



Rabbit Simulator

technical introduction

Made by Unreal Engine Client 김노운

목 차

1. 타일 맵 E

기능성 블록 만들기
타일의 심도 측정
고정 카메라 배치와 시점 이동

2. 토끼 E

목적지까지 길 찾기
유전 알고리즘, 난수 모델
토끼 데이터와 데이터 시각화

3. 포식자 E

개요
생성 시각화

4. 카메라 E

Trace Target & Picking
Custom Depth Stencil
Following Camera



타일 맵



1. 타일 맵

초기 맵 세팅

맵 가로 10 100 20

맵 세로 10 100 20

웅덩이 생성 백분율:

나무 생성 백분율:

덤불 생성 백분율:

덤불당 베리 평균 개수:

베리 생성 주기:

베리의 영양분:

GameInstance



BP_World

initialize

CreateMapAllGrass

맵 가로+2와 세로+2만큼 직사각형 형태의 Grass Tile Spawn

OutLineChecker

가장 외곽에 있는 타일들은 OutLine으로 체크하고, 투명하게 만든다.

MakeWaterBlock

맵에 정해진 수치만큼 Water Tile을 절차적으로 생성한다.

MakeTree

맵에 정해진 수치만큼 Tree Tile을 확률적으로 생성한다.

MakeBush

맵에 정해진 수치만큼 Grass Tile에 Bush을 확률적으로 생성한다.

SetTileColor

주변의 타일이 나와 같은 만큼 심도를 설정하고 색을 설정한다.

SpawnCamera

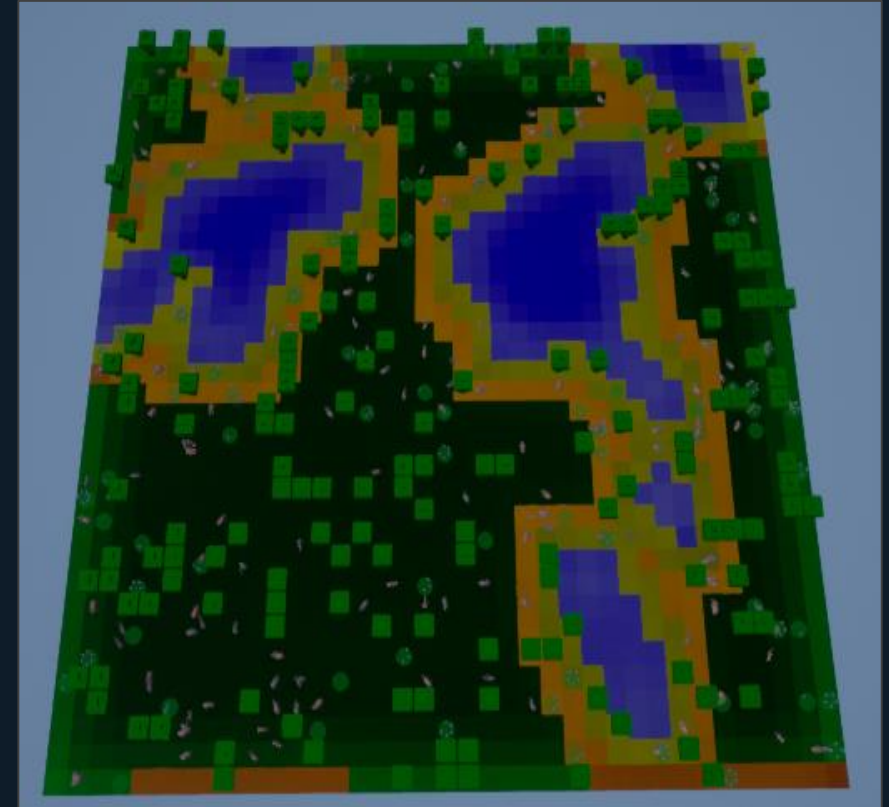
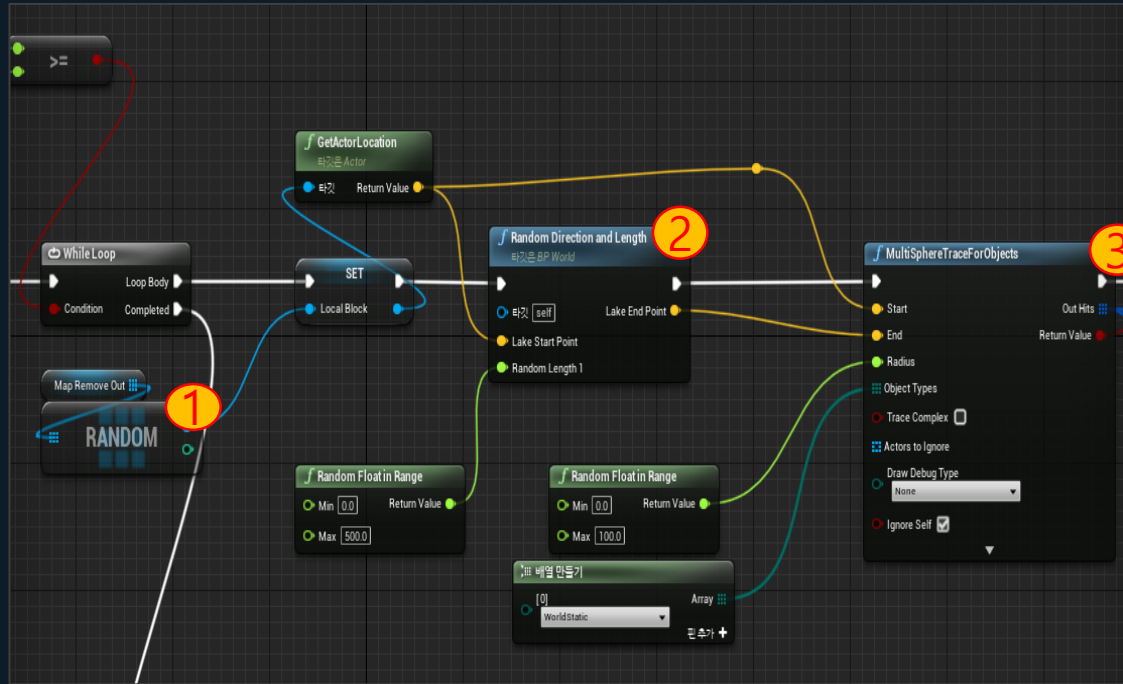
맵의 크기가 아무리 변하더라도 고정된 카메라를 관찰하기 좋은 위치에 스폰한다.

GameInstance로 세팅한 데이터를 가져와서 BP_World에 이식한다.

이식받은 데이터를 기반으로 Initialize를 실행한다.



