**캡스톤 디자인 I**

**종합설계 프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | *Tactic System을 적용한 수집형 RPG* |
| 팀 명 | *택틱RPG 팀* |
| 문서 제목 | 택틱 RPG 팀 계획서 |

|  |  |
| --- | --- |
| **Version** | 1.3 |
| **Date** | 2025-03-18 |

|  |  |
| --- | --- |
| **팀원** | 최치원 (팀장) |
| 강문정 |
| 김동휘 |
| 유민석 |
|  |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤 디자인I 수강 프로젝트를 수행하는 팀 “택틱RPG”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “택틱RPG”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 계획서-Tactic System을 적용한 영웅 수집형 RPG.doc |
| **원안작성자** | 유민석,강문정,최치원,김동휘 |
| **수정작업자** | 유민석,강문정,최치원,김동휘 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2025-03-15 | 유민석 | 1.0 | 최초 작성 | 맵/UI 연구 및 개발 내용 추가 |
| 2025-03-18 | 강문정 | 1.1 | 내용 수정 | 맵/UI 연구/개발 내용 추가 |
| 2025-03-18 | 최치원 | 1.2 | 내용 수정 | 개요,추진 배경 및 필요성 추가 |
| 2025-03-18 | 김동휘 | 1.3 | 내용 수정 | 전투 개발 및 결과 내용 추가 |

**목 차**

[1 개요 3](#_Toc1566926385)

[1.1 프로젝트 개요 3](#_Toc1121528947)

[1.2 추진 배경 및 필요성 4](#_Toc777631823)

[2 개발 목표 및 내용 6](#_Toc671925629)

[2.1 목표 6](#_Toc1492022252)

[2.2 연구/개발 내용 6](#_Toc1446921899)

[2.3 개발 결과 15](#_Toc7380766)

[3 프로젝트 팀 구성 및 역할 분담 17](#_Toc1051329756)

[3.1 개발 일정 17](#_Toc151550138)

[3.2 일정별 주요 산출물 17](#_Toc2133289748)

# **개요**

## 프로젝트 개요

본 프로젝트는 Unity엔진을 기반으로 한 TacticSystem을 활용한 영웅 수집형 RPG를 개발하는 것을 목적으로 한다. 플레이어는 **MapScene**에서 맵의 모든 지역을 점령하고, 최종적으로 보스를 처치하는 것을 목표로 한다. 반대로 보스는 맵을 점령하며 세력을 확장하고, 플레이어보다 먼저 모든 지역을 장악하는 것을 목표로 한다.   
 맵은 그래프로 연결되어 있으며, 노드는 **적군 점령지, 아군 점령지**로 구분된다. 플레이어는 한 턴마다 아군 점령지와 인접한 지역을 선택해 전투를 시작할 수 있다. 보스는 플레이어의 턴이 끝난 후, 적군 점령지와 인접한 무작위 노드를 일정 확률로 점령하면서 세력을 확장한다.   
 플레이어가 점령할 지역을 선택하면, 해당 지역에서 등장하는 몬스터의 종류와 승리 보상을 확인할 수 있는 **MapInfoUI**와, BattleScene에 영웅을 배치할 위치를 표시하는 **Grid UI**가 팝업된다. 보유한 영웅은 **RosterUI**에서 드래그 앤 드롭하여 원하는 위치에 배치할 수 있으며, 배치를 완료한 후 **출정 버튼**을 누르면 BattleScene으로 전환된다.

