



# AnimalPark

<the 동물 사냥꾼>

## < 목차 >

1. 게임의 배경
2. 게임의 구성
3. 게임의 진행
4. 코드구성

# AnimalPark

<the 동물 사냥꾼>

## 1. 게임의 배경

유전자 변형을 당한 동물들이 주민들을 위협하며 자신들의 영역을 넓히고 있습니다.

삶의 터전에서 고통받던 주민들은 전설적인 사냥꾼을 고용하게 되고,  
사냥꾼은 위협적인 능력을 가지게 된 동물들을 사냥하러 그들의 서식지로 출발합니다.

# AnimalPark <the 동물 사냥꾼>

## 2. 게임의 구성\_ 지역별 몬스터 리스트

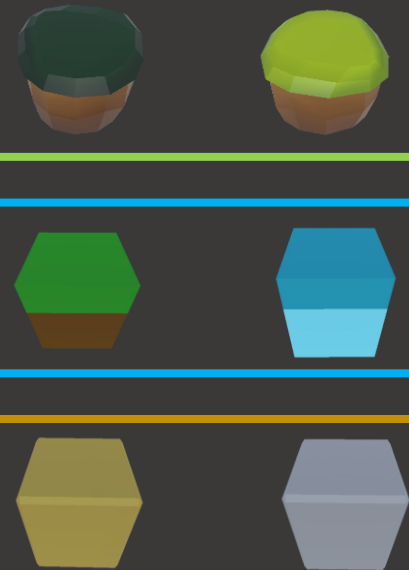
### 동물 List

- Mushroom(대왕버섯)
- Bomb(폭탄버섯)
- Elegator(악어)
- Flower(식인꽃)
- Cactus(선인장)
- Mole(두더지)

### Stage List

- Nature
- Island
- Desert

### 블록 List



# AnimalPark

<the 동물 사냥꾼>

## 2. 게임의 구성\_ 지역별 몬스터 리스트

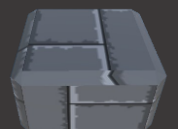
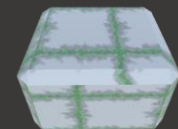
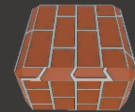
### 동물 List

- Bleed(출혈 펀처)
- Stun(기절 펀처)
- Muskrat(쥐)
- Colobus(원숭이)
- Chomper(외계생물)

### Stage List

- City
- Space

### 블록 List



# AnimalPark

<the 동물 사냥꾼>

## 2. 게임의 구성\_ 몬스터 정보

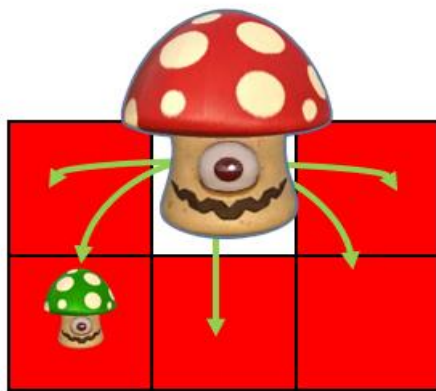
이름	스킬	공격	데미지
Mushroom(대왕버섯)	폭탄버섯 소환	X	0
Bomb(폭탄버섯)	X	자신을 폭발 시켜 피해를 줍니다.	2
Elegator(악어)	오징어 폭탄 소환	헌터를 넉백시킵니다.	1
Flower(식인꽃)	X	거대화하여 공격합니다.	3
Cactus(선인장)	전 범위 속박 마법	X	0
Mole(두더지)	X	땅 속에 숨어 접근하면 공격합니다.	2
Bleed(출혈 펀치)	X	출혈 디버프 부여한 공격을 합니다.	0
Stun(기절 펀치)	X	공격불가 디버프 부여한 공격을 합니다.	0
Muskrat(쥐)	X	범위에 있는 헌터에게 메테오를 확정으로 맞춥니다.	2
Colobus(원숭이)	X	유도 폭탄 바나나를 날립니다.	0.5
Chomper(외계생명)	X	부식침을 날려 이속 디버프를 부여합니다.	1

# AnimalPark <the 동물 사냥꾼>

## 2. 게임의 구성\_몬스터 이동 및 스킬 범위

### [ Mushroom ]

#### [ 스킬 ]



#### [ 이동 ]

#### [ 이동 범위 ]



[범위 중 가장 험터에 가까운 위치에 폭탄버섯을 소환합니다.]

### [ Bomb ]

#### [ 공격 ]

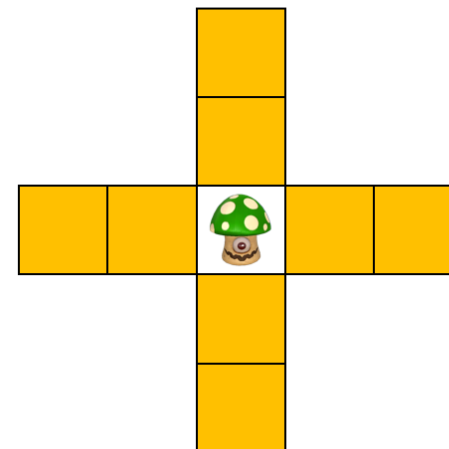
#### [ 공격 범위 ]



[공격 이후 오브젝트도 같이 파괴됩니다.]