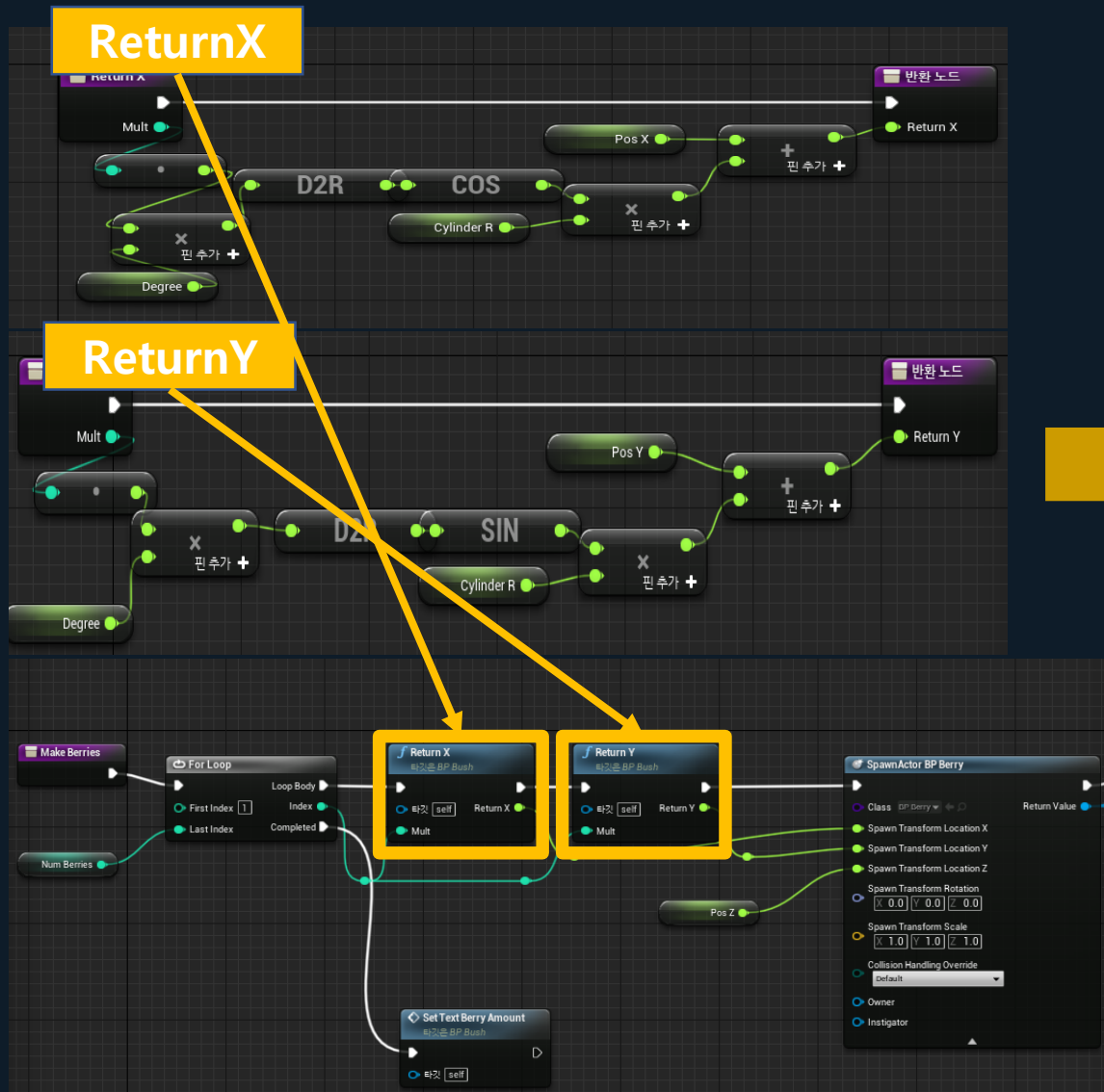
1-1. 타일 맵 - 기능성 블록 만들기 - Make Water Block



1. 무작위 타일을 선택한다.
2. 그 타일의 위치 값을 기준으로 무작위 방향과 길이를 받는다.
3. MultiSphereTraceForObject로 2의 정보를 바탕으로 타원형 콜리전을 생성, 기존 타일과 충돌을 체크하여 타일의 타입을 Water로 바꾼다.

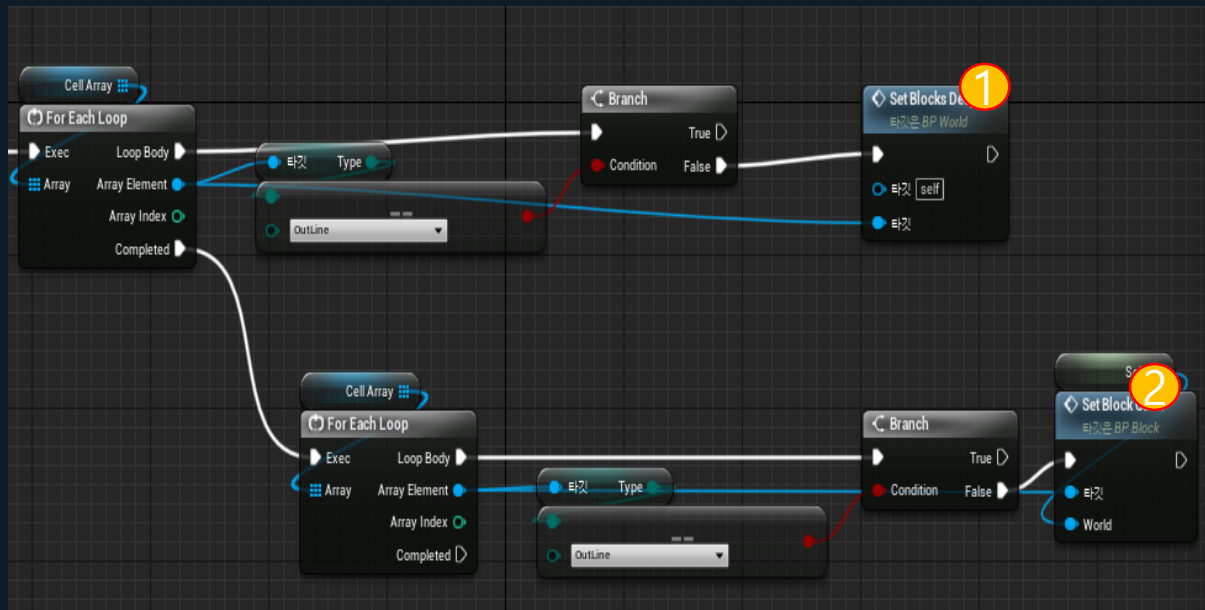


1-1. 타일 맵 - 기능성 블록 만들기 - Make Bush



삼각함수를 사용하여 360도를 NumBerries(평균 베리 개수) 만큼 나누고(Degree) X, Y 좌표를 구하여 Bush에 Berry를 Spawn한다.

1-2. 타일 맵 - 타일의 심도 측정



1. SetBlocksDeep: 사방 25칸의 타일 중 자신과 동일한 타일의 개수만큼 BP_Block의 변수 Deep에 저장한다.
2. SetBlockColor: 사방 9칸의 Deep 변수를 토대로 타일 속성에 따라 다른 색상을 입힌다.



