[유니티 기초] 15. 길찾기 A.I 구현

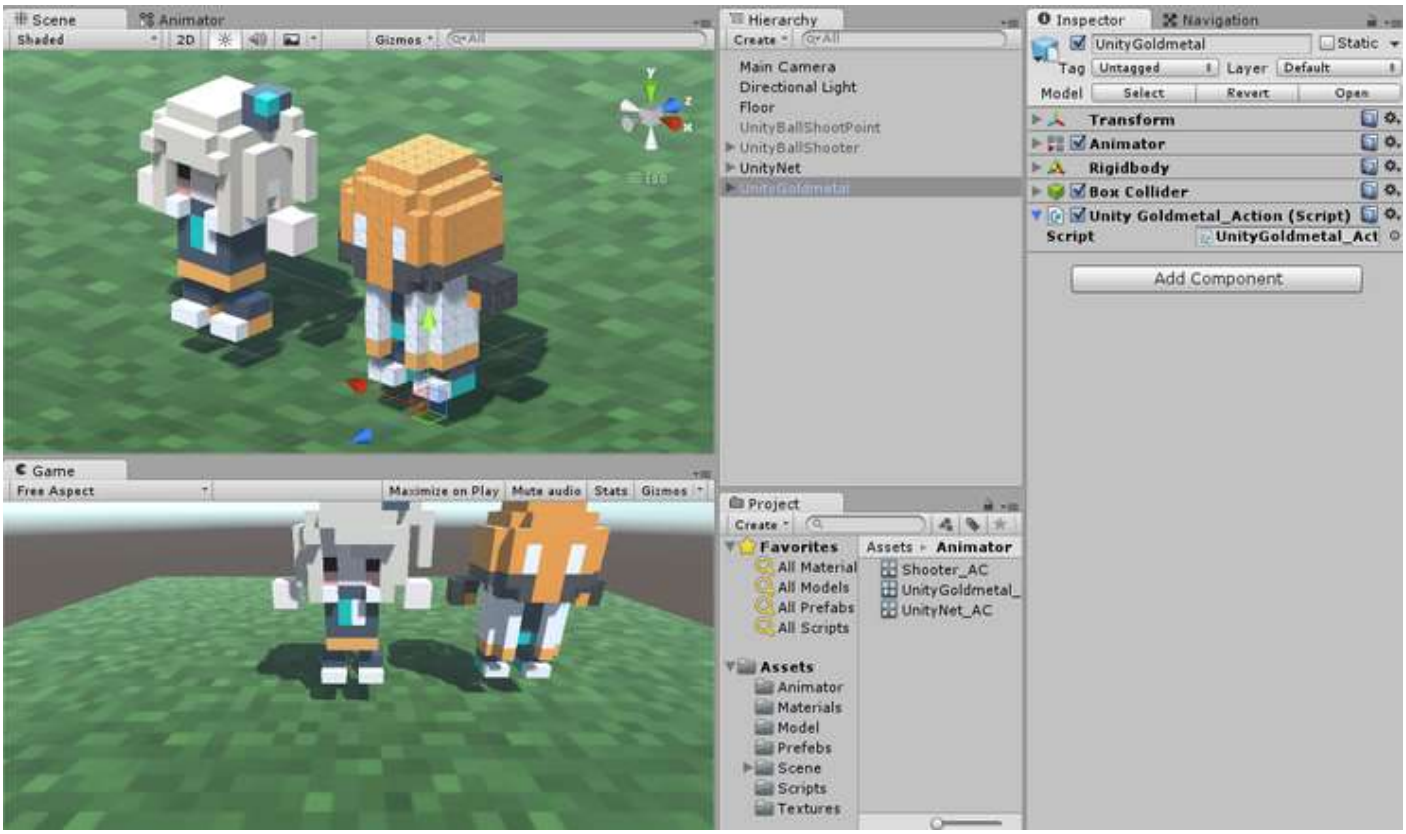
유니티·개발

2015. 10. 17. 22:42

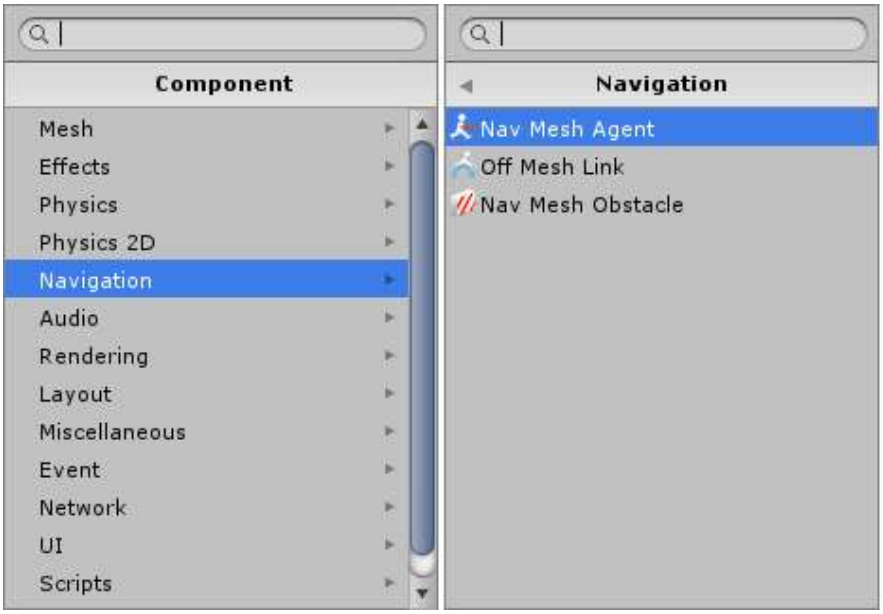
