MAPLE WAR

작성자 : 안승현

목차

- Maple War 소개
- Maple MOD 소개
- 계획
  - 1주차
  - 2주차
  - 3주차
  - 4주차
- 구현에 어려웠던 점
- 해결
- 개발 후 느낀점 혹은 추후 개선 방향

MAPLE WAR 소개

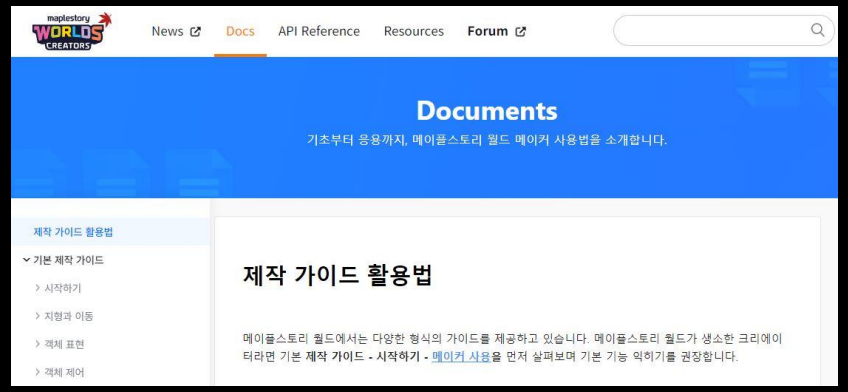
- NEXON의 게임 엔진인 Project MOD를 사용하여 구현
- 프래그래밍 언어 Lua 언어를 사용
- 횡 스크롤 디펜스 대전 게임 (전쟁 시대를 기반으로 재구성)
- 멀티플레이 기능 지원
- https://youtu.be/PsPQbX_L96o (YouTube 영상)

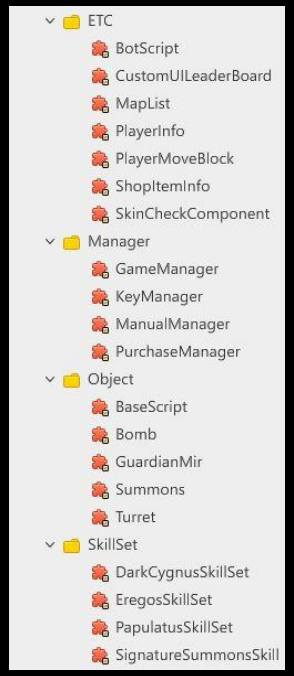
PROJECT MOD 소개

- 2021년 Nexon에서 Unity 엔진을 자체 커스텀하여 제작한 게임 엔진 입니다.
- 누구 쉽게 게임 개발을 할 수 있게 블록코딩과 쉽게 네트워킹 게임을 제작할 수 있는 특징이 있습니다.
- 현재 Maple World로 명칭이 변경되었습니다.

Maple World 공식 사이트
https://maplestoryworlds.nexon.com/ko/play

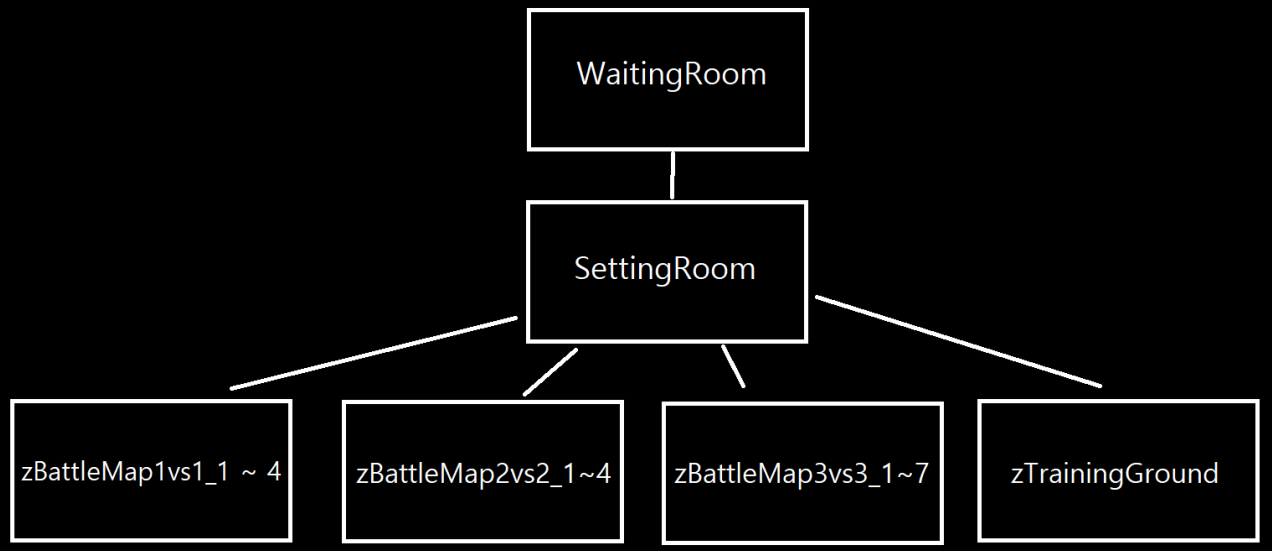
1주차 – 구조 설계





- Project MOD의 문서 살펴보기 (Lua 언어, 엔진 사용법)
- 컨텐츠에 사용될 리소스 찾기
- 게임 흐름, 클래스 구조 설계

1주차 – 구조 설계



씬 구조

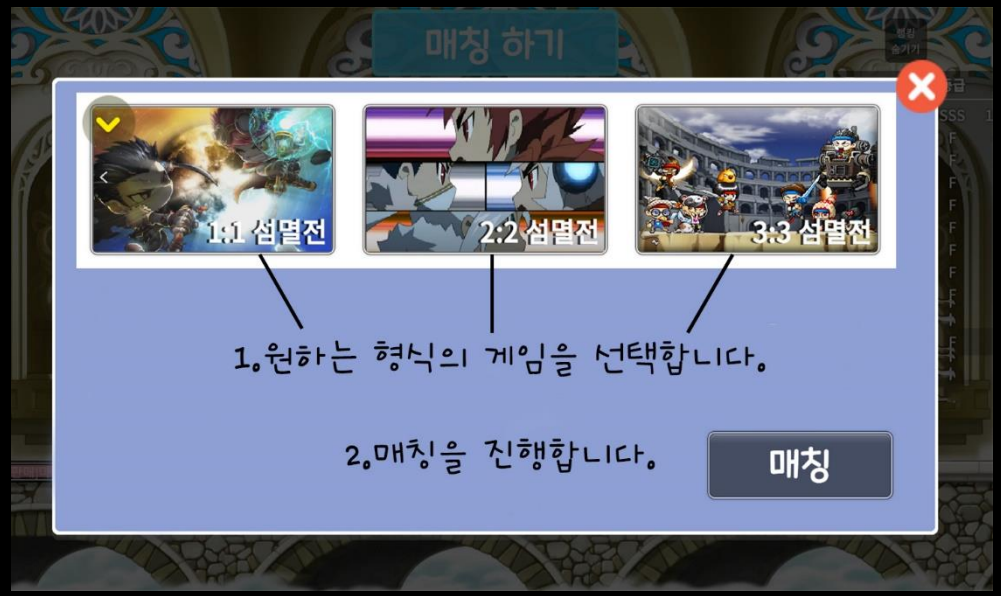
- WaitingRoom 시작 씬
- SettingRoom 밑으론 인스턴스 씬으로 동적으로 씬을 생성
- 인스턴스 씬 특징상 문자열 중 가장 빠른 문자열의 씬이 인스턴스의 시작 씬이 되므로 SettingRoom이 먼저 오고 나머진 씬은 씬 이름 앞에 z문자를 추가하여 설정

1주차 – 구조 설계

WaitingRoom 씬 제작

- 게임 시작시 입장 하는 씬
- 타일 및 NPC 배치
- WaitingRoom 씬 UI 제작

2주차 – 랜덤 매치 기능 및 인스턴스 씬 구현



- 매칭 UI

- 플레이할 컨텐츠를 선택해 등록하는 단계입니다.

