

김세현 포트폴리오

GitHub: <https://github.com/ArshesSH>

프로젝트 목차

1	프로젝트 요약.....	2
2	TOPES 교통 통합 관제 시스템.....	3
3	화재유형별 화재진압기법 개선을 위한 표준작전절차 및 실증 시스템 개발 과제.....	4
4	AWAS-XR.....	5
5	LAND400 HUMS.....	6
6	펄어비스 신작프로젝트.....	7
7	Project LUP.....	8
8	Deus Ex Machina.....	27
9	RockmanX5 모작.....	35
10	Deadlock.....	44
11	VRFlight Simulation.....	48
12	Sky Stability.....	51

1 프로젝트 요약

1. TOPES 교통 통합 관제 시스템

통합 교통 관제 플랫폼

프로젝트 기간: 2024.12 ~ 2025.04

2. 화재유형별 화재진압기법 개선을 위한 표준작전절차 및 실증 시스템 개발 과제

AR 화재 진압 교육 훈련 시뮬레이터

프로젝트 기간: 2024.10 ~ 2024.12

3. AWAS-XR

HoloLens2 MR 공정 교육 시나리오 제작 프로그램

프로젝트 기간: 2023.12 ~ 2024.05

4. LAND400 HUMS

한화시스템 납품 상태감시시스템

프로젝트 기간: 2023.06 ~ 2024.11

5. 페어비스 신작 프로젝트

페어비스 붉은사막 게임플레이 컨텐츠 개발

프로젝트 기간: 2023.03 - 2023.05

6. Project LUP

Behavior Tree를 이용한 방치형 RPG 게임 (<https://youtu.be/9qVIJFajaxc>)

프로젝트 기간: 2023.01~ 2023.02

7. Deus Ex Machina

기획자와 협업한 비대칭 PVP 게임 (<https://youtu.be/p3pPeP9O2TY>)

프로젝트 기간: 2022.09 ~ 2022.12

8. Rockman X5 모작

Windows API를 활용한 2D 슈팅 게임 (<https://youtu.be/lzxj7TzOfHA>)

프로젝트 기간: 2022.08 ~ 2022.09

9. Deadlock

콘솔 창에서 이미지 처리를 구현한 C 언어 2D 탱크 슈팅 게임 (<https://youtu.be/ym8-lvTHfhM>)

프로젝트 기간: 2022.05 ~ 2022.05

10. VR Flight simulation

UE4를 사용한 VR 항공기 조종 시뮬레이션 (<https://youtu.be/R9U9pKLASw0?t=942>)

프로젝트 기간: 2020.12 - 2021.06

11. Sky Stability

PID를 사용한 무인항공기 수평 자세제어 시스템 (Arduino, C)

프로젝트 기간: 2020.09 - 2020.12

2 TOPES 교통 통합 관제 시스템

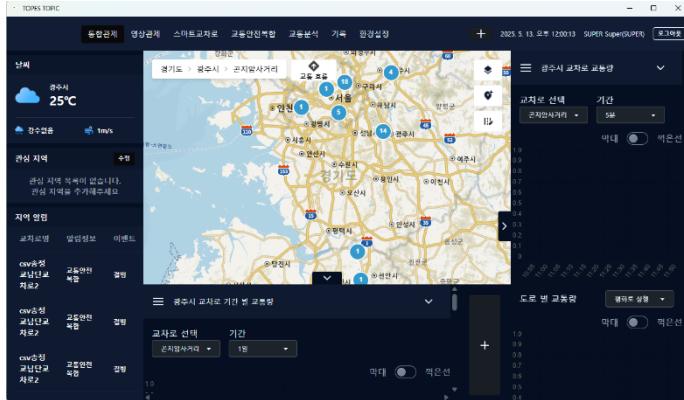


그림 2 TOPES 통합관제시스템



그림 1 스마트교차로 생성 모습

개요

- CCTV 관제, 스마트교차로 시스템, AI를 이용한 교통분석 등을 가진 교통 통합 관제 시스템
- 프로젝트 기간: 2024.12 ~ 2025.04
- 역할: Windows 데스크탑 클라이언트 개발
- 사용 기술: TypeScript, Svelte, Babylon.js, Tailwind CSS, Tauri, OpenLayers
- 주요 기여 내용
 - 데스크탑 클라이언트 개발
 - Svelte, Tauri를 이용한 UI 및 기능 구현
 - 2D 페인트 툴 기능 구현
 - 선 그리기, 패턴 그리기, 페인트 및 브러시, 이미지 추가 및 크기 조정
 - 2D 페인트 결과를 Babylon.js 3D 씬으로 연동
 - Babylon.js를 이용한 스마트교차로 및 교통분석 시스템
 - 차량 데이터를 이용한 차량 이동 및 신호등 신호 시뮬레이션
 - 2D 영상 좌표를 3D 가상 공간으로 변환하는 기능
 - 지역 위치와 시간에 따른 일출 및 일몰 효과
 - OpenLayers 지도 시스템 기능 구현
 - 아이콘 클러스터링 기능
 - 경로 생성 기능

3 화재유형별 화재진압기법 개선을 위한 표준작전절차 및 실증 시스템 개발 과제



그림 3 화재 진압 시뮬레이터 이미지

- 개요
 - Unreal Engine 5를 이용한 모바일 AR 화재 진압 시뮬레이터
- 프로젝트 기간: 2024.10 ~ 2024.12
- 역할: Android 클라이언트 개발
- 사용 기술: Unreal Engine 5, C++
- 주요 기여 내용
 - 교육 훈련 시나리오 제작을 위한 클래스 및 컨텐츠 제작
 - 시나리오 조작 컨트롤러
 - FSM에 따라 애니메이션을 재생하고, 이벤트를 발생하는 시나리오 액터
 - 소방호스를 이용한 불 끄기
 - 교육 훈련 UI 제작
 - 설정 UI, 시나리오 UI, 퀴즈 UI, PDF 뷰어

4 AWAS-XR



그림 4 AWAS-XR 시연 모습

- 개요
 - HoloLens2 MR 공정 교육 시나리오 제작 프로그램
- 프로젝트 기간: 2023.12 ~ 2024.05
- 역할: HoloLens2 클라이언트 개발
- 사용 기술: Unity, C#
- 주요 기여 내용
 - 컨텐츠 제작
 - Color Picker 제작
 - User Menu UI 제작
 - 프로젝트 전체 UI Physics based에서 UGUI로 전환
 - 시스템 제작 및 개선
 - 가상공간 물체 Grab 시스템 구현
 - Job system 을 이용한 모델 로딩 속도 향상
 - HoloLens2 에 적용 가능한 Outline shader 제작

5 LAND400 HUMS

- 개요
 - LAND400 상태감시시스템
- 프로젝트 기간: 2023.06 ~ 2023.12
- 역할: HUMS(Health and Usage Monitoring System) SBC 소프트웨어 설계 및 구현
- 사용 기술: Linux, C
- 주요 기여 내용
 - HUMS SBC 소프트웨어 전체 재설계 및 구현
 - 데이터 수집 - 정제 - 저장 파이프라인 설계 및 구현
 - UDP 통신 패킷 검사 및 복구 시스템 설계 및 구현
 - 정적 검사 및 동적 검사 통과

6 펄어비스 신작프로젝트



그림 5 붉은사막 퀵슬롯 UI



그림 6 붉은사막 적 체력 UI

- 개요
 - 펄어비스 붉은사막 게임플레이 컨텐츠 개발
- 프로젝트 기간: 2023.03 ~ 2023.05
- 역할: UI/UX 개발 및 게임 플레이 관련 기능 개발 및 유지보수
- 사용 기술: C++, HTML, CSS
- 주요 기여 내용
 - UI/UX 개발
 - 캐릭터 및 장비 교체를 위한 원형 퀵슬롯 UI 구현
 - 적 체력 UI 개선
 - 인벤토리 UI 및 탑승물 UI 개선

7 Project LUP

7.1 프로젝트 소개

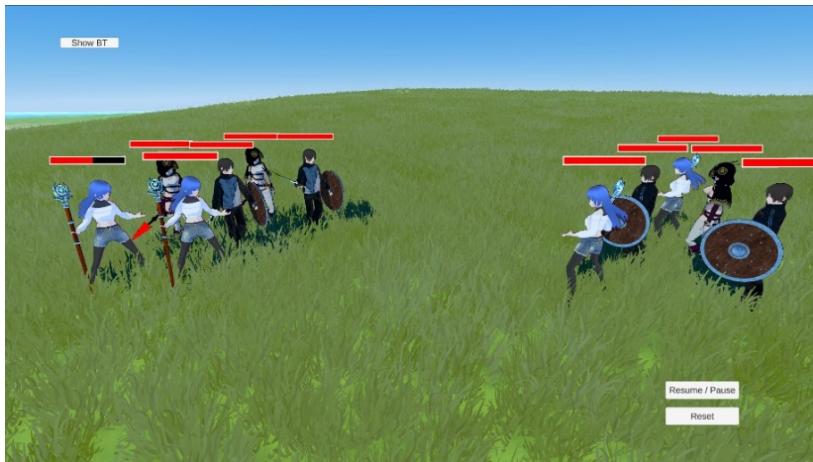


그림 7 게임 전투 대기 장면

- **프로젝트 기간:** 2023.01 ~ 2023.02
- **프로젝트 인원:** 1인
- **프로젝트 주제**
Last Fortress 및 세븐나이츠와 유사한
방치형 RPG 게임
- **사용 기술**
Unity, C#
- **참고 영상:**
<https://youtu.be/9gVlJFajaxc>
- **기획 의도**

- 전투를 시작하면 자동으로 캐릭터가 싸우는 게임
- 액션 및 타격감을 느낄 수 있는 게임
- 3D 카툰 렌더링 게임
- 각 캐릭터 별로 속성과 스테이터스가 존재
- 크리티컬 공격 시 강조 효과
- 속성에 따라 상성이 있어, 상성에 따른 데미지 보너스
- 드래그 & 드랍으로 3x3 그리드에 캐릭터 배치
- 전투 시 캐릭터의 런타임 스테이터스에 따라 공격 우선 순위 설정
- 캐릭터는 여러 개의 스킬 보유 및 전투 상황에서 가장 효율적인 스킬 선택
- 각 스킬은 스킬 범위를 가지고 있어, 지정한 타겟을 기준으로 범위에 스킬 수행
- 상대가 전부 쓰러지면 게임에서 승리
- 주요 개발 항목
 - Behavior Tree 를 이용한 자동 전투 AI 구현
 - 동적 스킬 및 타겟 결정 시스템 구현
 - Shader 를 이용한 Health Bar 구현
 - 액션 및 타격감을 위한 카메라 시스템