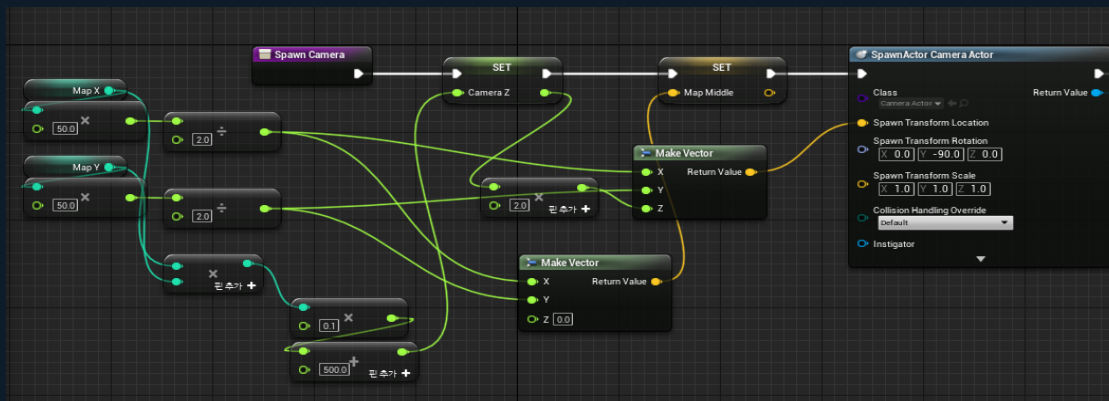
1-3. 타일 맵 - 고정 카메라의 배치와 시점 이동 - 카메라 배치



20 X 20



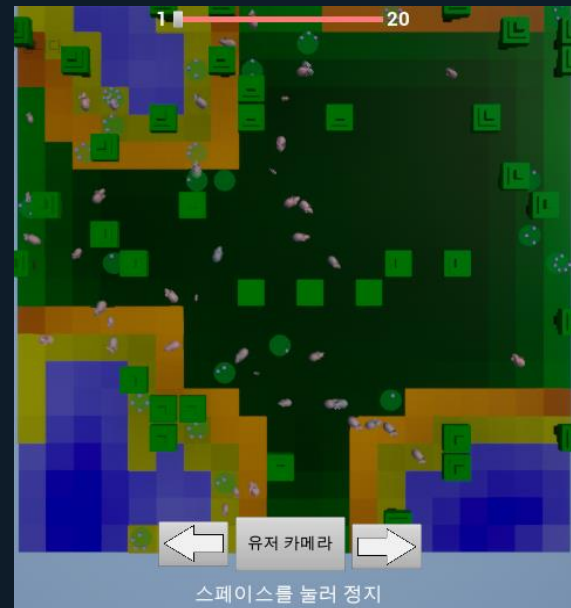
40 X 40



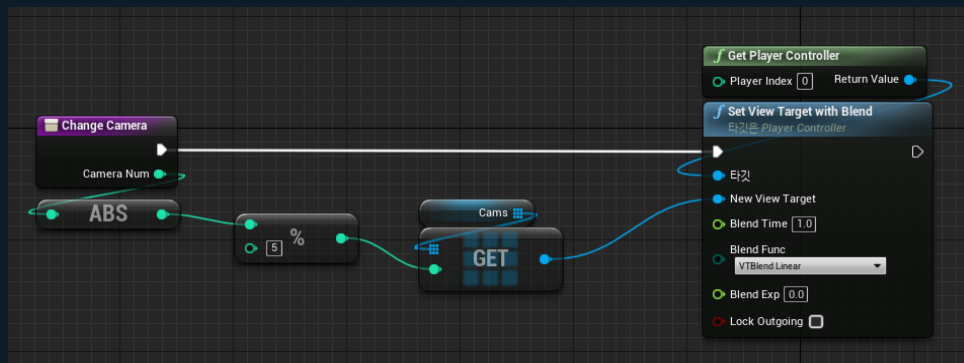
Map X와 Map Y 수치에 따라
유기적으로 카메라 위치를 보정함.



1-3. 타일 맵 - 고정 카메라의 배치와 시점 이동 - 카메라 시점 이동



고정된 카메라 뷰



SetViewTargetWithBlend를 사용하여
자연스럽게 카메라의 위치를 옮긴다.