http://blog.naver.com/gold_metal/220511730779

안녕하세요.
골드메탈입니다.

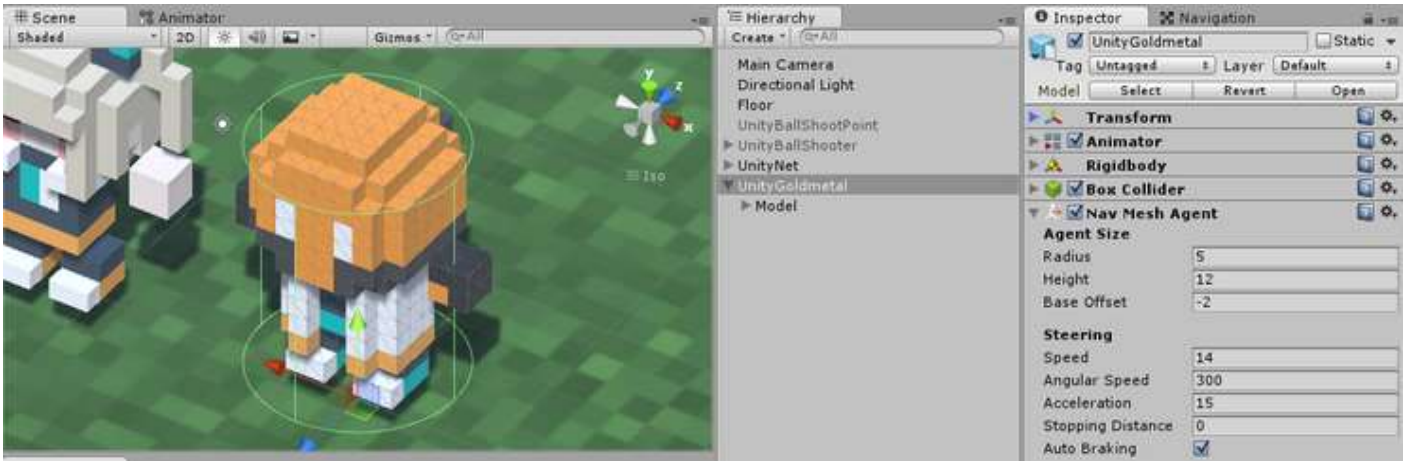
이번 편은 자동으로 최적화된 길을 찾아가는
A.I (인공지능)을 구현해보도록 하겠습니다.