**BattleScene**에서는 그리드UI의 정보를 기반으로 영웅들이 전장에 배치되며, 적 몬스터들은 미리 설정된 프리셋 중 랜덤하게 배치된다. 전투는 **Tactic System**을 활용하여 진행되며, 각 캐릭터(영웅 및 몬스터)는 자신의 Global Cooldown마다 Tactic System의 **Tactic List**를 순회하면서 조건에 맞는 행동을 수행한다.   
 **Tactic**은 **Target, Condition, Action**으로 구성된다. Target은 Ally, Self, Enemy 등 1차적으로 대상을 선택하는 역할을 하며, 선택된 대상 리스트에서 Condition(예: HP < 50% && 가장 가까운 대상)을 만족하는 캐릭터를 필터링한다. 이후 해당 캐릭터를 대상으로 Action(공격, 회복, 버프 등)을 실행한다. 또한, 런타임 중 **TacticSystemUI**를 통해 실시간으로 영웅의 Tactic을 수정할 수 있어 플레이어가 전황에 맞춰 전략을 조정할 수 있도록 한다.   
 하나의 **BattleScene**은 여러 개의 **Wave**로 구성되며, 각 Wave의 몬스터를 처치하면 다음 Wave로 전환된다. 모든 Wave의 몬스터를 처치하면 **ResultUI**가 승리를 표기하고 보상을 지급한다. 반대로, 전투 도중 모든 영웅의 체력이 0이 되면 **패배**로 처리되며, 해당 점령지는 적군 점령지로 변경된다.

승리 보상으로는 **영웅, 골드** 등이 있으며, 획득한 골드는 **MapScene**의 **ShopUI**를 통해 상점에서 영웅을 구매하는 데 사용할 수 있다. 또한, MapScene에서 적군 점령지 중에는 일정확률로, **특수한 적군 점령지**가 존재한다. 이를 제한된 턴 수 내에 점령하면 더욱 큰 보상을 획득할 수 있다.

## 추진 배경 및 필요성

**1.2.1 시장 및 기술 현황**

현재 수집형 RPG는 모바일 및 PC 게임 시장에서 매우 인기 있는 장르로 자리 잡고 있다. 이 장르는 플레이어가 다양한 영웅을 수집하고 성장시키며, 전략적인 전투를 펼치는 방식으로 구성된다. 대부분의 게임은 자동 전투 시스템을 채택하여 AI가 스킬을 사용하며 전투를 진행하거나, 유저가 턴마다 직접 캐릭터의 행동을 결정하는 턴제 방식으로 이루어진다. 또한, AI가 기본적인 전투를 수행하고, 중요한 스킬은 플레이어가 직접 조작하는 방식도 존재한다. 대표적인 작품으로는 *리니지M*, *에픽세븐*, *블루아카이브* 등이 있다. 현재 대부분의 수집형 RPG는 캐릭터 수집과 성장을 핵심 요소로 삼고 있으며, 각 영웅은 고유한 능력과 스킬을 보유하고 있어 플레이어가 전략적으로 조합하여 전투를 수행할 수 있다. 게임의 전략성은 캐릭터 간의 조합과 상성에 따라 크게 달라진다. 또한, 소셜 및 온라인 요소가 결합되어 다양한 콘텐츠와 플레이어 간의 상호작용을 제공하며, 지속적인 업데이트를 통해 새로운 영웅을 수집하고 성장시킬 기회를 제공한다.

**1.2.2 기존 시스템의 문제점 및 한계**

**자동 전투 시스템의 단조로움과 전술적 개입 부족**  
수집형 RPG는 자동 전투 시스템 혹은 턴제 시스템을 채택하고 있다. 자동 전투 시스템에서는 캐릭터들이 프로그래머가 사전에 설정한 AI에 따라 동일한 패턴으로 전투를 진행하며, 유저의 전략적 개입이 거의 불가능하다. 그로 인해 전투는 반복적이고 단조롭게 진행된다. 턴제 시스템을 사용한다면, 유저의 개입을 통한 전략적 요소가 증가하지만, 전투 진행 속도가 느리고, 반복적인 조작이 필요해진다. 이런 시스템은 게임의 재미를 더하기 위해 캐릭터 성장과 전투력 증가에만 의존하는 경향이 있다.

**맵 시스템의 미흡한 활용**  
대부분의 RPG에서는 맵 탐색 요소가 존재하지만, 이를 전략적인 점령 시스템으로 적극적으로 활용하는 게임은 드물다. 현재의 시스템에서는 맵 점령이 단순히 거점이나 지역을 차지하는 데 그치며, 점령된 지역이 게임 진행에 미치는 전략적 영향이 부족하다. 점령과 전투의 연계성이 제한적이어서 플레이어는 맵을 통해 전략적 선택을 할 기회가 적다. 맵 점령은 중요한 게임 요소로 자리잡을 수 있지만, 이를 보다 깊이 있게 활용하여 세력을 확장하는 목표를 제시하는 시스템이 부족하다.