2주차 – 랜덤 매치 기능 및 인스턴스 씬 구현

entity MatchingSettingButton (/ui/WaitingRoomGroup/MatchingSettingButton)

```
HandleButtonClickEvent ( ButtonClickEvent event )
{
   -- Parameters
   local Entity = event.Entity
   ------------------------------------------------------------
   _SoundService:PlaySound("7eda004ded944e459f1fbb16c7f0f3f4",1)
   self:InitSetting()
}
```

```
void InitSetting ( )
{
   self.UIChat.Enable=false
   self:AIModSelect(false)
   self:ModeSelect(1)
   self.MWBGMask.Enable=true -- 창 활성화
}
```

- 매칭 창
- 매칭 하기 버튼을 눌러 InitSetting()을 호출합니다.
- 채팅 비활성화, ai 모드 비활성화, 1:1 모드, 매칭 창 활성화를 수행합니다.

2주차 – 랜덤 매치 기능 및 인스턴스 씬 구현

```
void ModeSelect ( number index )
{
    self.modIndex = index
    if index == 0 then
        self.CheckIcon0.Enable = true
        self.CheckIcon1.Enable=false
        self.CheckIcon2.Enable=false
        self.CheckIcon3.Enable=false
        self.ExplanationText.Text = self.mode0Comment
        self.modeName = "훈련장"
    elseif index == 1 then
        self.CheckIcon0.Enable = false
        self.CheckIcon1.Enable=true
        self.CheckIcon2.Enable=false
        self.CheckIcon3.Enable=false
        self.ExplanationText.Text = self.mode1Comment
        if self.isAIMode == false then
            self.modeName = "1:1 섬멸전"
        else
            self.modeName = "1:1 섬멸전 (AI)"
        end

    elseif index ==2 then
        self.CheckIcon0.Enable = false
        self.CheckIcon1.Enable=false
        self.CheckIcon2.Enable=true
        self.CheckIcon3.Enable=false
        self.ExplanationText.Text = self.mode2Comment
        if self.isAIMode == false then
            self.modeName = "2:2 섬멸전"
        else
            self.modeName = "2:2 섬멸전 (AI)"
        end
    elseif index ==3 then
        self.CheckIcon0.Enable = false
        self.CheckIcon1.Enable=false
        self.CheckIcon2.Enable=false
        self.CheckIcon3.Enable=true
        self.ExplanationText.Text = self.mode3Comment
        if self.isAIMode == false then
            self.modeName = "3:3 섬멸전"
        else
            self.modeName = "3:3 섬멸전 (AI)"
        end
    end
}
```

- 매칭 모드 선택 코드
- ModeSelect()은 정수형 index을 받는 함수 입니다.
- Index 변수의 값에 따라 해당 게임 모드를 설정합니다.
- 0(훈련장), 1(1:1 섬멸전), 2 (2:2 섬멸전), 3 (3:3 섬멸전)

2주차 – 랜덤 매치 기능 및 인스턴스 씬 구현

```
void AIModSelect ( boolean isShow )
{
    if isShow == true then
        self.isAIMode = true
        self.TrainingBtn.ButtonComponent.Enable = true
        self.TrainingBtn:GetChildByName("ImageUISprite").SpriteGUIRendererComponent.Color.a = 1
        self.AIIcon1.Enable = true
        self.AIIcon2.Enable = true
        self.AIIcon3.Enable = true
        self.AImodeCheckIcon.Enable = true
    else
        self.isAIMode = false
        self.TrainingBtn.ButtonComponent.Enable = false
        self.TrainingBtn:GetChildByName("ImageUISprite").SpriteGUIRendererComponent.Color.a = 0.5
        self.AIIcon1.Enable = false
        self.AIIcon2.Enable = false
        self.AIIcon3.Enable = false
        self.AImodeCheckIcon.Enable = false
    end

    if self.modIndex == 0 then
        self:ModeSelect(1)
    else
        self:ModeSelect(self.modIndex)
    end
}
```

- AI 모드 선택
- AI 버튼을 눌러 AIModeSelect() 함수를 호출 할 수 있습니다.
- isShow 매개변수가 true일시 해당 모든 선택창의 이미지 하단에 AI 이미지가 활성화 됩니다.
- 만약 연습 모드일 경우 강제로 1:1 섬멸전으로 모드가 변경 됩니다.

2주차 – 랜덤 매치 기능 및 인스턴스 씬 구현

```
void SearchingStart ( )
{
   self.UIChat.Enable=true
   self.MWBGMask.Enable=false -- 창 닫고 매칭
   self.MatchingInfo.Enable=true
   self.matchTime = 0
   self.MatchingCancelButton.Enable=true
   _UserService.LocalPlayer.PlayerInfo:OnSearch(true,self.modIndex,self.isAIMode)
   self.isSerching=true
}
```

```
void SearchingCancel ( )
{
   self.UIChat.Enable=true
   self.MatchingInfo.Enable=false
   self.MatchingCancelButton.Enable=false
   self.MatchingSettingButton.Enable=true
   _UserService.LocalPlayer.PlayerInfo:OnSearch(false,-1,false)
   self.isSerching=false
}
```

- 매칭 시작
- 모드 선택 후 매칭 시작 버튼을 누르면 SearchingStart() 함수가 호출됩니다.
- 채팅창, 매칭 진행 UI 활성화, 매칭 UI 비활성화, 매칭 타이머, 현재 매칭 변수 초기화
- LocalPlayer의 PlayerInfo 클래스의 OnSearch() 함수를 서버에게 호출 요청 서버는 현재 매칭 정보를 저장합니다.

2주차 – 랜덤 매치 기능 및 인스턴스 씬 구현

```
void OnUpdate ( number delta )
{
   if self:IsClient() then
       if self.isSerching then
           local time = self.matchTime
           local mm = math.floor(self.matchTime / 60)
           local ss = math.floor(math.fmod(self.matchTime,60))
           self.MatchingInfo.TextComponent.Text = self.modeName.."\n"..string.format("%02d:%02d",mm,ss) .. ""
           self.matchTime = self.matchTime + delta
       end
   else
       if self.curTime >= self.refreshTime then
           self.curTime=0
           if self.isCheckSerahcingPlayer == false then
               self:CheckSearchingPlayer()
           end
       else
           self.curTime = self.curTime + delta
       end
   end


}
```

- 매칭 카운터(OnUpdate)
- 클라이언트는 만약 isSearching == true 면 현재 매칭 시간을 시분초로 변환하여 UI에 표시해 줍니다.
- 서버는 정해진 시간마다 매칭 요청한 플레이어를 감지하여 매칭시켜 줍니다.