#### [ 이동 ]

#### [ 이동 범위 ]

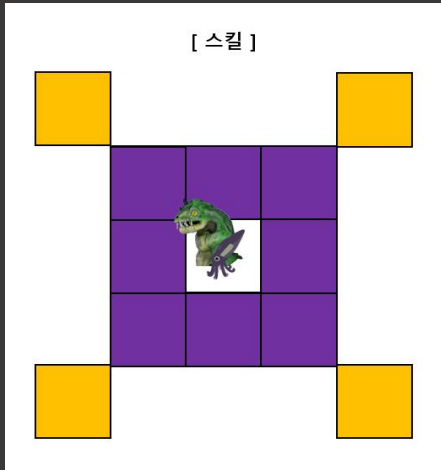


# AnimalPark <the 동물 사냥꾼>

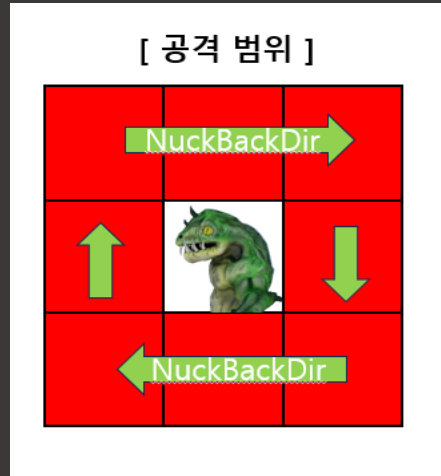
## 2. 게임의 구성\_몬스터 이동 및 스킬 범위

### [ Elegator ]

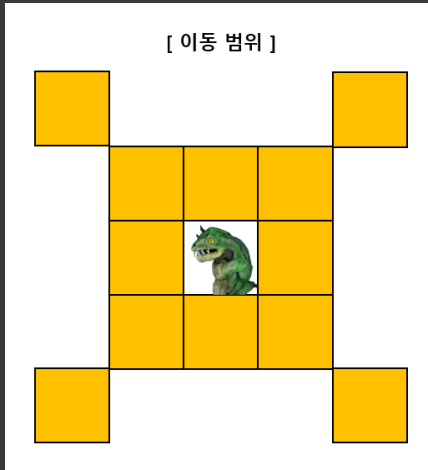
#### [ 스킬 ]



#### [ 공격 ]



#### [ 이동 ]



[보라색 범위의 공격을 하는 파괴할 수 없는 오징어 폭탄을 소환 후 이동합니다.]

[공격 범위내에 헌터의 위치에 따라 너 백 방향이 달라집니다.]

### [ Flower ]

#### [ 공격 ]



[제자리에서 헌터 접근 시 거대화하여 공격합니다.]



# AnimalPark <the 동물 사냥꾼>

## 2. 게임의 구성\_몬스터 이동 및 스킬 범위

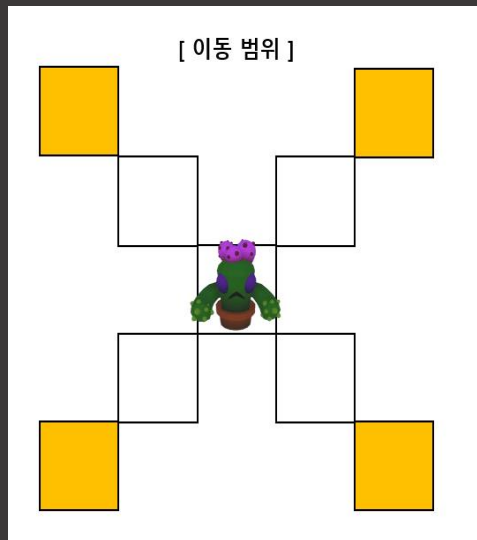
### [ Cactus ]

#### [ 스킬 ]



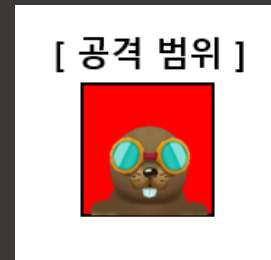
[헌터의 움직임을 1턴간 봉인하는 이동 디버프를 부여합니다.]  
[다른 몬스터와 달리 Cactus는 헌터에게서 멀어지는 이동을 합니다.]

#### [ 이동 ]



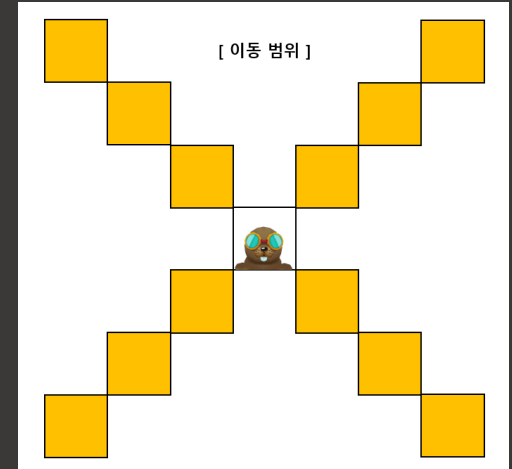
### [ Mole ]

#### [ 공격 ]



[공격 할 때 이외에는 땅 속에 숨어서 자신의 위치를 숨기고 있습니다.]