**낮은 영웅 수집 접근성**  
수집형 RPG에서는 새로운 영웅을 수집하는 데 있어 과금이 필수적인 경우가 많다. 게임 내에서 영웅을 얻는 주요 방법이 랜덤 뽑기 또는 제한된 자원을 통한 구매로 제공되며, 이러한 시스템은 플레이어에게 과금을 유도하는 구조로 설계되어 있다. 결과적으로, 과금을 하지 않으면 원하는 영웅을 얻기 어렵고, 이는 게임 진행에서 중요한 전략적 선택을 제한하는 요소로 작용한다. 새로운 영웅을 수집하기 위한 과금 의존도가 높아지면서, 유저는 영웅 수집과 성장에 있어 실질적인 재미보다는 금전적인 투자가 중요한 요소로 작용하게 된다.

**1.2.3 개발할 시스템의 필요성**

기존 수집형 RPG의 자동 전투 시스템은 유저의 전략적 개입이 제한되며, 동일한 패턴의 반복으로 인해 전투가 단조롭게 진행되는 문제가 있다. 턴제 시스템을 도입하면 전략적 요소는 증가하지만, 전투 속도가 느려지고 반복적인 조작이 필요해진다. 이러한 한계를 극복하기 위해 본 프로젝트에서는 Tactic System을 활용한 전투 방식을 도입하여 유저가 실시간으로 전술을 조정할 수 있도록 한다. 이를 통해 자동 전투의 단조로움을 해소하면서도, 전략적인 개입이 가능한 시스템을 구축하여 유저에게 전투의 재미를 제공하고자 한다.

또한, 기존 RPG의 맵 시스템은 탐색과 점령 요소가 포함되어 있지만, 전략적 활용이 부족하여 플레이어가 적극적으로 개입할 기회가 제한적이다. 맵 점령이 단순한 지역 확보에 그치고, 게임 진행에 미치는 영향이 미미하기 때문에 전략적인 플레이가 어렵다. 이에 본 프로젝트에서는 맵을 그래프 구조로 설계하여, 플레이어와 보스가 세력을 확장하는 방식으로 진행되도록 한다. 플레이어는 점령을 통해 전투를 선택할 수 있으며, 보스는 무작위로 세력을 확장함으로써 맵 점령의 중요성을 높이고 전략적 긴장감을 강화한다. 이를 통해 플레이어는 단순한 탐색이 아닌, 전술적인 선택을 기반으로 게임을 진행할 수 있다.

수집형 RPG의 또 다른 문제점은 영웅 수집의 접근성이 낮다는 점이다. 대부분의 게임에서는 랜덤 뽑기나 제한된 자원을 통한 구매 방식으로 영웅을 획득해야 하며, 이는 과금 유도를 목적으로 설계된 경우가 많다. 결과적으로 과금을 하지 않는 유저는 원하는 영웅을 얻기 어려워 전략적 선택의 폭이 제한된다. 본 프로젝트에서는 전투 승리 보상으로 영웅을 획득할 수 있도록 하며, 획득한 골드를 사용하여 상점에서 직접 영웅을 구매할 수 있도록 한다. 이를 통해 유저가 과금 여부와 관계없이 전략적으로 필요한 영웅을 확보할 수 있는 기회를 제공하며, 영웅 수집의 재미를 극대화하고자 한다.

# **개발 목표 및 내용**

## 목표

본 프로젝트에서 과금을 하지 않으면서 수집형 RPG를 즐기고 싶고, 전략게임을 좋아하는 사람을 위해, 유저가 실시간으로 전술을 조정할 수 있는 전투와 맵 세력 확장을 경험할 수 있는 **Tactic System 기반 수집형 RPG**를 개발한다.

