

기간 : 2025.06 ~ 진행 중
개발 인원 : 12인

- 카툰 렌더링 쉐이더 구현 및 적용

카툰 렌더링 쉐이더 적용

카툰 렌더링을 모든 액터에 입히는 것이 아닌 선택적으로 빠르게 적용 하기 위해 구현
캐싱된 마스터 쉐이딩 머티리얼과 텍스처를 합성해 툰 쉐이딩이 적용된 머티리얼 인스턴스를 생성

```
if (TWeakObjectPtr<UMaterialInstanceDynamic>* CachedInst = InstanceCache.Find(BaseTexture))
{
    if (CachedInst->IsValid()) return CachedInst->Get();
}

UMaterialInstanceDynamic* NewInstance = UMaterialInstanceDynamic::Create(CelShaderMaster, nullptr);
NewInstance->SetTextureParameterValue(BaseColorParamName, BaseTexture);
CopyMaterialParameters(SourceMat, NewInstance);

InstanceCache.Add(BaseTexture, NewInstance);
return NewInstance;
```

캐싱된 인스턴스 생성 및 원본의 파라미터를 복사하는 로직

인게임 적용

단순 적용 : 부자연스러운 쉐이딩

마스터 쉐이딩 머티리얼 설정 변경

자연스럽게 적용된 모습

기획서 - GOOGLE DRIVE 게임 플레이 - MAPLESTORY WORLDS

RESISTANCE VS BLACK WING MSW X SUPER HACKATHON 2022 프로젝트

장르 : 실시간 팀 대전 전략 게임
엔진 : MapleStory Worlds

기간 : 2022.09 ~ 2022.12 (4개월)
개발 인원 : 3인

- 거점 점령/건물 건설 기능 구현

거점 점령 및 건물 건설

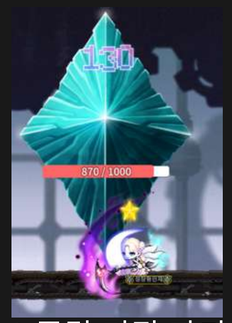
점령 판단 및 점령 시 건물 건설을 활성화하기 위해 구현
점령 시간에 의한 거점 점령 및 스탯에 따른 건물 건설 활성화

```
elseif(tem == 5)then
    self.is_left = false
    self.buildable = true
    self.Entity.BasicParticleComponent.Color = Color.red
    for key, value in pairs(self.right_list) do
        value.PlayerTimer:UI_off()
        value.PlayerBuilding:build_UI_on(value)
        value.StatComponent:GetEXP(15)
        value.PlayerBuilding:occupy_UI_on()
        self:put_E(value)
    end
else
    self.sec = tem
    self:update_time()
```

점령 시간 계산 (5초)

수료 및 최다질문상 수상

인게임 실행 과정



1.중립 거점 파괴

2. 중립 점령지 점령 중...

3. 점령 성공

4. 건물 건설 스탯을 통해 건물 해금

5. 건물 건설 완료

딴짓 방지 도구 소개 영상 - YOUTUBE 인피니톤 공식 영상 - YOUTUBE

ATTENTION 스마일게이트 데브 커뮤니티 해커톤, INFINITHON 2025

설명 : 개발 중 딴짓 방지 깜짝 퀴즈 플러그인
엔진 : Unreal Engine 5.4.4

기간 : 2025.08.08 ~ 10 (3일)
개발 인원 : 5인

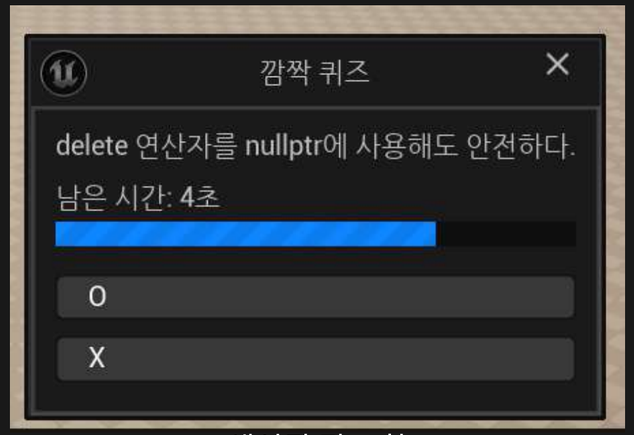
- SLATE UI를 활용한 언리얼 에디터 퀴즈 팝업 시스템
- WIN API를 활용한 외부 창 입력 감지

SLATE UI 팝업 창

팝업 퀴즈 창 UI의 형태 구성을 위해 구현
SCompoundWidget 기반의 Slate UI 퀴즈 창

```
switch (QuizType)
{
case EQuizType::MultipleChoice:
    ChildSlot
        [
            SNew(SBorder)
                .Padding(12)
                [
```

퀴즈 타입 별 팝업 창 생성



생성된 퀴즈 창

외부 창 입력 감지

외부 프로그램에서 개발 중 퀴즈 팝업 방지를 위해 구현
윈도우 전역 입력을 후킹해 활동 감지

```
bool FGlobalInputWatcher::ShouldFireForCurrentForeground()
{
    const FString Proc = GetForegroundProcessNameLower();
    if (Proc.IsEmpty()) return false;
    return Whitelist.Contains(Proc);
```

화이트리스트 필터링으로 지정된 앱에서만 반응

▶ 지정한 앱(Visual Studio 등)에서 작업 시 퀴즈가 뜨지 않음