Mobile Programming Final Report

총장님 타이쿤 Character Pathfinding

B577027 이재형

목차

[Mobile Programming Final Report 1](#_Toc27462518)

[총장님 타이쿤 Character Pathfinding 1](#_Toc27462519)

[개요 2](#_Toc27462520)

[작동 2](#_Toc27462521)

[목적지 설정 2](#_Toc27462522)

[BFS PathFind 2](#_Toc27462523)

[경로 추적 2](#_Toc27462524)

[구현 3](#_Toc27462525)

[GameManager 3](#_Toc27462526)

[MouseManager 4](#_Toc27462527)

[CharacterFSM 6](#_Toc27462528)

[Enums 6](#_Toc27462529)

[CharacterFSM 7](#_Toc27462530)

[PathNode 13](#_Toc27462531)

[어려웠던 점들 14](#_Toc27462532)

개요

총장님 타이쿤은 제가 현재 진행중인 졸업작품 프로젝트입니다.

이 게임 내에 등장하는 조작 불가능한 캐릭터 (학생, 교수 등)에 BFS Pathfinding을 구현하여 목적지까지 걸어가도록 구현하였습니다. 해당 코드는 CharacterFSM.cs에 구현되어 있습니다.

또한 추가적으로 게임의 건설 시스템과 캐릭터들 사이에 옵저버 패턴을 구현하여 캐릭터들이 경로를 재설정하도록 구현하였습니다.

작동

목적지 설정

Start시 모든 생성된 건물들은 GameManager가 관리합니다.   
건물은 각자 Building 태그를 가집니다.   
각 캐릭터는 이렇게 생성된 건물들 중 무작위 하나를 목적지로 설정합니다.  
이후 해당 목적지까지의 경로를 탐색합니다.

BFS PathFind

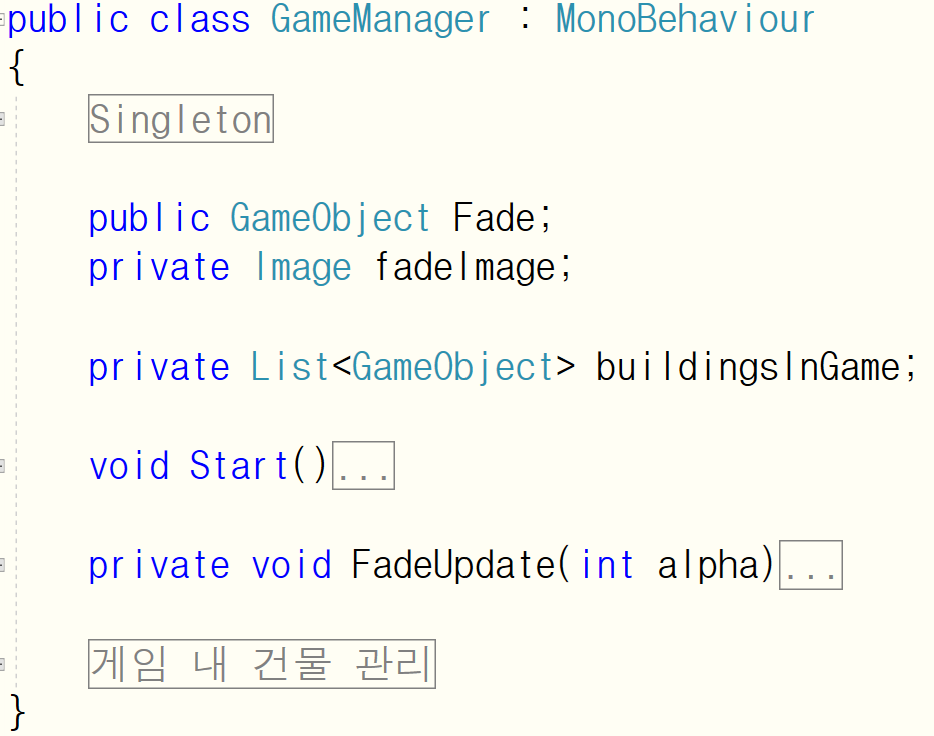
캐릭터의 위치로부터 노드를 생성하여 n방향 레이캐스트를 실행합니다.  
충돌체가 없는 지점은 새로운 노드를 생성하며 이 때 겹치는 노드가 없는지 탐색합니다.  
캐스트가 목적지인 건물에 닿으면 경로를 설정하고 캐스트를 종료합니다.

경로 추적

설정된 경로 노드를 따라 이동합니다.  
도중에 Subject인 MouseManager에게 연락이 오면 이동을 중단하고 경로를 재설정 후 이동합니다.