2주차 – 랜덤 매치 기능 및 인스턴스 씬 구현

```
server only
void CheckSearchingPlayer ( )
{
    local waitMap = _EntityService:GetEntityByPath("/maps/WaitingRoom")
    if waitMap == nil then
        return
    end

    local users = _UserService.UserEntities
    if users == nil then
        return
    end
    self.isCheckSerahcingPlayer = true
    wait(1)

    -- PVE 테이블
    -- 준비된 플레이어 수와
    -- 플레이어를 담을 테이블 생성
    -- 훈련장
    local TGTable1 = {}
    local PVETable1 = {}
    local PVETable2 = {}
    local PVETable3 = {}

    -- PVP 테이블
    -- 준비된 플레이어 수와
    -- 플레이어를 담을 테이블 생성
    -- 1:1
    local PVPTable1 = {}
    -- 2:2
    local PVPTable2 = {}
    -- 3:3
    local PVPTable3 = {}
```

```
    -- 준비된 플레이어 테이블에 삽입후 카운트
    for k,v in pairs(users) do
        local user = v.PlayerInfo
        if user.isSearching then
            if user.modIndex == 0 then
                table.insert(TGTable1,user)
            elseif user.modIndex == 1 then
                if user.isAIMode == false then
                    table.insert(PVPTable1,user)
                else
                    table.insert(PVETable1,user)
                end
            elseif user.modIndex == 2 then
                if user.isAIMode == false then
                    table.insert(PVPTable2,user)
                else
                    table.insert(PVETable2,user)
                end
            elseif user.modIndex == 3 then
                if user.isAIMode == false then
                    table.insert(PVPTable3,user)
                else
                    table.insert(PVETable3,user)
                end
            end
        end
    end

    self:SeperateMatchUser(TGTable1,#TGTable1,1,true) -- 훈련
    self:SeperateMatchUser(PVETable1,#PVETable1,1,true) -- 1:봇
    self:SeperateMatchUser(PVETable2,#PVETable2,2,true) -- 2:봇
    self:SeperateMatchUser(PVETable3,#PVETable3,3,true) -- 3:봇
    self:SeperateMatchUser(PVPTable1,#PVPTable1,2,false) -- 1:1
    self:SeperateMatchUser(PVPTable2,#PVPTable2,4,false) -- 2:2
    self:SeperateMatchUser(PVPTable3,#PVPTable3,6,false) -- 3:3

    self.isCheckSerahcingPlayer = false
}
```

- 매칭 유저 체크
- OnUpdate 함수에서 정해진 시간마다 호출 됩니다.
- 유저정보를 통해 매칭 요청한 플레이어를 해당 테이플에 넣어줍니다.

2주차 – 랜덤 매치 기능 및 인스턴스 씬 구현

```
server only
void SeperateMatchUser ( table userTable, number readyCnt, number maxPlayer, boolean isAI )
{
    -- 유저 매칭시
    if readyCnt >= maxPlayer then
        table.sort(userTable,function(a,b) return a.matchTime > b.matchTime end)
        local addUser = 0
        self.roomCnt = self.roomCnt +1

        for k,v in pairs(userTable) do
            local user = v

            if addUser < maxPlayer then
                addUser = addUser + 1
                self:SettingTeam(addUser,v.Entity.Name,isAI)
                user:MoveToInstanceMap(self.roomCnt)
            elseif  addUser == maxPlayer then
                if readyCnt - maxPlayer >= maxPlayer then
                    readyCnt = readyCnt - maxPlayer
                    self.roomCnt = self.roomCnt + 1
                    addUser = 1
                    self:SettingTeam(addUser,v.Entity.Name,isAI)
                    user:MoveToInstanceMap(self.roomCnt)
                else
                    break
                end
            end
        end
    end
}
```

- 매칭 카운터
- 각 테이블별로 해당 유저들을 매칭시켜 인스턴스로 이동시켜주는 함수 입니다.
- 매칭 우선 순위는 매칭 시간이 오래된 순으로 내림차순 정렬을 실행 합니다.

2주차 – 랜덤 매치 기능 및 인스턴스 씬 구현

```
server only
void SettingTeam ( number count, string playerName, boolean isAI )
{
local playerData = _DataStorageService:GetDataStorage(playerName.."_PlayerInfoData")

   if isAI == false then
       if count % 2 ~= 0 then
           playerData:SetAsync("teamNum",tostring(1),nil)
       else
           playerData:SetAsync("teamNum",tostring(2),nil)
       end
   else
       playerData:SetAsync("teamNum",tostring(1),nil) -- 1팀 고정
   end
}
```

- 팀 설정
- 매칭 성공시 해당 유저의 팀을 설정 하는 함수 입니다.
- 해당 플레이어의 DB를 가져와 teamNum에 해당 팀 정보를 저장합니다.

2주차 – 랜덤 매치 기능 및 인스턴스 씬 구현

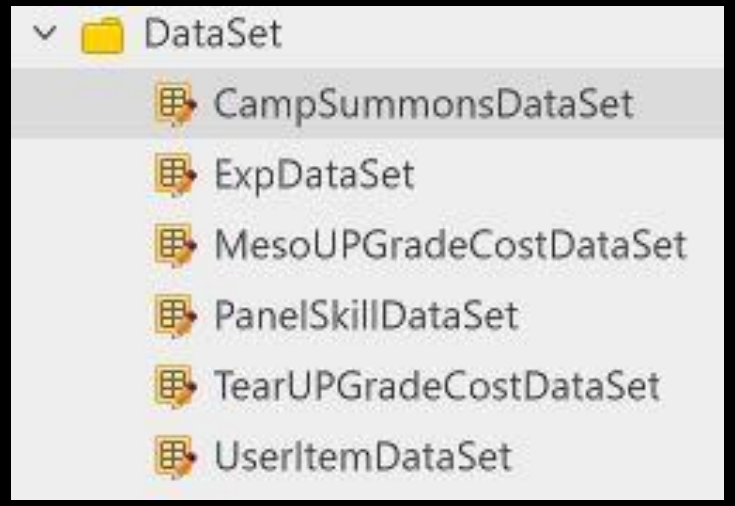
```
void MoveToInstanceMap ( number roomNum )
{
    self:MoveToInstanceMapServer(self.Entity.Name,roomNum)
}

void MoveToInstanceMapServer ( string requestedUserId, string roomNum )
{
    local instanceMap = _InstanceMapService:GetOrCreateInstanceMap("Room"..roomNum)
    instanceMap:MoveUser(requestedUserId)
}
```

```
server
void GoWaitRoom ( )
{
    _InstanceMapService:MoveUserToParentMap(self.Entity.Name)
}
```

- 인스턴스 씬
- 해당 roomNum를 이용해 고유한 인스턴스 씬으로 이동합니다.
- 각 인스턴스씬은 “Room” + 숫자로 구분합니다.
- GoWaitRoom은 부모 맵으로 돌아 갑니다. (WaitingRoom으로 이동)

3주차 – 진영 선택, 소환수, 및 기능 구현



DataSet

- CampSummonsDataSet

- 각 진영의 소환수들의 데이터를 정의한 데이터셋

- ExpDataSet

- 플레이어의 Level에 따른 Exp를 정의한 데이터셋

- Meso,Tear UpgradeCostDataSet

- 메소, 티어를 업그레이드 하기 위한 비용을 설정한 데이터셋

- PanelSkillDataSet

- 상단 패널의 스킬에 대한 값을 정의한 데이터셋

- UserItemDataSet

- 유저의 NameTag, DamageSkin 을 아이템으로 정의한 데이터 셋

3주차 – 진영 선택, 소환수, 및 기능 구현

SettingRoom

- 매칭이 성사되고 플레이어가 플레이할 진영을 선택하는 씬 입니다.
- 각 진영을 선택하여 시그니처와, 생성 가능한 소환수를 확인할 수 있습니다.
- 약 60초의 선택 시간이 있으며 전원 선택 후 10초 뒤 특정 씬으로 이동합니다.