#### [ 이동 ]



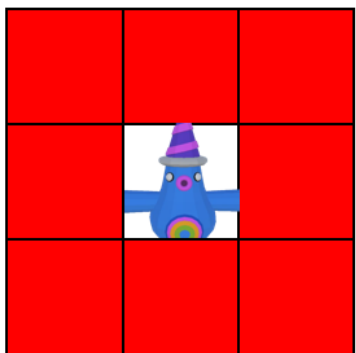
# AnimalPark <the 동물 사냥꾼>

## 2. 게임의 구성\_몬스터 이동 및 스킬 범위

### [ Bleed ]

#### [ 공격 ]

##### [ 공격 범위 ]

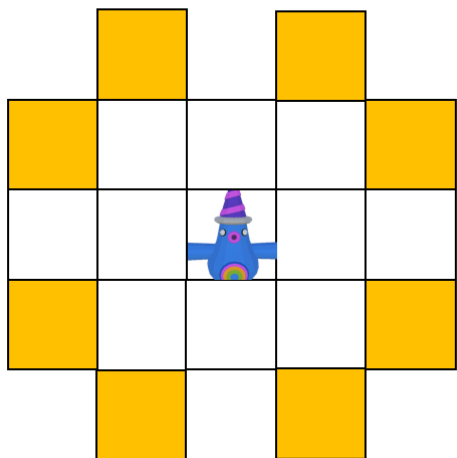


출혈 디버프 부여

[헌터에게 3턴간 출혈 디버프를 부여하는 공격을 합니다.]

#### [ 이동 ]

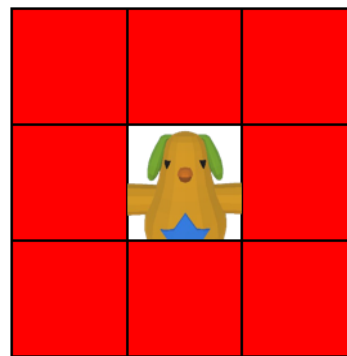
##### [ 이동 범위 ]



### [ Stun ]

#### [ 공격 ]

##### [ 공격 범위 ]

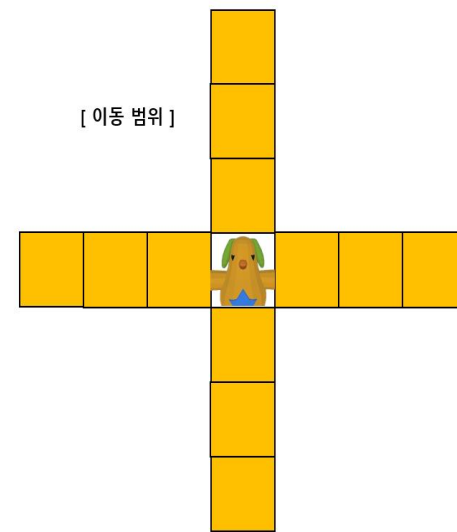


공격불가 디버프 부여

[헌터에게 1턴간 공격을 제한하는 디버프를 부여하는 공격을 합니다.]

#### [ 이동 ]

##### [ 이동 범위 ]



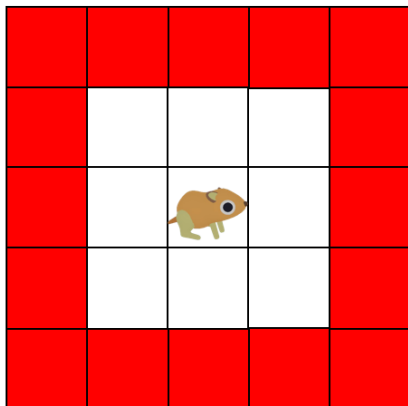
# AnimalPark <the 동물 사냥꾼>

## 2. 게임의 구성\_몬스터 이동 및 스킬 범위

[ Muskrat ]

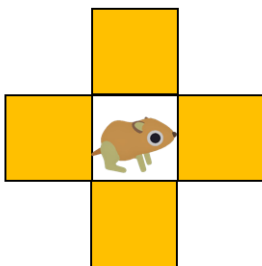
[ 공격 ]

[ 공격 범위 ]



[ 이동 ]

[ 이동 범위 ]



[공격범위에 닿은 헌터에게 확정적으로 피격되는 메테오를 소환합니다.]

# AnimalPark <the 동물 사냥꾼>

## 2. 게임의 구성\_몬스터 이동 및 스킬 범위

### [ Colobus ]

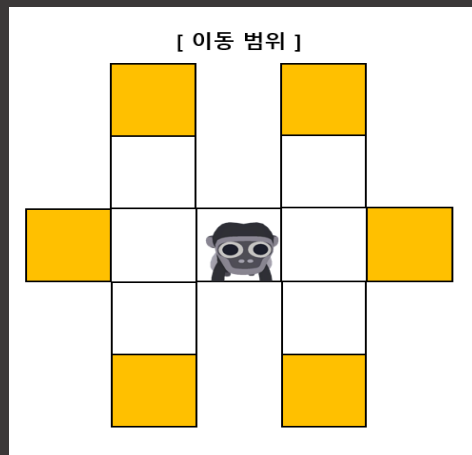
#### [ 공격 ]



[피할 수 없는 폭탄 바나나를 투척합니다.]