구현

GameManager



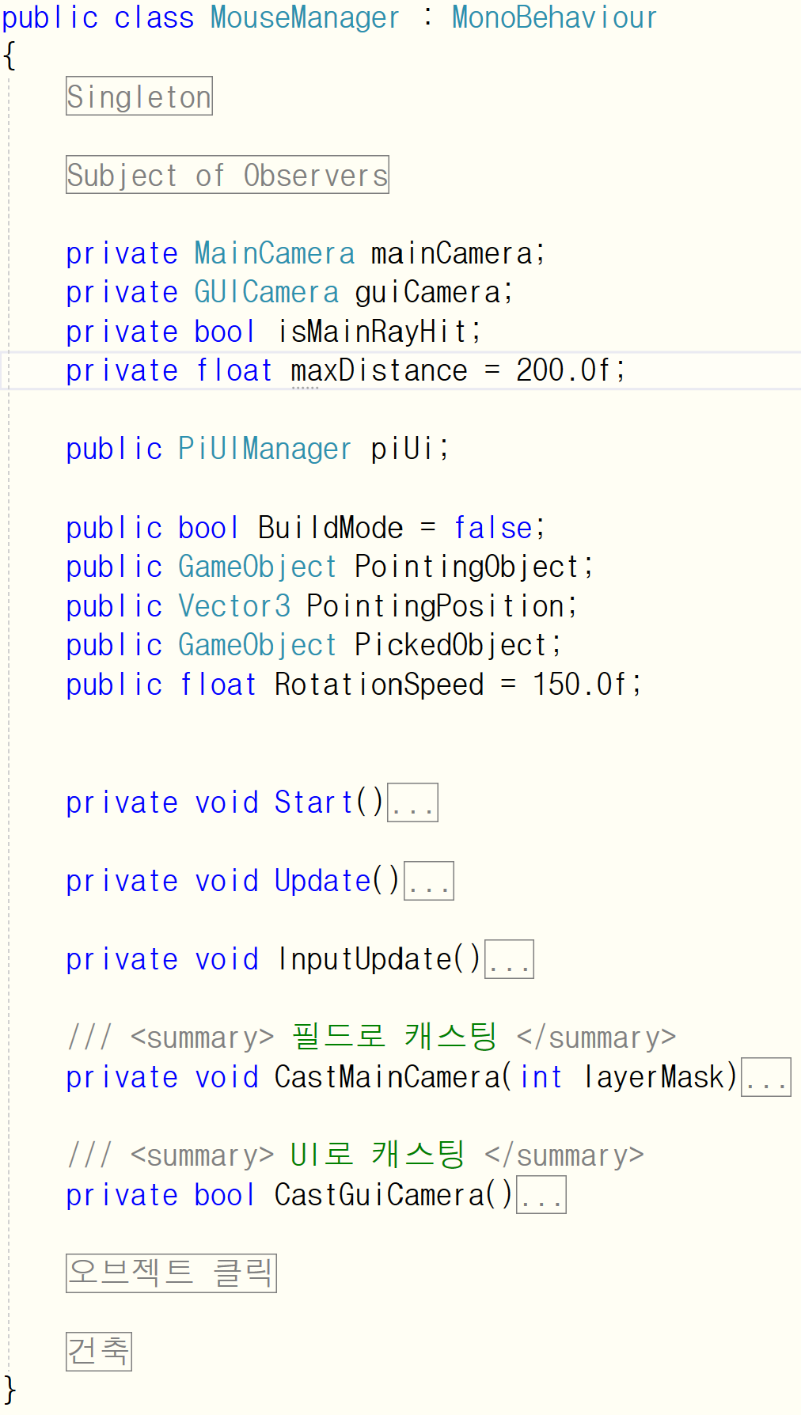
건물들을 관리하는 GameManager 클래스입니다.



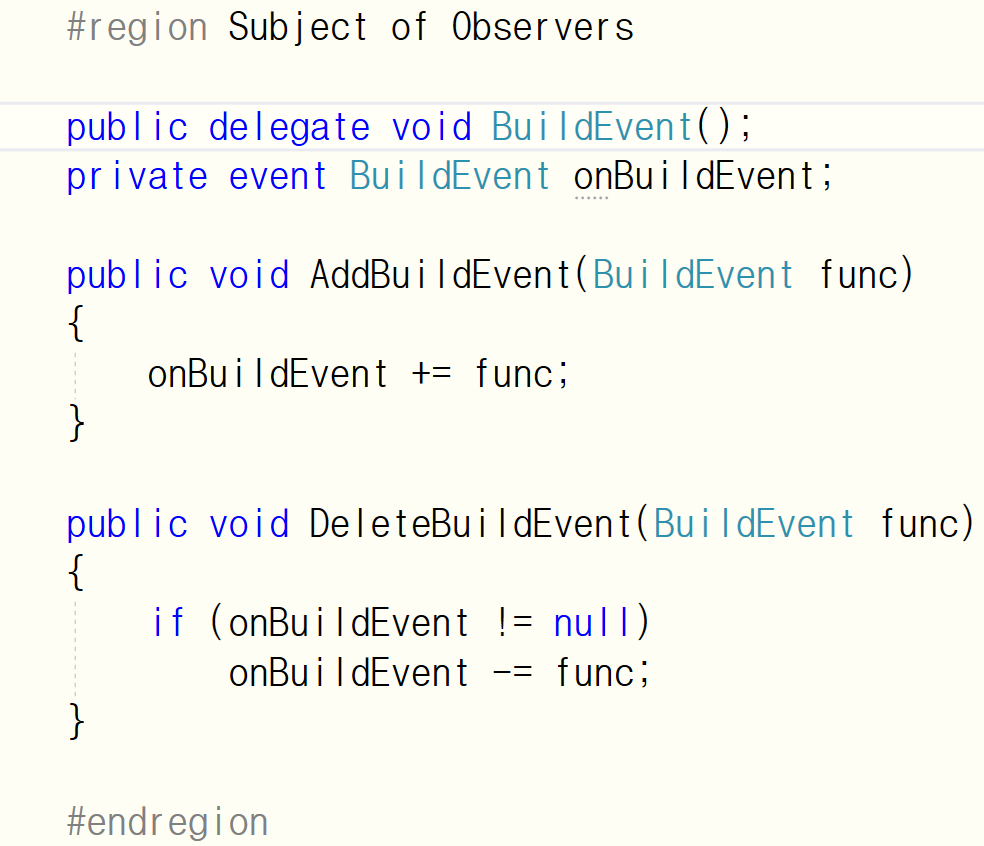
새로운 건물이 건설될 때 마다 건물정보를 리스트에 보관하며 이 중 무작위 건물정보를 보내줄 수 있도록 구현했습니다.

이는 Character가 목적지를 설정할 때 사용되며 추후 무작위가 아닌 특정 건물을 주도록 변경할 예정입니다.

MouseManager



건물의 건설을 관리하는 MouseManager입니다.

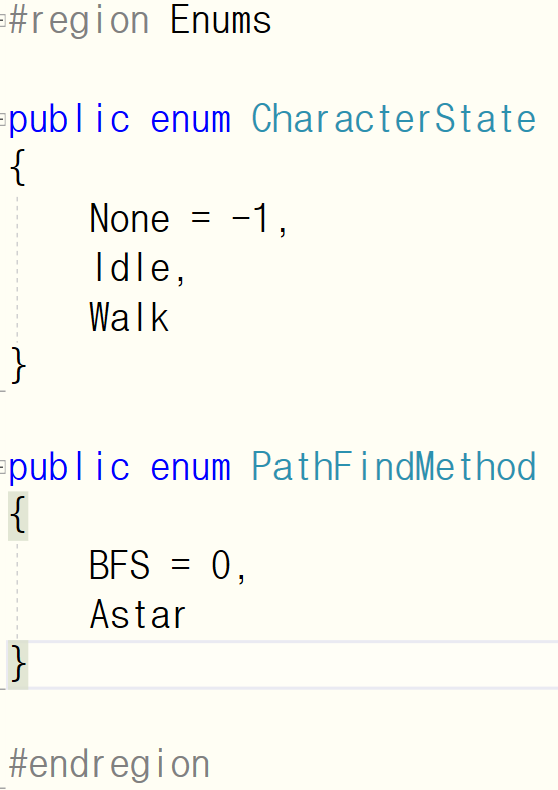


MouseManager가 관리하는 Observer들의 Subject 구현 부분입니다.