7.2 Behavior Tree 를 이용한 자동 전투 AI 구현

7.2.1 Behavior Tree 구조

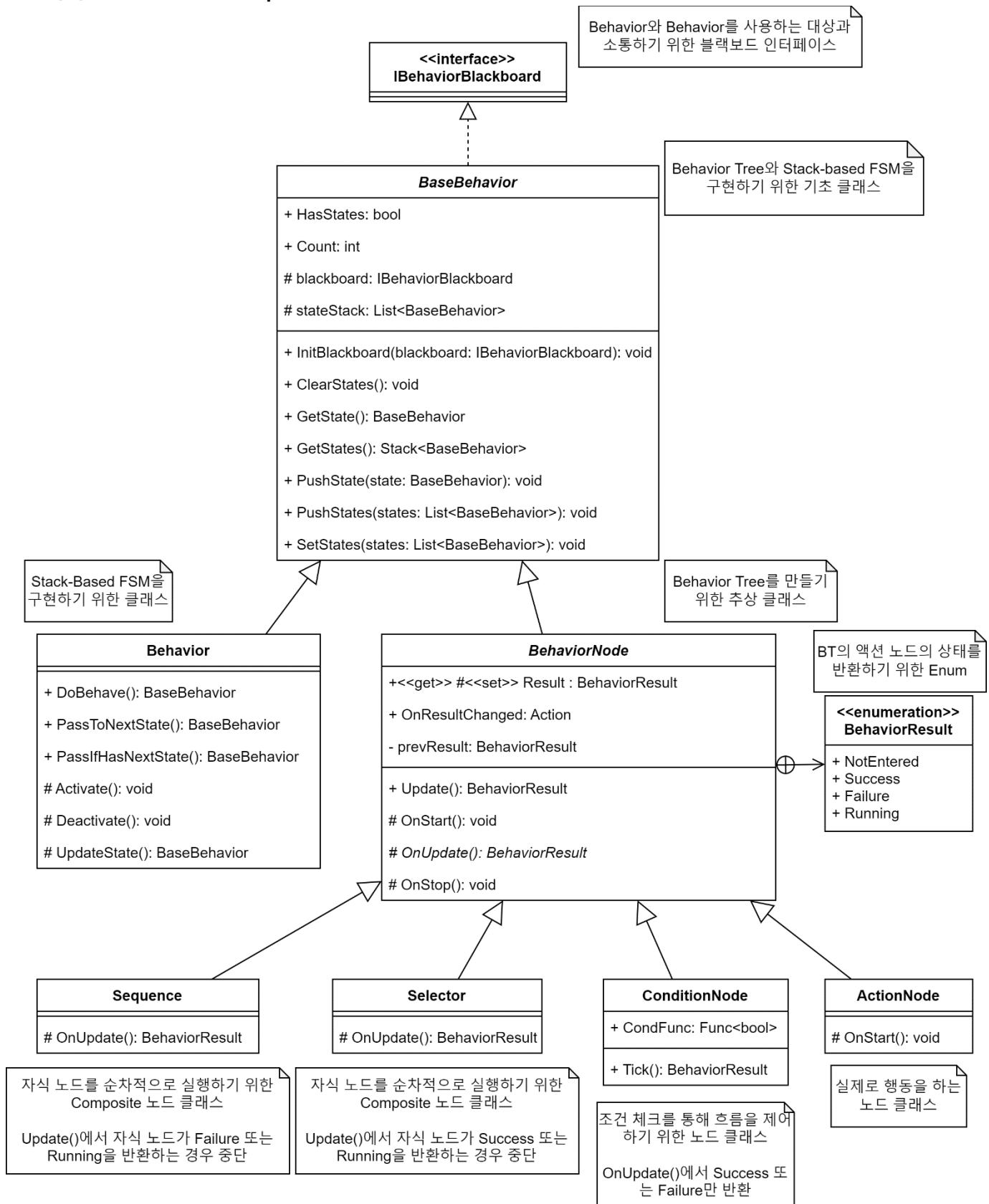


그림 8 Behavior Tree 클래스 다이어그램

- Behavior Tree 를 구성하는 클래스는 위와 같이 제작됨

```

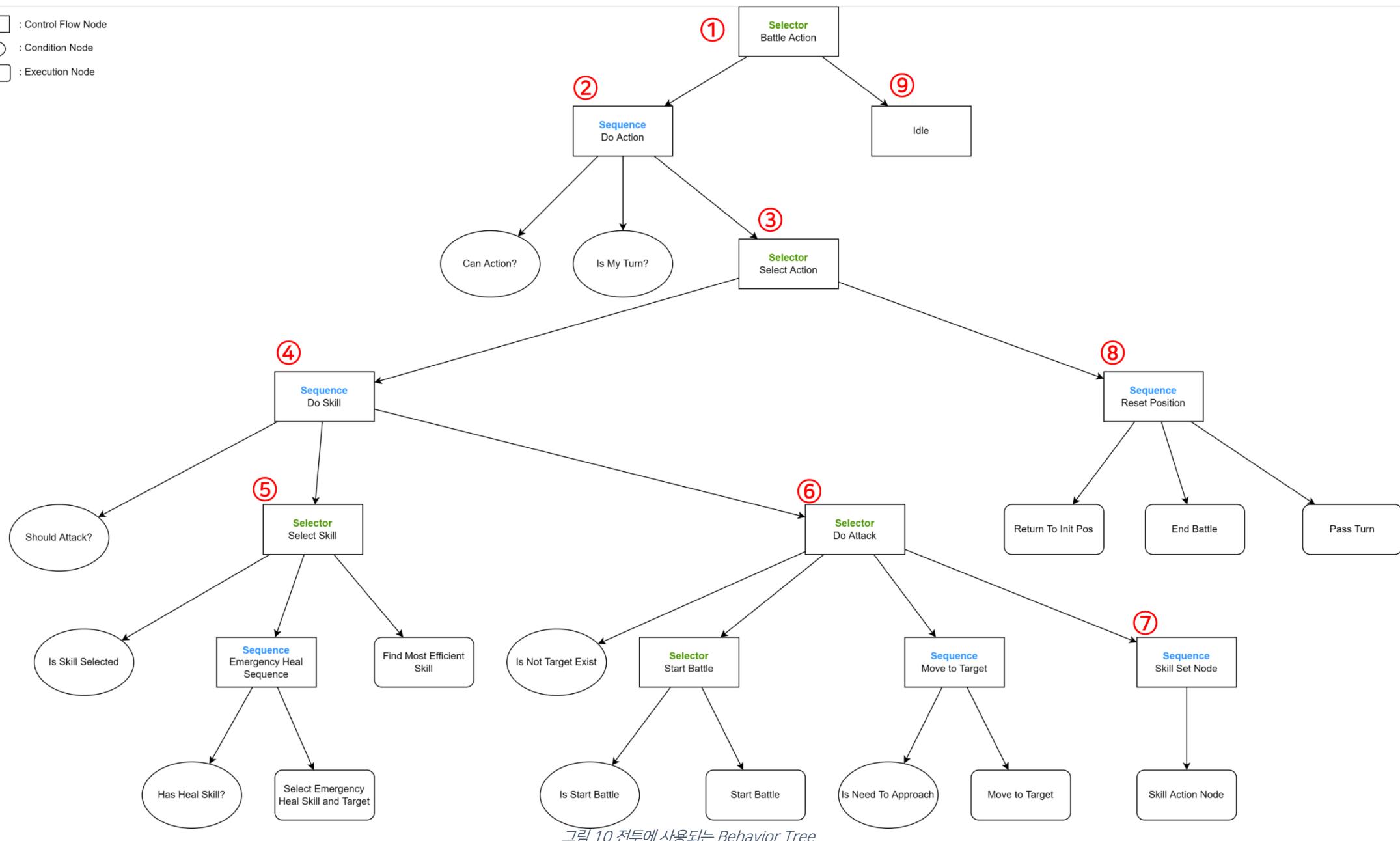
1  using System.Collections.Generic;
2  using System;
3
4  namespace ArshesSH.BehaviorTree
5  {
6      public abstract class BehaviorNode : BaseBehavior
7      {
8          public enum BehaviorResult
9          {
10              NotEntered,
11              Success,
12              Failure,
13              Running,
14          }
15          public BehaviorResult Result { get; protected set; } = BehaviorResult.NotEntered;
16          public bool IsDebug { get; protected set; }
17          public uint Level { get; protected set; }
18          public uint Order { get; protected set; }
19          public event Action OnResultChanged;
20
21          bool isStarted;
22          BehaviorResult prevResult = BehaviorResult.NotEntered;
23
24          public BehaviorNode( IBehaviorBlackboard blackboard, List<BaseBehavior> successors = null )
25          :
26          base( successors )
27          {
28              Blackboard = blackboard;
29          }
30
31          public BehaviorResult Update()
32          {
33              if(Blackboard == null)
34              {
35                  Debug.Debugger.LogError( $"Blackboard was not init in {GetType().Name}" );
36                  return BehaviorResult.Failure;
37              }
38              if(!isStarted)
39              {
40                  OnStart();
41                  isStarted = true;
42              }
43
44              Result = OnUpdate();
45
46              if(Result != BehaviorResult.Running)
47              {
48                  OnStop();
49                  isStarted = false;
50              }
51              if(prevResult != Result)
52              {
53                  OnResultChanged?.Invoke();
54                  prevResult = Result;
55              }
56              return Result;
57          }
58          protected virtual void OnStart() { }
59          protected abstract BehaviorResult OnUpdate();
60          protected virtual void OnStop() { }
61      }
62  }

```

- Behavior Tree 의 노드의 기초 클래스인 BehaviorNode 는 위와 같이 구성됨
- 행동 트리의 노드들은 BehaviorNode 를 상속받아서 구현
- 노드는 OnStart(), OnUpdate(), OnStop()의 흐름으로 구성
- OnStart()는 해당 노드에 처음 접근했을 때에만 호출, 하위 클래스에서는 노드가 처음 시작될 때 수행할 행동을 구현할 수 있음
- OnUpdate()는 노드에서 수행해야 할 작업을 처리하는 함수로 노드의 결과를 반환, 모든 노드는 작업을 처리해야함으로 추상 함수로 구성
- OnStop()은 해당 노드가 Running 이 아닌 상태를 반환할 때 한 번만 호출, 하위 클래스에서는 노드가 끝날 때 수행할 행동을 구현할 수 있음

7.2.2 Behavior Tree 흐름

- : Control Flow Node
- : Condition Node
- : Execution Node



- 각 캐릭터는 전투를 수행할 수 있는 Behavior Tree를 가지고 있으며, Update()에서 Behavior Tree를 갱신함
- Behavior Tree를 순회하면서 해당 조건에 맞는 행동을 수행
- 위 Behavior Tree의 흐름은 다음과 같음

- 전투를 수행하기 위한 Battle Behavior Tree의 root는 Selector로 구성

```

1 public class BattleBT : Selector
2 {
3     public BattleBT( IBehaviorBlackboard blackboard ) : base( blackboard )
4     {
5         PushState( new Action_Idle(blackboard) );
6         PushState( new Sequence_DoAction( blackboard ) );
7     }
8 }
```

그림 11 전투를 담당하는 행동 트리 BattleBT 클래스 코드

- 행동을 할 수 있고, 자신의 차례인 경우 수행할 액션을 선택
- 스킬을 수행하고, 스킬 수행이 끝나면 자신의 자리로 돌아가도록 Selector 구성
- 공격을 할 수 있다면 스킬을 선택하고, 이후 공격 수행

5. 스킬이 한 번만 선택될 수 있도록 조건을 처리하며, 가장 효율적인 스킬을 선택하도록 구성
응급 상황에 놓인 아군이 존재하는 경우 해당 아군에 대한 힐 처리

그림 12 가장 효율적인 스킬을 찾는 Action_FindEfficientSkill 클래스 코드

```

1  using System.Collections.Generic;
2  using ArshesSH.BehaviorTree;
3
4  namespace ArshesSH.Test.GridBattle
5  {
6      public class Action_FindEfficientSkill : ActionNode
7      {
8          readonly ICharacterBlackboard board;
9
10         public Action_FindEfficientSkill( IBehaviorBlackboard blackboard ) : base( blackboard )
11         {
12             board = Blackboard as ICharacterBlackboard;
13         }
14
15         protected override BehaviorResult OnUpdate()
16         {
17             if(!FindMostEfficientSkillAndTarget())
18             {
19                 return BehaviorResult.Failure;
20             }
21             board.Actor.IsSkillSelected = true;
22             return BehaviorResult.Success;
23         }
24
25         bool FindMostEfficientSkillAndTarget()
26         {
27             int finalEfficiency = 0;
28             bool result = false;
29
30             var skillSets = board.Actor.SlotOfSkills.SkillSets;
31             for (var skillIdx = 0; skillIdx < skillSets.Count; ++skillIdx )
32             {
33                 SkillSet curSkill = skillSets[skillIdx];
34                 var targetSet = board.Actor.SelectTargetSetBySkill( curSkill );
35
36                 for (var characterIdx = 0; characterIdx < targetSet.Count; ++characterIdx)
37                 {
38                     Character target = targetSet[characterIdx];
39                     if (!board.Actor.CanApproachTarget(curSkill, target)) continue;
40
41                     int skillEfficiency = curSkill.GetEfficiency(board.Actor, target, board.Handler);
42                     if (finalEfficiency >= skillEfficiency) continue;
43
44                     finalEfficiency = skillEfficiency;
45                     board.Actor.SelectedSkillIdx = skillIdx;
46                     board.Actor.SelectedTargetGridPos = target.GridPos;
47                     board.Actor.SelectedTargetFaction = target.Factions;
48                     result = true;
49                 }
50             }
51             return result;
52         }
53     }
54 }
```

6. 게임의 흐름을 관리하는 Mediator 인 Battle Handler에게 전투의 시작을 알리고, 스킬 대상에게 스킬 사거리만큼 이동 및 스킬을 사용하도록 구성

7. 자신이 선택한 스킬에 따라 동적으로 스킬 노드 생성 및 수행

```
1  using System.Collections.Generic;
2  using ArshesSH.BehaviorTree;
3
4  namespace ArshesSH.Test.GridBattle
5  {
6      public class SkillSetNode : Sequence
7      {
8          ICharacterBlackboard board;
9          IBTContext_Target targetContext;
10
11         public SkillSetNode(IBehaviorBlackboard blackboard, IBTContext_Target targetContext) : base(blackboard)
12         {
13             board = blackboard as ICharacterBlackboard;
14             this.targetContext = targetContext;
15         }
16
17         protected override void OnStart()
18         {
19             base.OnStart();
20
21             var skillSet = board.Actor.GetSelectedSkillSet;
22
23             for (int i = skillSet.Skills.Length - 1; i >= 0; --i)
24             {
25                 stateStack.Add( new SkillActionNode(Blackboard, skillSet.Skills[i], targetContext) );
26             }
27
28             if(board.Actor.GetSelectedSkillSet.HasHealingSkill) return;
29             targetContext.Target.transform.LookAt(board.Actor.transform.position);
30         }
31
32         protected override void OnStop()
33         {
34             targetContext.ResetTarget();
35             board.Actor.ShouldAttack = false;
36             stateStack.Clear();
37             base.OnStop();
38         }
39     }
40 }
41 }
```

그림 13 SkillSetNode 클래스 코드

8. 자신의 초기 위치로 복귀하는 절차

9. 자신의 차례가 아닌 경우 대기 상태 유지

7.2.3 Behavior Tree Debugger



그림 14 Behavior Tree Debugger의 인게임 모습

ObjectPool을 사용하고 등록하기 위해 참조

Pool에 등록할 UI 데이터

디버깅 할 BT의 루트

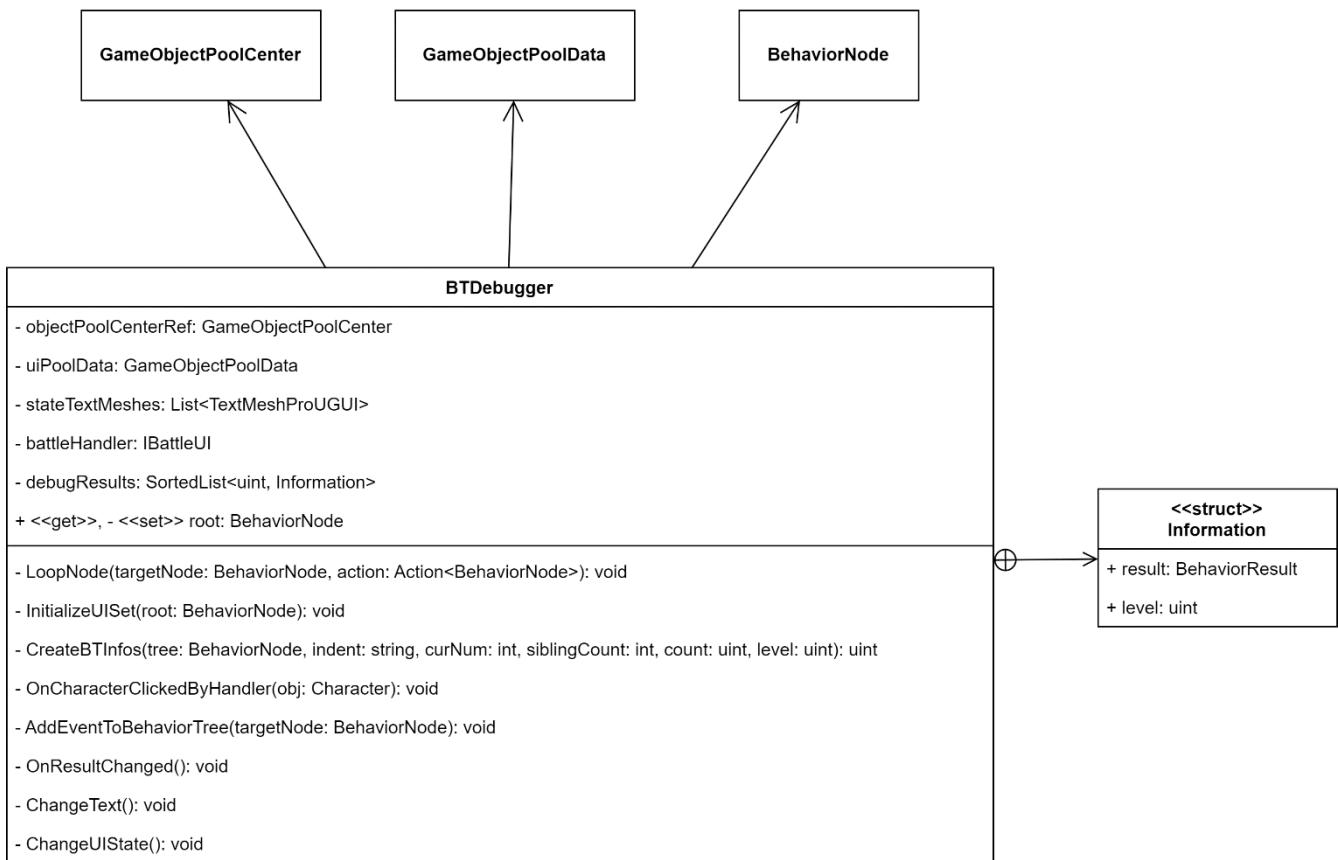


그림 15 BTDebugger 클래스 디자인 그램

- Object Pool 을 이용해 추적할 행동 트리의 노드 수만큼 UI 생성
- Root 를 통해 노드를 전체 순회하면서 각 노드의 반환 값을 저장

7.3 동적 스킬 및 타겟 결정 시스템 구현

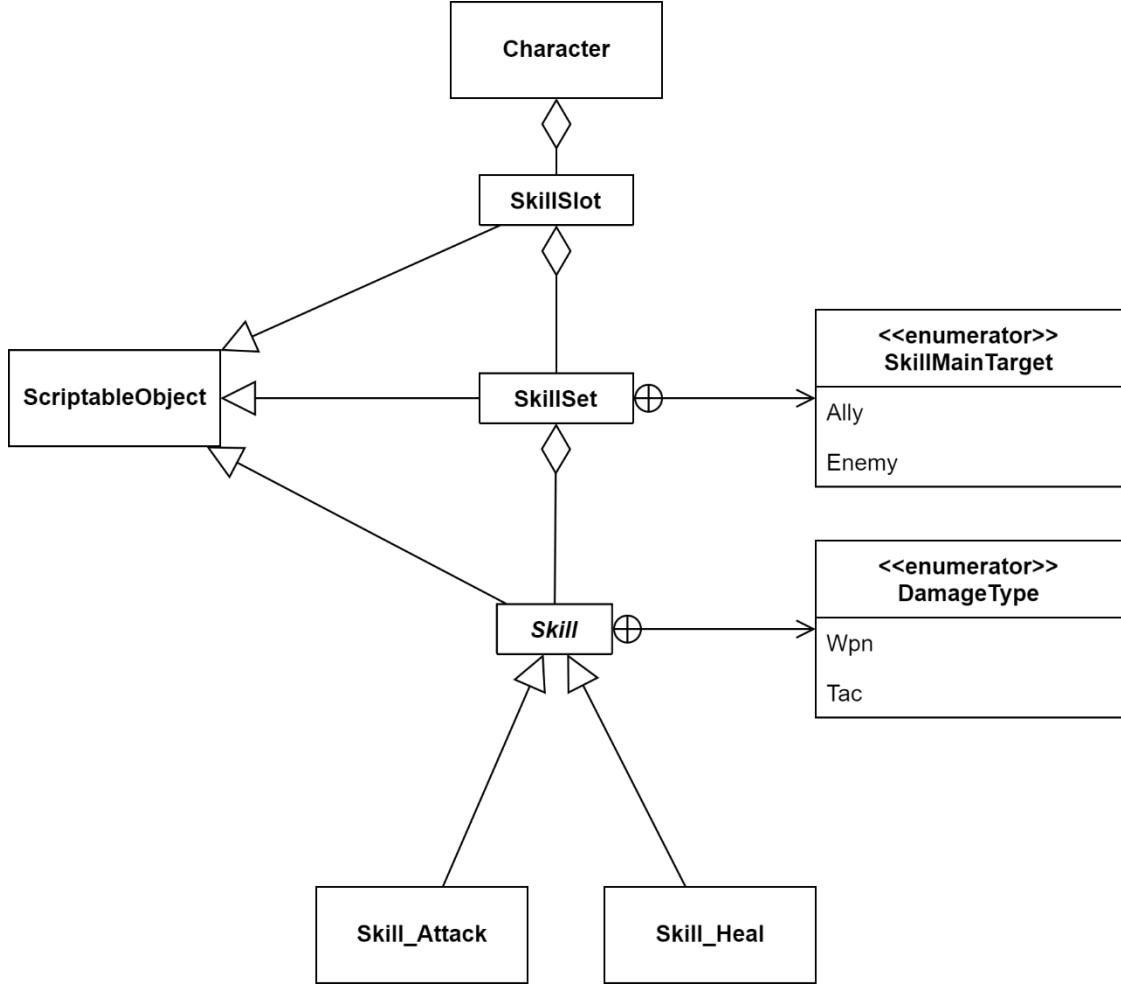


그림 16 Skill 클래스 다이어그램

7.3.1 스킬 구성

- 캐릭터(Character)는 스킬 슬롯(SkillSlot)을 가지고 있으며, 스킬 슬롯에는 여러 개의 스킬 조합(SkillSet)이 들어갈 수 있음
- 스킬 조합은 여러 개의 스킬로 구성되어 있음
- 스킬에는 데미지, 공격 범위, 애니메이션, 스킬 이펙트 등 스킬을 구성하는 데이터가 존재
- 전투 시 Behavior Tree를 통해 사용할 스킬 조합을 선택하고, 선택된 스킬 조합을 기반으로 SkillSetNode에서 스킬의 수만큼 SkillActionNode를 생성
- Behavior Tree 상에서 SkillActionNode의 상위 노드인 SkillSetNode는 Sequence 노드를 상속받기 때문에, 하위 노드의 SkillActionNode들을 순차적으로 실행
- 각 SkillActionNode은 선택한 스킬의 애니메이션 및 효과를 처리함

```

1  using System.Collections.Generic;
2  using UnityEngine;
3
4  namespace ArshesSH.Test.GridBattle
5  {
6      [CreateAssetMenu( fileName = "SkillSlot", menuName = "ScriptableObjects/GridBattle/Skill/Skill Slot" )]
7      public class SkillSlot : ScriptableObject
8      {
9          [field: SerializeField] public List<SkillSet> SkillSets { get; private set; }
10         public bool IsSortedByPureDamage { get; private set; }
11
12         int totalPureDamage;
13         bool isPureDamageSet;
14         void OnDisable()
15         {
16             IsSortedByPureDamage = false;
17             isPureDamageSet = false;
18         }
19         public int GetSkillsPredictedDamage(Character owner)
20         {
21             if ( !IsSortedByPureDamage) SortSkillAsPureDamage(owner);
22             return isPureDamageSet ? totalPureDamage : totalPureDamage = SkillSets[0].CalcSkillsPureDamage(owner);
23         }
24         void SortSkillAsPureDamage( Character owner )
25         {
26             SkillSets.Sort( ( lhs, rhs ) => lhs.CalcSkillsPureDamage(owner).CompareTo( rhs.CalcSkillsPureDamage(owner) ) );
27             IsSortedByPureDamage = true;
28         }
29     }
30 }