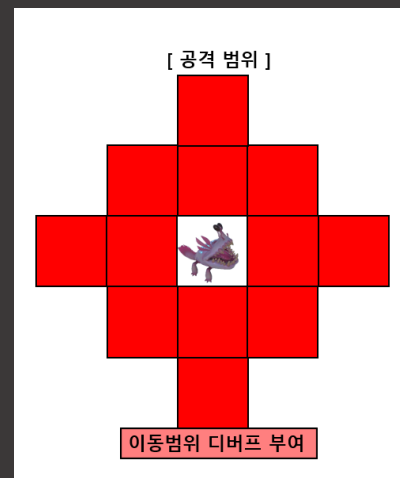
[Colobus는 헌터에게서 멀어지는 이동을 합니다.]

#### [ 이동 ]



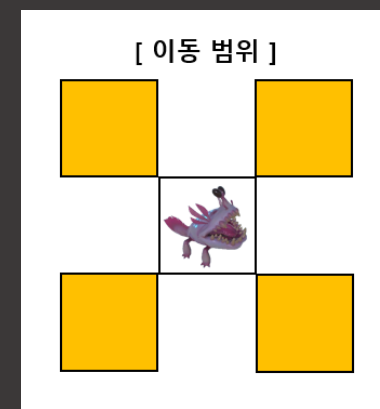
### [ Chomper ]

#### [ 공격 ]



[헌터에게 1턴간 이동범위를 제한하는 디버프를 부여하는 공격을 합니다.]

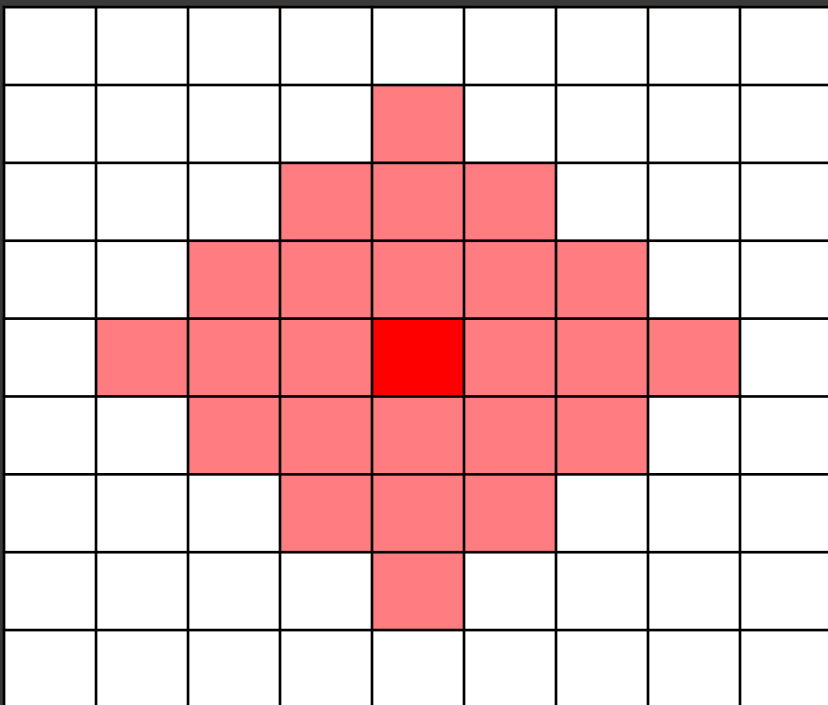
#### [ 이동 ]



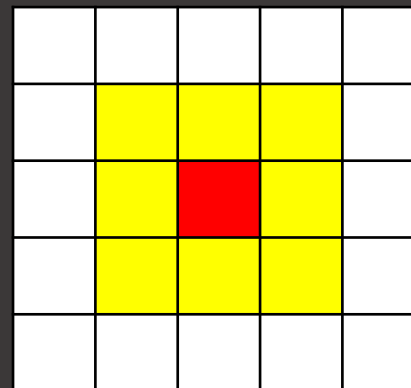
# AnimalPark <the 동물 사냥꾼>

## 2. 게임의 구성\_헌터 이동 및 스킬 범위

[ 이동범위 ]



[ 공격 ]