옵저버들의 함수를 delegate에 등록해 건물 정보의 변경을 알립니다.

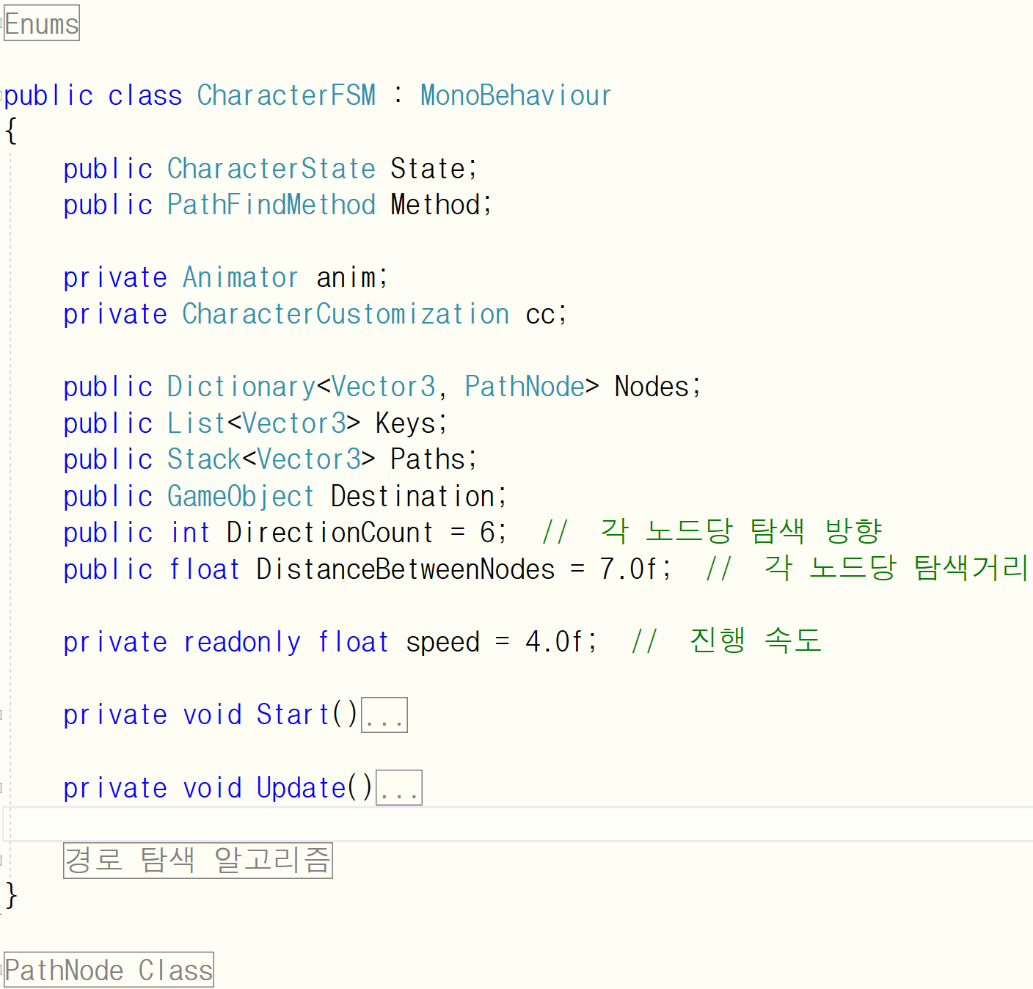
CharacterFSM

Enums



캐릭터 클래스의 상태와 길 찾기 방식 Enum입니다.