## 연구/개발 내용

**BattleScene**

|  |  |  |
| --- | --- | --- |
| **목표** | **세부목표, 연구/개발 방법** | **결과물** |
| GameManager | 게임 전반에 걸친 변수,함수 관리  Singleton 패턴 적용 | GameManager.cs |
| Player 제작 | 사용자 입력 반응 함수 호출  1. Ehanced InputSystem 사용  2. PlayerController Component로 적용 | InputActionAsset,  PlayerController.cs |
| 탑뷰 시점 카메라 제작  1. Cinemachine 사용  2. look,follow,offset 조정 | Virtual Camera |
| 캐릭터 마우스 호버링 시 Outline 렌더링  1.Outline.cs 적용  2. RayCast를 이용, MousePosition에 있는 Character 참조  3. 마우스 호버링 시 Character의 Outline을 렌더링. | Outline.cs,  MouseHoverOutline.cs |
| MapScene에서 BattleScene으로 전환될때, GridUI 정보를 토대로 SimpleGrid의 위치에 배치  1. MapScene의 GridUI의 GridSize 만큼 Transform을 가진 객체 생성  2. Grid에 맞게 위치 조정 | SimpleGrid.cs |
| TacticSystem 제작 | Character(Hero,Enemy)를 관리할 Manager 생성 | CharacterManager.cs |
| Character에 적용할 TacticSystemComponent 생성  1. Tactic을 List로 관리.  2. Character의 GlobalCooldown마다 List를 순회,  조건에 맞는 Tactic 수행 | TacticSystem.cs |
| TacticSystem에서 사용할 Tactic 제작  1. ScriptableObject를 상속  2. Target ,Condition, Action을 변수로 가짐 | Tactic.cs |
| 캐릭터를 첫번째로 필터링 하는 Target제작  1.ScriptableObject를 상속  2. 추상클래스로 제작, 이를 상속받는 Ally,Enemy 등 구현  3. CharacterList를 반환하는 Filter함수 보유 | Target.cs |
| Target에서 온 CharacterList를 두번째로 필터링하는 Condition 제작  1.ScriptableObject를 상속  2. 추상클래스로 제작,이를 상속받는 HP<50 , Nearlest 등 제작  3. CharacterList를 반환하는 Filter함수 보유  4. 여러 Condition을 조합할 수 있는 MixedCondition제작 | Condition.cs  MixedCondition.cs |
| 실질적으로 애니메이션, 근접공격등의 행동을 지시하는 Action 제작.  1.ScriptableObject를 상속  2. 추상클래스로 제작, 이를 상속받는 MeleeAttack ,Heal 등 제작  3. 실질적인 동작을 하는 Execute함수를 보유 | Action.cs |
| Character 제작 | Character의 전반적인 변수를 보유하고 있는 CharacterStat 생성  1. isEnemy 변수를 통해 Hero/Enemy분리  2. target,condition,action의 리스트를 보유  3. HP,GloobalCoolDown등의 변수 보유 | CharacterStat.cs |
| Character가 길을 찾으며 이동할 수 있는 Navigation System 적용.  1. NavMeshAgent를 Character에 추가  2. NavMesh Surface를 Scene에 적용  3. 이동하는 벽에는 NavMeshObstacle 추가  4. 이동불가 지역 설정 NavMeshModifierVolume을 추가, AreaType 조정 | NavMesh  GameObjects |