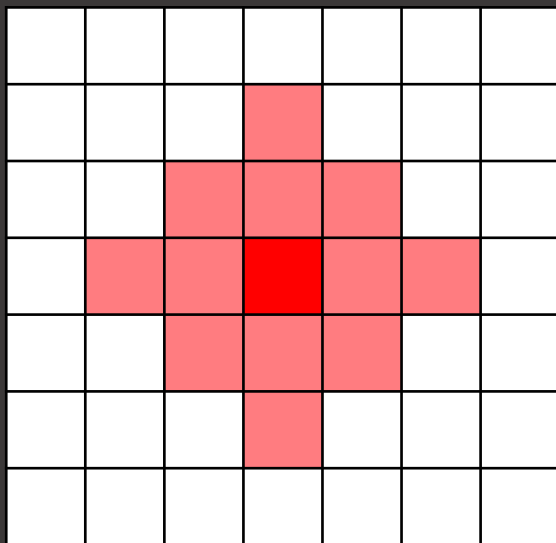
-  : 헌터 현재 위치
-  : 이동 가능 위치
-  : 공격 범위

# AnimalPark

<the 동물 사냥꾼>




## 2. 게임의 구성\_헌터 이동 및 스킬 범위

[ 스킬 이동 범위]



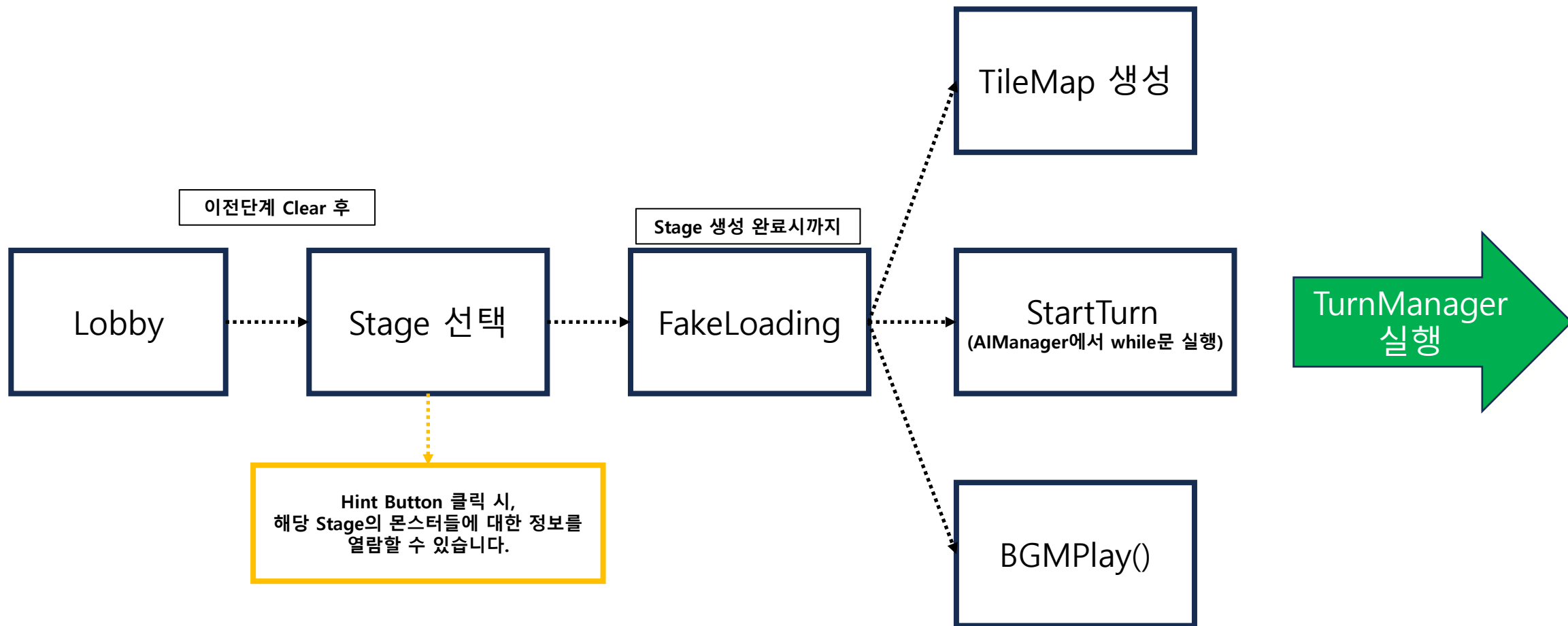
[ 스킬 공격 범위]