```

그림 17 SkillSlot 클래스 코드

```

1  using System.Collections.Generic;
2  using UnityEngine;
3
4  namespace ArshesSH.Test.GridBattle
5  {
6      [CreateAssetMenu( fileName = "SkillSet", menuName = "ScriptableObjects/GridBattle/Skill/SkillSet" )]
7      public class SkillSet : ScriptableObject
8      {
9          public enum SkillMainTarget { Ally, Enemy };
10         [field: SerializeField] public int GridRange { get; protected set; }
11         [field: SerializeField] public float MoveRange { get; protected set; }
12         [field: SerializeField] public bool HasHealingSkill { get; protected set; }
13         [field: SerializeField] public SkillMainTarget Target { get; protected set; }
14
15         public Skill[] Skills;
16         [HideInInspector] public bool IsSkillAccessible => Skills != null;
17
18         public int CalcSkillsPureDamage(Character owner)
19         {
20             int totalPure = 0;
21             for (int i = 0; i < Skills.Length; ++i)
22             {
23                 if(Skills[i] is Skill_Attack)
24                     totalPure += Skills[i].GetPureDamage(owner);
25             }
26             return totalPure;
27         }
28         public int GetHealAmount(Character owner)
29         {
30             int healAmount = 0;
31             for (int i = 0; i < Skills.Length; ++i)
32             {
33                 if(Skills[i] is Skill_Heal)
34                     healAmount += Skills[i].GetPureDamage(owner);
35             }
36             return healAmount;
37         }
38
39         public int GetEfficiency(Character owner, Character target, IBattleHandler handler)
40         {
41             int efficiency = 0;
42             for (int i = 0; i < Skills.Length; ++i)
43             {
44                 efficiency += Skills[i].GetEfficiency(owner, target, handler);
45             }
46             return efficiency;
47         }
48
49         public HashSet<GridVec> GetSkillTargetGrids(GridVec targetPos, Faction faction)
50         {
51             HashSet<GridVec> targetGrids = new HashSet<GridVec>();
52
53             foreach (var skill in Skills)
54             {
55                 HashSet<GridVec> skillGrids = skill.GetSkillTargetPositions(targetPos, faction);
56                 foreach (var gridVec in skillGrids)
57                 {
58                     targetGrids.Add(gridVec);
59                 }
60             }
61             return targetGrids;
62         }
63     }
64 }

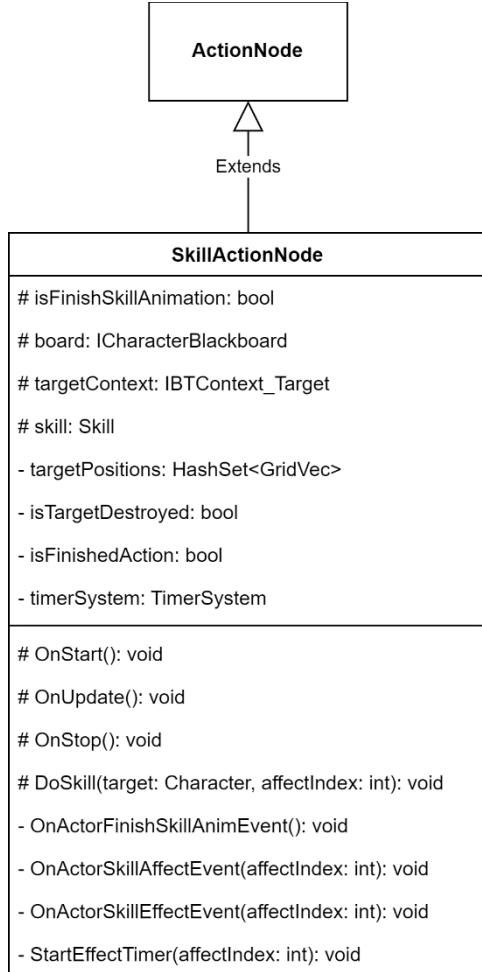
```

그림 18 SkillSet 클래스 코드

7.3.2 스킬 및 타겟 결정 기준

1. 자신이 회복 스킬을 가진 경우, 아군이 죽을 것으로 판단 시 해당 아군을 대상으로 설정 및 회복 스킬 사용 준비
2. 만약 회복을 해도 아군이 죽을 것으로 판단되면, 회복을 선택하지 않고 다음 노드로 이동해 가장 효율적인 스킬 탐색 진행
3. 회복 스킬이 없는 경우, 자신이 가진 스킬 중 가장 효율적인 스킬 탐색 진행
4. 가장 효율적인 스킬은 HP를 변화량이 가장 많은 스킬을 기준으로 함

7.3.3 SkillActionNode



- 해당 노드에 들어오면 선정한 스킬의 애니메이션을 실행하고, 끝날 때 Successor를 반환하도록 구성
- 애니메이션에서 효과, 공격 적용, 종료 이벤트가 발생 시 이벤트 처리

그림 19 SkillActionNode 클래스 다이어그램

```

1  using System.Collections.Generic;
2  using ArshesSH.BehaviorTree;
3  using ArshesSH.Utilities.Timer;
4  using UnityEngine;
5
6  namespace ArshesSH.Test.GridBattle
7  {
8      public class SkillActionNode : ActionNode
9      {
10         const string UseSkillStr = "DefaultSkill";
11         const string AffectedBySkillStr = "DefaultAffected";
12         readonly ICharacterBlackboard board;
13         readonly IBTContext_Target targetContext;
14         readonly Skill skill;
15         HashSet<GridVec> targetPositions;
16         bool isFinishSkillAnimation;
17         bool isTargetDestroyed;
18         bool isFinishedAction;
19         TimerSystem timerSystem;
20         public SkillActionNode(IBehaviorBlackboard blackboard, Skill skill, IBTContext_Target targetContext) : base(blackboard)
21         {
22             board = blackboard as ICharacterBlackboard;
23             if (board == null) return;
24             this.skill = skill;
25             this.targetContext = targetContext;
26             timerSystem = new TimerSystem();
27         }
28         protected override void OnStart()
29         {
30             if(isFinishedAction) return;
31
32             base.OnStart();
33             if (targetContext.Target == null)
34             {
35                 isTargetDestroyed = true;
36                 return;
37             }
38             board.Actor.SkillAffectEvent += OnActorSkillAffectEvent;
39             board.Actor.FinishSkillAnimEvent += OnActorFinishSkillAnimEvent;
40             board.Actor.SkillEffectEvent += OnActorSkillEffectEvent;
41
42             targetPositions = skill.GetSkillTargetPositions(targetContext.Target.GridPos, targetContext.Target.Factions);
43             board.Actor.ChangeAnimationClipTo(UseSkillStr, skill.ActorAnimClip);
44             board.Actor.PlayAttackAnimation();
45         }
46         protected override BehaviorResult OnUpdate()
47         {
48             if (isTargetDestroyed) return BehaviorResult.Failure;
49             if (isFinishedAction) return BehaviorResult.Success;
50             return isFinishSkillAnimation ? BehaviorResult.Success : BehaviorResult.Running;
51         }
52         protected override void OnStop()
53         {
54             if(isFinishedAction) return;
55
56             isFinishSkillAnimation = false;
57             board.Actor.SkillAffectEvent -= OnActorSkillAffectEvent;
58             board.Actor.FinishSkillAnimEvent -= OnActorFinishSkillAnimEvent;
59             board.Actor.SkillEffectEvent -= OnActorSkillEffectEvent;
60             isFinishedAction = true;
61
62             base.OnStop();
63         }
64         protected virtual void DoSkill(Character target, int affectIndex)
65         {
66             SkillAffectData affectData = skill.DataSkillAffect[affectIndex];
67             if (!affectData.IsNotDamagedAffect) return;
68             int damage = skill.GetHealthChangeAmount(board.Actor, targetContext.Target, board.Handler, affectIndex);
69             target.HandleHealth(damage);
70
71             Color targetColor = skill.IsCritical ? affectData.CriticalDamageColor : affectData.NormalDamageColor;
72
73             board.Handler.DoFloatingDamage(target.transform.position, target.transform.rotation, Mathf.Abs(damage),
74                 targetColor);
75             target.ShineSystem.ActivateShine(affectData.ShineSpeed, affectData.AffectShineColor);
76         }
77         void StartEffectTimer(EffectPoolData effectData, Vector3 position, Quaternion rotation)
78         {
79             var effectLifeTimer = board.Handler.ObjectPoolCenter.GetObjectFromPool<LifeTimer>(effectData.Tag,
80                 position,
81                 rotation,
82                 effectData.IsDynamicPool
83             );
84             if(effectLifeTimer == null) {Debug.Debugger.LogError("Put LifeTime Class to effect prefab!");}
85             effectLifeTimer.SetObjectPoolCenter(board.Handler.ObjectPoolCenter);
86             effectLifeTimer.StartLifeTimer();
87         }

```

```

88     void OnActorFinishSkillAnimEvent()
89     {
90         board.Actor.ResetAnimationSpeed();
91         isFinishSkillAnimation = true;
92     }
93     void OnActorSkillAffectEvent(int affectIndex)
94     {
95         foreach (GridVec targetPos in targetPositions)
96         {
97             string prevAnimName = AffectedBySkillStr;
98             if(!board.Handler.TryFindTarget(targetContext.Target.Factions, targetPos, out Character curTarget)) return;
99
100            if ( skill.DataSkillAffect[affectIndex].AffectAnimation != null )
101            {
102                curTarget.ChangeAnimationClipTo(prevAnimName, skill.DataSkillAffect[affectIndex].AffectAnimation);
103                prevAnimName = skill.DataSkillAffect[affectIndex].AffectAnimation.name;
104                curTarget.PlayAffectedAnimation();
105            }
106            DoSkill(curTarget, affectIndex);
107
108            if(skill.DataSkillAffect[affectIndex].CameraShakeData != null)
109                board.Handler.ShakeCamera( skill.DataSkillAffect[affectIndex].CameraShakeData );
110            if(skill.DataSkillAffect[affectIndex].HitStopData != null && skill.IsCritical)
111                board.Handler.DoHitStop(skill.DataSkillAffect[affectIndex].HitStopData);
112        }
113    }
114    void OnActorSkillEffectEvent(int index)
115    {
116        if ( skill.EffectData == null || skill.EffectData.Length <= 0) return;
117        if ( index >= skill.EffectData.Length ) return;
118        EffectPoolData effectData = skill.EffectData[index];
119        board.Handler.ObjectPoolCenter.InitializePool(effectData);
120        Transform baseTransform =
121            (effectData.IsSpawnToTarget) ? targetContext.Target.transform : board.Actor.transform;
122        if ( effectData.IsSpawnToTarget )
123        {
124            foreach ( var gridVec in skill.GetSkillTargetPositions( targetContext.Target.GridPos, targetContext.Target.Factions ) )
125            {
126                if ( board.Handler.TryFindTarget( targetContext.Target.Factions, gridVec, out var eachTarget ) )
127                {
128                    Quaternion rotation = Quaternion.Euler(eachTarget.transform.rotation.eulerAngles + effectData.EffectRotation);
129                    Vector3 position = eachTarget.transform.position + rotation * effectData.EffectPosition ;
130
131                    StartEffectTimer(effectData, position, rotation);
132                }
133            }
134        }
135        else
136        {
137            Quaternion rotation = Quaternion.Euler(baseTransform.rotation.eulerAngles + effectData.EffectRotation);
138            Vector3 position = baseTransform.position + rotation * effectData.EffectPosition ;
139            StartEffectTimer( effectData, position, rotation );
140        }
141    }
142 }
143 }
144

```

그림 20 SkillActionNode 클래스 다이어그램

- 스킬의 데미지 처리, 스킬의 애니메이션, 스킬의 이펙트는 모두 해당 노드에서 이벤트가 발생 시 처리함
- 위 이벤트들은 애니메이션에서 등록
- 스킬 애니메이션에 종료 이벤트를 등록해, 종료 시점 시 SkillActionNode 를 탈출하도록 구성

7.4 Shader 를 이용한 Health Bar 구현

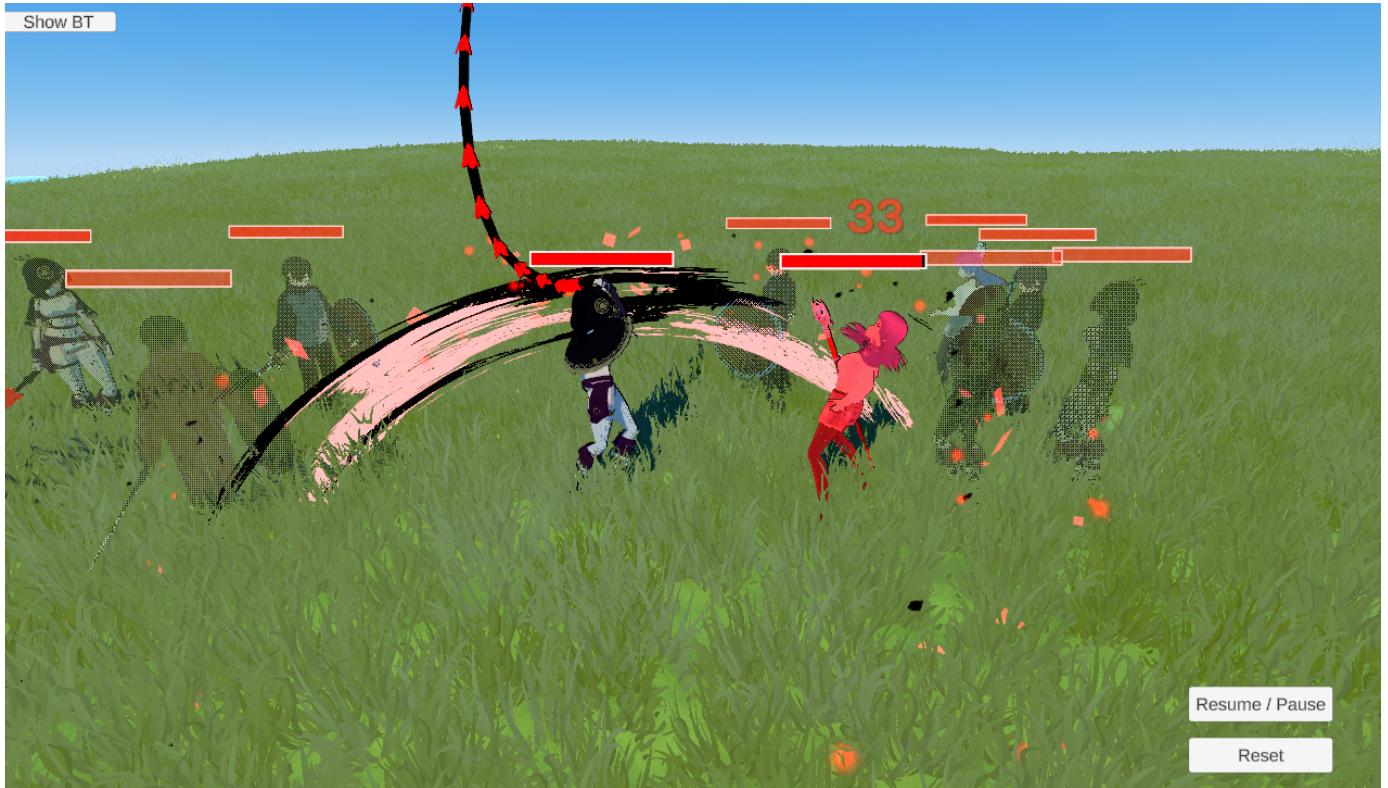


그림 21 Health Bar 이미지

- UGUI를 이용해서 Health Bar 를 구현하는 경우 각 캐릭터의 Health Bar 하나 당 Draw Call 발생
- 캐릭터가 한 게임에 최대 18 개까지 존재할 수 있으므로, Draw Call 이 18 개 증가할 수 있음
- 불필요한 Draw Call 을 줄이기 위해 Shader 를 이용하여 Health Bar 제작
- 카메라 거리에 따라 Health Bar 의 Alpha 값을 조절하기 위해 Shader 에 Near Fade Dithering 옵션 추가

