(플레이 화면에 구현되어야 할 내용만을 담았습니다)

(OOP 개념을 사용하지 않기 위해서 (Method/Field) 대신 (Function/Variable)이라는 어휘를 사용한 점 양해 부탁드립니다)

(Public이 붙은 것은 다른 Script에서 접근할 수 있는 것을 의미하며, Private가 붙은 것은 다른 Script에서 접근할 수 없는 것을 의미합니다)

# GameManager.cs

* 게임의 전반적인 수치를 관리하는 매니저 역할입니다.
* 게임 수치에 맞춰 UI 요소를 관리합니다(Private 함수)
* 플레이어가 버튼을 누르면 그걸 감지해서 HeadQuarter.cs의 Unit 소환 함수를 호출해야 합니다.
* 게임 내에서 GameManager Object의 Component로 작용합니다.
* 다음과 같은 기능을 제공하는 public 함수를 구현해야 합니다.

Public Function:

**Public Vector3 Get\_RGBValue()**

* + Unit 및 UI가 호출하며, 현재 환경의 RGB 값을 Vector3형으로 반환합니다.
  + Input Parameter: None
  + Output Parameter
    - Vector3(int Red, int Green, int Blue): 현재 환경값 수치를 반환합니다.

**Public void Destroyed(int Team)**

* + 호출되면 Team 번호의 HQ가 파괴되었다고 간주하고 승리 판정을 내립니다.
  + Input Parameter
    - int Team: HQ가 파괴된 Team의 번호입니다.
  + Output Parameter: None

Private Function:

**Private void Start()**

* + 게임이 시작될 때 단 한번 호출되는 함수
  + RGB값을 초기화 합니다.

**Private void Update()**

* + 매 Frame마다 호출되는 기본 함수
  + 플레이어의 키 입력을 감지하고, 입력된 키에 따라서 적절한 행동을 취합니다.
  + Esc: 일시정지
  + Player 1은 Q, W, E를, Player 2는 U, I, O를 누르면 각 HeadQuarter.cs의 Summon\_Unit Function을 호출한다. (인자로 각각 1,2,3을 넘겨주며, 미리 할당된 세 종류의 크리처를 소환한다)
  + Player 1은 1, 2, 3을 Player 2는 7, 8, 9를 누르면 각 HeadQuarter의 SpellManager.cs의 Spell\_Load을 호출합니다. (인자로 각각 1,2,3을 넘겨주며, 미리 할당된 세 종류의 주문을 사용할 준비를 합니다.)

Public Variable

* Int MAX\_R, MAX\_G, MAX\_B: 각 환경요소의 최대값을 나타냅니다. Editor에서 편집합니다.
* Int MIN\_R, MIN\_G, MIN\_B: 각 환경요소의 최소값을 나타냅니다. Editor에서 편집합니다.

Private Variable

# HeadQuarter.cs

* 각 HQ를 관리하는 서브 매니저 역할입니다. (HQ는 각 플레이어별로 한 개씩 있습니다)
* HQ의 체력, 오브젝트 소환, 피격시 체력 감소, 파괴되었을 때 GameManager에게 파괴되었음을 알려야 합니다. (GameManager의 Destroyed 함수를 호출)
* HQ에는 ‘에너지’ 자원이 매 초마다 1씩 생성되며, 이 자원을 소비해서 스킬 사용/ 크리쳐 소환을 수행합니다.
* 모든 종류의 Unit을 Prefab화 하여 HeadQuarter.cs에서 정해진 종류의 Prefab을 소환할 수 있게 합니다.

Public Function:

* **Public bool Summon\_Unit(int ID)**
  + GameManager에 의해 호출되며, 호출되면 ID 번호에 맞는 Unit GameObject를 생성합니다. 생성한 후 생성된 GameObject를 내부 변수(List)에 저장합니다.
  + (Summonable == True && Is\_Pause == False) == True 일 때만 소환합니다.
  + 에너지가 부족하면 소환하지 않습니다.
  + Input Parameter:
    - Int ID: 소환할 Unit의 미리 지정된 ID를 의미합니다. ID에 해당하는 종류의 Unit을 소환합니다.
  + Output Parameter:
    - 소환이 성공적으로 이루어졌는지를 나타내는 Boolean 값을 리턴합니다.
* **Public void Hit(float damage)**
  + 호출되면 피격되었다고 간주하고 체력을 damage만큼 경감합니다.
  + Input Parameter
    - Float damage: 얼마의 데미지를 입었는가를 나타냅니다, 체력이 이 수치만큼 감소합니다.
  + Output Parameter: None
* **Public float Health()**
  + 현재 체력을 Return 합니다.
  + Input Parameter: None
  + Output Parameter
    - 현재 체력(health)를 리턴합니다.