-  : 헌터 현재 위치
-  : 이동 가능 위치
-  : 얼음 디버프 부여 범위

# AnimalPark <the 동물 사냥꾼>

## 3. 게임의 진행

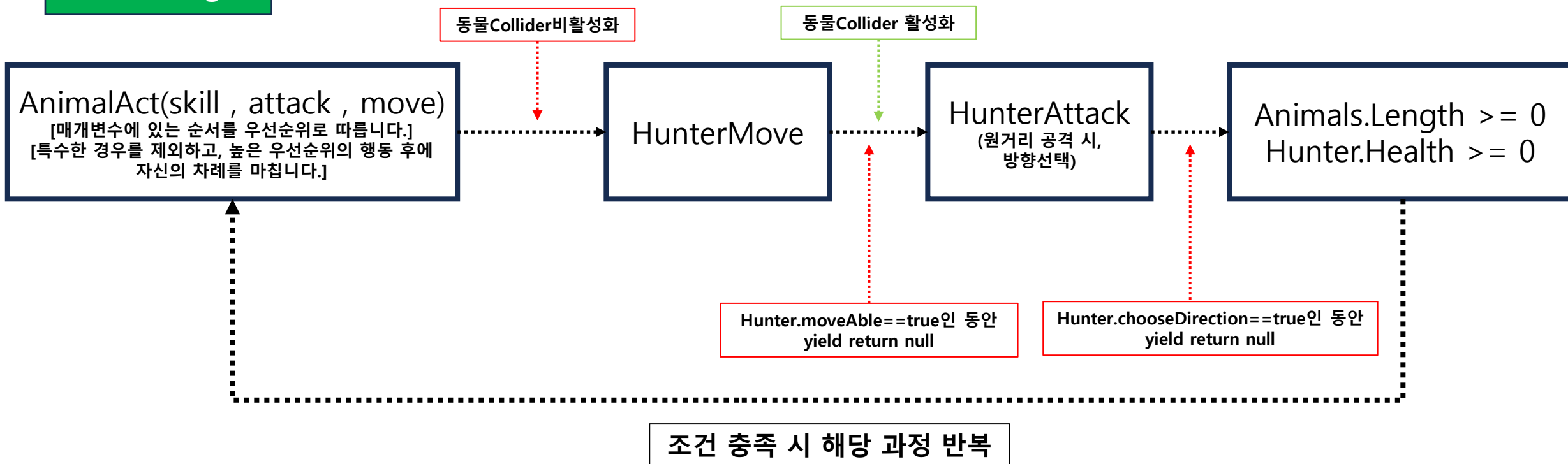


# AnimalPark

<the 동물 사냥꾼>

## 3. 게임의 진행

TurnManager

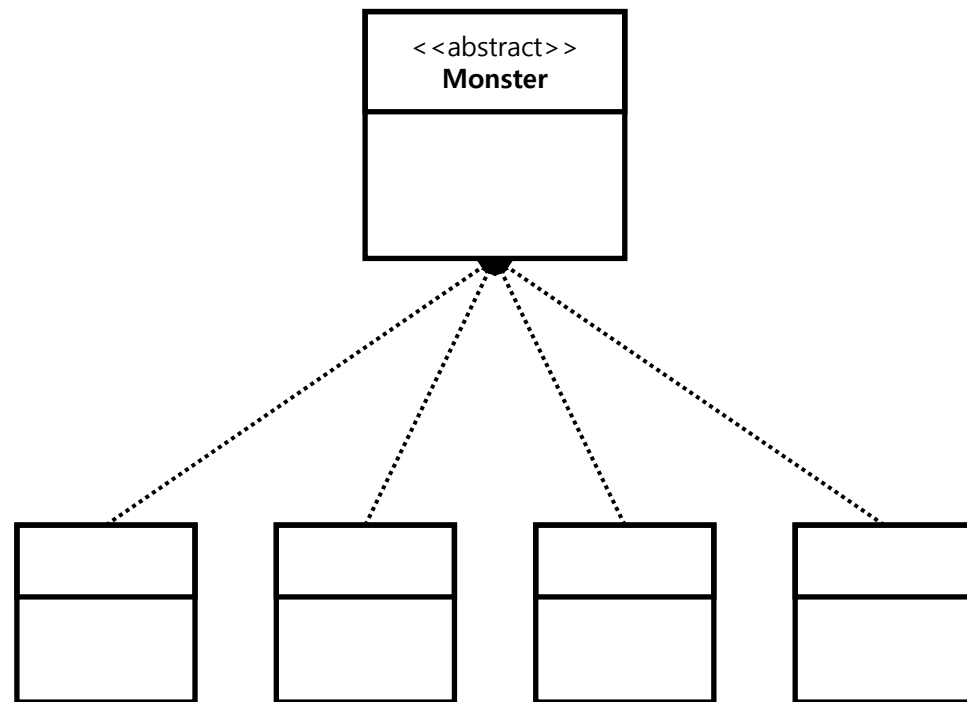
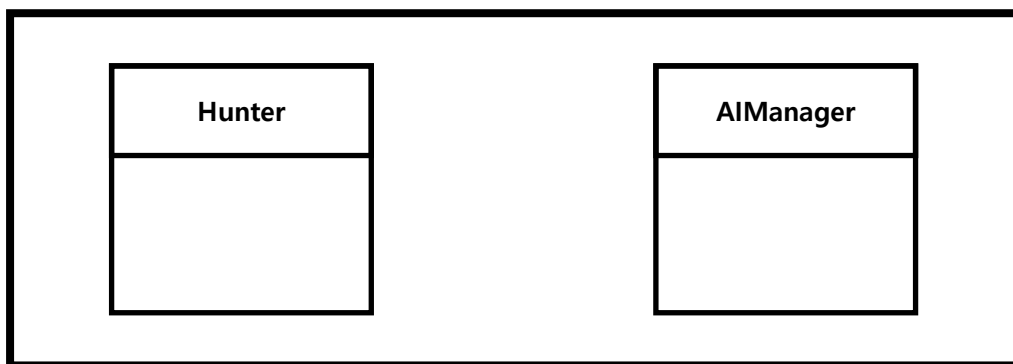
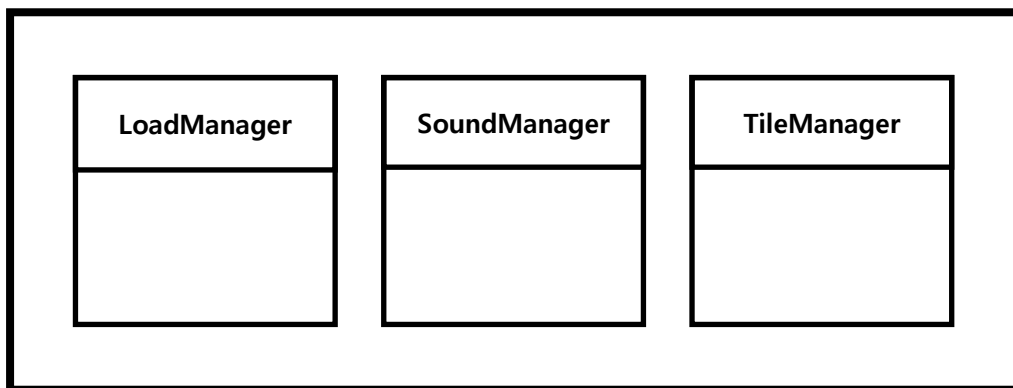