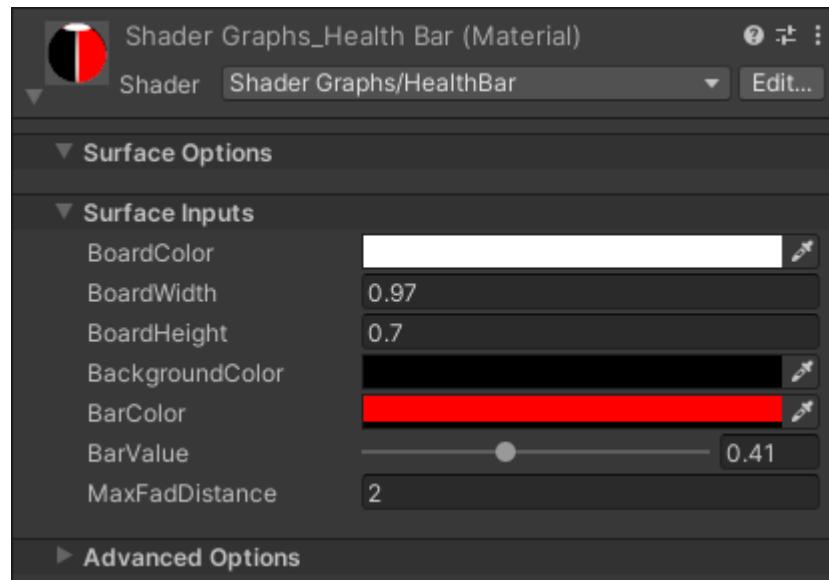


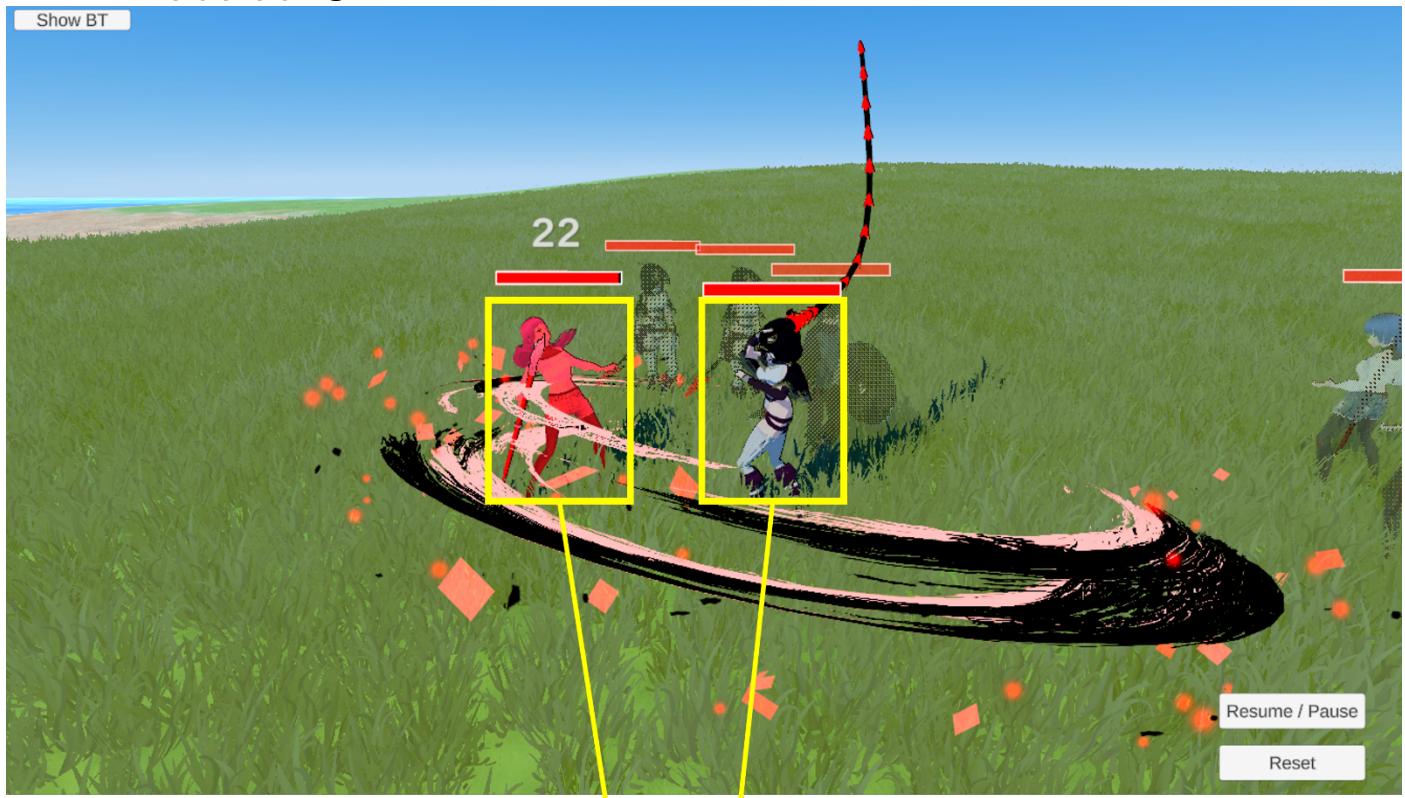
그림 22 Shader 옵션 이미지



그림 23 Health Bar Shader Graph

7.5 액션 및 타격감을 위한 카메라 처리

7.5.1 카메라 위치 조정



추적할 Transform들을 기준으로 경계 계산 후, 카메라 위치 변경

그림 24 전투 상태의 캐릭터들을 추적하는 카메라 모습

- 전투중인 캐릭터들을 대상으로 선정하여 화면의 범위를 계산하여 움직이는 기능
- 최소거리와 최대거리를 입력 받아 카메라 범위 조절
- 카메라 중심에 대한 위치 및 회전에 대한 오프셋 제공

7.5.2 카메라 흔들림

- 캐릭터 공격 시 카메라 흔들림 적용
- 흔들리는 정도는 Scriptable Object로 만든 CameraShakeData를 통해 미리 설정 가능
- 흔들리는 방식은 Random, Perlin-Noise, Sinusoidal 방식 제공

```

1  using System.Collections;
2  using UnityEngine;
3  using Random = UnityEngine.Random;
4
5  namespace ArshesSH.Utilities
6  {
7      public class CameraShake : MonoBehaviour
8      {
9          public enum ShakeMode
10         {
11             Random,
12             Perlin,
13             Sinusoidal
14         }
15         [SerializeField] Transform camTransform;
16         Vector3 originalPos;
17         Vector3 shakePos;
18         float shakeTime = 0f;
19         void Awake()
20         {
21             if (camTransform != null) return;
22             if (Camera.main != null) camTransform = Camera.main.transform;
23         }
24         void OnDisable()
25         {
26             StopAllCoroutines();
27         }
28
29         public void Shake( float shakeAmount, float shakeDuration, ShakeMode shakeMode = ShakeMode.Random,
30                         float decreaseFactor = 1.0f )
31         {
32             StartCoroutine( ShakeCoroutine(shakeAmount, shakeDuration, shakeMode, decreaseFactor) );
33         }
34         public void Shake( CameraShakeData cameraShakeData )
35         {
36             StartCoroutine(
37                 ShakeCoroutine( cameraShakeData.ShakeAmount, cameraShakeData.ShakeDuration, cameraShakeData.ShakeMode,
38                                 cameraShakeData.DecreaseFactor ) );
39         }
40
41         IEnumerator ShakeCoroutine( float shakeAmount, float shakeDuration, ShakeMode shakeMode = ShakeMode.Random,
42                                     float decreaseFactor = 1.0f )
43         {
44             originalPos = camTransform.localPosition;
45             shakePos = originalPos;
46             float duration = shakeDuration;
47             while (duration > 0)
48             {
49                 switch (shakeMode)
50                 {
51                     case ShakeMode.Random:
52                         RandomShake( shakeAmount );
53                         break;
54                     case ShakeMode.Perlin:
55                         PerlinShake( shakeAmount );
56                         break;
57                     case ShakeMode.Sinusoidal:
58                         SinusoidalShake( shakeAmount );
59                         break;
60                 }
61                 duration -= Time.deltaTime * decreaseFactor;
62                 camTransform.localPosition = Vector3.Lerp( camTransform.localPosition, shakePos, Time.deltaTime * 10f );
63                 yield return null;
64             }
65             camTransform.localPosition = originalPos;
66         }
67         void RandomShake( float shakeAmount )
68         {
69             shakePos = originalPos + Random.insideUnitSphere * shakeAmount;
70         }
71         void PerlinShake( float shakeAmount )
72         {
73             shakeTime += Time.deltaTime;
74             float shakeDelta = shakeTime * shakeAmount * 10f;
75
76             float x = Mathf.PerlinNoise( shakeDelta, 0f );
77             float y = Mathf.PerlinNoise( 0f, shakeDelta );
78             shakePos = new Vector3( originalPos.x + x - 0.5f, originalPos.y + y - 0.5f, originalPos.z );
79         }
80         void SinusoidalShake( float shakeAmount )
81         {
82             shakeTime += Time.deltaTime;
83             float x = Mathf.Sin(shakeTime * shakeAmount);
84             float y = Mathf.Sin(shakeTime * shakeAmount);
85             shakePos = new Vector3( originalPos.x + x, originalPos.y + y, originalPos.z );
86         }
87     }
88 }

```

그림 25 CameraShake 클래스 코드

8 Deus Ex Machina

8.1 프로젝트 소개



그림 26 Deus Ex Machina 인게임 장면

<https://youtu.be/p3pPeP9O2TY>

- 프로젝트 기간
2022.09 ~ 2022.12
- 프로젝트 인원
8인 (프로그래머 3인, 기획자 5인)
- 프로젝트 주제
Dead by Daylight 를 모작한 1:4
비대칭 PVP 게임
- 참고 영상:

- 사용 기술
Unity, C#, Google Apps Script
- 기획 의도
 - 소수의 플레이어에게는 강력한 능력치를 주어 적을 사냥하는 재미 부여
 - 다수의 플레이어에게는 어려운 환경에서 협력을 통해 시련을 극복하는 성취의 재미 부여
 - PC 플랫폼에서의 5인 멀티플레이 게임
 - 서바이벌 호러 장르
 - 20분의 플레이 타임
- 게임 설명
 - 퇴마사와 악령이 들린 인형, 두 가지 진형으로 나뉘어 목표를 달성하는 것이 게임의 목표
 - 인형 플레이어들은 퇴마사 플레이어의 공격을 피해, 맵에서 탈출하는 것이 목표
 - 퇴마사 플레이어는 인형 플레이어의 탈출을 저지하는 것이 목표
- 주요 개발 항목
 - 계정 및 네트워크 시스템
 - 플레이어 데이터 매니저
 - 상호작용 시스템
 - 캐스팅 시스템

8.2 계정 및 네트워크 시스템

8.2.1 계정 관리 데이터베이스

Data Table

1	ID	Password	Nickname
2	-	-	-
3	Test1	MTIz	Test1
4	inha1234	MTIzNA==	inha1234
5	Exorcist	MTIzMtIz	Hunter

Log

2022-11-30 16:24	{"order": "login", "result": "OK", "message": "로그인 완료", "index": "17", "id": "Exorcist1234", "nickname": "ExorcistPlayer"}
2022-11-30 16:25	{order=login, id=inha1234, password=1}
2022-11-30 16:25	{"order": "login", "result": "ERROR", "message": "로그인 실패", "index": "17", "id": null, "nickname": "ExorcistPlayer"}
2022-11-30 16:26	{order=register, password=1234, nickname=inha, id=inha1004}
2022-11-30 16:26	{"order": "register", "result": "OK", "message": "회원가입 완료"}
2022-11-30 16:26	{id=inha1004, password=1234, order=register, nickname=1234}

그림 27 Google Sheets 의 데이터 테이블 및 로그 테이블의 모습

- DBMS 를 사용할 수 없는 환경이어서 Google Sheets 로 DB 제작
- 계정 데이터 및 접속 로그 관리 테이블 제작
- 통신 및 데이터 가공은 Google Apps Script 를 이용
- 클라이언트와 DB 와의 통신은 Post 를 통해 이루어짐
- 데이터는 JSON 파일로 가공하여 전달