| Character Animation 제작 | Character Tactic list에 따른 Character animation을 적용한다.  1. Charcter의 이동 animation 적용  2. Character의 공격 animation 적용  3. Character의 스킬 animation 적용  4. Character의 군중제어 animation 적용  5. Charater의 사망 animation 적용  6. 전투후 승리/패배의 Character animation 적용 | CharaterAnimator  (Animator Controller),  Action.cs |
| Effect 제작, Projectile 제작 | Character의 공격과 스킬의 이펙트 효과를 추가한다. 투사체의 경우 투사체 추가 animation을 추가한다. | Effect.cs  Projectile.cs |
| TacticSystemUI  제작 | 런타임에 캐릭터의 TacticSystem을 수정 UI 팝업  1. TacticUIManager를 제작해서, TacticUI 팝업 및 초기화, 업데이트를 관리. | TacticUIManager.cs |
| DropDown UI를 통해 Tactic의 target,condition,action 수정  1. 팝업 시, CharacterStat의 target,condition,action  리스트를 드랍다운 리스트에 추가.  2. null을 나타내는 Empty 속성 추가  3.Priority가 1인 Tactic은 Enemy ,Nearlest, MeleeAttack으로 고정.  4. Dropdown의 interactable을 조정 | DropDownUI |
| Toggle을 통해 각 Tactic의 Enable 속성 변경  1.Toggle의 IsOn과 Tactic의 Enable을 연동하여 적용. | ToggleUI |
| DragDropUI를 통해 Tactic의 Priority를 변경.  1. DraggableTactic에서 EventHandler를 통해  드래그 시작, 도중, 종료를 감지한다.  2. TacticSlot에서는 Slot이 비어있으면 Tactic을 배치하고, 아니면 Tactic의 Priority 교체. | DraggableTactic.cs  TacticSlot.cs |
| Wave 시스템 | BattleManager는 Wave에 대한 관리 및 진행 담당  1. Character가 죽었을때 이벤트를 발생시킴.  2. BattleManager는 이벤트를 구독한다. | BattleManager.cs |
| MapScene에서 GameManager를 통해 얻어온 WaveData를 통해 Monster를 BattleScene에 배치 | WaveData.cs를 이용해 Scene에 배치 |
| BattleManager가 Wave가 끝났음을 판단한다.  1. Wave가 끝나면 다음 Wave가 진행될 위치로 Player 및 캐릭터들을 강제 이동.  2. 이동하기 전에 Character의 TacticSystem을 끈다.  3. Player와 Character가 Wave가 진행될 위치에 도착하면 TacticSystem을 켠다. | WaveSpot,  BattleManager 기능추가 |
| ResultUI 제작 | BattleManager에서 승리 혹은 패배를 판단한다.  1. CharacterManager에 있는 CharacterList를 통해, Hp가 0인지 판단하여, 승리 혹은 패배를 결정.  2. MapScene으로 돌아가는 버튼을 팝업하고,  3. 승리시에는 보상 목록도 팝업.  4. MapScene에서 보상목록에 있는 보상을 Player 에게 추가. | ResultUI |
| Character Save 기능 | Json 을 이용해서 데이터를 저장한다.  저장할 데이터는 CharacterStat, TacticSystem에 있는 정보. | Save\_n.json |
| Sound 추가 | 플레이 상황에 맞는 소리를 설정한다.  1. map sound : 일반모드와 맵환경에 따른 전투모드의 BGM을 추가  2. character sound : 캐릭터들의 이동, 공격, 피격, 목소리의 SFX 추가  3. projectile sound : 투사체들의 sound 추가  4. effect sound : 이펙트들의 sound 추가 | AudioManager.cs  SoundManager.cs  AudioMixer  Sound(.wav .mp3 .ogg) |
| Character UI 추가 | 전투시 Character의 체력과 마나,현재 실행중인 tactic 상태를 알려주는 UI를 추가한다. | CharacterUI.cs |