# AnimalPark <the 동물 사냥꾼>

## 4. 코드 구성

### <DontDestroyObject>



# AnimalPark

<the 동물 사냥꾼>

## 4. 코드 구성\_ DontDestroyObject

### LoadManager

- ...

```
+ LoadScene(string sceneName) : void
+ <<coroutine>>LoadNextScene(string next) :
IEnumerator
+ ActiveHintPanel() : void
+ ActivePausePanel() : void
```

### TileManager

```
- Blocks : GameObject[]
- SceneNumber : Dictionary<string , int>
- SceneName : string[]
```

```
+ CreateMap() : void
- stage에 맞는 타일쌍을 찾아 맵 생성
+ insertTileMap() : void
- 오브젝트 이동 후에 현재 위치 저장
+ CheckTileMap() : bool
- 몬스터 이동 전 이동 가능한 위치인지 확인
```

### SoundManager

```
- bgm : AudioClip[]
- audioClips : AudioClip[]
- backgroundAudioSource : AudioSource
- effectAudioSource : AudioSource
```

```
+ BGMPlay() : void
+ SoundPlay(string name) : void
```

# AnimalPark

<the 동물 사냥꾼>

## 4. 코드 구성

### AIManager

- Animals : GameObject[]
- TileMap : int[ , ]

- + StartTurn() : void
- + <<coroutine>> ActiveAiManager() : IEnumerator
- + <<coroutine>> TurnManager() : IEnumerator
  - TileManager를 호출해서 맵을 생성하고, 코루틴 함수내부에 while문을 사용하여 각 오브젝트간 행동간격을 조절합니다.
- + UpdateAnimalList() : void
- + RemoveAnimal(GameObject animal) : void
- + AddAnimal (GameObject animal) : void
  - 몬스터의 생성과 제거 시, 각 몬스터들에 대한 썸네일 정보를 갱신해 줍니다.

<<abstract>>

### Animal

- 
- + <<virtual>> SetAnimalStatus() : void
  - 최초 씬 전환 시 Json파일로부터 해당 Stage의 몬스터들에 대한 정보를 받아옵니다.
- + <<virtual>> AnimalAct(int skill , bool attack, bool move) : void
  - switch문을 통해 skill , attack , move 중 가능한 행동 하나를 실행합니다.
- + <<virtual>> Move(Vector3 cur , Vector3[] MovePoint) : void
- + <<coroutine>> JumpToPosition() : IEnumerator
  - 몬스터의 이동시에 애니메이션 동작을 자연스럽게 이어줍니다.
- + <<virtual>> Damage() : void
- + <<virtual>> Die() : void

# AnimalPark <the 동물 사냥꾼>

## 4. 코드구성 \_AIManager

```
public void SetStageData(){
    TextAsset StatusList = Resources.Load<TextAsset>("AnimalStatus");

    if (animalStatus != null)
    {
        AnimalData[] animals = JsonUtility.FromJson<AnimalDataArray>(StatusList.ToString()).animal;
        // 동물 데이터 리스트를 Dictionary에 저장
        animalDictionary = new Dictionary<string, AnimalData>();

        foreach (var animal in animals)
        {
            animalDictionary.Add(animal.name, animal);
        }

        foreach (var animal in Animals)
        {
            if (animalDictionary.ContainsKey(animal.name))
            {
                AnimalData animalData = animalDictionary[animal.name];
                animal.GetComponent<Monster>().SetAnimalStatus(
                    animalData.AttackDMG,
                    animalData.Health,
                    animalData.SkillCount
                );
            }

            Vector3 animalPosition = animal.transform.position;
            FindAnyObjectByType<TileManager>().GetComponent<TileManager>().insertTileMap(
                (int)animalPosition.x / 2, (int)animalPosition.z / 2, 1);
        }
    }
}
```

```
[System.Serializable]
참조 7개
public class AnimalData
{
    public string name; // 동물의 이름
    public float Health; // 체력
    public float AttackDMG; // 공격력
    public int SkillCount;
}

[System.Serializable]
참조 1개
public class AnimalDataArray
{
    public AnimalData[] animal;
}
```

Stage의 동물리스트가 저장된 배열 Animals

Key 값인 name을 통해, 각 오브젝트에 맞는 Json정보를 할당 받습니다.