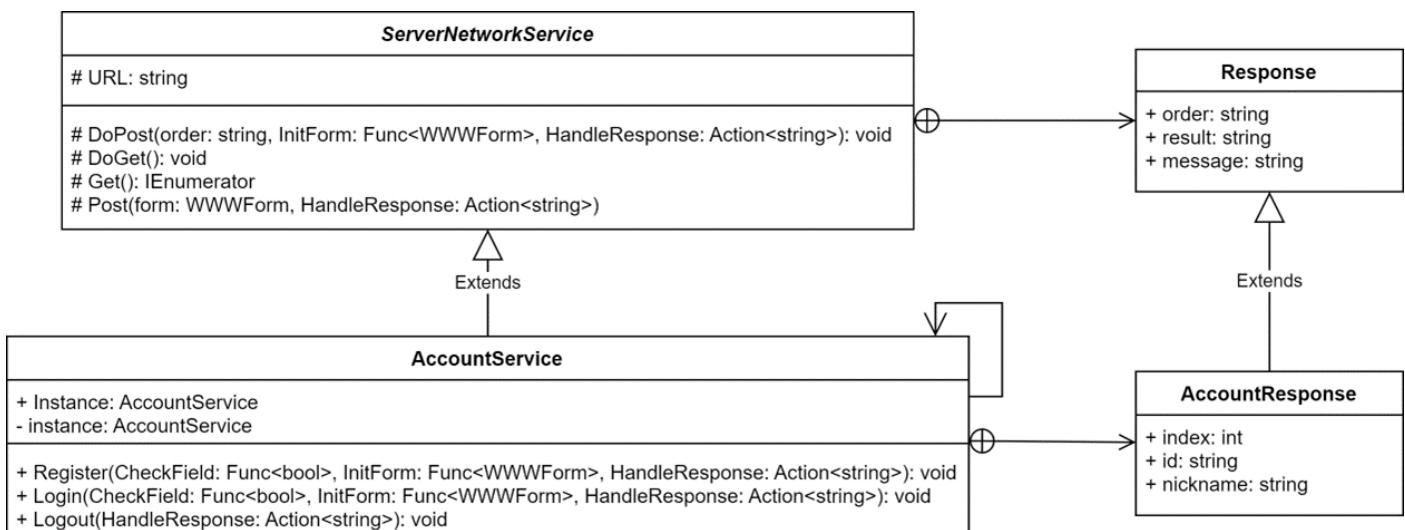


그림 28 DB 네트워크 서비스 클래스 디자인 그램

8.2.2 플레이어 데이터 매니저

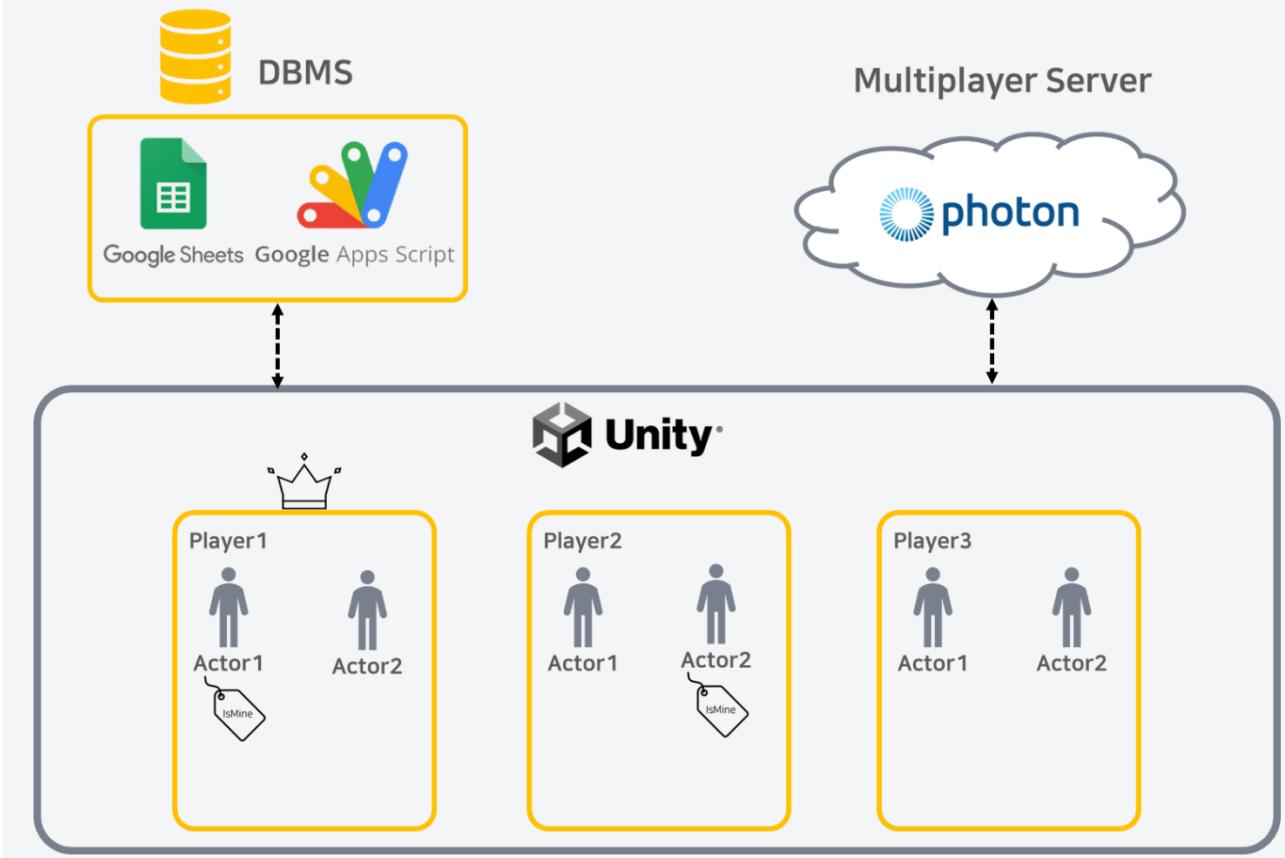


그림 29 게임 데이터 흐름 구성도

- 플레이어의 데이터는 계정 데이터와, 게임 캐릭터의 데이터가 존재
- 계정 데이터는 DB에 저장하며, 게임 캐릭터 데이터는 정보로서 csv 파일로 저장
- 플레이어 데이터 매니저는 게임 시작 전, 위 정보들을 읽어 해당 클라이언트에게 할당
- 다른 클라이언트의 데이터는 플레이어 데이터 매니저를 통해 획득 가능
- 자신이 조종하는 캐릭터의 데이터는 오직 해당 클라이언트에서만 수정 가능

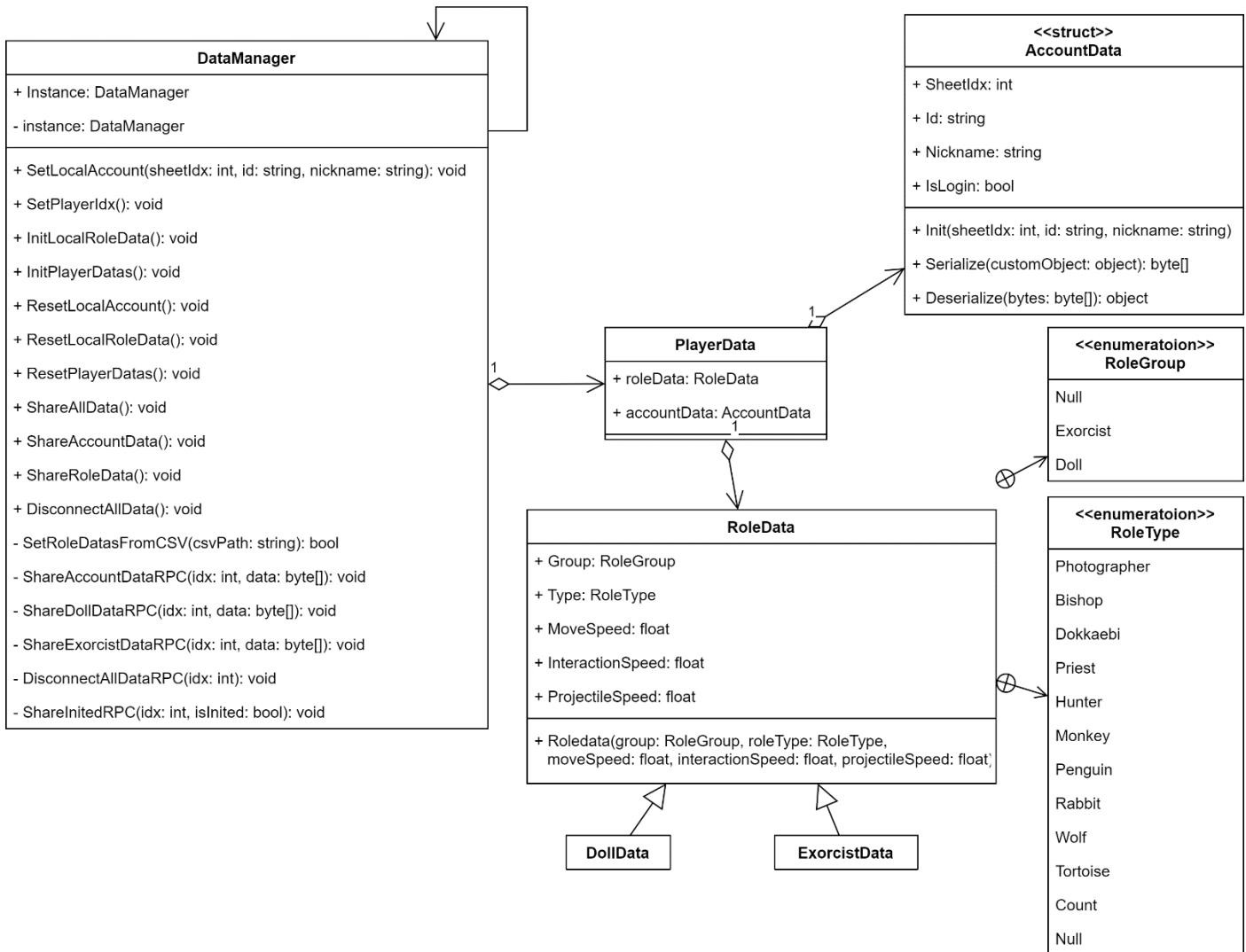
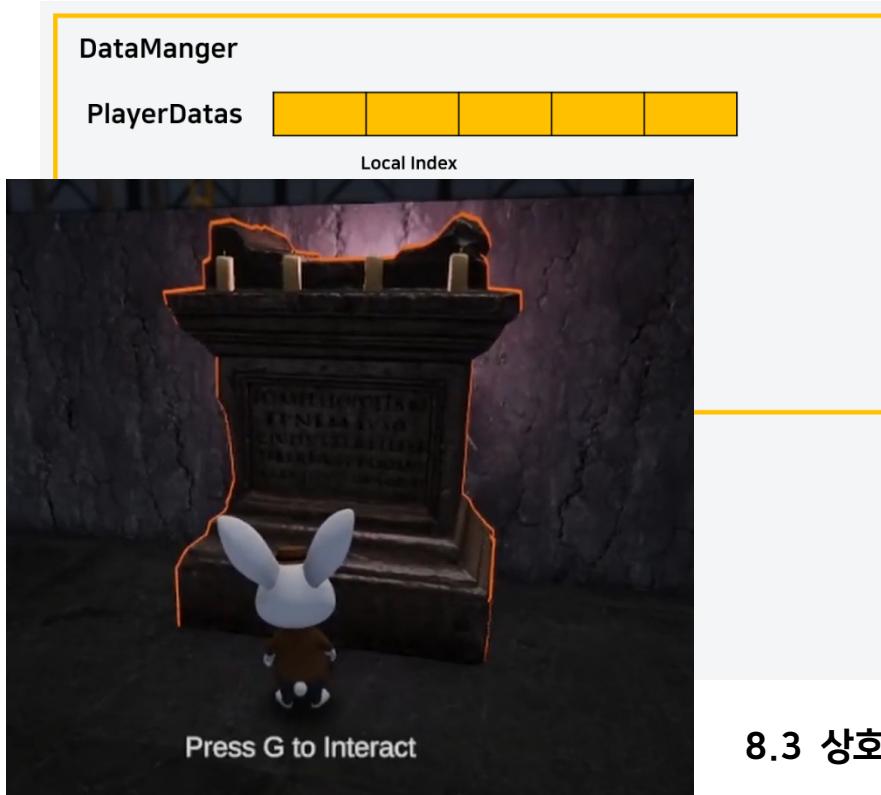


그림 30 `DataManager` 클래스 다이어그램

- `DataManager` 는 Façade 패턴으로 Photon API 를 통한 데이터 공유 과정을 래핑



- 플레이어 데이터를 수정하면 해당 클래스에서 데이터 매니저가 제공하는 인터페이스 함수를 호출

- 해당 인터페이스 함수는 RPC 함수를 호출하여 해당 클라이언트의 수정된 데이터를 다른 클라이언트에 공유

8.3 상호작용 시스템

- 상호작용 가능한 오브젝트와 플레이어가 상호작용하기 위한 시스템
- 상호작용 가능 조건을 만족하는 경우 플레이어에게 상호작용에 관한 설명을 보여줌
- 상호작용 가능할 시 림 라이트 효과로 해당 오브젝트 강조

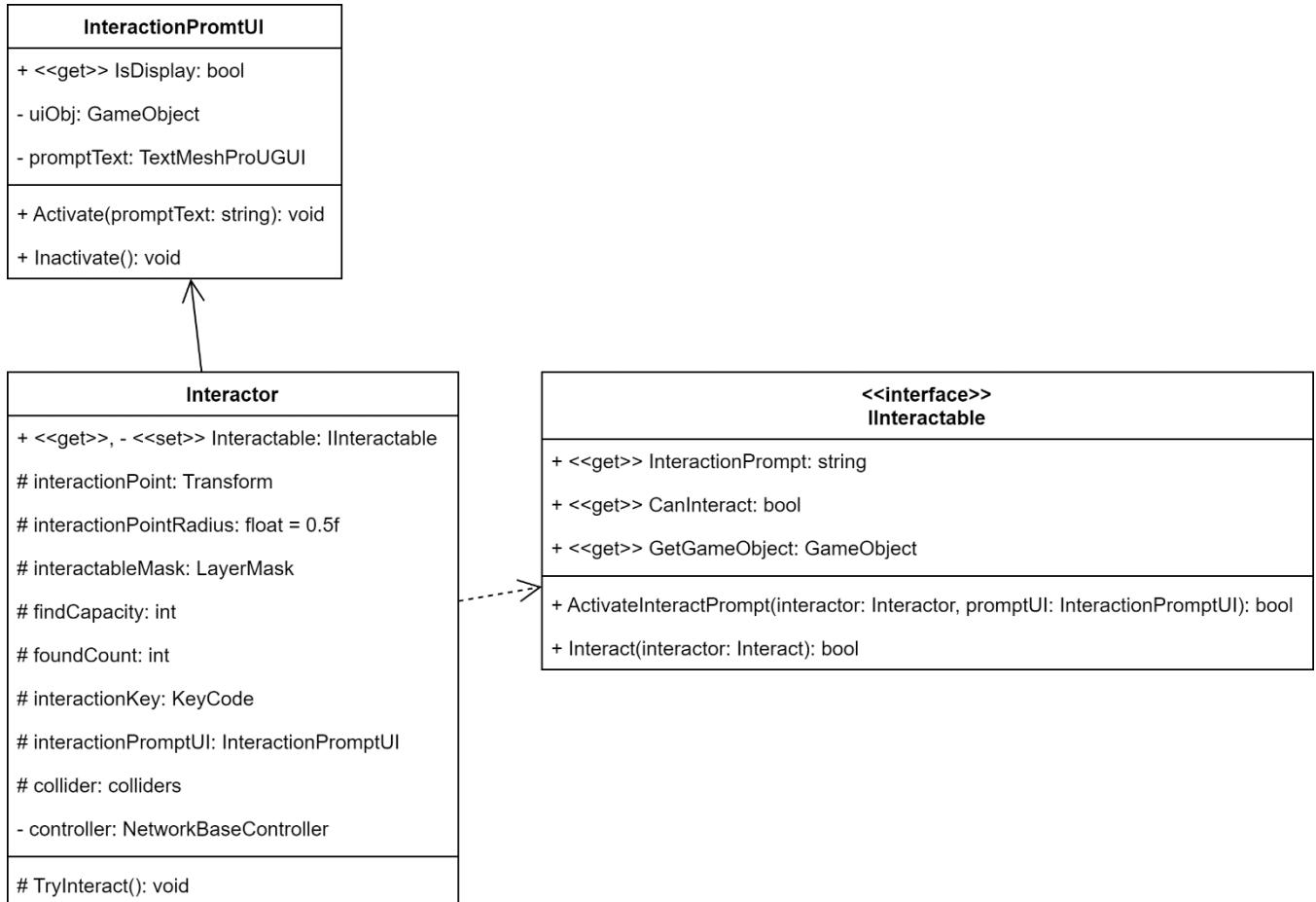


그림 32 Interaction System 클래스 디아그램

- Interactor는 상호작용의 가능 여부를 판단하는 컴포넌트 클래스
- Interactor는 IInteractable을 구현하는 클래스를 찾은 경우 TryInteract()를 호출
- 상호작용이 가능한 경우 InteractionPromptUI에서 상호작용이 가능 메시지 출력

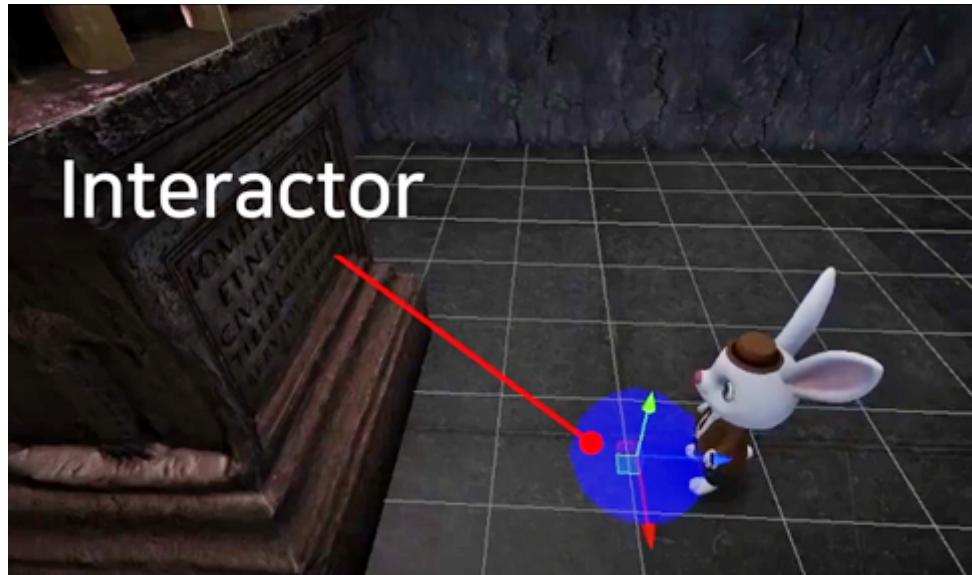


그림 33 Interactor의 범위를 Gizmo를 통해 확인한 모습

- 캐릭터는 상호작용을 위해 Interactor 를 컴포넌트로 소유

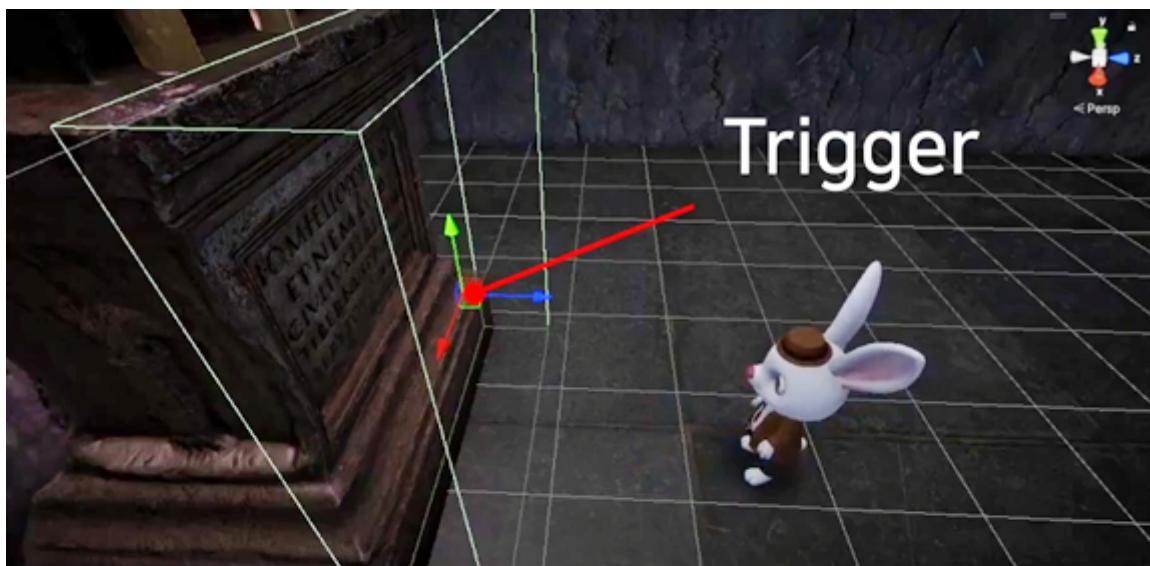


그림 34 상호작용 가능한 오브젝트의 트리거

- Interactor 가 IInteractable 을 구현한 오브젝트의 클래스의 트리거와 충돌하면 상호작용 가능

8.4 캐스팅 시스템



그림 35 캐스팅 시스템의 진행 모습

- 상호작용하는 오브젝트의 충전 값을 목표 값까지 변경시키기 위한 시스템
- 게임의 모든 상호작용에서 캐스팅을 원하기 때문에 제작



그림 36 캐스팅 유형

- 캐스팅은 자동으로 충전되는 타입과, 키를 누르고 있을 때에만 충전되는 타입 두 가지가 존재
- 캐스팅은 데이터가 유지되는 타입과, 캐스팅을 취소할 경우 데이터가 사라지는 타입 두 가지가 존재