**MapScene**

|  |  |  |
| --- | --- | --- |
| UiManager | GameManager에서 instance로 생성 | GameManager.cs |
| MapUiManger | UiManger을 상속받아서 Map을 관리할 수 있도록 함 | UiManagers.cs |
| Node 관련 디자인 | Random.Range(n,m)을 활용하여 좌표를 생성 | CreateNode.cs |
| 생성된 좌표를 들로네 삼각분할법을 이용하여 최대한 균일하게 연결 | CreateNode.cs |
| 노드 클릭시 노드 관련 ui active(true) | UiEvent.cs |
| X 버튼 클릭 시 active(false) | UiEvent. |
| Node UI | Hero 목록을 list로 관리하여 Roaster에서 나타남 | List.cs |
| PositionGrid에서 선택된 hero 무리와 위치 정보를 담고있음 | PositionGrid.cs |
| Creep 목록을 list로 관리 하여 나타냄 | List.cs |
| Local Boss 정보를 나타냄 | Icon.cs |
| EnemyNode라고 정해지면 Creep과 Boss의 정보를 Data에 기록된 set중 무작위로 결정 | SetNodeSide.cs  WaveData.cs  SetWave.cs |
| 지역이 특수적이 나오는지에 대한 정보 | GameManager.cs |
| Start를 누르면 Grid내 정보와 해당 유닛의 정보를 전투로 전달 | GameManager.cs |
| MapUi | 지역에 관한 정보와 이미지 출력 | Icon.cs |
| Map 이동 및 확대축소 | CameraController.cs |
| Map에 있는 상점 아이콘을 누르면 상점 ui 등장 | UiEvent.cs |
| 게임 내 시간 변화를 시계 ui의 변화를 통해 나타냄 | UiEvent.cs |
| 특정 EnemyNode에 특수 적을 할당 | GameManger.cs |
| 시간이 밤이 되면 적이 아군 Node를 한곳 점령 | TakeEvnet.cs |
| Player가 적 Node를 점령하면 ally 설정 | TakeEvent.cs |
| 상점 UI | 구매 가능한 Hero 목록을 list로 관리하여 나타냄 | List.cs |
| 가지고 있는 Gold를 ui상에 표현 | ResourceManager.cs |
| 설정 UI | Map에 있는 설정 버튼을 누르면 Game을 멈추고 설정 ui 등장 | GameManager.cs  UiEvent.cs |
| 오디오 설정 관련 | ResourceManager.Cs |
| 화면 관련 설정 | ResourceManager.Cs |
| Save & Load | SaveAndLoad.cs |
| 게임종료 | GameManger.cs |
| Option 버튼을 통해서 오디오 설정 ui와 화면 설정 ui로 넘어 가기 | UiManager.cs  UiEvent.cs |
| Playerdata관리 | Player의 획득 골드 목록과 획득 hero 목록 | PlayerData.cs  Resource.cs |
| UnitData관리 | Unit의 스탯관련 data | UnitData.cs |
| Hero의 현재 채력,마나 상태 | ResourceManger.cs  GameManger.cs |

**LoginScene**

|  |  |  |
| --- | --- | --- |
| 게임 시작 시 나오는 화면 | Load | SaveAndLoad.cs |
| Start | GameManager.cs |
| Option과 누르면 audio , screen 설정 ui 출력과 조절 가능 | UiManager.cs  Resource.cs |
| Quit | GameManager.cs |

## 개발 결과

### 결과물 목록 및 상세 사양

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 대분류 | 소분류 | 기능 | 비고 | 비고 |
| 맵 | *맵 화면 이동* | 화면을 드래그해서 이동하거나 줌인 줌아웃 가능 |  |  |
| *노드 열기* | 노드 버튼 클릭시 노드 ui 출력 |  |  |
| *상점 이동* | 상점 버튼 클릭시 상점 ui 출력 |  |  |
| *설정 창 이동* | 설정 버튼 클릭시 설정 ui 출력 |  |  |
| *점령Event* | 적이나 아군이 점령시 점령한 대상의 점령지로 교체 | 특수적이 등장하는것도 특수적이 해당 노드를 점령했다고 표시 |  |
| 상점 | *영웅 구매* | 플레이어가 보유하지 않은 영웅 캐릭터 구매 |  |  |
| 설정 | *옵션* | 오디오 화면 관련 설정 가능 |  |  |
| *저장,로드* | 현재 상태를 저장하거나 가장 최근에 저장한 정보를 가져옴 |  |  |
| *현재 게임 종료* | 현재 게임을 종료함 |  |  |
| 초기 화면 | *새 게임* | 새로운 게임 시작가능 |  |  |
| *이어하기* | 저장한 가장 최신 정보를 로드 |  |  |
| *옵션* | 오디오 , 화면 관련 설정 가능 |  |  |
| *게임 종료* | 게임 종료 |  |  |
| 노드 | *유닛 배치,선택* | Hero를 목록에서 dragdrop을 통해 선택과 배치를 한번에 할 수 있음 |  |  |
| *hero목록* | 보유 중인 hero 목록을 보여줌 |  |  |
| *creep 목록* | 해당 지역의 creep의 정보와 개체수를 보여줌 |  |  |
| *보스 정보* | 보스의 정보를 보여줌 |  |  |
| *지역 정보* | 지역의 상태와 특성을 보여줌 |  |  |
| *전투 시작* | 전투 시작시 유닛 배치 밑 선택 영웅 목록 데이터를 전투 scene으로 전전달 |  |  |
| 플레이어 | *플레이어 제작* | 화면이동 및 캐릭터 선택 |  |  |
| 캐릭터 | *택틱시스템* | Target,Condition,Action을 조정하는 TacticSystem 제작 |  |  |
| *택틱시스템UI* | 드랍다운으로 캐릭터의 Tactic 수정 |  |  |
| *캐릭터 제작* | 택틱시스템을 통해 동작하고  캐릭터 스탯을 가지고 있다 |  |  |
| *캐릭터 애니메이션 디자인* | 이동, 공격, 피격, 군중제어, 사망, 승리/패배 애니메이션 제작 |  |  |
| *상태창 UI* | 캐릭터들의 상태를 보여주는 UI를를 제작 |  |  |
| 액션 | *이펙트 제작* | 캐릭터 공격, 피격 이펙트 효과 제작 |  |  |
| *투사체 제작* | 캐릭터 투사체 애니메이션 제작 |  |  |
| 전투 | *웨이브 시스템 제작* | Wave 종료 판정 및 플레이어 Wave위치로 이동 |  |  |
| *전투결과 확인* | 결과 및 보상 확인,  MapScene으로 이동 |  |  |
| 소리 | *소리 생성* | *BGM, SFX 설정* |  |  |
| 저장 | *캐릭터 저장* | CharacterStat, TacticSystem 저장 |  |  |