이번에는 필자의 캐릭터를 넣어보았습니다.
이 새로운 캐릭터가 플레이어를 따라가도록 만들 것입니다.

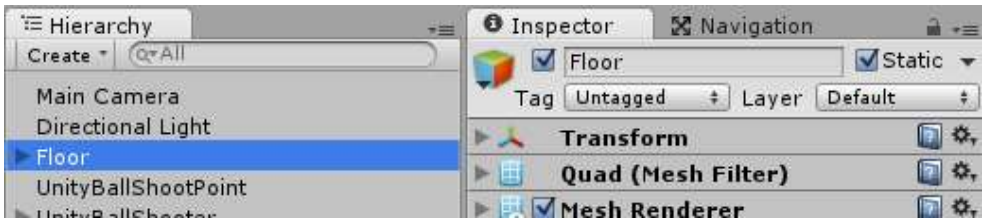


Add Compoenet - Navigation - Nav Mash Agent 메뉴로
내비 에이전트를 만들어줍니다.



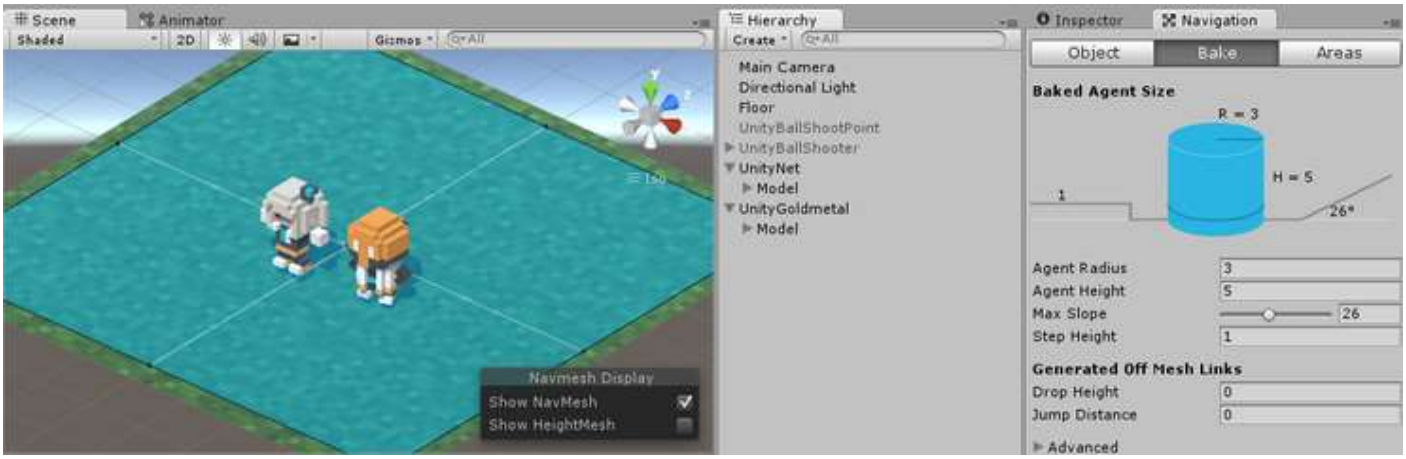
내비 에이전트가 추가되는 순간부터,
충돌기준이 충돌체(Collider)에서 내비 에이전트로 바뀝니다.
때문에 크기를 오브젝트에 맞추어야겠지요.

이 외에도 이동 속도와 회전 속도,
가속력을 설정할 수 있습니다.



그럼 이 인공지능이 돌아다닐 구역을 설정하도록 합시다.
먼저 지형인 오브젝트는 속성 창 위쪽의
정적(Static) 에 체크해줍니다.

구역을 만들기 위해선
지형을 미리 올려두어 계산하게끔 해야하기 때문이죠.
이 계산을 **Bake** 라고 합니다.