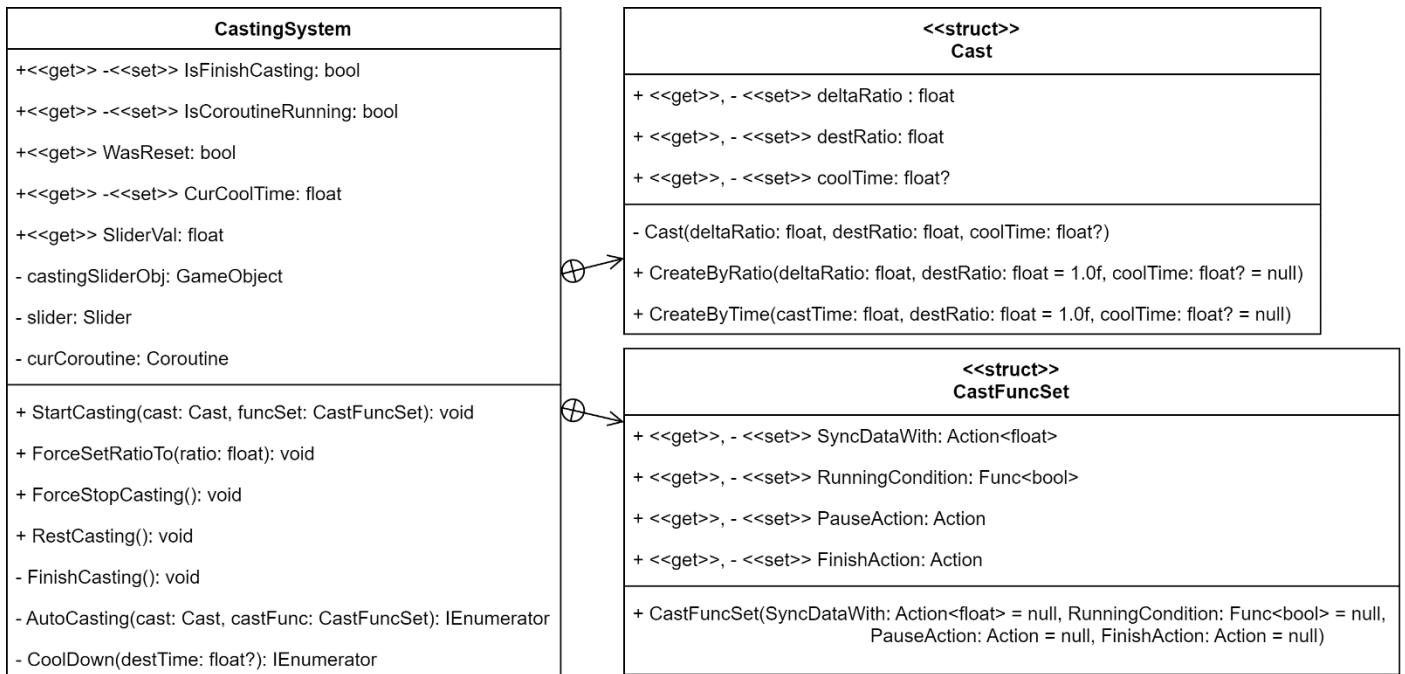


그림 37 Casting System 클래스 다이어그램

- CastingSystem 클래스는 타입을 나타내는 Cast 와, 캐스팅을 위한 함수의 조합인 CastFuncSet 을 빌더 패턴으로 사용

9 RockmanX5 모작

9.1 프로젝트 소개



그림 38 인게임 전투 장면

<https://youtu.be/lzxj7Tz0fHA>

- 프로젝트 기간
2022.08 ~ 2022.09 기
- 프로젝트 인원
1인
- 프로젝트 주제
Windows API 를 활용한 C++
게임
- 참고 영상

- 사용 기술
C++, Windows API
- 기획 의도
 - 2D 횡스크롤 게임
 - 록맨 X5 모작
 - 록맨 X5 의 오프닝 스테이지 구현 목표
- 게임 설명
 - 플레이어 캐릭터의 기본적인 움직임 구현
 - 적 캐릭터의 공격 패턴 구현
- 주요 개발 항목
 - Framework 구성
 - SAT 를 이용한 충돌 처리
 - 스프라이트 에디터
 - FSM 을 통한 캐릭터 상태 관리 및 적 AI

9.2 Framework 구성

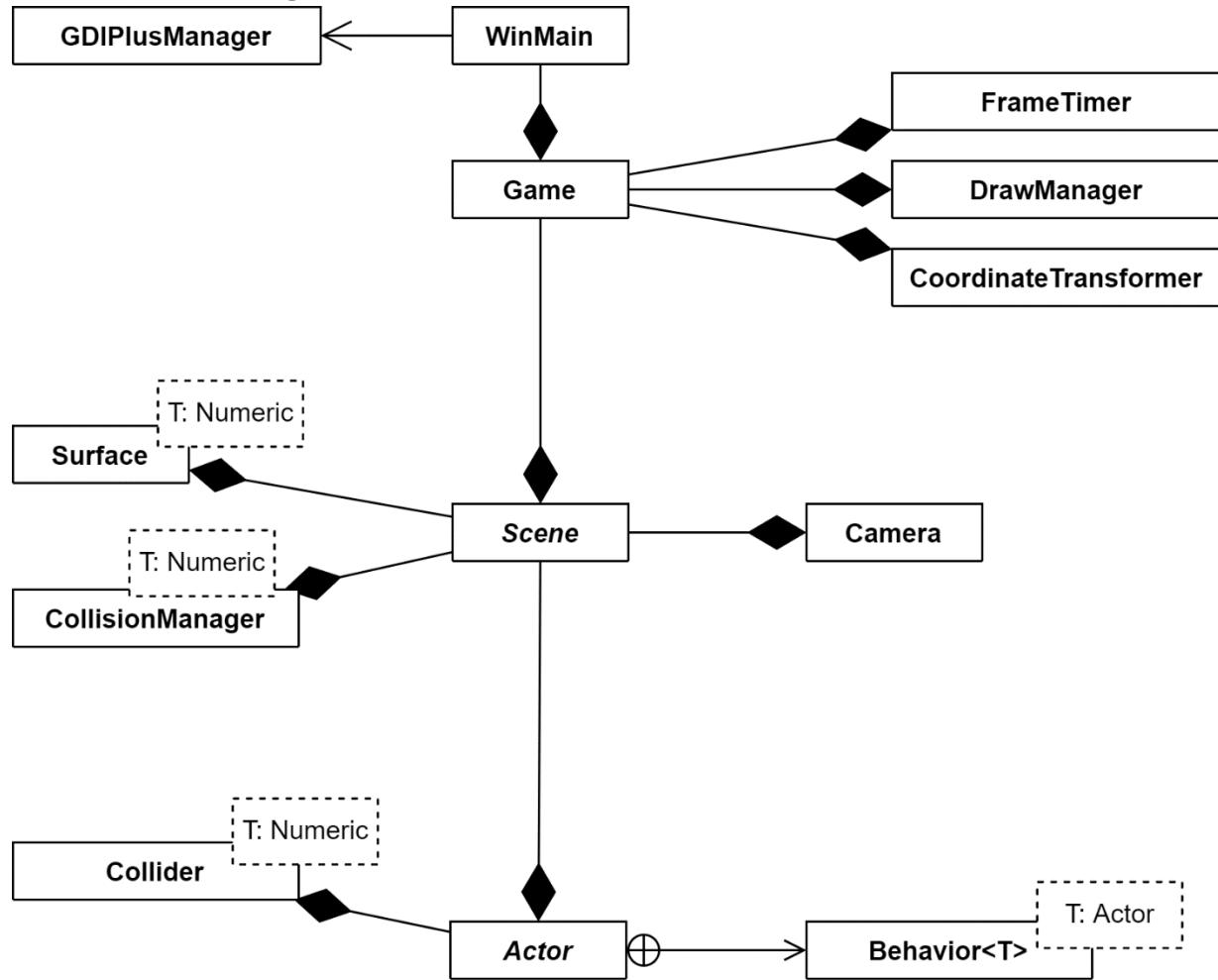


그림 39 Game Framework 클래스 다이어그램

- **WinMain**: Windows API 메인 클래스
- **GDIPlusManager**: GDIPlus 를 사용하기 위해 Semaphore 를 이용한 클래스
- **Game**: 모델 업데이트와 프레임 구성을 별도로 작성하도록 만든 클래스
- **FrameTimer**: DeltaTime 을 측정하는 타이머 클래스
- **DrawManager**: Windows API 를 통해 그림을 그리기 위한 클래스
- **CoordinateTransformer**: 화면 좌표계를 수학 좌표계로 변환하는 클래스
- **Scene**: 게임 씬 클래스
- **Camera**: 화면을 움직일 수 있는 카메라 클래스
- **Surface**: 화면에 글씨 또는 도형을 출력할 수 있는 클래스
- **CollisionManager**: 충돌을 처리하는 클래스
- **Collider**: 충돌할 수 있는 모양과 범위를 지정하는 클래스
- **Actor**: 게임 오브젝트 클래스
- **Behavior**: FSM 을 통해 Actor 의 행동을 관리하는 클래스

```

1 #pragma once
2
3 #include <memory>
4 #include <vector>
5 #include "FrameTimer.h"
6 #include "DrawManager.h"
7 #include "CoordinateTransformer.h"
8 #include "Scene.h"
9
10 class Game
11 {
12 public:
13     Game();
14     void ComposeFrame( HDC hdc );
15     void UpdateModel();
16
17     void RefreshScreen();
18     void SetScreenSize( HWND hWnd )
19     {
20         GetClientRect( hWnd, &screenRect );
21     }
22     void SetClientSize( HWND hWnd, int width = screenWidth, int height = screenHeight );
23 private:
24     void CycleScenes();
25     void ReverseCycleScenes();
26
27 public:
28     RECT screenRect;
29 private:
30     static constexpr int screenWidth = 1600;
31     static constexpr int screenHeight = 900;
32
33     RECT oldScreenSize = screenRect;
34     FrameTimer ft;
35     DrawManager drawManager;
36     CoordinateTransformer ct;
37
38     bool isScreenChanged = true;
39
40     std::vector<std::unique_ptr<Scene>> pScenes;
41     std::vector<std::unique_ptr<Scene>>::iterator curScene;
42 };

```

그림 40 Game.h 코드 예시

```

1 #include "Game.h"
2 #include "OpeningScene.h"
3
4 Game::Game()
5 {
6     pScenes.push_back( std::make_unique<OpeningScene>( screenWidth, screenHeight, ct ) );
7     curScene = pScenes.begin();
8 }
9
10 void Game::ComposeFrame(HDC hdc)
11 {
12     drawManager.DrawMain( hdc, screenRect, isScreenChanged,
13         [this]( HDC hdc )
14     {
15         (*curScene)->Draw( hdc );
16     }
17 );
18     if ( isScreenChanged )
19     {
20         isScreenChanged = false;
21     }
22 }
23
24 void Game::UpdateModel()
25 {
26     float dt = ft.Mark();
27     RefreshScreen();
28     (*curScene)->Update( dt, *this );
29 }
30 void Game::RefreshScreen()
31 {
32     if ( screenRect.right != oldScreenSize.right || screenRect.bottom != oldScreenSize.bottom )
33     {
34         isScreenChanged = true;
35         oldScreenSize.left = screenRect.left;
36         oldScreenSize.top = screenRect.top;
37         oldScreenSize.right = screenRect.right;
38         oldScreenSize.bottom = screenRect.bottom;
39     }
40 }
41
42 void Game::SetClientSize( HWND hWnd, int width, int height )
43 {
44     SetWindowPos( hWnd, nullptr, 0, 0, width, height, SWP_NOMOVE );
45     RECT winRect;
46     RECT screenRect;
47     GetWindowRect( hWnd, &winRect );
48     GetClientRect( hWnd, &screenRect );
49
50     const int clientWidth = screenRect.right - screenRect.left;
51     const int clientHeight = screenRect.bottom - screenRect.top;
52     int winWidth = winRect.right - winRect.left;
53     int winHeight = winRect.bottom - winRect.top;
54     winWidth += winWidth - clientWidth;
55     winHeight += winHeight - clientHeight;
56     const int resolutionX = GetSystemMetrics( SM_CXSCREEN );
57     const int resolutionY = GetSystemMetrics( SM_CYSCREEN );
58
59     SetWindowPos( hWnd, nullptr, ((resolutionX / 2) - (winWidth / 2)),
60                 ((resolutionY / 2) - (winHeight / 2)), winWidth, winHeight, NULL );
61 }