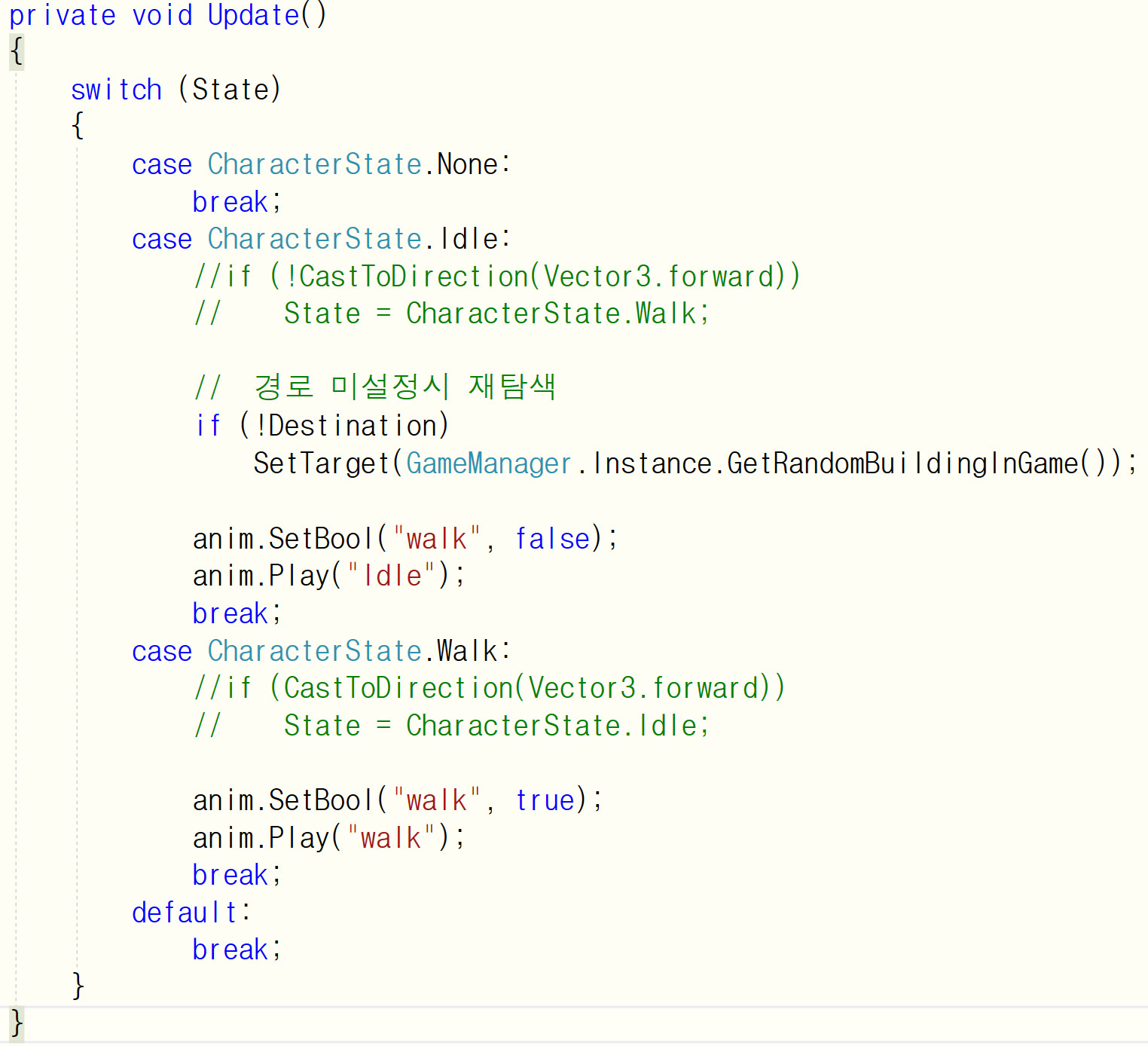
CharacterFSM



각 캐릭터가 갖는 CharacterFSM 클래스입니다.

Nodes를 Dictionary로 구현하고 그 키 값들을 List로 저장하여 읽을 때 속도를 향상시켰습니다.

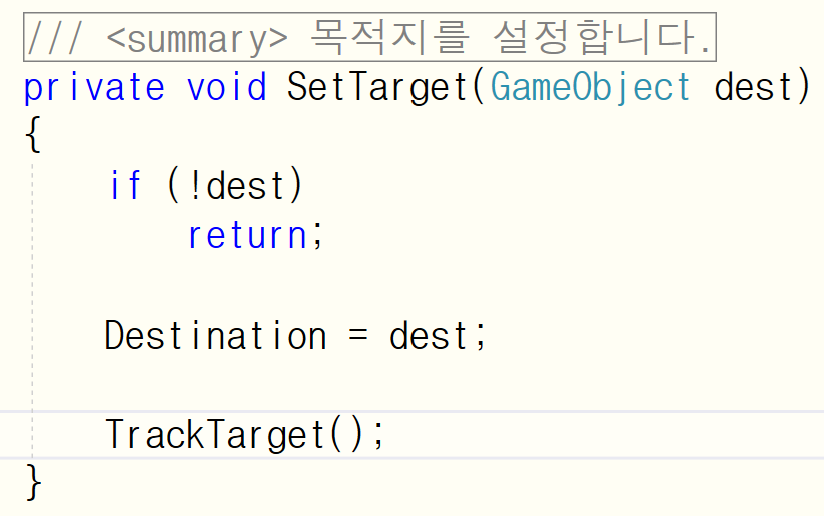
Pathfind를 통해 생성된 경로는 Paths 스택에 저장됩니다.



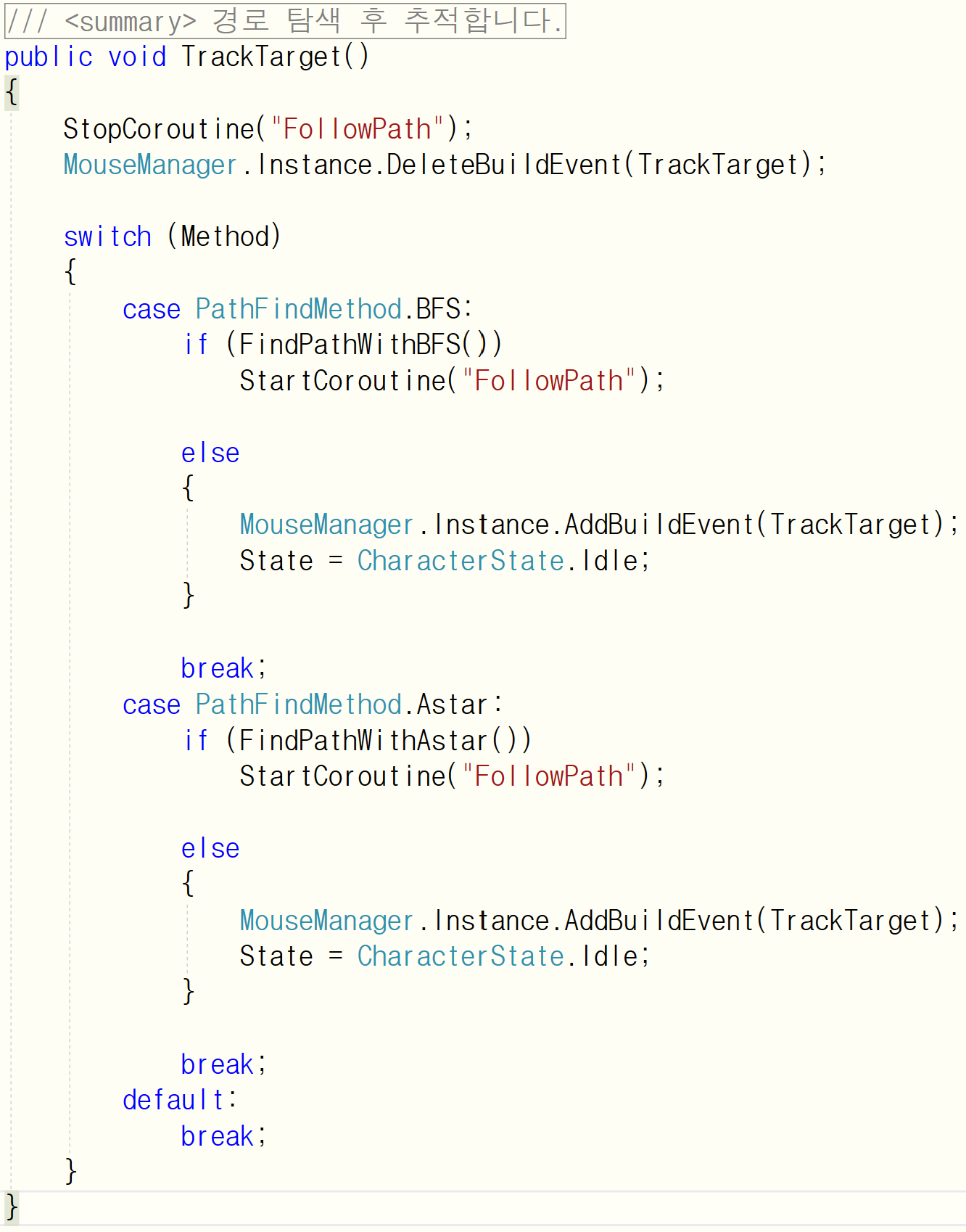
캐릭터는 Idle상태일 때 목적지를 탐색하여 저장합니다.