3주차 – 진영 선택, 소환수, 및 기능 구현

```
void CampBtnDown ( number index )
{
   self.Entity.SoundComponent.Enable = false
   self.Entity.SoundComponent.AudioClipRUID = self.campSummonsDataSet:GetCell(index,3)
   self.Entity.SoundComponent.Enable = true

   self.campName.Text = self.campSummonsDataSet:GetCell(index,1)
   self.campImage.ImageRUID = self.campSummonsDataSet:GetCell(index,2)
   for i=1,#self.TierImageArray do -- 15
       self.TierImageArray[i].ImageRUID = self.campSummonsDataSet:GetCell(index,5 + 4*(i-1))
   end

self:SyncCamp(_UserService.LocalPlayer.PlayerInfo,index)
}

server
void SyncCamp ( Component PlayerInfo, number Index )
{
   PlayerInfo.myCamp = Index
}
```

진영 선택

- 해당 진영을 클릭시 HandleButtonClickEvent로 CampBtnDown()을 호출합니다.
- 해당 진영의 DataSet을 참조하여 BGM을 실행합니다.
- 해당 진영의 DataSet을 참조하여 ImageRUID를 업데이트 합니다.
- 클라이언트가 SyncCamp함수를 서버에서 호출하도록 요청하여 PlayerInfo의 myCamp변수를 변경합니다.

3주차 – 진영 선택, 소환수, 및 기능 구현

```
client
void ReadyOK ( boolean isCheck )
{
   if self.isReady == false then
       self.isReady = true
       self.DisabelPanel.Enable=true
       self.ReadyBtn.Enable=false

       _UserService.LocalPlayer.PlayerInfo:OnReady(true)

       if isCheck == true then
           self:checkReadyPlayer()
       end
   end
}
```

```
server
void checkReadyPlayer ( )
{
   local users = _UserService.UserEntities
   local totalUser = 0
   local count= 0

   for k,v in pairs(users) do
       local user = v.PlayerInfo
       if user.isReady == true then
       count = count + 1
       end
       totalUser = totalUser + 1
   end

   if count == totalUser then
       self.Time = 0
   end
}
```

진영 선택

- 클라이언트는 진영 선택 시 Ready() 함수를 호출하며 서버에게 checkReadyPlayer() 호출해 게임 시작 여부를 확인합니다.
- checkReadyPlayer()호출 요청을 받은 서버는 해당 씬의 유저들을 순회하면서 게임 준비 여부를 확인하여 만족 시 타이머 변수 값을 0으로 설정합니다.

3주차 – 진영 선택, 소환수, 및 기능 구현

```
server only
void HoldTime ( )
{
if self.Entity.GameManager:PlayerAllJoin() == true then
    function GInit()
        self.Entity.GameManager:Init()
    end
    _TimerService:SetTimer(self.Entity.GameManager,GInit,0.00000001,false)

    local startTime = 10
    for i=1,startTime do
        _UIToast:ShowMessage(string.format("%d초 후 %s",startTime - i + 1,"게임이 시작됩니다."))
        wait(1)
    end

    local modeIndex = self.Entity.GameManager.mode
    local mapObjectPath = self.Entity.MapList:SearchOutRandomMap(modeIndex)
    self:MoveMap(mapObjectPath)

    function CEndGame()
        self.Entity.GameManager:CheckEndGame()
    end

    _TimerService:SetTimer(self.Entity.GameManager,CEndGame,0.00000001,false)
  else
    self:ReturnWR()
  end
}

client
void MoveMap ( string teleportObject )
{
  self.SettingRoomGroup.Visible=false
  self.BG.Enable=false
  self.ChatEntity.Enable=true
_UserService.LocalPlayer.PlayerInfo:Teleport(teleportObject)
}
```

진영 선택

- 타이머 변수가 0이 되면 HoldTime() 함수가 호출 됩니다.
- 만약 게임 시간 전 인원이랑 맞지 않으면 게임을 강제 종료 합니다.
- 게임 매니저 함수를 초기화하며 10초의 타이머가 지나고 랜덤 맵을 선택해 모든 플레이어를 이동시키는 로직을 수행하는 함수 입니다.
- 게임 종료를 탐지하는 타이머를 호출합니다.

3주차 – 진영 선택, 소환수, 및 기능 구현

인 게임

- 첫 입장시 자신 소유의 기지에서 카메라가 위치하게 됩니다.
- 게임은 5초 후 시작 하며 게임 시작 시 메소가 쌓이며 정해진 시간과 차례가 되면 소환이 가능합니다.
- 상단의 기술과 하단의 소환수 목록으로 상대방의 전력을 막고 기지를 부수면 되는 간단 규칙인 컨텐츠 입니다.

3주차 – 진영 선택, 소환수, 및 기능 구현

턴 제

- 매턴 마다 소환수를 소환 할 수 있습니다.
- 소환 – 휴식 – 소환 으로 진행됩니다.
- 휴식 시간일땐 훈련된 소환수들은 소환리스트에 저장되며 소환 리스트가 가득차면 더 이상 생산이 불가능 합니다.

3주차 – 진영 선택, 소환수, 및 기능 구현

```
server only
void TrainingTurnCheck ( )
{
    local redTeam = false
    local blueTeam = false
    local redCount = 0
    local blueCount = 0

    local tTurn = self.nowTurn
    ::continue::
    if self.isGameSet == true then
        return false
    end
    local users = _UserService.UserEntities
    if #users == 0 then
        return false
    end

    for k,v in pairs(users) do
        local pInfo = v.PlayerInfo
        local GUC = v.GameUIController
        if pInfo.teamNum == 1 then
            redCount = redCount + 1
            if redTeam == false then
                if pInfo.myTurn == tTurn then
                    pInfo.isTrainingTurn = true
                    GUC:TurnInfo(1,v)
                    if GUC.curSaveQueueCount == 0 then
                        GUC.isSave = false
                        GUC:SetSaveText(false,v.OwnerId)
                    else
                        GUC:SaveQueueSingal(v.OwnerId)
                    end
                    redTeam = true
                end
            end
        elseif pInfo.teamNum == 2 then
            blueCount = blueCount + 1
            if blueTeam == false then
                if pInfo.myTurn == tTurn then
                    pInfo.isTrainingTurn = true
                    GUC:TurnInfo(2,v)
                    if GUC.curSaveQueueCount == 0 then
                        GUC.isSave = false
                        GUC:SetSaveText(false,v.OwnerId)
                    else
                        GUC:SaveQueueSingal(v.OwnerId)
                    end
                    blueTeam = true
                end
            end
```

```
        end
    end

    if self.isAI == false then
        -- 한팀이 없을시 함수 종료
        if redCount == 0 or blueCount == 0 then
            return
        end

        -- 플레이어가 탈주하여 true가 아닐시 tTurn의 인덱스를 1 올리고 다시 반복문 돌린다
        if redTeam == false or blueTeam == false then
            local turn = self.maxPlayerCount / 2
            if tTurn  == turn then
                tTurn = 1
            else
                tTurn = tTurn + 1
            end
            goto continue
        end
    else
        -- 플레이어가 탈주하여 true가 아닐시 tTurn의 인덱스를 1 올리고 다시 반복문 돌린다
        if redTeam == false then
            local turn = self.maxPlayerCount
            if tTurn  == turn then
                tTurn = 1
            else
                tTurn = tTurn + 1
            end
            goto continue
        end

    end

    -- 15초 동안 소환 가능하다
    function TrainingTimeOverTS()
        self.isBreakTime = true
        self:TrainingTimeOver()
    end

    for k,v in pairs(users) do
        local GUC = v.GameUIController
        GUC.TurnTimer = self.TrainingTime
    end

    _TimerService:SetTimer(self,TrainingTimeOverTS,self.TrainingTime,false)
}
```