**Public void Pause\_Toggle(bool Is\_Pause)**

* + GameManager에 의해 호출되며. 호출되면 게임이 일시정지됩니다. 모든 Unit의 동작을 일시정지합니다.(Unit.cs의 Pause\_Toggle을 호출)
  + Input Parameter:
    - Bool Is\_Pause: 일시정지 여부를 나타냅니다, True면 모든 Unit을 일시정지하며, False면 일시정지를 해제합니다.
  + Output Parameter: None
* **Public void Finish\_Game()**
* **Public int Unit\_Count()**
  + 현재 Unit 수를 리턴합니다.
  + Input Parameter: None
  + Output Parameter:
    - Int Unit\_Count: 현재 유닛 수
* **Public void Unit\_Dead(GameObject Dead\_Target)**
  + Unit.cs에 의해 호출되며, 체력이 0 이하가 된 Unit을 비활성화 하고 Unit\_Count를 1 감소시킵니다.
* **Public void Toggle\_Summonable()**
  + GameMamager.cs나 같은 Object에 붙어있는 SpellManager.cs 에서 호출하며 Summonable의 값을 현재 값의 반대 값(True 🡨🡪 False)으로 변경합니다.
  + Input / Output Parameter: None

Private Function:

**Private void Start()**

* + 오브젝트가 생성이 완료된 후 단 한번 호출되는 함수
  + Summonable의 값을 True로 변경합니다.

**Private void Update()**

* + 매 Frame마다 호출되는 기본 함수
  + 매초마다 에너지를 1씩 추가합니다. 에너지가 최대치만큼 있으면 추가하지 않습니다.
    - Update가 호출되고 다음 Update가 호출될 때까지 걸린 시간에 1을 곱한 값만큼 에너지를 추가합니다. (결과적으로 초당 1씩 에너지가 추가됩니다).

Public Variable

Private Variable

Bool Summonable: 새로운 크리쳐를 소환 가능한지 여부를 나타냅니다. Default값은 True입니다.

Bool Is\_Paused: 현재 게임이 일시정지 되었는지를 나타냅니다. Default값은 False입니다.

Float Health: HQ의 현재 체력을 나타냅니다.

Int Resource, Max\_Resource: HQ가 현재 소유한 자원의 수와 최대 자원의 수를 나타냅니다.

Int Unit\_Count: 현재 Unit의 수를 나타냅니다. Summon\_Unit() 함수를 호출할 때 마다 1씩 증가하며, Removed() 함수가 호출될 때 마다 1씩 감소합니다.

# Unit.cs

* 모든 유닛의 AI 역할을 하는 코드입니다.
* 이동 / 공격 / 피격을 관장합니다.
* 각 Unit별로 고유의 공격Collider / 피격Collider가 있으며, 공격Collider와 충돌한 Unit 중 특정 Unit의 Hit 함수를 호출합니다. Collider는 Unity 편집기에서 직접 조율합니다.

Public Function:

**Public void Init(int Team, GameObject HeadQuarter)**

* + Unit이 생성된 직후 HeadQuarter에 의해 호출되며, Unit이 속한 팀의 번호를 할당합니다.
  + Input Parameter
    - Int Team: Unit이 소속된 팀의 번호입니다. 피아식별 할때 사용됩니다.
    - GameObject HeadQuarter: Unit을 소환한 HeadQuarter GameObject를 나타냅니다. Unit이 죽을 때 사용됩니다.
  + Output Parameter: None

**Public void Hit(float damage)**

* + 특정 Collider와 충돌했을 때 그 Collider가 붙어있는 Object의 Script가 호출합니다, 호출되면 피격되었다고 간주하고 체력을 damage만큼 경감합니다.
  + Input Parameter
    - Float damage: 얼마의 데미지를 입었는가를 나타냅니다, 체력이 이 수치만큼 감소합니다.
  + Output Parameter: None

Public Float Get\_Cost()

* + 이 Unit을 소환하는데 필요한 자원의 양을 Return 합니다.
  + HeadQuarter.cs에서 Object Pooling을 진행할 때 한번씩 호출되며, 게임이 시작되고 나서는 호출되지 않습니다.
  + Input Parameter: None
  + Output Parameter: Float Cost

Private Function:

**Private void Start()**

* + 오브젝트가 생성이 완료된 후 단 한번 호출되는 함수. 편집기에서 초기화되지 않은 변수를 초기화합니다.

**Private void Update()**

* + 매 Frame마다 호출되는 기본 함수
  + 현재 체력을 확인하고, 체력이 0보다 낮으면 Is\_Moveable을 False로 변경한다.
  + 공격 가능한 Object가 사정거리 내에 있는지 확인한 후, 없으면 Is\_Moveable을 True로 변경한 뒤 Move()를 호출합니다. (있으면 False로 변경합니다)

**Private void Move()**

* + 호출될 때 마다 정해진 양식의 움직임을 나타내는 함수.
  + Private void Update에서 호출되며, Is\_Moveable이 True일때만 Unit Object를 이동시킵니다.