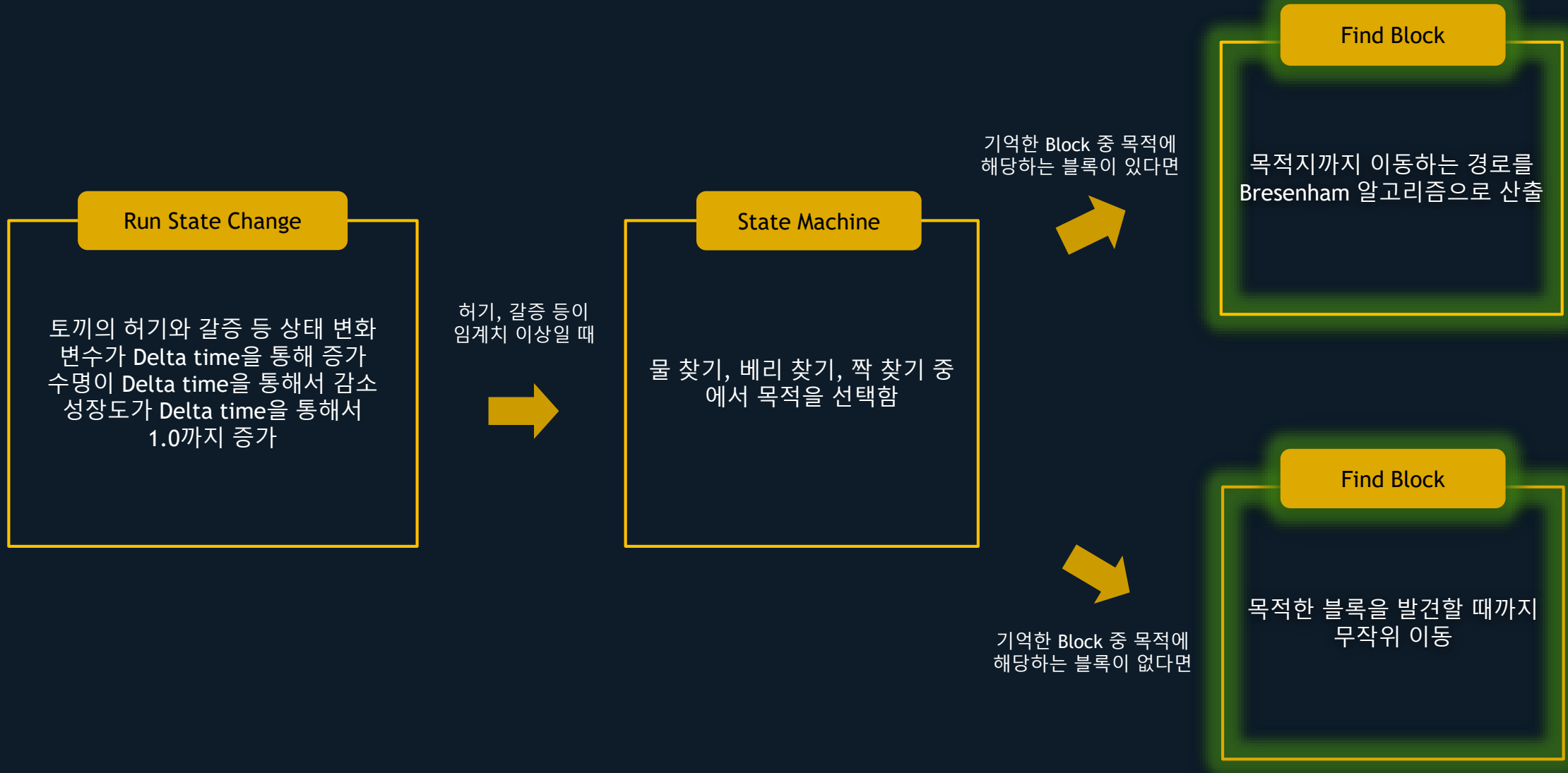




토끼



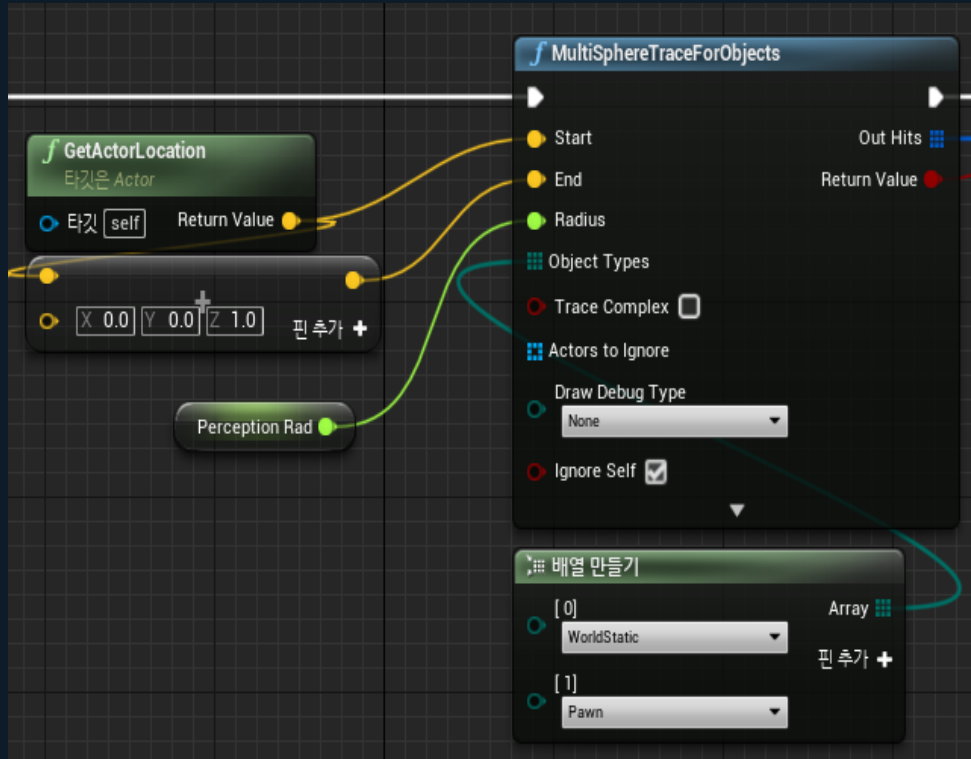
2. 토끼



2-1. 토끼 - 목적지까지 길찾기 - Find Block

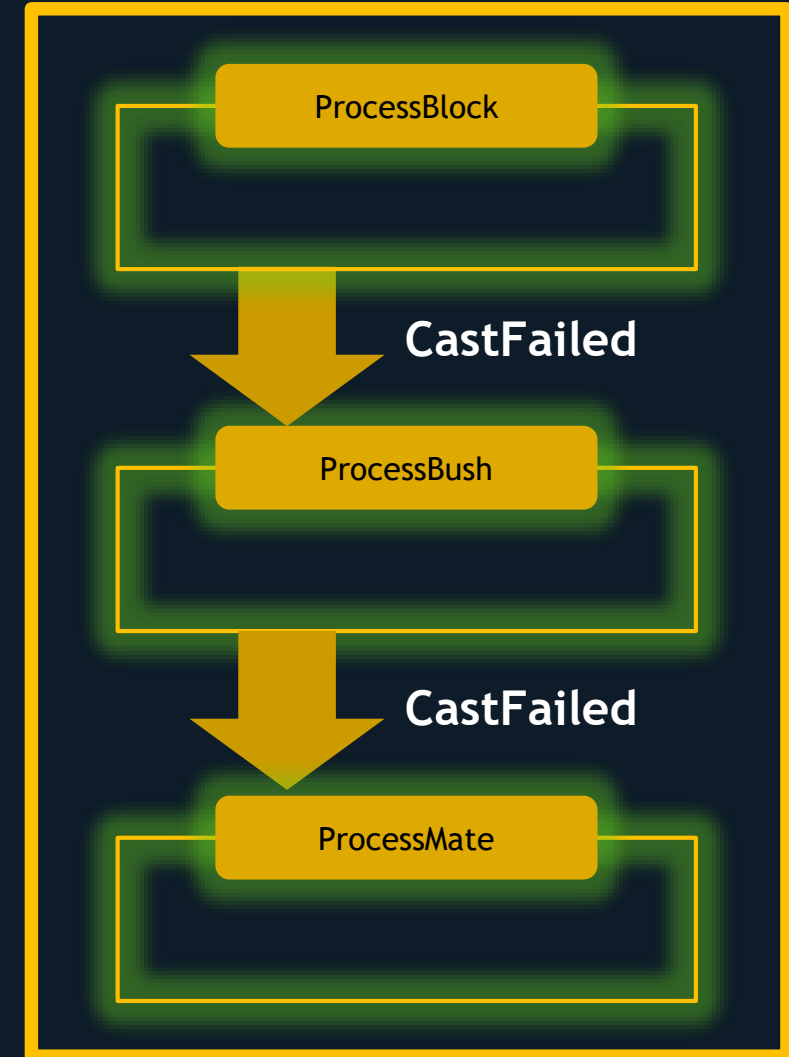


2-1. 토끼 - 목적지까지 길찾기 - Find Block - Do Perception



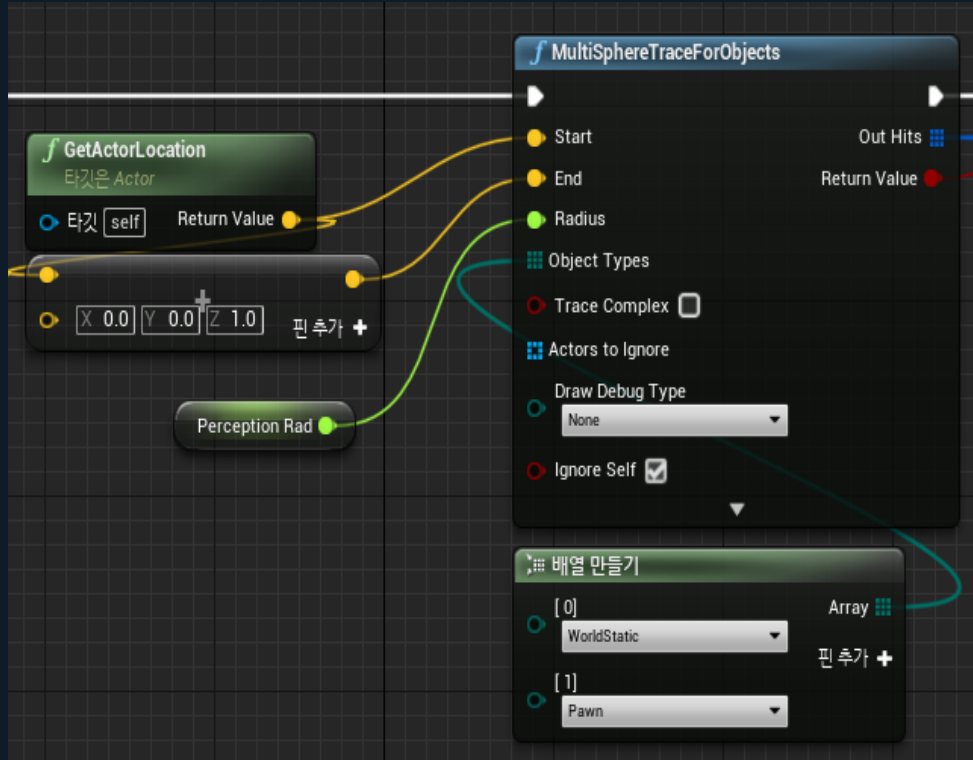
현재 위치를 기준으로
MultiSphereTraceForObject로 충돌 체크

HitResult



2-1. 토끼 - 목적지까지 길찾기 - Find Block - Do Perception

- Process Block



OutHit를 기준으로 For 문을 돌려 다수의 HitResult를 반환하여 각 함수에 전달한다.

Hit Result

ProcessBlock

BP_Block으로 Cast하고 성공하면, 타입에 따라
Grass: PerceptedBlock (이동 가능한 땅)
에 추가하거나
Water: MemorizedWaterBlock (물 블록 기억)
에 세팅한다. 이후 목이 마를 때
MemorizedWaterBlock을 타겟으로 반환한다.

Outline과 Tree 는 갈 수 없는 땅과 장애물의
개념이므로 함수를 종료한다.

목이 마르고, 근처에 물 블록이 있다면
DrinkWater 한다.

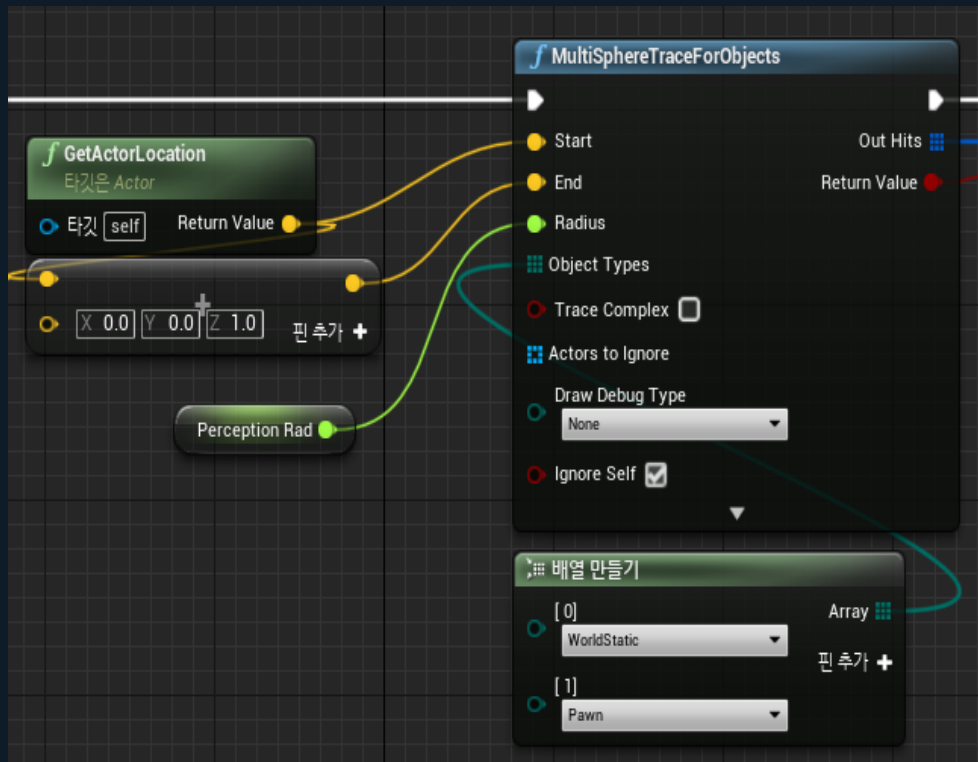
Cast Failed

DrinkWater

Thirsty 수치를 0으로 만든다.