턴 제

- TrainingTurnCheck()를 통해 소환 턴을 제어 합니다.
- _TimerService:SetTimer를 통해 예약 호출 됩니다.
- 각 팀의 PlayerInfo의 myTurn변수를 통해 현재 Turn 변수의 값과 동일할 시 소환 권한을 할당합니다.
- 현재 턴 변수가 최대 플레이어 수면 다시 턴 변수를 1로 변경시켜 순환시켜 줍니다.
- _TimerService:SetTimer로 TraningTimeOverTS()가 호출되면 더 이상 소환수를 호출 할 수 없는 상태 즉 휴식 상태로 들어갑니다.

3주차 – 진영 선택, 소환수, 및 기능 구현

```
server only
void TrainingTimeOver ( )
{
    if self.isGameSet == true then
        return false
    end

    -- 소환 시간이 아니므로 false로 세팅
    local users = _UserService.UserEntities
    for k,v in pairs(users) do
        local pInfo = v.PlayerInfo
        local GUC = v.GameUIController
        pInfo.isTrainingTurn = false
        GUC.isSave = true
        GUC:SetSaveText(true,v.OwnerId)
    end

    -- 20초 정도 쉬고 다시 소환타임
    function TrainingTurnCheckTS()
        -- 현재 턴을 사이클 맞게 계산
        local maxTurn = 0
        if self.isAI == false then
            maxTurn = self.maxPlayerCount / 2
        else
            maxTurn = self.maxPlayerCount
        end

        if self.nowTurn == maxTurn then
            self.nowTurn = 1
        else
            self.nowTurn = self.nowTurn + 1
        end

        self.isBreakTime = false
        self:TrainingTurnCheck()
    end
    for k,v in pairs(users) do
        local GUC = v.GameUIController
        GUC.TurnTimer = self.HoldTime
        GUC:TurnInfo(3,nil,v.OwnerId)
    end
    _TimerService:SetTimer(self,TrainingTurnCheckTS,self.HoldTime,false)
}
```

턴 소환 끝

- 소환시간이 끝났을때 호출되는 함수 입니다.
- 해당 인스턴스에 있는 모든 유저 정보를 가져와서 훈련 불가 상태로 전환합니다.
- _TimerService::SetTimer 함수로 HoldTime후 TraningTurnCheckTS() 를 호출 하도록 합니다.
- 이 함수가 호출되면 다시 플레이어가 소환수를 소환할 수 있게 됩니다.

3주차 – 진영 선택, 소환수, 및 기능 구현

1티어

2티어

3티어

스킬

- 상단 패널에서 게임 진행에 유용한 스킬을 사용할 수 있습니다.
- LV를 올려 태크를 올리면 더 강력한 스킬을 사용하실 수 있습니다.

3주차 – 진영 선택, 소환수, 및 기능 구현

	1	2	3	4	5	6	7	8
ID	id	ImageRUID	Cost	CoolTime0	CoolTime	CoolTime2	CoolTime3	value
1	시간 증폭	00a2e9a99e314		50	105	210	315	2
2	몽키 포탑 Lv 1	09821c31e9f741	50	1	1	1	1	
3	몽키 포탑 Lv 2	05396735ae0b4	100	1	1	1	1	
4	몽키 포탑 Lv 3	0cf75431d1294(	150	1	1	1	1	
5	수호 미르 Lv 1	0d590811c1d54	50	45	90	130	240	
6	수호 미르 Lv 2	0bc2df2f99074a	100	50	100	150	270	
7	수호 미르 Lv 3	0327692fc8964(	150	65	110	180	300	
8	폭탄	094e5dddf6da4	0	5	5	5	5	
9	시그너스	0017858b7d9d4	600	70	140	280	420	
10	파풀라투스	02d581a8c88c4	700	75	150	290	430	
11	에레고스	02db3052746144	550	60	120	260	400	

```
server
void TimeAmplification ( )
{
   if self.curPS0CoolTime == 0.0 and self.isPS0SkillActive == false then
       _SoundService:PlaySound("3b153c4fb6ff438aab70ae67e8ad3034",1,self.Entity.OwnerId)
       self.isPS0SkillActive = true
       self.trainingSpeed = self.panelDataSet:GetCell(1,8)
       wait(10)
       self.trainingSpeed = 1
       self.isPS0SkillActive = false
       self.curPS0CoolTime =  self.panelDataSet:GetCell(1,4+self.pInfo.modIndex)
       self.isPS0CoolStart = true
   end
}
```

스킬

- 스킬 정보는 PanelSkillDataSet에서 관리 됩니다.
- 비용, 티어별 쿨타임이 기록되어 있습니다.

시간 증폭(스킬)

- 10초 동안 소환수의 생산 시간이 2배로 빨라집니다.

3주차 – 진영 선택, 소환수, 및 기능 구현

```
server
void TurretConstruction ( )
{
   if self.pInfo.MyBase.BaseScript.isDead == true then
       return
   end

   if self.curPS1CoolTime == 0.0 then
       local socketNum = self:SocketCheck()
       if socketNum ~= 0 then
           local curTierLevel = self.pInfo.curTierLevel
           local cost = tonumber(self.panelDataSet:GetCell(1 + curTierLevel,3))
           local setCool = tonumber(self.panelDataSet:GetCell(1 + curTierLevel,4+self.pInfo.modIndex))
           if self.pInfo.curMeso >=  cost then
               self.curPS1CoolTime = setCool
               self.pInfo:SetMeso(-cost)
               -- 포탑 소환
               local spawnPosition = nil
               if socketNum == 1 then
                   spawnPosition =  self.pInfo.MyBase.BaseScript.TurretPos
                   self.pInfo.MyBase.BaseScript.TurretSocekt = true
               elseif socketNum == 2 then
                   spawnPosition = self.pInfo.MyBase.BaseScript.TurretPos2
                   self.pInfo.MyBase.BaseScript.TurretSocekt2 = true
               elseif socketNum == 3 then
                   spawnPosition = self.pInfo.MyBase.BaseScript.TurretPos3
                   self.pInfo.MyBase.BaseScript.TurretSocekt3 = true
               end

               local parsent = self.pInfo.MyBase.CurrentMap
               local Id = self.panelDataSet:GetCell(1 + curTierLevel,9)
               local Name = "spawnEntity"
               local spawnEntitiy =
               _SpawnService:SpawnByModelId(Id,Name,spawnPosition,parsent,self.Entity.OwnerId)

               spawnEntitiy.Turret:SetInit(self.Entity,socketNum)
               self.isPS1CoolStart = true
           else
               self:NotEnoughMeso(self.Entity.OwnerId)
           end
       end
   end
}
```

포탑 건설(스킬)

- 자신의 기지가 존재하며, 최대 3까지 설치 가능 합니다.
- 현재 티어, 메소, 쿨타임, 포탑 설치 가능 여부를 확인해 해당 조건을 만족시 해당 포탑을 설치 합니다.
- 포탑은 30초 동안 유지하여 기지를 보호합니다.