Window - Navigation 으로 내비게이션 속성창을 추가합니다.
그 다음 Bake 탭으로 가면 윗 그림처럼 나타나지요.

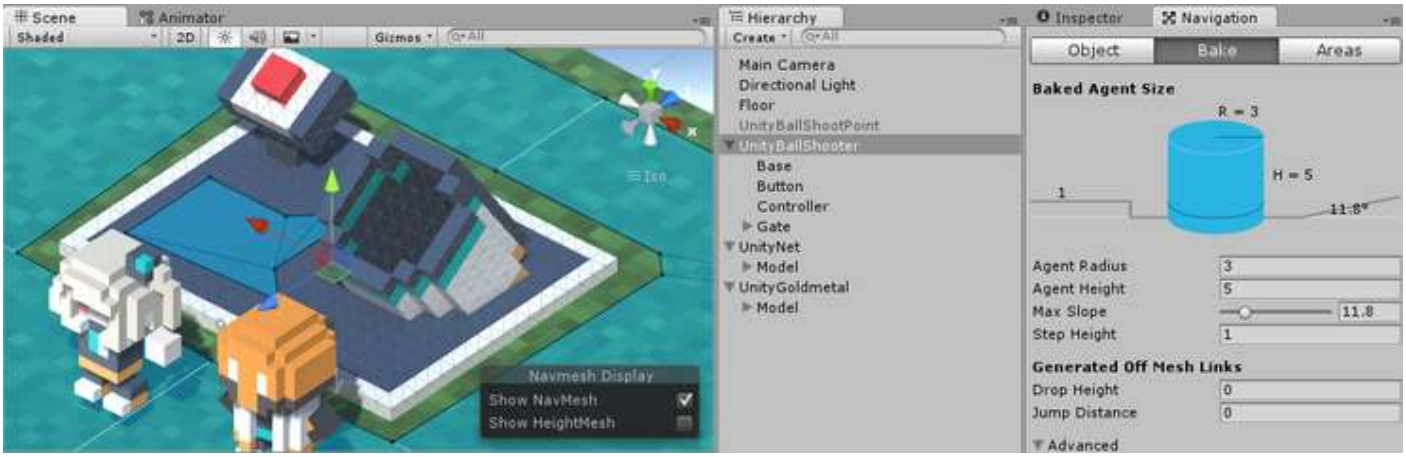
---

1. Agent Radius
: 내비 에이전트의 반지름.
: 이 값만큼 외각 여백이 결정됨.
2. Agent Height
: 내비 에이전트의 높이.
: 최대 등산 각도와 계단 높이에 영향을 줌.

3. Max Slope
: 최대 등산 각도.
:각도가 높을수록 그만큼 각진 지형도 구역에 포함.

4. Step Height
: 계단 높이.
: 이 값만큼 높이 차가 있는 지형도 구역에 포함.

여러 속성 중에서도 계단 높이는
구역 설정에 많은 영향을 주는데요.



계단 높이가 1일 때
그 이상의 지형을 벽으로 인식하는 반면,

계단 높이를 2로 올리자
벽으로 인식되던 지형도 올라가도록 바뀌었네요.

이렇게 구역을 재설정 할 때에는
속성창 아래의 Clear 한번 해주고 난 후,
Bake로 재생성 해주어야합니다.