```

9.3 SAT를 이용한 충돌 처리



경사면 충돌

그림 42 디버그 모드에서 확인한 충돌 모습

- SAT(Separating Axis Theorem)을 이용해 AABB 및 OBB 처리

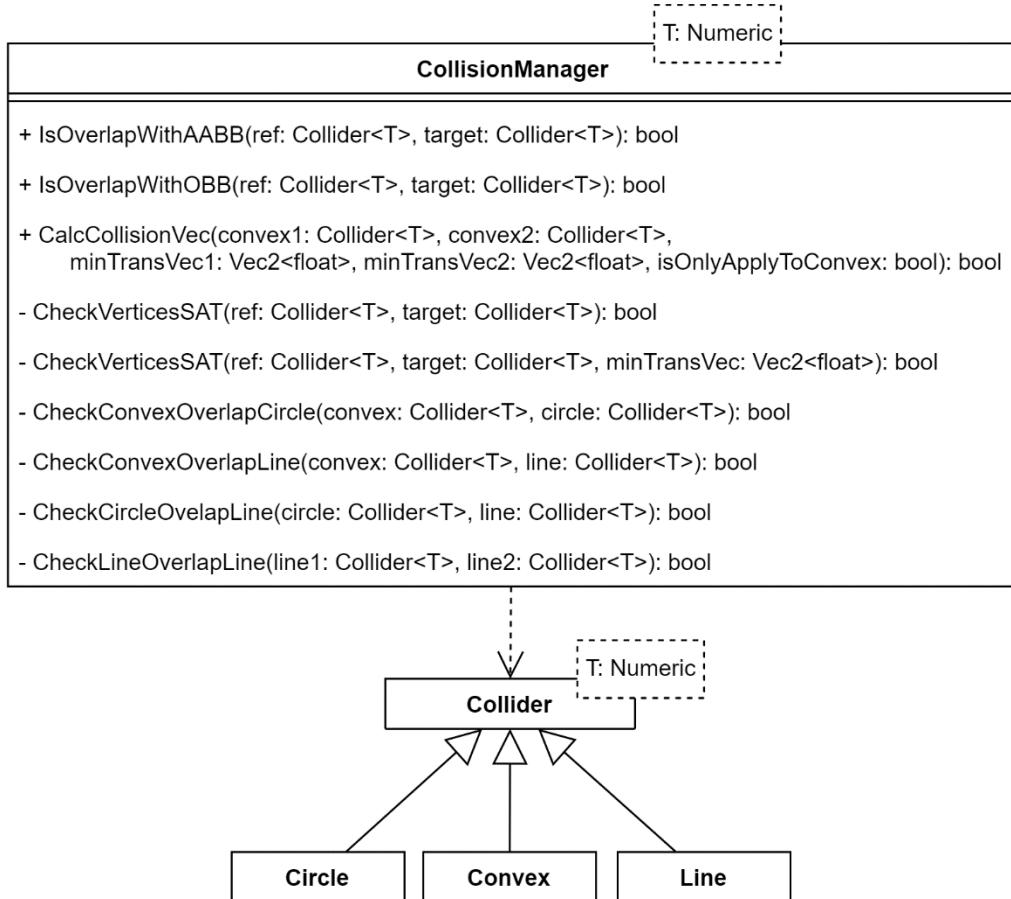


그림 43 충돌 처리 클래스 다이어그램

9.4 스프라이트 에디터

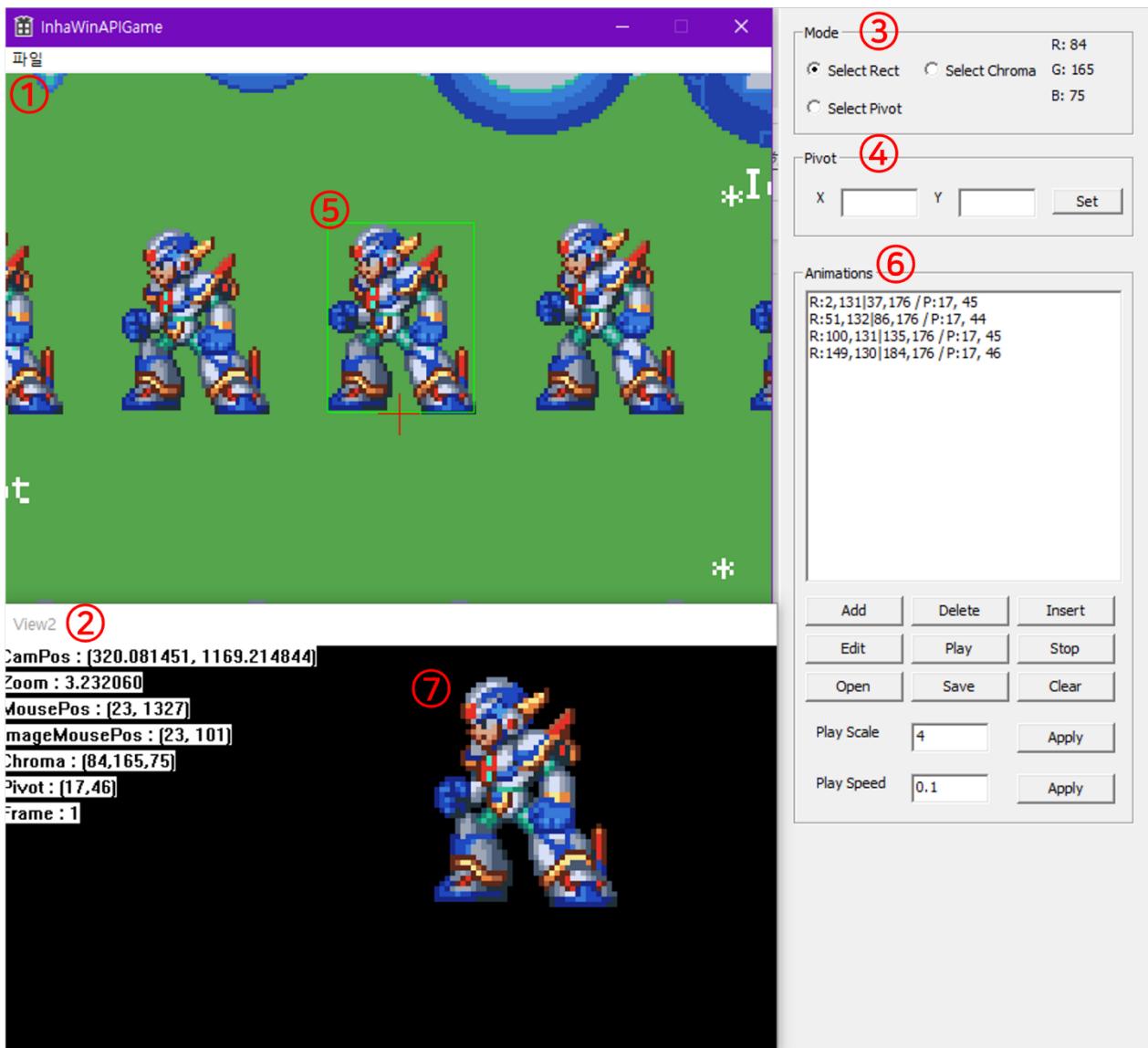


그림 44 스프라이트 에디터 모습

- 록맨 X5의 레퍼런스 이미지는 하나의 큰 스프라이트 파일로 구성되어, 스프라이트의 위치를 코드로 직접 지정하기 어려움
- 애니메이션을 편집하기 위해 스프라이트 에디터 제작
- 이미지 로딩, 범위 지정, 프레임 생성, 애니메이션 재생, 애니메이션 로딩, 애니메이션 저장 기능 제공

9.4.1 기능 설명

1. 스프라이트 이미지 창
2. 애니메이션 재생 및 디버깅 창
3. 마우스 액션 모드 설정
 - Select Chroma: 마우스 클릭을 스포이트로 작동하여, 배경색 추출
 - Select Rect: 마우스 드래그로 스프라이트 범위 지정, 드래그 이후 드래그 범위에서 가장 최적화 된 스프라이트 범위로 변경
 - Select Pivot: 마우스 클릭으로 Pivot 위치 조정
4. Pivot
 - 0~1로 정규화 된 좌표로 이미지의 Pivot 지정, 기본 (0.5, 1)
 - Set 버튼으로 Pivot의 위치를 지정
버튼 클릭 시 스프라이트 이미지 창에 Pivot의 위치가 빨간색 십자로 표기됨
5. 드래그로 지정된 스프라이트 범위와 기본 Pivot 위치
6. 애니메이션 창
 - R: 사각형 픽셀 좌표
 - P: Pivot 픽셀 좌표
 - 위에서 아래 순으로 애니메이션 프레임 배열로 구성

Add: 버튼 클릭 시 애니메이션 창에 해당 스프라이트 범위를 애니메이션 프레임에 추가함
Delete: 선택한 애니메이션 프레임을 애니메이션에서 제거
Insert: 선택한 애니메이션 프레임 다음에 현재 선택된 스프라이트 범위 추가
Edit: 선택한 애니메이션 프레임을 현재 선택된 스프라이트 범위로 변경
Play: 애니메이션 재생 창에서 현재 애니메이션을 재생
Stop: 애니메이션 재생 창에서 플레이 중인 애니메이션을 정지
Open: 파일 시스템을 통해 애니메이션 파일 로딩
Save: 파일 시스템을 통해 애니메이션 파일 저장
Clear: 애니메이션 초기화
Play Scale: 애니메이션 이미지크기 조정
Play Speed: 애니메이션 재생 간격 시간 조정
7. 현재 플레이 중인 애니메이션의 모습

9.5 FSM을 이용한 캐릭터 상태 관리 및 적 AI



그림 45 벽타기 상태 및 디버깅 화면

- 캐릭터의 상태 표현을 위해 FSM을 사용
- 캐릭터의 이동 상태를 FSM을 통해 사용하며, 각 상태에서 공격 상태에 따라 애니메이션 변경

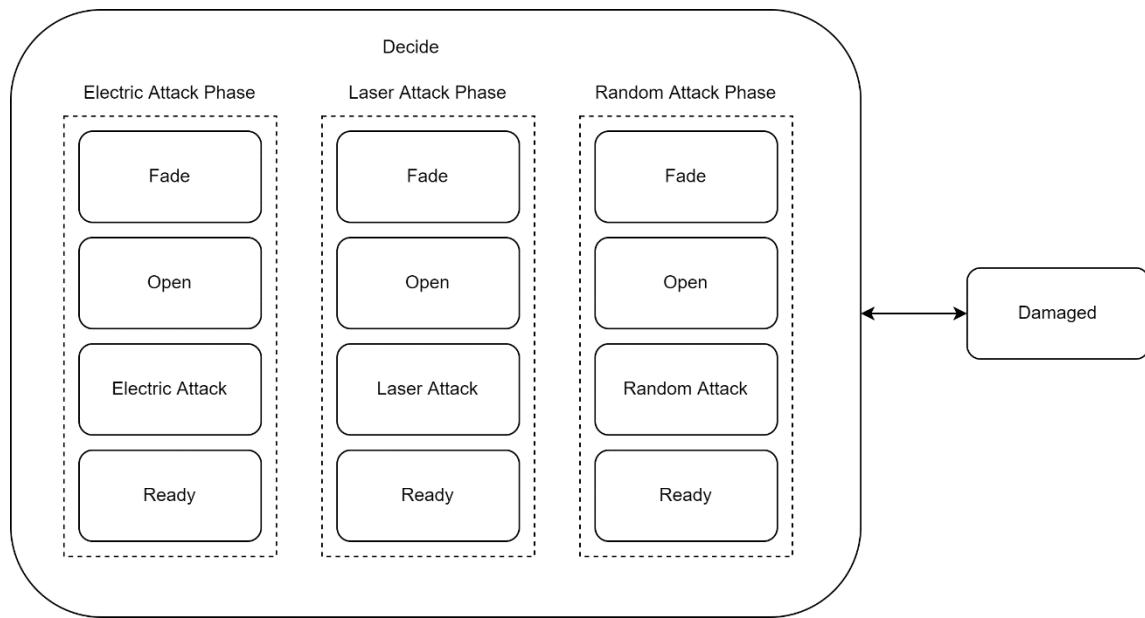


그림 46 적 AI FSM 구성

- 적의 공격 AI 또한 FSM을 사용
- 행동을 결정하는 메인 상태인 Decide에서, 서브 상태로 공격 행동을 결정

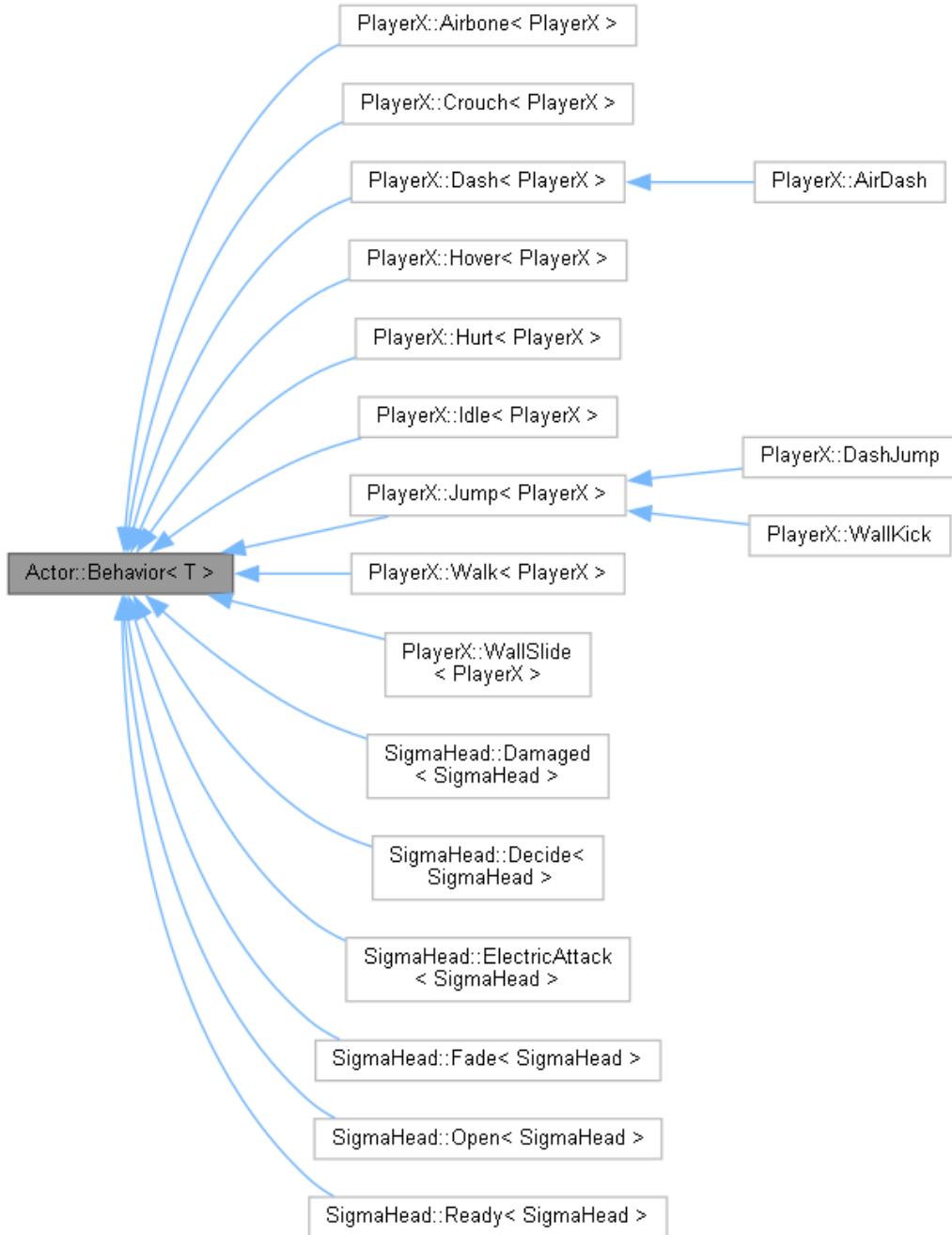
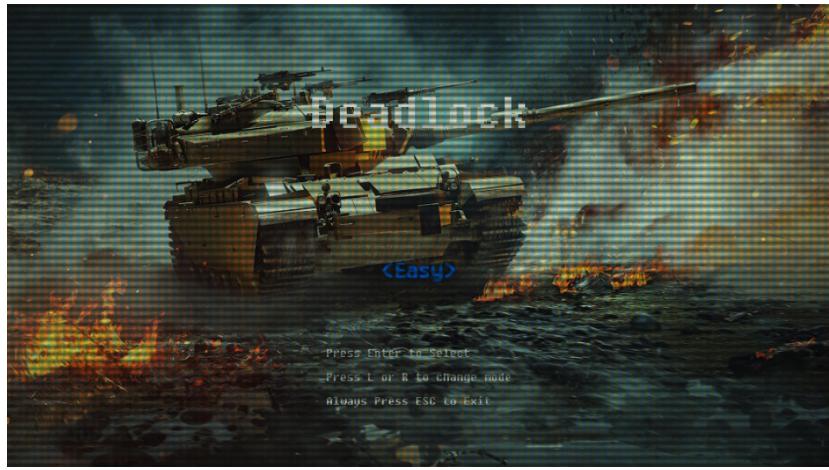


그림 47 Behavior 클래스를 상속받는 행동 Class 들

10 Deadlock



- 프로젝트 기간: 2022.05 ~ 2022.05
- 프로젝트 인원: 1인
- 프로젝트 주제:
C를 이용한 텍스트 게임
- 사용 기술: C
- 참고 영상:
<https://youtu.be/ym8-lvTHfhM>
- 기획 의도

- 웜즈, 포트리스와 같은 2D 탱크 슈팅 게임
- 그래픽 라이브러리 없이 이미지 파일을 읽어 콘솔 창에 띄우기
- 주요 개발 항목
 - BMP 이미지 출력 시스템
- 게임 설명
 - 게임 시작 시 난이도 설정 (Easy or Hard)
 - 탱크 종류 선택 (경전차, 중형전차, 중전차, 자주포)
 - 3 스테이지 구성
 - 플레이어 턴 – AI 턴으로 턴 시스템 구성
 - 플레이어 턴에서 플레이어는 이동, 각도 및 힘 조절, 사격을 할 수 있으며, 사격 시 포탄 발사 후 턴 종료
 - 이동 거리 및 범위는 제한되어 있으며, 화면 밖으로 나갈 수 없음
 - 포탄은 포물선으로 발사되며 탱크 종류에 따라 데미지 및 그래픽이 다름
 - 포탄이 상대를 맞추면 상대 체력을 데미지만큼 감소
 - AI는 랜덤으로 입력 받은 값만큼 움직이며, 이동 이후 사격함
 - AI는 시작 화면에서 설정한 난이도에 따라 명중률을 보정 받음
 - 플레이어의 체력이 0 이되는 경우 패배하고 시작 화면으로 이동
 - AI의 체력이 0 이되는 경우 다음 스테이지로 이동, 3 스테이지 클리어 시 게임 승리