3주차 – 진영 선택, 소환수, 및 기능 구현

```
server
void SummonGuardianMir ( )
{
   if self.curPS2CoolTime == 0.0 then
        local curTierLevel = self.pInfo.curTierLevel
        local cost = tonumber(self.panelDataSet:GetCell(4 + curTierLevel,3))
        local setCool = tonumber(self.panelDataSet:GetCell(4 + curTierLevel,4+self.pInfo.modIndex))
        if self.pInfo.curMeso >= cost then
            self.curPS2CoolTime = setCool
            self.pInfo:SetMeso(-cost)
            local spawnPosition = self.pInfo.MyBase.TransformComponent.Position
            local parsent = self.pInfo.MyBase.CurrentMap
            local Id = self.panelDataSet:GetCell(4 + curTierLevel,9)
            local Name = "spawnEntity"
            local spawnEntitiy = _SpawnService:SpawnByModelId(Id,Name,spawnPosition,parsent,self.Entity.OwnerId)
            spawnEntitiy.GuardianMir:SetInit(self.Entity)
            self.isPS2CoolStart = true
        else
            self:NotEnoughMeso(self.Entity.OwnerId)
        end
   end
}
```

수호자 미르(스킬)

- 현재 설정된 기지에서 소환 되어 적을 강하게 밀쳐내는 스킬입니다.
- 현재 티어, 메소, 쿨타임을 확인해 해당 조건을 만족시 미르가 소환 됩니다.
- 해당 스킬 사용시 쿨타임이 적용 됩니다.

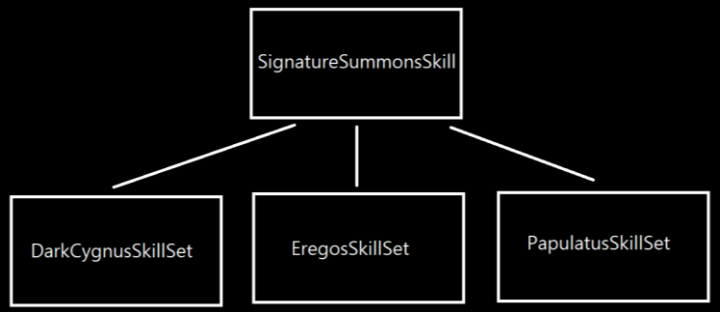
3주차 – 진영 선택, 소환수, 및 기능 구현

```
server
void SummonSignatureSummon ( )
{
   if self.pInfo.SignatureSummon == false and self.curPS3CoolTime == 0.0 then
       if self.pInfo.curTierLevel == 3 then
           if self.pInfo.SignatureSummon == false then
               local myCamp =  self.pInfo.myCamp
               local cost = tonumber(self.panelDataSet:GetCell(8 + myCamp,3))
               if self.pInfo.curMeso >= cost then
                   self.pInfo.SignatureSummon = true
                   self.pInfo:SetMeso(-cost)
                   -- 시그니처 소환수 소환
                   local playerStartPos =  Vector3(self.pInfo.MyBase.BaseScript.playerStart.x,
                       self.pInfo.MyBase.BaseScript.playerStart.y+1,
                       self.pInfo.MyBase.BaseScript.playerStart.z)
                   local spawnPosition = playerStartPos
                   local parsent = self.pInfo.MyBase.CurrentMap
                   local Id = self.panelDataSet:GetCell(8 + myCamp,9)
                   local Name = "spawnEntity"
                   local spawnEntitiy =
                   _SpawnService:SpawnByModelId(Id,Name,spawnPosition,parsent,self.Entity.OwnerId)
                   spawnEntitiy.Summons:SetInit(self.Entity, -1, -1)
                   self:UIUPdate(3,self.Entity.OwnerId)
               else
                   self:NotEnoughMeso(self.Entity.OwnerId)
               end
           end
       end
   end
}
```

시그니처 소환수 (스킬)

- 각 진영에서 뽑을 수 있는 최강의 소환수를 소환합니다.
- 플레이어당 1기씩 소유할 수 있으며 시그니처 소환수가 사망시 다시 소환할 수 있습니다.
- 시그니처 소환수는 3티어 부터 소환이 가능합니다.

3주차 – 진영 선택, 소환수, 및 기능 구현




```
server
void skill1 ( )
server
void skill2 ( )
```

시그니처 소환수 스킬

- 각 시그니처 소환수의 스킬은 SignatureSummonsSKill 클래스를 상속받아 skill1 (),skill2()를 구현합니다.
- 각 스킬은 시그니처 소환수를 클릭하거나, c키로 스킬창을 열 수 있습니다.
- 각 스킬은 소환시 1번씩만 사용이 가능합니다.

3주차 – 진영 선택, 소환수, 및 기능 구현

소환

- 하단의 버튼들을 눌러 비용을 지불하여 소환수를 소환합니다.

3주차 – 진영 선택, 소환수, 및 기능 구현

```
server
void TrainingRegister ( number index )
{
   if self.curSQueueCount < self.maxSQueueCount then
       local campIndex = self.pInfo.myCamp
       local UPIndex = 4*(index-1)
       local tierIndex = (self.pInfo.curTierLevel -1) * 20
       local cost = tonumber(self.campSummonsDataSet:GetCell(campIndex,6+UPIndex+tierIndex))

       if self.pInfo.curMeso >= cost then
           self.pInfo:SetMeso(-cost)
           local count = _LinkedListLogic:AddElement(self.summonsLinkedList,{campIndex,4+UPIndex+tierIndex})
           self.curSQueueCount = self.curSQueueCount + count

           if self.isFirstQueue == false then
               self:InitTrainingSetting()
           end

           local imageRUID = self.campSummonsDataSet:GetCell(campIndex,5+UPIndex+tierIndex)
           _LinkedListLogic:QueueImage(true,imageRUID,-1,self.Entity.OwnerId) -- 대기열 등록 -1
       else
           self:NotEnoughMeso(self.Entity.OwnerId)
       end
   end
}
```

```
server only
void TrainingSummons ( )
{
local count = _LinkedListLogic:DeleteElement(self.summonsLinkedList,0)
   self.curSQueueCount = self.curSQueueCount + count
       if self.isSave == true then
           if self.curSaveQueueCount < self.maxSaveQueueCount and self.isSaveSummonsPopup == false then
               local sCount = _LinkedListLogic:AddElement(self.saveQueue,{self.tCampIndex,self.tSIndex})
               self.curSaveQueueCount = self.curSaveQueueCount + sCount
               _LinkedListLogic:SaveQueueInitImage(true,self.tImageRUID,self.Entity.OwnerId)
           else
               self:Summons(self.tCampIndex,self.tSIndex)
           end
       else
           self:Summons(self.tCampIndex,self.tSIndex)
       end
       _LinkedListLogic:LinkedListImage(false,nil,0,self.Entity.OwnerId)
       if self.summonsLinkedList.head ~= self.summonsLinkedList.tail then
           self:InitTrainingSetting()
       else
           self.isFirstQueue = false
           self:CloseTrainingGroup(self.Entity.OwnerId)
       end
}
```