**Private void Is\_Dead()**

* + Update()에서 호출하며, 체력이 0 이하로 떨어졌는지 확인합니다. 0 이하로 떨어졌으면 GameObject HQ의 HeadQuarter.cs Component의 Unit\_Dead(this)를 호출합니다.

Public Variable

* Int Max\_Health, Health: 최대 체력수치/현재 체력수치를 나타냅니다, Unity 편집기에서 각 Unit별로 고유의 값을 지정합니다.
* Int Max\_Energy, Energy: 최대 에너지 수치/현재 에너지 수치를 나타냅니다, Unity 편집기에서 각 Unit별로 고유의 값을 지정합니다.
* Float Cost: 이 Unit을 소환하는데 필요한 자원(에너지)의 양입니다. Unity 편집기에서 각 Unit별로 별도의 값을 지정한 뒤 Prefab화 합니다.

Private Variable

* Int MAX\_R, MAX\_G, MAX\_B: 각 환경요소의 최대값을 나타냅니다. GameManager에서 받아옵니다.
* Int MIN\_R, MIN\_G, MIN\_B: 각 환경요소의 최소값을 나타냅니다. GameManager에서 받아옵니다.
* Int CUR\_R, CUR\_G, CUR\_B: 현재 환경요소의 값을 나타냅니다. GameManager에서 받아옵니다.
* Bool Is\_Moveable: 현재 Unit.cs가 붙어 있는 Object가 움직일 수 있는지를 나타냅니다. 기본값은 True이며, 앞에 공격할 수 있는 다른 Object가 있거나 health가 0보다 작아지면 False로 변경됩니다.
* GameObject HQ: Unit을 소환한 HeadQuarter GameObject를 저장합니다.
* Float Speed: 각 Unit의 고유 속도입니다, 단위는 Unity Meter입니다.

# UI\_Enviroment.cs

* UI 요소 중 환경 수치를 나타내기 위한 스크립트입니다. RGB 속성에 따라 UI의 모양을 바꾸며, UI의 모양이 **확정되면** 관련 함수를 작성합니다.
* GameManager.cs의 Get\_RGBValue() 함수를 통해 RGB값을 받아와서 UI에 반영합니다.

Public Function:

Private Function:

**Private void Start()**

* + 오브젝트가 생성이 완료된 후 단 한번 호출되는 함수
  + GameManager의 public int MAX\_R, MAX\_G, MAX\_B, MIN\_R, MIN\_G, MIN\_B를 받아옵니다.

**Private void Update()**

* + 매 Frame마다 호출되는 기본 함수

Public Variable

Private Variable

* Int MAX\_R, MAX\_G, MAX\_B: 각 환경요소의 최대값을 나타냅니다. GameManager에서 받아옵니다.
* Int MIN\_R, MIN\_G, MIN\_B: 각 환경요소의 최소값을 나타냅니다. GameManager에서 받아옵니다.

# SpellManager.cs

* 각 HQ에서 시전되는 주문을 관리하는 서브 매니저 역할입니다.
* 플레이어가 주문을 사용하고자 할 때 SpellManager.cs에서 플레이어의 입력을 받고 주문을 시전합니다.

Public Function:

**Public void Spell\_Load(int Spell\_ID)**

* + GameManager에 의해 호출되며, 호출되면 ID 번호에 맞는 Spell을 사용할 준비를 합니다. 그와 동시에 HeadQuarter.cs의 ToggleSummonable()을 호출하여 크리처 소환을 막습니다.
  + Input Parameter:
    - Int ID: 소환할 Unit의 미리 지정된 ID를 의미합니다. ID에 해당하는 종류의 주문을 사용할 준비를 합니다.

Private Function:

**Private void Start()**

* + 오브젝트가 생성이 완료된 후 단 한번 호출되는 함수

**Private void Update()**

* + 매 Frame마다 호출되는 기본 함수
    - 주문이 특정 좌표에 캐스팅되는 경우면 Player 1은 A, D를 Player 2는 J, L을 이용해 주문의 좌표를 좌우로 이동한다.
    - Player 1와 Player 2는 각각 S, K를 누름으로써 주문을 발동시킬 수 있다.
    - 주문이 좌표를 지정할 필요가 없다면 바로 발동한다.
      * *주문을 발동하면 어떤 함수를 호출할지 여부는 주문의 종류가 다 정해지면 하나하나 설계할 예정입니다.*
    - 주문 발동이 끝나면 같은 Script의 Spell\_UnLoad()를 호출합니다.
  + Input / Output Parameter: None

**Private void Spell\_UnLoad()**

* + Update()에서 주문 시전이 완료되면 호출되며, HeadQuarter.cs의 ToggleSummonable()을 호출하여 크리처 소환을 다시 허용합니다.
  + Input / Output Parameter: None

Public Variable

Private Variable

* Bool Is\_Toggled: 주문 사용 준비 중인지를 나타냅니다. Default는 False입니다.
* Int Spell\_ID: 현재 사용 대기중인 Spell의 식별번호를 저장합니다. Default는 1입니다.