2-1. 토끼 - 목적지까지 길찾기 - Find Block - Do Perception

- Process Block - Process Bush



ProcessBlock에서 CastFailed했다면
ProcessBush를 실행하고 HitResult를
전달한다.



ProcessBush

BP_Bush로 Cast하고 성공하면
MemorizedBush에 세팅한다.

이후 배가 고플 때 MemorizedBush로
이동하여 bush에 FindBerry 요청을
넣는다.

성공시 Bush의 Berry가 1개 줄어들고
Berry의 BerrySatiety 만큼 배고픔이
줄어든다.

실패시 Bush를 IgnoreBush에 기억하고
IgnoreBushTime 만큼 무시한다.

FindBerry

타겟 Bush의 BerryCount가 0이면
false를 리턴한다. BerryCount가 0
이상이면 true와 BerrySatiety를
리턴한다. BerryCount는 남은
Berry의 개수이다.

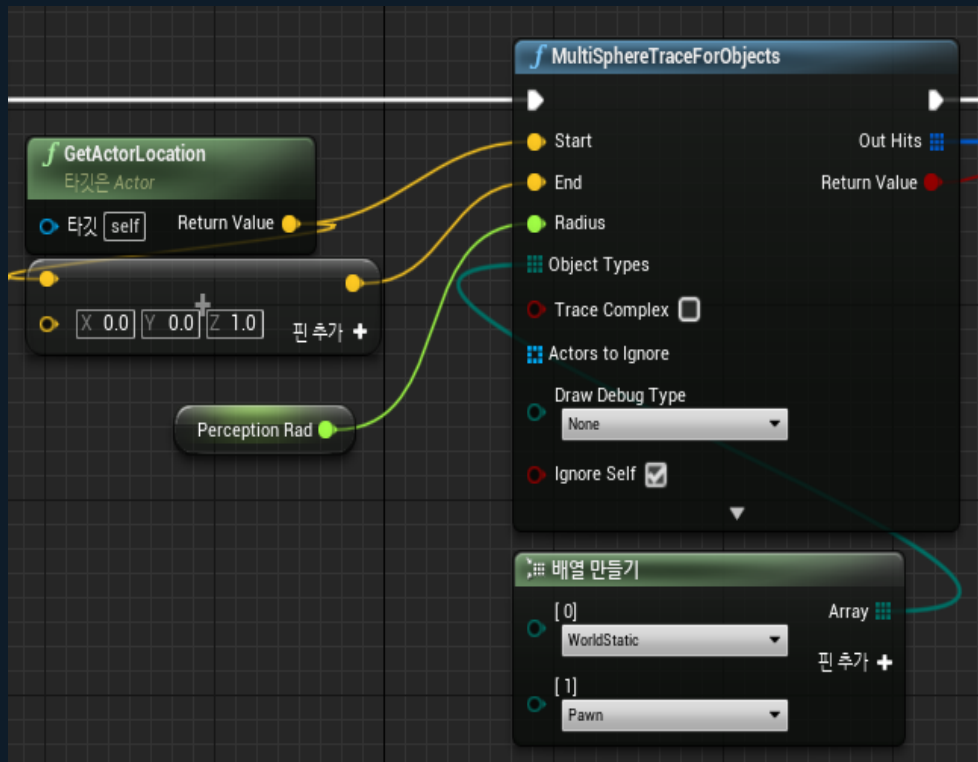
Cast
Failed

And

1. Not male.
2. Growth \geq 1.0
3. Not old.
4. Not young
5. 임의의 사유로
StopMate가 아니어야
한다.

2-1. 토끼 - 목적지까지 길찾기 - Find Block - Do Perception

- Process Block - Process Bush - ProcessMate



ProcessBush에서 CastFailed했고, HitResult - Actor를 BP_Rabbit으로 Cast한 후 대상이
1. Not male, 2. Growth ≥ 1.0 , 3. Not old, 4. Not young,
5. 임의의 사유로 StopMate가 아니라면 ProcessMate를
실행하고 HitResult를 전달한다.



ProcessMate

전달받은 대상이 임신 중이 아니고, 대상의 Attractive가 0에서부터 100까지의 값을 가진 임의의 float 보다 높고 충분히 가깝다면 타겟에게 TryMate 한다.

대상의 위치가 자신과 가깝지 않다면 TargetRabbit으로 세팅한다. 이후 Bresenham 알고리즘을 사용하여 TargetRabbit까지 가는 길을 반환한다.

TryMate가 성공하면 DoMate를 하고 PC_Genetic에 MakingChildRabbitData를 요청한다. 데이터가 만들어지면 PC_Genetic에서 FemaleRabbit에게 ChildRabbit의 데이터를 2개 전달한다. FemaleRabbit은 ChildRabbit 데이터를 임신 기간 동안 소유한다. 임신 기간이 만료되면 FemaleRabbit의 주변에 ChildRabbit의 데이터를 가진 BP_Rabbit을 둘 Spawn 한다.

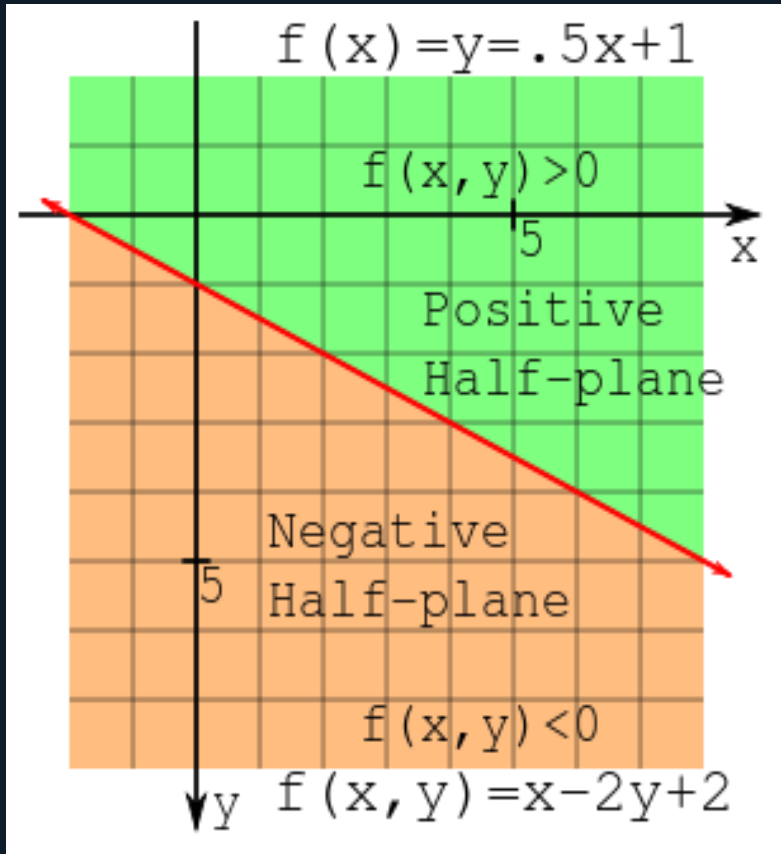
TryMate

요청을 넣은 Male Rabbit의 Attractive가 0에서부터 100까지의 값을 가진 임의의 float 보다 높다면 true, 아니면 false한다.

DoMate

Mate 수치를 0으로 만들고 배고픔과 목마름 수치를 늘린다.

2-1. 토끼 - 목적지까지 길찾기 - Bresenham [길찾기 알고리즘]



BP_Block은 X와 Y 변수를 가지고 있다. 이 변수를 사용해서 현재 CurrentBlock에서 목표 지점까지 직선 그리기 알고리즘을 수행한다. 그 결과 나오는 X,Y 쌍을 저장한다.

Bresenham 직선 그리기 알고리즘은 직선의 기울기로 판별식을 만들어 가상의 선이 해당 점을 지나고 있는지 지나지 않고 있는지 정수 연산만으로 빠르게 판별할 수 있는 알고리즘이다.

EndX-StartX = 0 일 때와 기울기에 따라 Xweight, Yweight를 각기 다르게 적용하여 모든 기울기에 대한 판별식을 제작했다.