소환

- 소환시 TrainRegister를 통해 조건을 만족할 시 LinkedList 자료구조에 들어간 후 훈련에 들어 갑니다.
- 만약 소환수 저장 상태일 경우 훈련 시간이 끝나도 바로 소환되지 않고 저장 LinkedList에 저장됩니다.
- 만약 저장LinkedList까지 차버리면 더 이상 훈련이 진행되지 않습니다.
- 소환시 대기열은 자료구조 LinkedList를 직접 구현 하여 사용했습니다.

3주차 – 진영 선택, 소환수, 및 기능 구현

```
server
void SaveSummons ( )
{
  if self.pInfo.isTrainingTurn == false then
      _SoundService:PlaySound("e6341acb5d4746a2be12ea6676970c0b",1,self.Entity.OwnerId)
      _UIToast:ShowWarningMSG("소환 턴이 아닙니다.",1,self.Entity.OwnerId)
      return
  end

  if self.isSaveSummonsPopup == true then
      return
  end

  if self.isSave == true then -- 리스트에 저장하고 있었다는 소리이므로 dequeue한다
      self.isSave = not self.isSave
      if self.saveLinkedList.head ~= self.saveLinkedList.tail then -- 1개라도 있다는 소리
          self:AllSaveSummonsPopup()
          _LinkedListLogic:SaveLinkedListInitImage(false,nil,self.Entity.OwnerId)
      end
  else
      self.isSave = not self.isSave
  end

  self:SetSaveText(self.isSave,self.Entity.OwnerId)
}
```

```
server only
void AllSaveSummonsPopup ( )
{
   self.isSaveSummonsPopup = true
   while true do
       if self.saveQueue.head == self.saveQueue.tail then
           self.isSaveSummonsPopup = false
           break
       end
       local value = _LinkedListLogic:ReadElement(self.saveQueue,0)

       local campIndex = 0
       local sIndex = 0

       for k,v in pairs(value) do
           if k == 1 then
               campIndex = v
           elseif k == 2 then
               sIndex = v
           end
       end
       local count = _LinkedListLogic:DeleteElement(self.saveQueue,0)
       self.curSaveQueueCount = self.curSaveQueueCount + count
       wait(0.1)
       self:Summons(campIndex,sIndex)
   end
}
```

소환수 저장 및 방출

- SaveSummons()은 훈련이 완료된 소환수들을 LinkedList에 저장하는 함수 입니다.
- AllSaveSummonsPopup()은 저장된 소환수들을 한번에 소환하는 함수 입니다.

3주차 – 진영 선택, 소환수, 및 기능 구현

```
LinkedListLogic {
    Property:    +
        [None]
        SpriteGUIRendererComponent  QueueSprite0 = /ui/GameUIGroup/TrainingGroup/QueueBtn1/QueueImage0
        [None]
        SpriteGUIRendererComponent  QueueSprite1 = /ui/GameUIGroup/TrainingGroup/QueueBtn2/QueueImage1
        [None]
        SpriteGUIRendererComponent  QueueSprite2 = /ui/GameUIGroup/TrainingGroup/QueueBtn3/QueueImage2
        [None]
        SpriteGUIRendererComponent  QueueSprite3 = /ui/GameUIGroup/TrainingGroup/QueueBtn4/QueueImage3
        [None]
        SpriteGUIRendererComponent  QueueSprite4 = /ui/GameUIGroup/TrainingGroup/QueueBtn5/QueueImage4
        [None]
        SyncTable<Component>  QueueSpriteArr
        [None]
        Entity  SummonsImageGroup = /ui/GameUIGroup/SaveSummonsList/SummonsImageGroup
        [None]
        any  SummonsSaveImageArr = nil
        [None]
        string  emptyImage = "eb25237c9c3d2f642a84bfb3c0cfd830"

    Function:    +
        server
        void  AddElement ( table queue, SyncTable <number,  number> value )
        server
        void  ReadElement ( table queue, number index )
        server
        void  DeleteElement ( table queue, number index )
        client only
        void  OnBeginPlay ( )
        client
        void  LinkedListImage ( boolean isInput, string imageRUID, number clearImageIndex )
        client
        void  SaveLinkedListInitImage ( boolean isInput, string imageRUID )

    Entity Event Handler:    +
}
```

LinkedList

- STL 라이브러리가 지원되지 않아 직접 구현하였습니다.
- 대기열 기능을 구현할 때 삽입, 삭제가 빠르고, 임의 접근도 만약 값이 캐싱되어 있으면 빠르기에 이에 적합한 LinkedList를 선택하여 구현하였습니다.

4주차 – 훈련장 및 AI 구현

훈련장 구현

- Maple War의 게임 기능을 테스트 할 수 있는 씬 입니다.
- 레벨, 메소를 임의적으로 추가할 수 있고 전적에는 반영되지 않습니다.

4주차 – 훈련장 및 AI 구현

```
server only
void BotInit ( )
{
   if self.isInit == false then
       self.isInit = true
       self.Entity.Enable = true
       self.isBot = true
       self.Master = nil
       self.botMaster =nil
       self:SetHp()

       if(#self.Entity.Children > 0) then
           local e = self.Entity:GetChildByName("HPBarBG");
           self.HPBarBG = e;
           self.HPBarBG.SpriteRendererComponent.SortingLayer = "Npc"
           if(#e.Children > 0) then
               self.HPBar = e:GetChildByName("HP")
               self.HPBar.SpriteRendererComponent.SortingLayer = "Effect";
               self.HPBarFoll = e:GetChildByName("FollowHP");
               self.HPBarFoll.SpriteRendererComponent.SortingLayer = "Player";
           else
               self.HPBar = nil;
           end
       end

       self.playerStart = self.Entity:GetChildByName("StartPos").TransformComponent.WorldPosition
       self.TurretPos = self.Entity:GetChildByName("TurretPos").TransformComponent.WorldPosition
       self.TurretPos2 = self.Entity:GetChildByName("TurretPos2").TransformComponent.WorldPosition
       self.TurretPos3 = self.Entity:GetChildByName("TurretPos3").TransformComponent.WorldPosition

       if self.teamNum == 1 then
           self.addPosX = 1.5
       elseif self.teamNum ==2 then
           self.addPosX = -1.5
       end
       local entity = _SpawnService:SpawnByModelId("model://55f57fb2-134c-4962-8e47-6d27704249f4",
           "Bot",Vector3.zero,self.Entity,nil)
       entity.BotScript:SetBotInit()
   end
}
```

AI 구현

- BaseScript 스크립트

- AI 모드면 BotInit()함수 호출하여 관련 변수 초기화
- BotScript를 가지고 있는 게임 오브젝트를 생성하여 BotScript를 초기화 합니다.