경로 탐색과 추적에 사용되는 함수들입니다.

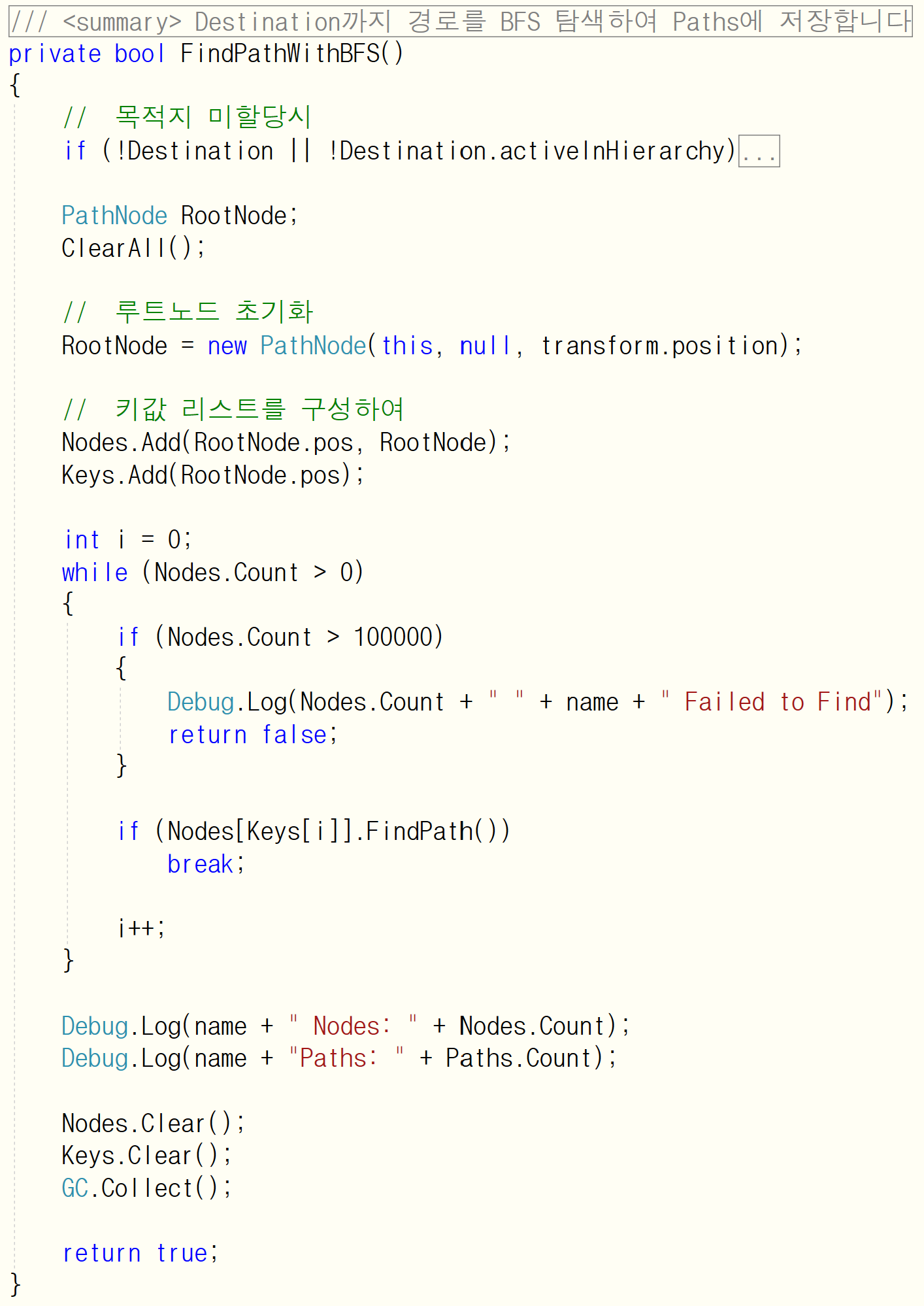


목적지가 설정되면 경로를 탐색합니다.



도중에 경로를 재탐색하는 경우를 대비하여 경로 추적 코루틴을 종료하고 마우스 매니저의 연락을 잠시 받지 않습니다.