2-2. 토끼 - 유전 알고리즘

유전 알고리즘은 자연을 모방한 진화 연산으로, 최적화 기법 중 하나이다. **해를 유전자 형식으로 표현할 수 있어야 하며, 선택, 교차, 변이, 대치의 과정을 가진다.**

선택

BP_Rabbit으로부터 DoMate 호출 후 PC_Genetic의 MakeChildRabbitData 함수를 호출하여 Male Rabbit과 Female Rabbit을 인자로 전달한다.



교차

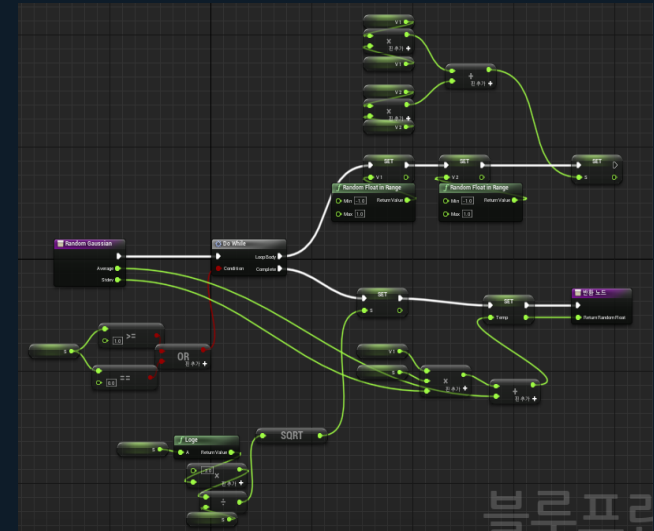
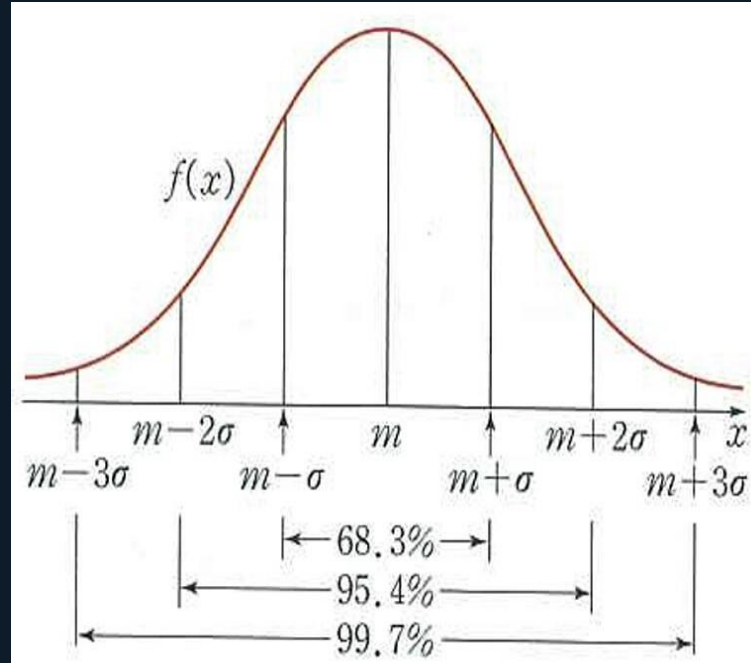
MakeChildRabbitData에서 MakeChildData 함수를 호출하며 부모의 데이터를 전달한다. 이를 통해서 자식 토끼의 능력치를 만드는데 이는 부모의 능력치 중 가장 높은 것과 낮은 것의 사이의 값으로 결정한다. 이후, 능력치에 돌연변이를 일으킬지 여부를 결정하고 돌연변이를 만든다. 모든 난수는 정규분포로 결정한다.

변이

교차로부터 도출된 자식의 능력치를 정규분포의 Average로 입력하여 함수 실행 결과값을 ChildData의 데이터에 세팅한다. 세팅이 완료된 데이터를 Female Rabbit에게 전달한다. Female Rabbit은 전달된 데이터 2개를 임신기간 동안 지닌 후 주변에 ChildRabbit을 Spawn 한다.

2-2. 토끼 - 난수 모델 [GaussianRandomDistribution]

```
double gaussianRandom(void) {  
    double v1, v2, s;  
  
    do {  
        v1 = 2 * ((double) rand() /  
RAND_MAX) - 1;  
        // -1.0 ~ 1.0 까지의 값  
        v2 = 2 * ((double) rand() /  
RAND_MAX) - 1;  
        // -1.0 ~ 1.0 까지의 값  
        s = v1 * v1 + v2 * v2;  
    } while (s >= 1 || s == 0);  
  
    s = sqrt( (-2 * log(s)) / s );  
  
    return v1 * s;  
}
```






상기의 내용을 블루프린트로 구현하였음.

자연적인 확률 분포는 대부분 정규 분포를 따르기 때문에 난수 모델로 설정하였다.



2-3. 토끼 – 토끼 데이터

0세대 토끼 세팅

0세대 토끼 백분율:	15.0
토끼 이동 속도:	200.0
토끼 매력:	50.0
임신기간:	30.0
인식반경:	100.0
욕구임계치:	20.0
목마름 배율:	0.5  2.0
배고픔 배율:	0.5  2.0
짜릿기 배율:	0.5  2.0
토끼 수명:	300.0

시작시, 전체 타일 당 한 마리를 기준으로 전체 맵에 몇 퍼센트를 생성할지?

토끼의 타일 이동속도

토끼가 배우자를 선택할 확률에 관여함

토끼는 임신기간을 조절할 수 있음. 적을수록 더 어린 토끼가 나온다.

토끼가 한번에 인식할 수 있는 반경

목마름, 배고픔, 짜릿기 욕구가 욕구임계치 이상이 될 때 욕구에 해당하는 행동을 이행함.

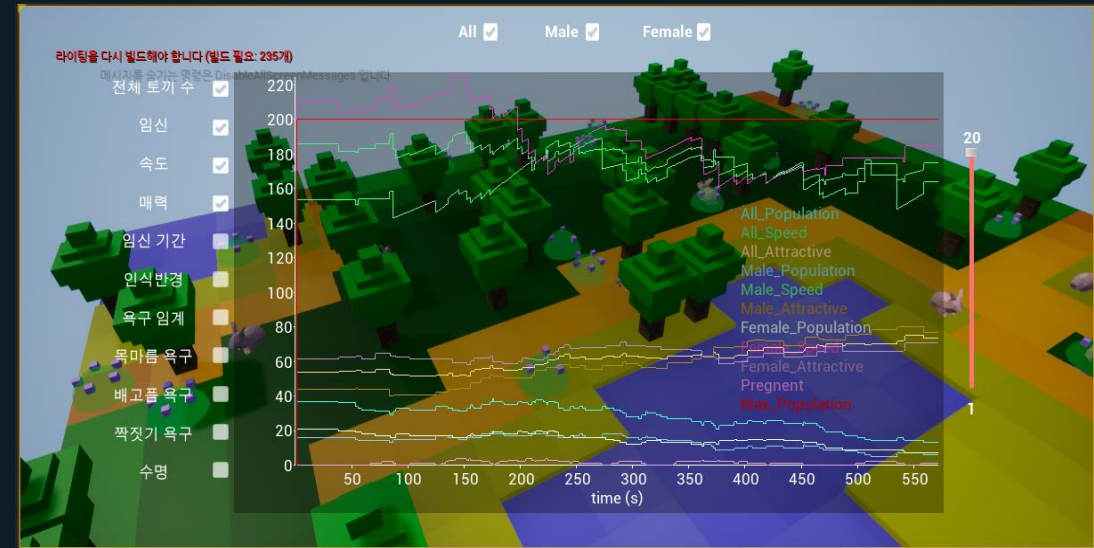
0이 되면 토끼가 자연사 한다.