4주차 – 훈련장 및 AI 구현

```
server
void SetBotInit ( )
{
   if self.isInit == false then
        self.isInit = true
        self.GM = _EntityService:GetEntityByPath("/maps/SettingRoom").GameManager
        self.botCamp = math.random(1,3)
        self.MyBase = self.Entity.Parent
        self.MyBase.BaseScript.botMaster = self.Entity
        self.teamNum = self.MyBase.BaseScript.teamNum
        self.botTurn = self.MyBase.BaseScript.baseTurn
        self.botName = "훈련용 봇"..tostring(math.floor(self.botTurn))
        self.MyBase.NameTagComponent.Name = self.botName
        self.campSummonsDataSet = _DataService:GetTable("CampSummonsDataSet")
        self.panelDataSet = _DataService:GetTable("PanelSkillDataSet")
        self.MUPGC = _DataService:GetTable("MesoUPGradeCostDataSet")
        self.expDataSet = _DataService:GetTable("ExpDataSet")
        self.isSummons = true
        wait(5)

        function MesoUPStart()
            self:MesoUPTimer()
        end
        _TimerService:SetTimer(self,MesoUPStart,0.000001,false)

        function MesoCoroutinStart()
            self:MesoCoroutine()
        end
        _TimerService:SetTimer(self,MesoCoroutinStart,0.000001,false)

        -- 메인 업데이트
        function MainTimerStart()
            self:MainTimer()
        end

        _TimerService:SetTimer(self,MainTimerStart,0.000001,false)

        -- 티어 증가 시간
        function TierUPTimerStart()
            self:TierUPTimer()
        end

        _TimerService:SetTimer(self,TierUPTimerStart,0.000001,false)

        -- 누적 데미지 초기화 시간
        function CDamageResetStart()
            self:CumulativeDamageReset()
        end
        _TimerService:SetTimer(self,CDamageResetStart,0.000001,false)
   end
}
```

AI 구현

- BotScript 스크립트

- Bot의 진영, 이름, 차례 등 게임 관련 변수를 초기화 합니다.
- 메소, 메소업, 티어업, 행동 함수, 누적 데미지 함수들을 타이머를 통해 호출 되도록 설계했습니다.
- Bot은 일정시간이 지나면 메소, 티어가 증가 하도록 설계했습니다.
- CDamageResetStart() 함수를 통해 기지에 들어온 누적 데미지를 통해 봇이 스킬을 사용하도록 구현하였습니다.

4주차 – 훈련장 및 AI 구현

```
server only
void MainTimer ( )
{
   while self.GM.isGameSet == false do
       wait(1)
       self.Timer = self.Timer + 1

       if self.isSummons == true and self.SummonsCool <= self.curCool then
           if self.GM.isBreakTime == false and self.botTurn == self.GM.nowTurn then
                   self.isSummons = false
                   function SummonsTimer()
                       self:Summons()
                   end
                   _TimerService:SetTimer(self,SummonsTimer,0.000001,false)

                   self:ThinkTime()
                   self.curCool = 0
           end
       else
           self.curCool = self.curCool + 1
       end
   end

   wait(5)
   self.Entity:Destroy()
}
```

```
server only
void ThinkTime ( )
{
   self.SummonsCool = math.random(0,1)
}
```

AI 구현

- BotScript 스크립트

- 매 1초마다 반복문을 순회합니다.
- 소환 조건을 충족하면 _TimerServer:SetTimer로 SummonsTimer를 예약 실행합니다.
- 예약 실행하는 이유는 Summons()함수에서 wait함수 실행할 때 MainTimer함수에 영향을 받지 않기 위함 입니다.
- ThinkTime()함수를 호출해 랜덤으로 소환 쿨타임을 적용합니다.

4주차 – 훈련장 및 AI 구현

```
if self.isBot == true and isvalid(self.botMaster) == true  then
    self.botMaster.BotScript:BotSkill(TotalDamage)
end
```

```
server only
void BotSkill (number TotalDamage )
{
    local usePercent = math.random(1,10)
    self.TurretUseDamage = self.TurretUseDamage + TotalDamage

    if self.TurretUseDamage >= 100 and usePercent <= 3 then
        self.TurretUseDamage = 0
        if self.turretCount < 3 then
            -- 포탑 설치
            self:TurretConstruction()
        end
    end

    usePercent = math.random(1,10)
    -- 미르
    self.MirUseDamage = self.MirUseDamage + TotalDamage
    if self.GM.nowTurn == self.botTurn and self.MirUseDamage >= 100 and usePercent <= 1 then
        if self.isMirUse == false then
            self:SummonGuardianMir()
        end
    end

    if self.BombCount ~= 0 then
        self.BoomUseDamage = self.BoomUseDamage + TotalDamage
        if (self.MyBase.BaseScript.curHp / self.MyBase.BaseScript.maxHp) * 100 <= 50 and
         (self.MyBase.BaseScript.curHp / self.MyBase.BaseScript.maxHp) * 100 > 5 then
            if self.BoomUseDamage >= 300 then
                -- 폭탄 투하
                self.BoomUseDamage = 0
                self:Bomb()
            end
        elseif (self.MyBase.BaseScript.curHp / self.MyBase.BaseScript.maxHp) * 100 <= 5 then
            -- 폭탄 투하
                self.BoomUseDamage = 0
                self:Bomb()
        end
    end


}
```

AI 구현

- BaseScript의 HandleHitEvent

- HandleHitEvent를 통해 들어오는 Damages 변수를 통해 BotSkill을 호출합니다.

- BotScript의 BotSkill

- HandleHitEvent를 통해 호출되며 들어온 TotalDamage 변수를 각각의 스킬 변수에 누적합니다.

- 랜덤 함수의 값과 누적된 변수를 통해 만약 일치하면 해당 스킬을 사용하는 로직 입니다.

- 일정 시간동안 대미지가 들어오지 않으면 각 변수들은 0으로 초기화 됩니다.

구현에 어려웠던 점

- Project MOD가 테스트 단계이다 보니 예상치 못한 에러가 많이 발생해 구현에 어려움이 있었습니다.
- Unity에서 지원하는 기능이 Project MOD에는 없거나 제대로 작동하지 않아서 구현에 어려움이 있었습니다. (ex 타일 맵, 애니메이션 이벤트, STL)
- 게임 리소스 및 사운드 리소스를 검색할 때 해당 이름으로 찾을 수는 없고 RUID로 찾아야하기 때문에 일일이 페이지 넘겨가며 확인하며 찾는게 너무 어려웠습니다.

해결

- 발생한 에러를 DM을 통해 문의하여 해결하였습니다. (ex 상속 문제)
- 해당 기능을 직접 구현하거나, 대체할 기능을 찾아보았습니다. (ex LinkedList)
- 시간을 투자하여 리소스를 직접 다 검색하였습니다.

개발 후 느낀점 혹은 추후 개선 방향

- 네트워크 요소를 넣으니까 구조 설계의 어려움 그리고 코드 버그 테스트 시간이 너무 오래걸렸다.
- Project MOD 설계 방식(이벤트) 대해 알게 되었다.
- 좀 더 구조를 명확하게 설계하고 코딩 해야겠습니다.