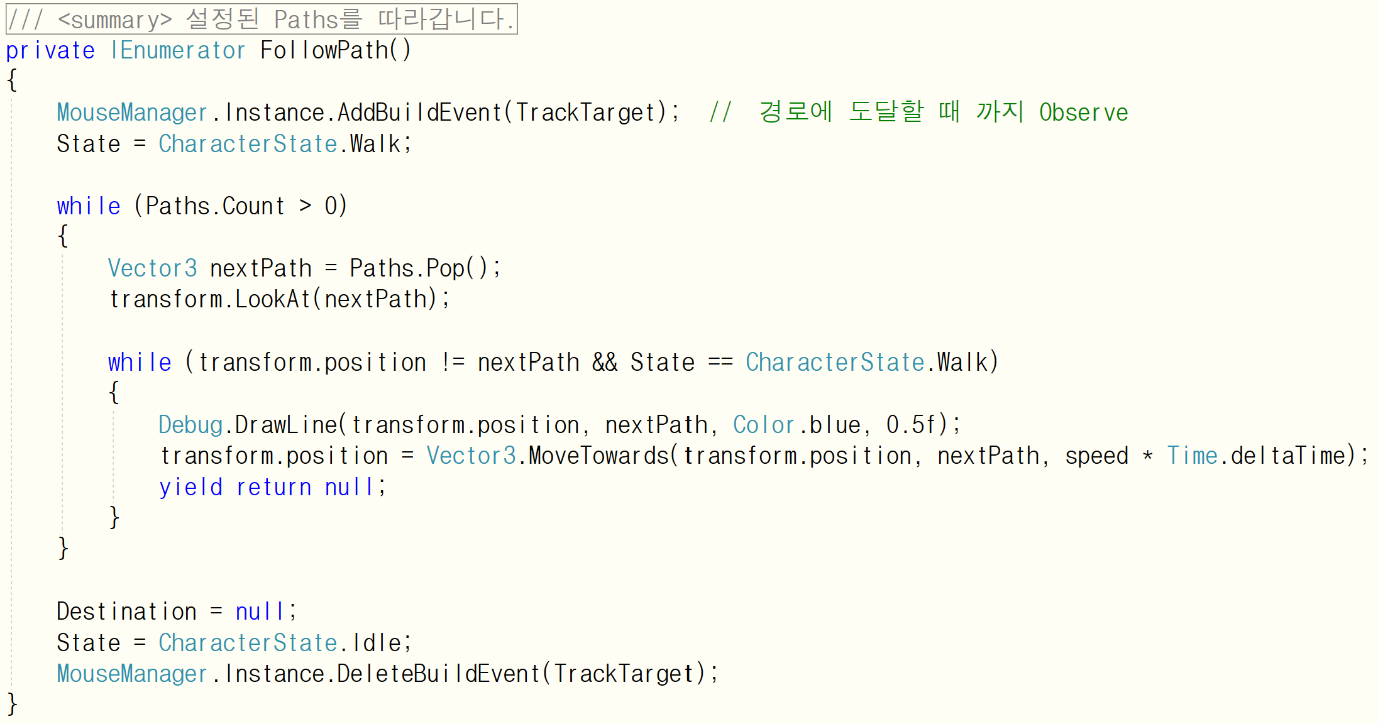
경로 설정에 성공하면 추적 코루틴을 수행하고, 실패하면 목적 건물의 수정을 기다리며 가만히 있습니다.



목적지가 없거나 비활성 상태면 경로를 찾지 않습니다.

Collection들을 정리 후 캐릭터의 위치에 RootNode를 생성하고 FindPath함수를 통해 생성된 자식 노드들의 FindPath함수를 실행합니다.

노드가 10만개 이상 생길 경우 탐색을 포기하도록 하였습니다.



예제로 주어진 코드와 흡사한 경로 추적 코루틴입니다.

Paths 스택에서 경로를 Pop하여 경로들을 Reverse하는 불필요한 소요를 없앴습니다.

경로를 추적하는 동안 건물의 배치가 수정되는 연락을 받도록 옵저버로 등록합니다.

PathNode

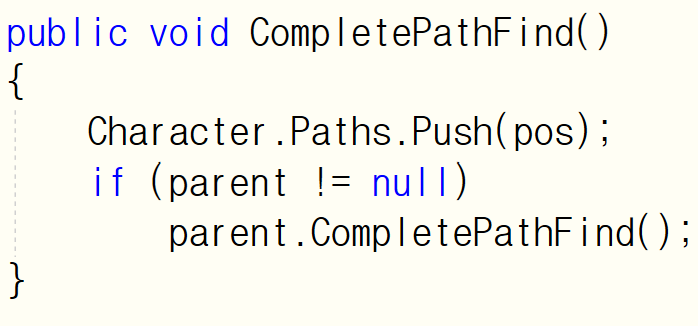


경로 노드 클래스입니다.