# **프로젝트 팀 구성 및 역할 분담**

**프로젝트에 참여하는 멤버의 역할을 구체적으로 명시한다.**

| 이름 | 역할 |
| --- | --- |
| 최치원 | - TacticSystem 개발  - TacticSystemUI, ResultUI 디자인  - Character TacticSystem Action 추가 |
| 김동휘 | * Character Asset 결정 * Character Animation 적용 * Enemy TacticSystem 디자인 * Character UI 디자인 |
| 유민석 | * MapUi , Node 개발 및 디자인 * 로그인 화면 , Setting 개발 및 디자인 |
| 강문정 | * MapUi , NodeUi 개발 및 디자인 * Ui 관련 Asset 결정 * Shop 개발 및 디자인 |

## 개발 일정

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **항목** | **세부내용** | **3월** | **4월** | **5월** | **6월** | **비고** |
| 요구사항분석 | 요구 분석 |  |  |  |  |  |
| SRS 작성 |  |  |  |  |  |
| 관련분야연구 | 주요 기술 연구 |  |  |  |  |  |
| 관련 시스템 분석 |  |  |  |  |  |
| 설계 | 시스템 설계 |  |  |  |  |  |
| 구현 | 코딩 및 모듈 테스트 |  |  |  |  |  |
| 테스트 | 시스템 테스트 |  |  |  |  |  |

## 일정별 주요 산출물

|  |  |  |  |
| --- | --- | --- | --- |
| 마일스톤 | 개요 | 시작일 | 종료일 |
| 계획서 발표 | 개발 환경 완성 (GCC 설치, 기본 응용 작성 및 테스트 완료)  **산출물 :**   1. 프로젝트 수행 계획서 2. 프로젝트 기능 일람표 | 2025-03-06 | 2025-03-20 |
| 설계 완료 | 시스템 설계 완료  **산출물 : prototype**   1. 시스템 설계 사양서 2. 게임 초기 단계 | 2025-03-06 | 2025-03-31 |
| 1차 중간 보고 | **산출물 : 1차 중간 보고서**   1. 프로젝트 1차 중간 보고서 2. 프로젝트 진도 점검표 3. 1차분 구현 소스 코드 | 2025-03-21 | 2025-04-10 |
| 2차 중간 보고 | **산출물 : 2차 중간 보고서**   1. 프로젝트 2차 중간 보고서 2. 프로젝트 진도 점검표 3. 2차분 구현 소스 코드 | 2025-04-11 | 2025-04-24 |
| 구현 완료 | 시스템 구현 완료  **산출물: Game v1.0** | 2025-04-25 | 2025-05-10 |
| 테스트 | 시스템 통합 테스트  **산출물: 테스트 결과 보고서** | 2025-05-11 | 2025-05-26 |
| 최종 보고서 | 최종 보고  **산출물: 최종 보고서** | 2025-06-03 | 2025-06-10 |