2-3. 토끼 - 데이터 시각화

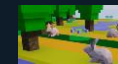


토끼의 개체 상태를 보여주는 막대 차트



토끼의 전체 상태를 보여주는 그래프

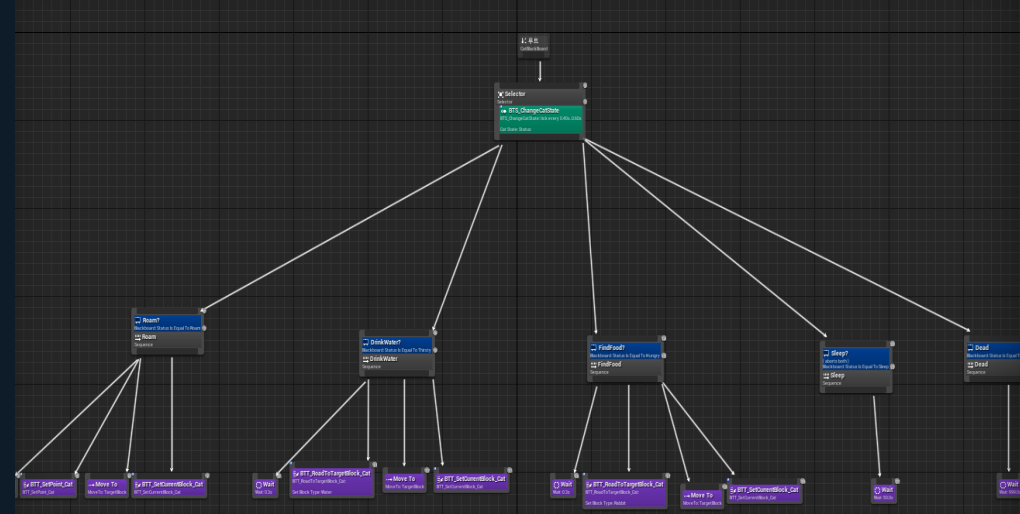
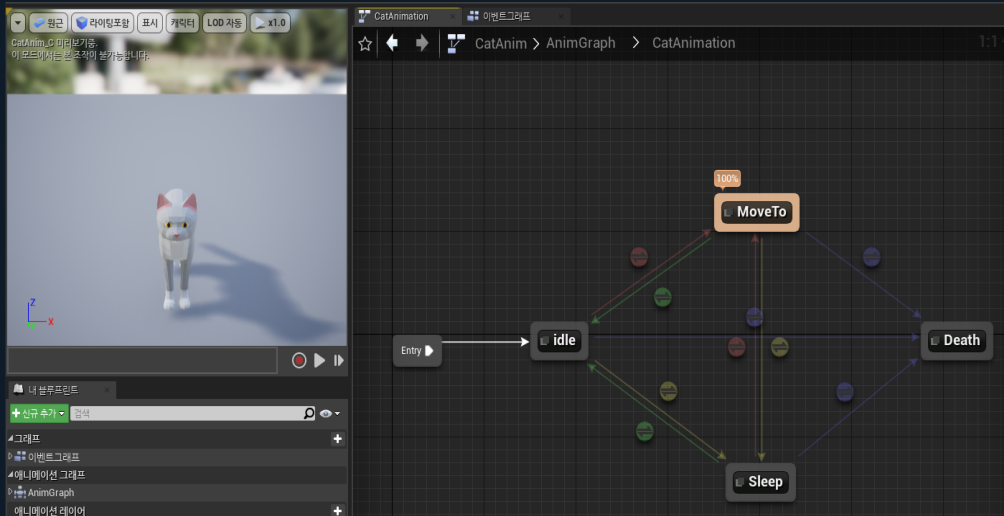
AddDataPoint와 AddSeries로 지속적으로 Data를 Input하여 현재 보이는 결과를 얻음
이를 통해 시뮬레이션 결과 분석이 용이해진다.



포식자



3-1. 포식자 - 개요



비헤비어 트리를 구현하고, 스테이트 머신의 상황에 맞는 기초적인 고양이 애니메이션 구현.

RunStateChange, StateMachine , FindBlock 등의 알고리즘은 토끼와 동일한 원리를 가짐

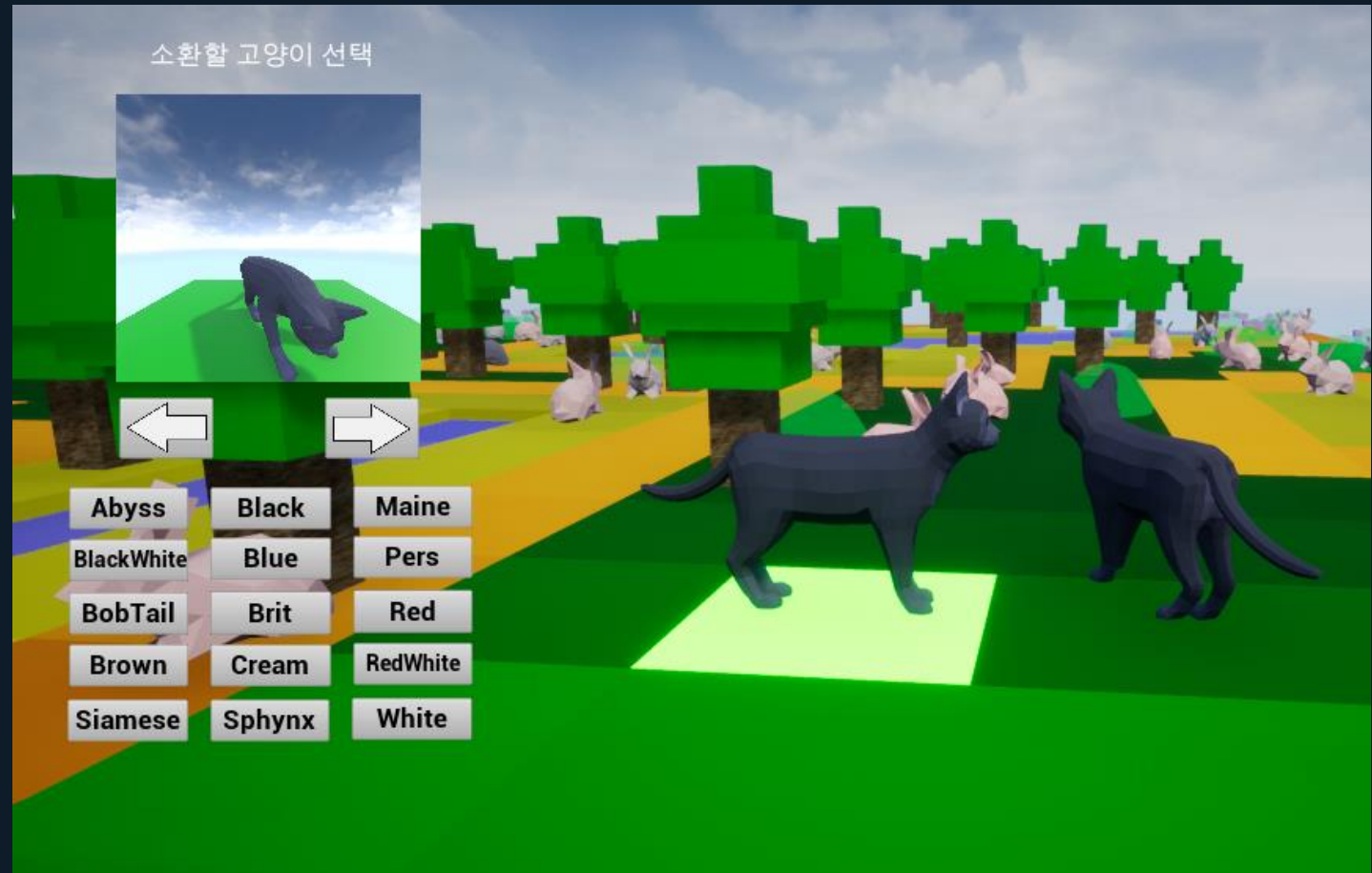
단, 고양이들은 Mate 기능이 없음.



3-2. 포식자 - 생성 시각화



RenderTarget을 사용하여 Widget의 Image에
RenderTarget을 사용하여 만든 Material을 세팅한다.



고양이를 선택하고, 마우스로 타일을 클릭하여 고양이를 소환하는 장면.
사용자의 편의성과 선택의 다양성을 증진하였다.