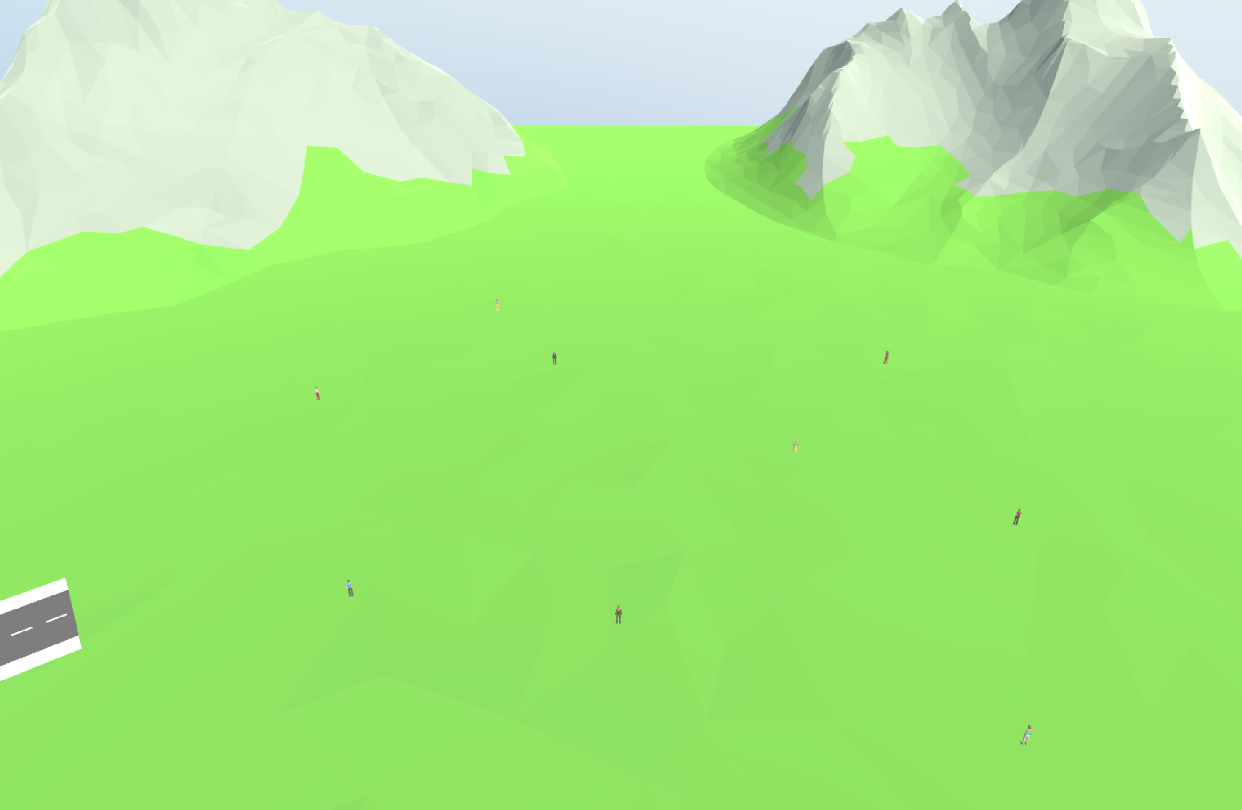
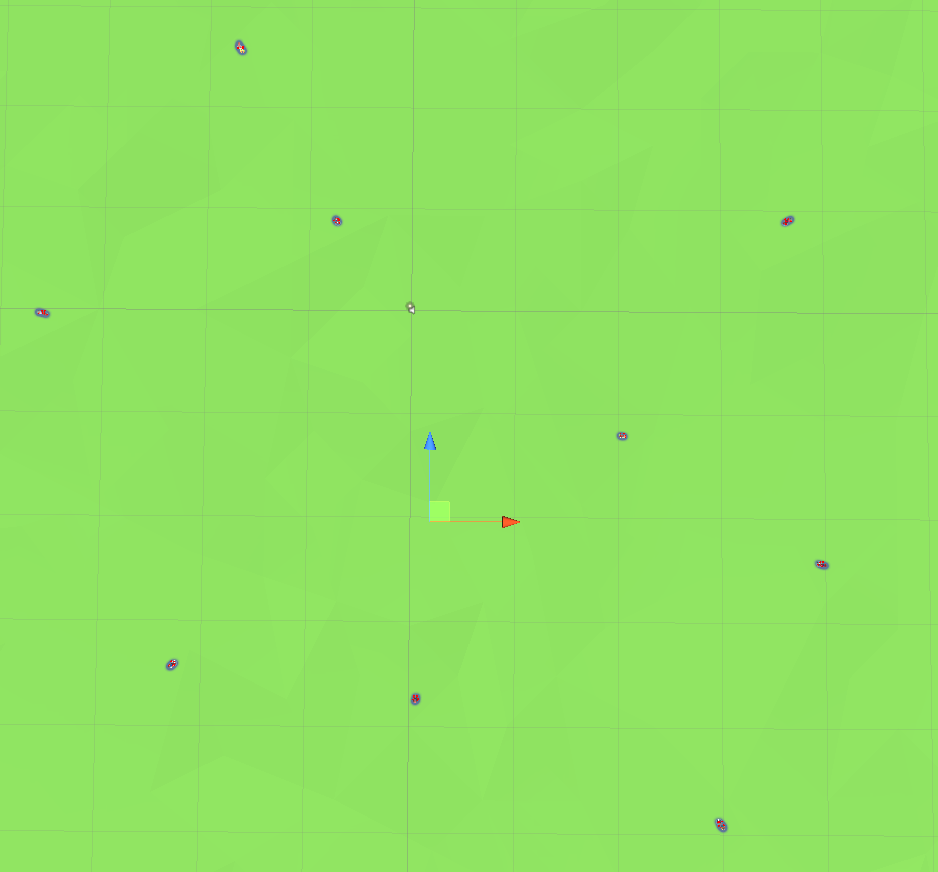
각 노드로부터 SphereCast로 탐색합니다.  
이 때 탐색할 방향의 수와 거리를 임의로 조작할 수 있게 구현했습니다.  
예를 들어, dir값이 4이면 전후좌우, 6이면 0, 60, 120, 180, 240, 300도 방향으로 캐스트합니다.

탐색 중에 목적지를 발견하거나 다른 노드가 목적지를 발견하면 탐색을 중단합니다. 그렇지 않을 경우, 탐색한 위치에 노드를 생성하여 탐색시킵니다.

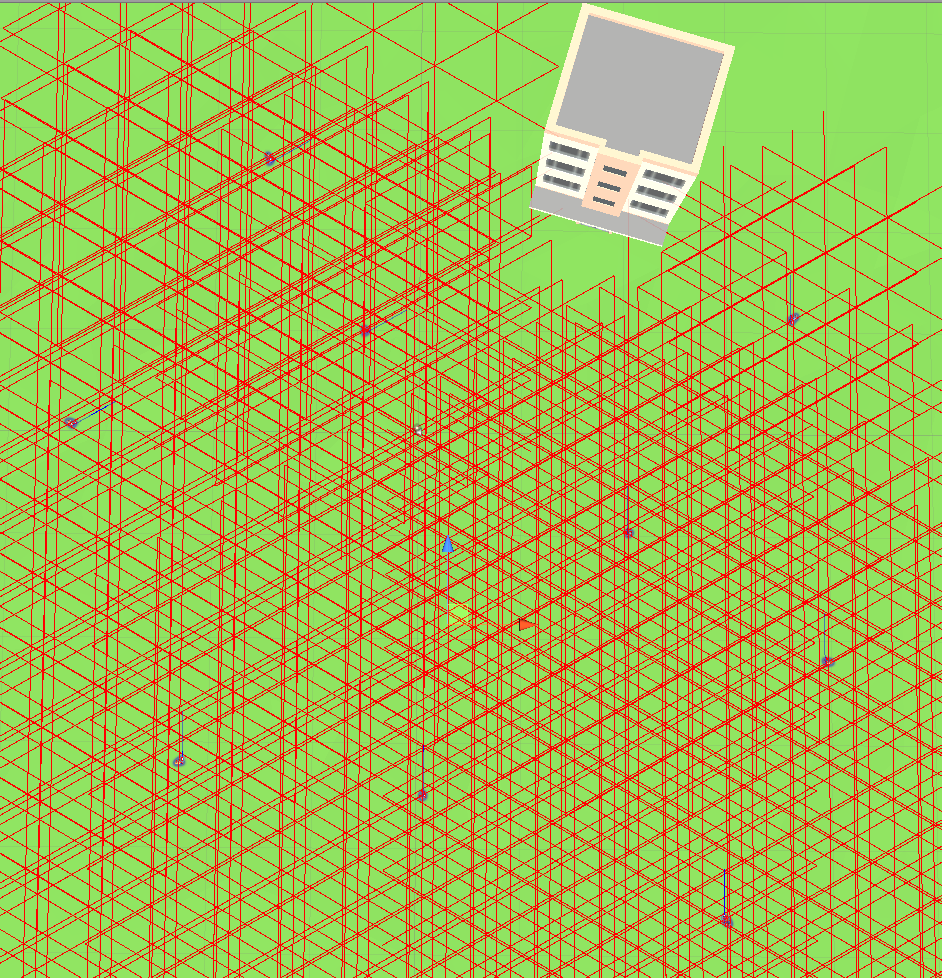


경로를 찾은 노드는 Paths스택에 부모 노드들을 타고 올라가 경로를 저장합니다.