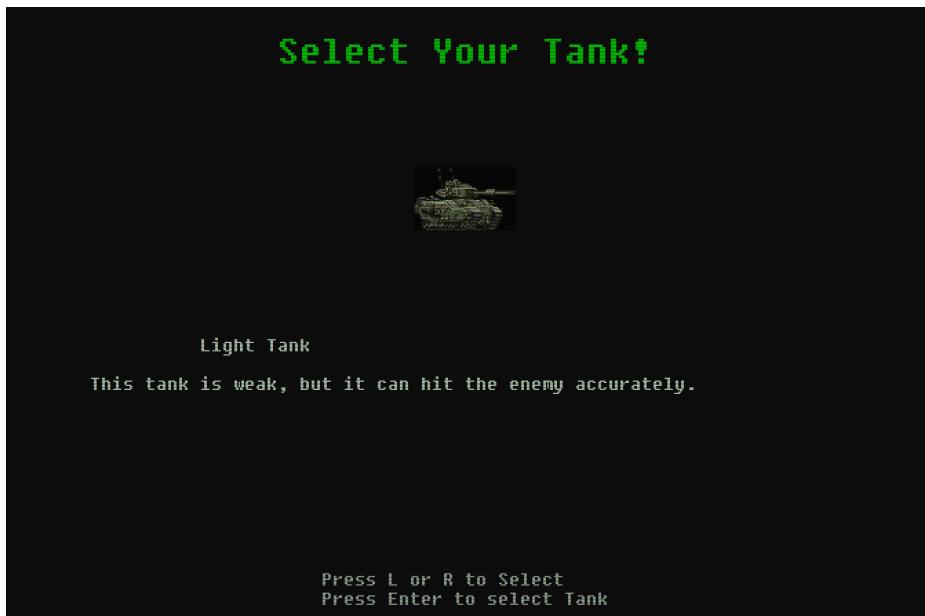


그림 49 탱크 선택 화면

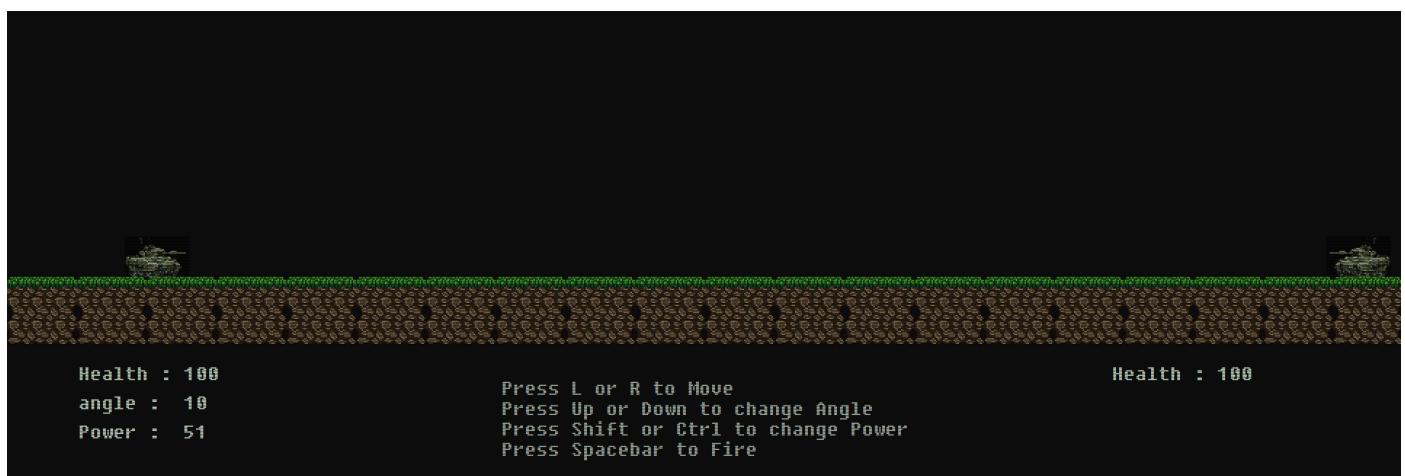


그림 50 게임 스테이지 화면

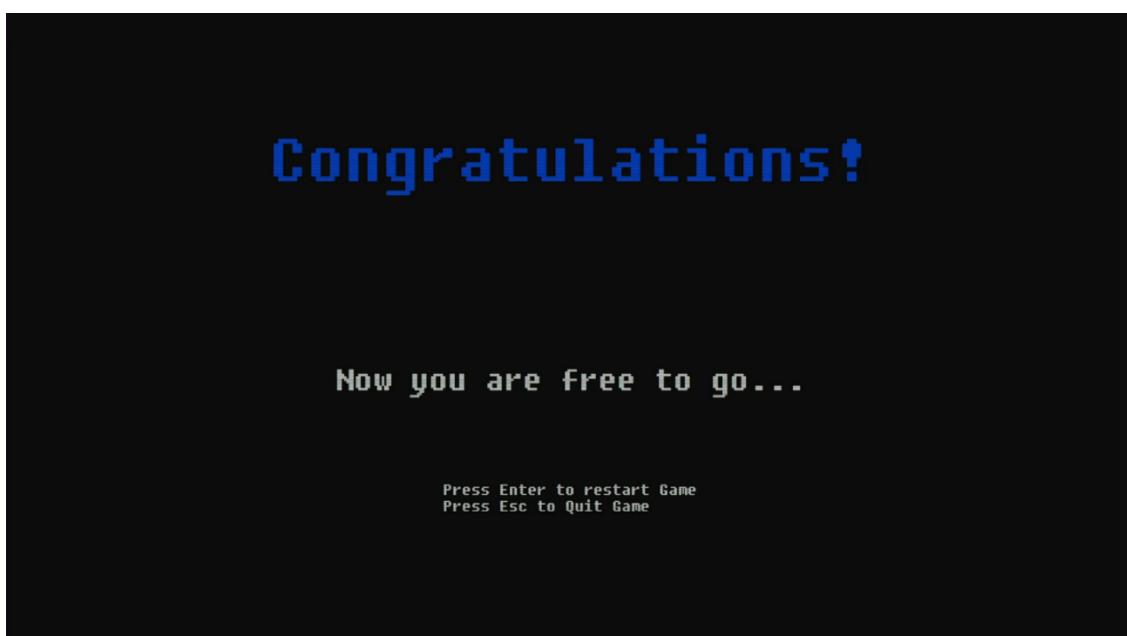


그림 51 게임 승리 화면

10.1 BMP 이미지 출력 시스템



그림 52 게임에서 사용하는 맵 타일 및 탱크 원본 이미지

- 콘솔 창에서 하나의 글자를 하나의 픽셀로 취급
- '■'를 하나의 픽셀로 취급
- RGB 색상으로 픽셀의 색 표현
- 화면의 좌표에 픽셀 하나를 출력할 수 있는 PutPixel() 함수 제공
- 파일 경로를 읽어 해당 BMP 파일을 출력하는 함수 DrawSprite() 제공
- 읽어온 이미지를 레퍼런스로 두어 실제 이미지는 참조하여 사용하도록 설계
- 스프라이트 이미지를 읽어 폰트를 사용할 수 있도록 설계
- 화면 간격을 늘려 픽셀 간격을 늘릴 수 있음
- 픽셀 간격이 넓을수록 출력 속도 증가

```

1 void MakeSurface ( const char* filename, Surface* pSurface )
2 {
3     FILE* pFile;
4
5     if (fopen_s( &pFile, filename, "rb" ) != 0)
6     {
7         //fclose( pFile );
8         return;
9     }
10    assert ( pfile != NULL );
11
12    // Read the bitmap file header
13    BITMAPFILEHEADER bmFileHeader;
14    fread( (char*)(&bmFileHeader), sizeof( bmFileHeader ), 1, pFile );
15
16    // Verify that this is a .BMP file by checking bitmap id
17    if (bmFileHeader.bfType != 0x4D42)
18    {
19        fclose( pFile );
20        return;
21    }
22
23    // Read the bitmap info header
24    BITMAPINFOHEADER bmInfoHeader;
25    fread( (char*)(&bmInfoHeader), sizeof( bmInfoHeader ), 1, pFile );
26
27    // Check 32bit Img
28    assert( bmInfoHeader.biBitCount == 24 || bmInfoHeader.biBitCount == 32 );
29    const bool is32b = bmInfoHeader.biBitCount == 32;
30
31    pSurface->width = bmInfoHeader.biWidth;
32
33    int yStart;
34    int yEnd;
35    int dy;
36    if (bmInfoHeader.biHeight < 0)
37    {
38        pSurface->height = -bmInfoHeader.biHeight;
39        yStart = 0;
40        yEnd = pSurface->height;
41        dy = 1;
42    }
43    else
44    {
45        pSurface->height = bmInfoHeader.biHeight;
46        yStart = pSurface->height - 1;
47        yEnd = -1;
48        dy = -1;
49    }
50
51    pSurface->pPixels = (Color*)malloc( sizeof( Color ) * pSurface->height * pSurface->width );
52    if ( pSurface->pPixels == NULL )
53    {
54        exit( 0 );
55    }
56
57    //move file pointer to the beginning of bitmap data
58    fseek( pFile, bmFileHeader.bfOffBits, SEEK_SET );
59    const int padding = (PIXEL_ALIGN - (pSurface->width * PIXEL_SIZE) % PIXEL_ALIGN) % PIXEL_ALIGN;
60
61    for (int y = yStart; y != yEnd; y += dy)
62    {
63        for (int x = 0; x < pSurface->width; x++)
64        {
65            SurfacePutPixel( x, y, MakeRGB( fgetc(pFile), fgetc( pFile ), fgetc( pFile ) ), pSurface );
66            if (is32b)
67            {
68                fseek( pFile, 1, SEEK_CUR );
69            }
70        }
71        if (!is32b)
72        {
73            fseek( pFile, padding, SEEK_CUR );
74        }
75    }
76
77    fclose( pFile );
78
79    pSurface->wasDrew = false;
80 }

```

그림 53 BMP 파일을 읽어 픽셀들을 저장하는 Surface로 변환하는 함수 코드

11 VRFlight Simulation



그림 54 VRFlight 시연 사진

t=942

- 프로젝트 기간: 2020.12 – 2021.06
- 프로젝트 인원: 4인
- 담당 업무: 프로젝트 팀장 및 개발자
- 프로젝트 주제:
UnrealEngine4를 이용한 VR 항공기 시뮬레이션

- 참고 영상:
<https://youtu.be/R9U9pKLASw0?>

- 사용 기술

Unreal Engine 4, C++, Blueprint, Blender

- 기획 의도

- 항공기 내부를 실제 기체와 유사하게 제작
- VR을 통해 직접 항공기를 조종하는 경험 제공
- 계기비행을 통한 착륙 경험 제공

- 주요 개발 항목

- 비행 시스템 구현
- 항공기 조종사 시점 및 항공기 시점 개발
- 항공 전자 장비 개발

구현 항목:

HUD, Air Speed Indicator, Altimeter Indicator, Attitude Director Indicator, AOA Indicator, Vertical Speed Indicator, Horizontal Situation Indicator, Fuel Flow, Indexer, Integrated Control Panel, Data Entry Display, Oil Pressure Indicator, Nozzle Open Indicator, RPM Indicator, VHF Omni Range, Aeronautical Light Aids

- 핸드 트래킹으로 항공 전자장비와 상호작용
- 항공기 모델 및 애니메이션 제작

11.1 비행 시스템 구현

표면적인 항공역학을 위해 개발해야 할 요소

Flight System

항공기에 작용하는 4가지 힘: 양력, 항력, 추력, 중력

중력 : Unreal Engine 자체 지원 → 직접 구현 X

직접 구현해야 할 힘 : 양력, 항력, 추력

고도에 따른 공기밀도, 마하 계수, 지면 효과

조종면, 3차원 좌표에 움직임 표현

Flight Assists

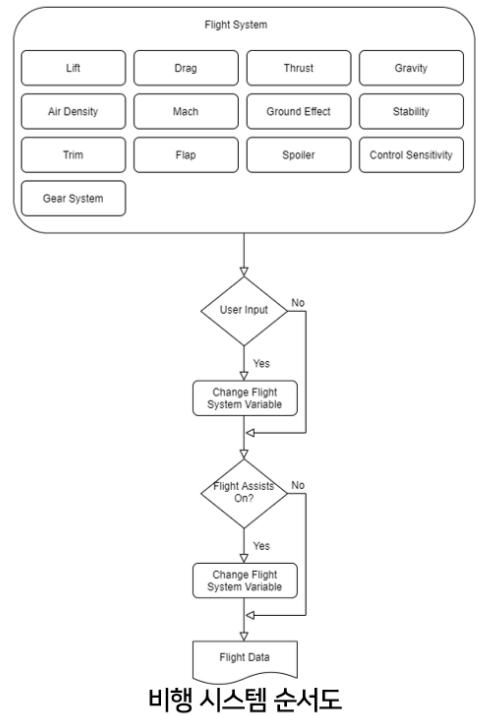


그림 55 항공역학 구현목록

기존 공식에 따른 양력, 항력, 추력, 공기밀도 제작

Lift

$$L = \frac{1}{2} \rho V^2 C_L S$$

$$L_F = \text{플랩의 양력계수} \times \text{플랩의 위치} \times \text{Flap Position} \times \text{Curve Value}$$

$$L_T = (L + L_F) \times \text{Ground Effect}_{\text{지면효과 계수}}$$

$$L_S = \text{Side Lift Multiplier} \times \text{Curve Value from Side slip} \times \text{Speed}$$

Drag

$$D_T = (D_M + D_S + D_F + D_B + D_A) \times \text{Speed} \times \text{Ground Effect}$$

$$D_M = \text{Drag Multiplier} \times \text{Curve Value}$$

$$D_S = \text{Side Drag Multiplier} \times \text{Curve Value}$$

$$D_F = \text{Flap } D_c \times \text{Flap Position} \times \text{Curve Value}$$

$$D_B = \text{Spoiler } D_c \times \text{Spoiler Position}$$

공기밀도

$$\rho = P \div (T \times \text{Specific Gas Constant})$$

$$P = P_{Sea Level} \times \{1 - (0.0000225577 \times Alt)^{5.25588}\}$$

$$T = T_{Sea Level} - \{\gamma \times \text{Min}(Alt, Alt_{Tropopause})\} + 273.15$$

Thrust

$$Thrust = \text{Thrust Curve from Air Density} \times \text{Total Speed} \times \sqrt{\frac{\text{Air Density}}{\text{Sea Level Air Density}}}$$

계산한 힘을 좌표계에 구현

$$F = F_{Aero} + F_{Engine}$$

$$F_{Aero} = \{(\mathbf{V}_{Drag} \times F_x) + (\mathbf{V}_{Side Lift} \times F_y) + (\mathbf{V}_{Lift} \times F_z)\} \times \text{Air Density}$$

$$F_{Engine} = \mathbf{V}_{Engine} \times \text{Thrust Amount}$$

그림 56 항공역학 계산식

11.2 항공 전자 장비

Avionics Model & Avionics Animation

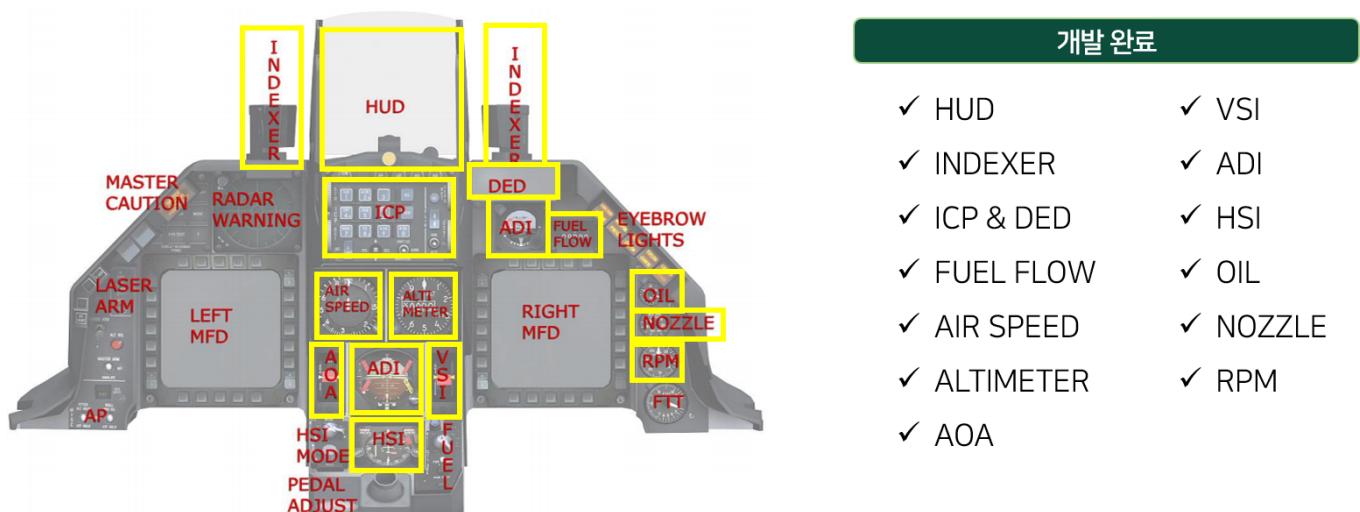


그림 57 개발한 항공 전자장비 목록



그림 58 인게임 조종석 화면

12 Sky Stability



- PID 를 이용한 무인항공기 수평자세제어 시스템
- 프로젝트 기간
2020.09 – 2020.12
- 프로젝트 인원
4인
- 담당 업무
프로젝트 팀장 및 소프트웨어 개발
- 프로젝트 주제
PID 를 활용한 모터 동작 캡스톤 프로젝트
- 사용 기술
Arduino, C
- 프로젝트 목표
 - PID 제어를 통한 수평 자세 제어
- 참고 자료
 - https://github.com/s-kim-mcg/Fixed-wing_FlightController