카메라



4-1. 카메라 -Trace Target & Picking



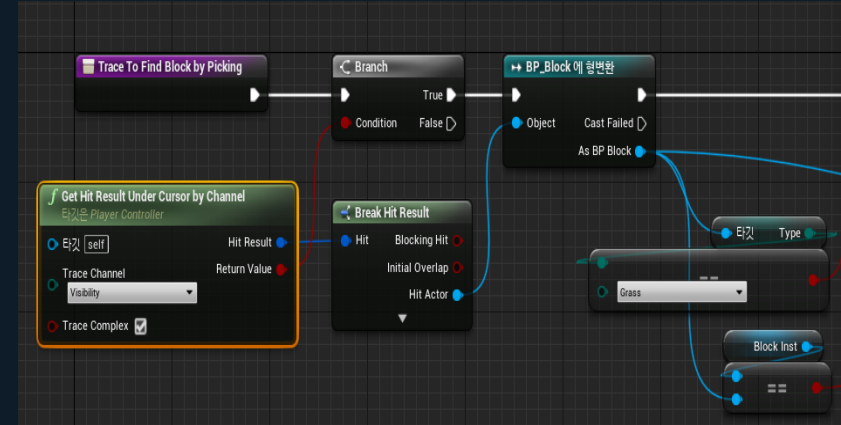
Mouse Overlap



Mouse Click

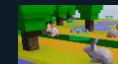


고양이도 동일하게
Overlap과 Click이 가능하다



Get Hit Result Under Cursor By Channel을 사용하여 마우스 포인터의 X,Y 값과 동일한 위치에 있는 픽셀을 가지고 있는 액터와 컴포넌트를 Hit Result로 리턴함.

이를 마우스가 Overlap 되었을 때, 그리고 Click 했을 때와 연동하여 TraceTarget & Picking 한다.



4-2. 카메라 -Custom Depth Stencil

엔진 - 렌더링

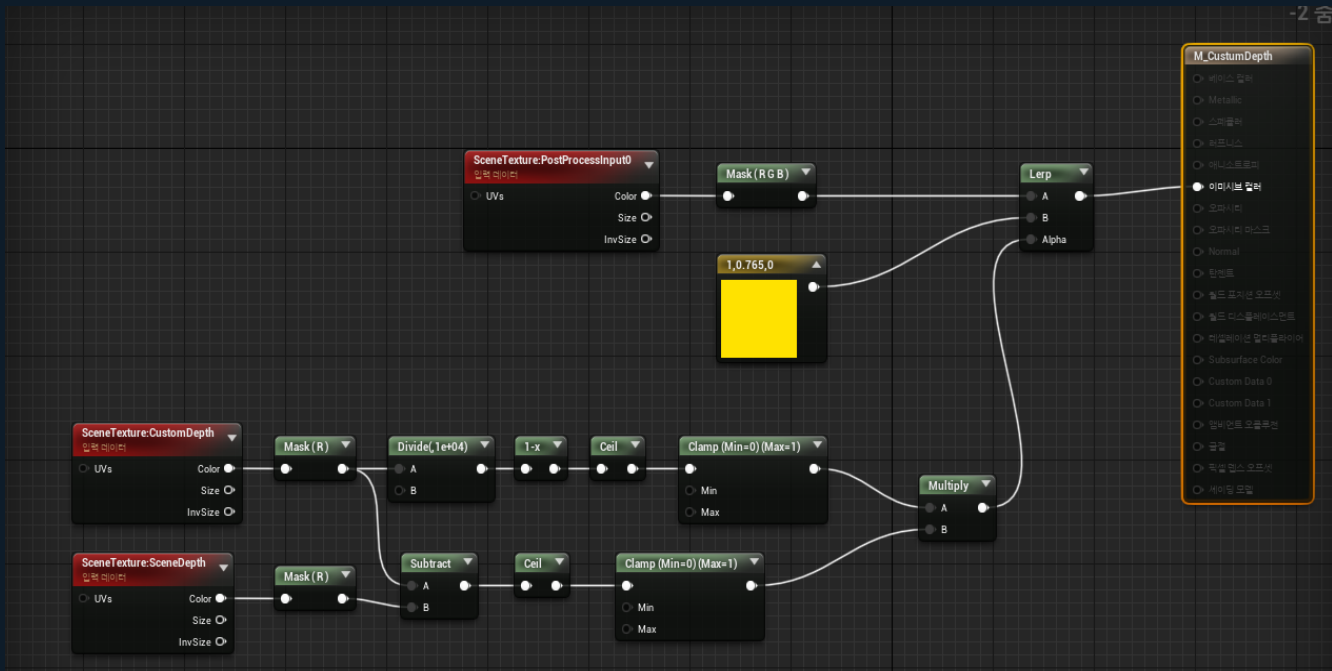
렌더러 설정 - 렌더링 설정들입니다.

▲ 포스트 프로세싱

커스텀 덤스-스텐실 패스

Enabled

커스텀 깊이 TAA 지터



CustumDepthStencil을 활용하여 가려진 부분을 실루엣으로 드러나게 하였다.

엔진-렌더링에서 커스텀덱스-스텐실 패스를 Enabled로 바꾸고 구현한 Material을 PostProcess에 적용하였다.

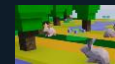
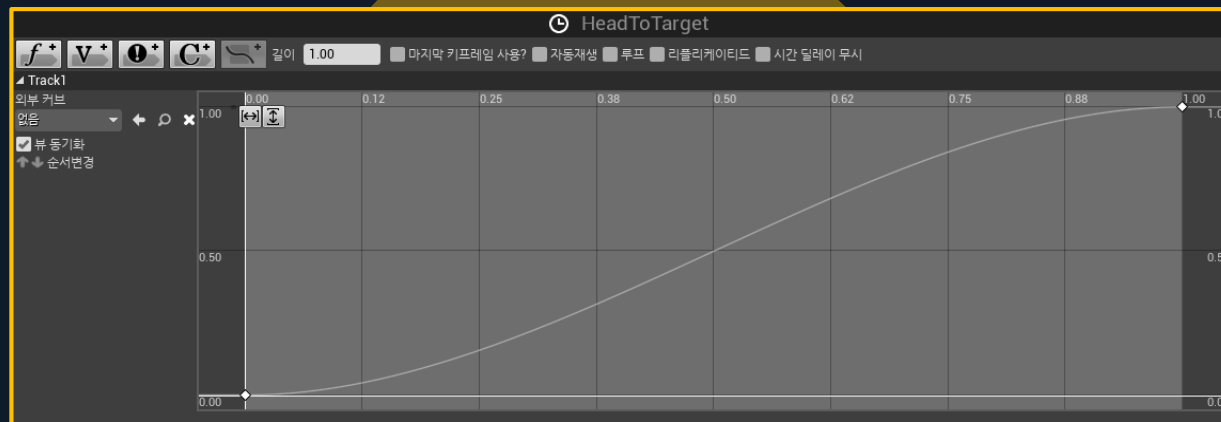
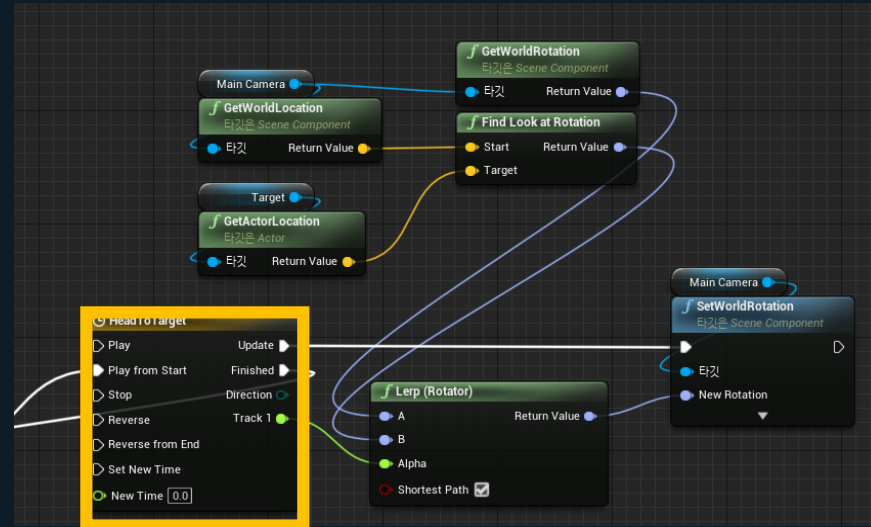


Rabbit Simulator

4-3. 카메라 -Following Camera



이후 Timeline을 사용하여 Following Target의 Rotation과 Location, Pivot의 Rotation과 Location을 부드럽게 오갈 수 있다.



4-3. 카메라 -Following Camera



타겟의 적응 대상이 된 토끼를 일정한 고도에서 추격하고 있다.





Rabbit Simulator technical introduction

Made by Unreal Engine Client 김노운

Tel: 010-7689-0904

Mail: nounkim2@gmail.com