테스트 스크린샷



건물을 배치하지 않은 상태로 캐릭터 10명을 배치해 시작하였습니다.

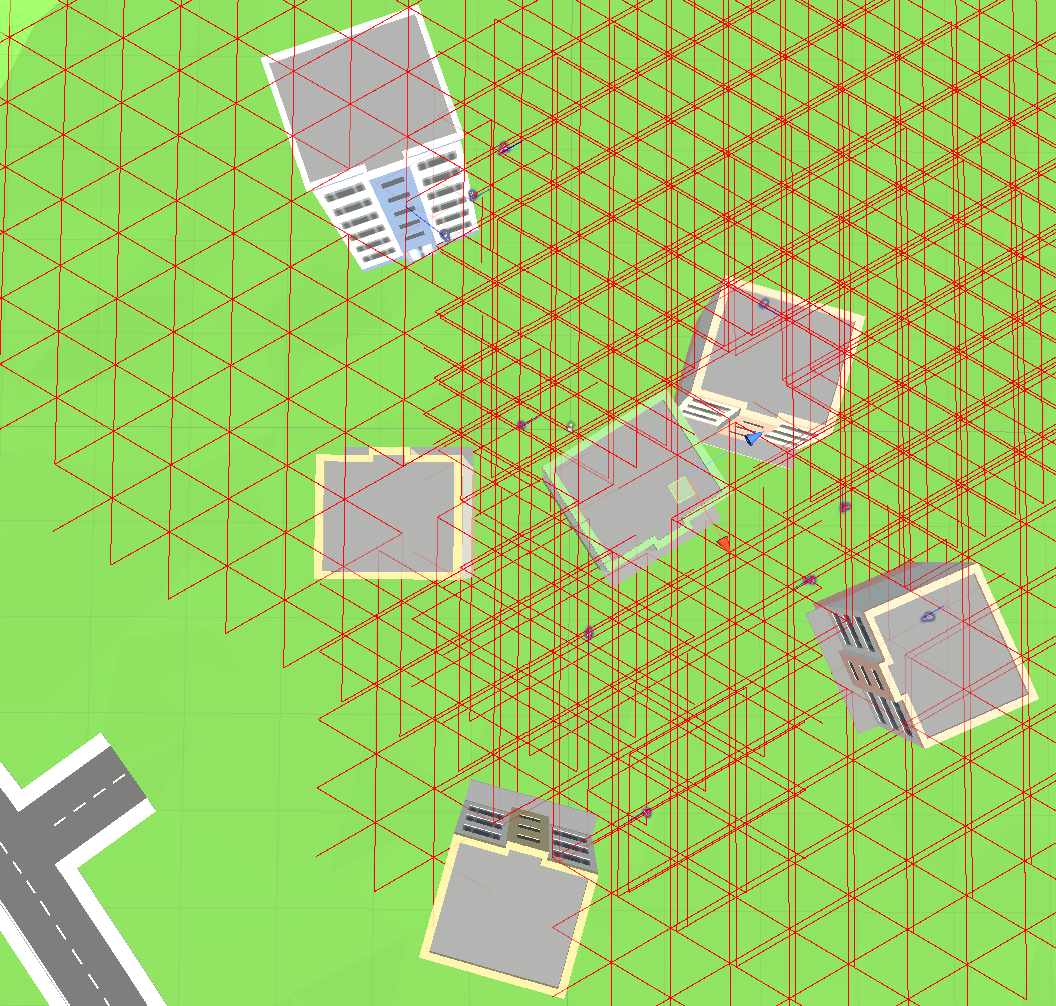


건물을 배치하면 일제히 경로를 탐색합니다.   
건물이 하나뿐이라 전부 한곳으로 몰리는 모습입니다.

캐릭터 10명은 모두 6방향 7거리 탐색으로 세팅해두었습니다.



건물을 여러 채 배치한 화면입니다.  
각자 목표로 하는 건물에 도달하면 다른 건물을 목표로 재설정하므로 각자 알아서 왔다갔다 하는 모습입니다.



건물 배치를 수정하니 캐릭터들이 경로를 재탐색하는 모습입니다.

어려웠던 점들

처음에는 그리드 좌표에서 벗어나 유기적으로 길찾기를 수행할 방법을 고민했습니다.

RPG게임 검은사막의 경우 지도에 미리 생성된 노드들이 잔뜩 있어서 그걸 통해 길찾기를 수행한다고 얼핏 얘기를 들은 적 있는데 그렇게 하면 속도는 엄청 빠르고 캐릭터가 접근 불가능한 곳은 탐색을 수행하지 않으니까 꽤 좋은 방법이라고 생각했습니다. 다만 노드를 사전에 구성하는데 노가다를 많이 해야 할 것 같아 포기했습니다.

2차적으로 생각한 방법 Raycast를 활용하여 Capsule Collider를 가지는 노드 GameObject를 생성하여 캐릭터끼리도 부딪히지 않는 길찾기를 고민했으나 Instanciate 과정에서 오버헤드가 너무 많이 발생해서 그냥 클래스만 생성되도록 변경했습니다.

또 노드 컬렉션을 Queue로 구성했는데 겹치는 노드에 대한 탐색속도가 너무 느려서 방법을 고민했습니다. 인터페이스를 명시적으로 구현해서 탐색비교를 pos만 하도록 했지만 이 마저도 너무 느렸습니다. 그래서 Dictionary에 노드들을 저장해서 Key값을 pos로 저장하면 좋을 것 같았습니다. 마침 Key값에 대한 탐색이 O(1)에 수렴한다는 MSDN 글을 보고 냉큼 채용해보니 속도차이가 많이 나더라구요.

그렇게 길찾기를 구현하고 이후에 매니저 스크립트와 연동해서 캐릭터가 유기적으로 이동하도록 구현하였습니다.

이후 사람이 많아지면 오버헤드가 더 커질 것을 대비해 다른 길찾기 방법을 적용해보려고 합니다.