# AnimalPark <the 동물 사냥꾼>

## 4. 코드구성 \_AIManager

참조 1개  
private IEnumerator TurnManager()  
{

while (Hunter.Health>=0)

{  
for (int i = 0; i < 8; i++)

{  
for (int j = 0; j < 8; j++)

{  
TileMap[i, j] = 0;

};

yield return null;

AnimalMove();

yield return new WaitForSeconds(1.0f);

AnimalAttack();

yield return new WaitForSeconds(1.0f);

if (CheckAnimalAttack()){yield return new WaitForSeconds(3.0f);}

else{ yield return new WaitForSeconds(1.5f); }

AnimalUnAttackBox();

HunterMove();

while (Hunter.Moveable) { yield return null;}

hunter.Attack();

while (Hunter.Attackable) { yield return null; }

if (Animals.Length <= 0 || Hunter.Health<=0)

{  
break;

};

}

동물의 움직임 관리는 애니메이션  
시간을 고려해서 시간을  
상이하게 설정합니다.

헌터의 위치선택을 방해하지 않기 위해 동물들의  
collider를 비활성화 시켜놓는 함수입니다.

헌터가 가지고 있는  
Attackable 과 moveAble을 이용하여  
게임의 진행을 판단합니다.

# AnimalPark

<the 동물 사냥꾼>

## 4. 코드 구성 \_LoadManager

참조 1개

```
IEnumerator LoadNextScene(string next)
{
    float loadTime = 2f;
    float elapsedTime = 0f;

    float changeInterval = 0.1f; //로딩 시 이미지 전환 속도
    float nextChangeTime = changeInterval;

    AsyncOperation asyncLoad = SceneManager.LoadSceneAsync(next);
    asyncLoad.allowSceneActivation = false; .....
    soundManager.MoveStage(); // 로딩 BGM 설정

    // 로딩 시 사용할 움직이는 이미지 설정
    while (elapsedTime < loadTime) .....
    {
        yield return null;
        elapsedTime += Time.deltaTime;

        MoveDucks((int)(elapsedTime / changeInterval) % loadingImages.Length);
        nextChangeTime += changeInterval;

        loadingBar.value = Mathf.Lerp(loadingBar.value, elapsedTime / loadTime, Time.deltaTime);
    }
}
```

씬 전환을 지연시키고, 조건이 만족되면 씬을 활성화합니다.

로딩이 진행되는 동안 움직이는 오리 이미지를 구현할 while문

# AnimalPark <the 동물 사냥꾼>

## 4. 코드 구성\_LoadManager

```
while (!asyncLoad.isDone)
{
    yield return null;

    // 진행률 0.9 미만일 때
    if (asyncLoad.progress < 0.9f)
    {
        loadingBar.value = asyncLoad.progress;
    }
    else
    {
        // 마지막 로딩 완료
        loadingBar.value = 1f;

        // 씬 전환 허용
        asyncLoad.allowSceneActivation = true;
        aiManager.ActiveHintPanel();
        yield return new WaitForSeconds(0.5f);
        tileManager.CreateTileMap();
        aiManager.StartTurn();
        loadingBar.value = 0;
        loadingCanvas.SetActive(false);
        soundManager.BGMPlay();
    }
}
```



로딩이 끝나고 나면 기본 세팅을 진행합니다.

1. 힌트 패널 활성화
2. Stage에 맞는 블록타일로 맵 생성
3. AIManager의 게임진행함수 실행
4. Stage에 맞는 BGM Play

# AnimalPark <the 동물 사냥꾼>

## 4. 코드 구성 \_Monster



Monster(부모클래스)



동물이름(자식클래스)

```
public virtual void AnimalAct() { }
```

함수 오버라이딩 활용



```
public override void AnimalAct()  
{  
    skillCount--;  
    if (skillCount < 0) { skillCount = totalSkillCount; }  
  
    base.AnimalAct(skillCount, attackable, true);  
}
```

함수 오버로딩 활용



```
public void AnimalAct(int skillcount, bool attackAble, bool moveAble)  
{  
    if (ice)  
        return;  
    transform.LookAt(Hunter.HunterPosition);  
    transform.rotation = Quaternion.Euler(0, transform.rotation.eulerAngles.y % 360, 0);  
    if (skillcount == 0)  
    {  
        Skill();  
    }  
  
    else if (attackAble)  
    {  
        Attack();  
    }  
  
    else if (moveAble)  
    {  
        Move();  
    }  
}
```

각 동물의 컨셉에 맞게 (skill, attack, move)의 매개변수들을 변경해서 동작을 수행시킵니다.



# AnimalPark

<the 동물 사냥꾼>

**END**

[블로그 주소 : Game | 코딩일지](#)

깃 허브 주소 : <https://github.com/BaeGWoo>