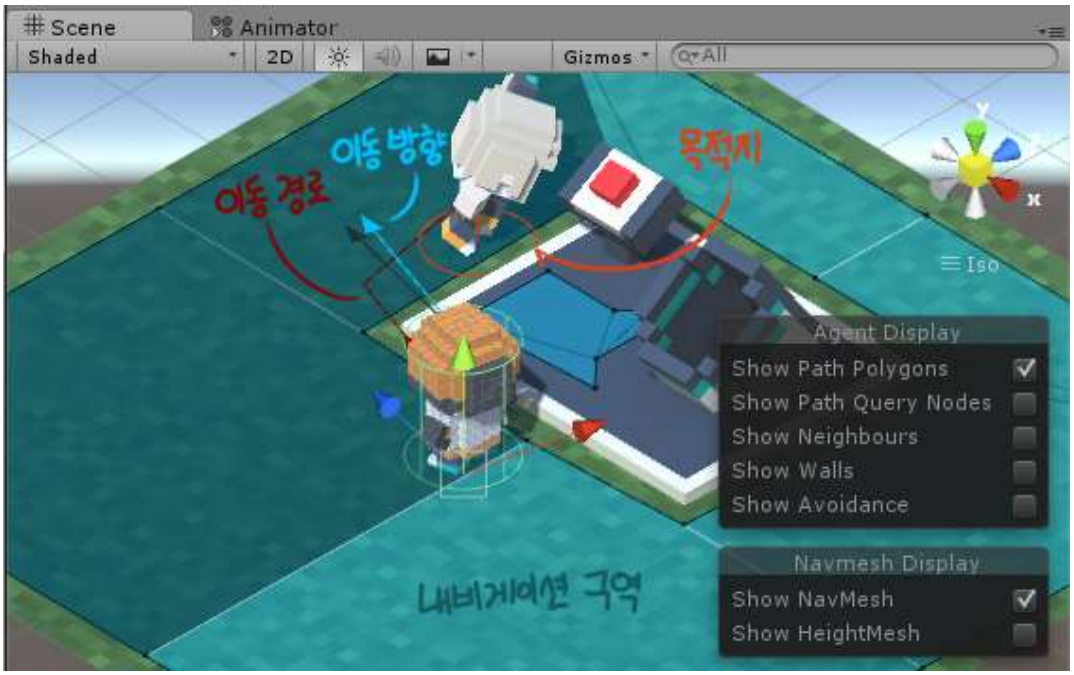
이제 마지막으로
스크립트에서 이동 로직을 만들면 됩니다.

```
public class UnityGoldmetal_Action : MonoBehaviour {

    GameObject player;

    Animator animator;
    NavMeshAgent nav;

    //-------------------------------------------------------[ Override Function ]

    void Awake ()
    {
        player = GameObject.FindGameObjectWithTag ("Player");

        animator = GetComponent<Animator> ();
        nav = GetComponent<NavMeshAgent> ();
    }

    void Update ()
    {
        nav.SetDestination(player.transform.position);
    }
}
```

내비 에이전트를 불러온 다음,
업데이트 사이클에서 **SetDestination (Vector3)** 함수를 통하여
목적지를 설정합니다.

필자는 플레이어를 따라다니는 인공지능을 만들기 위해
플레이어의 위치 값을 넣어주었습니다.

게임 상에서 보니
만들어진 구역에서 플레이어를 따라오는군요.

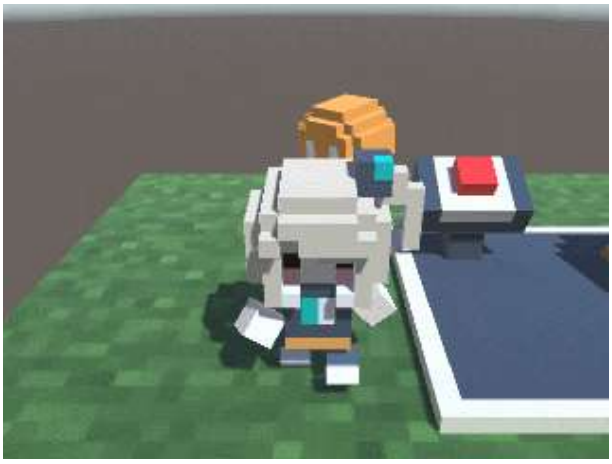


오브젝트 속성창에서 내비 에이전트를 펼치고.
내비게이션 속성창으로 바꾼 상태로 프로젝트를 재생해보면
이렇게 내비 에이전트가 어떻게 이동하는지 보여줍니다.

```
void Update ()
{
    nav.SetDestination(player.transform.position);

    AnimationUpdate();
}

void AnimationUpdate ()
{
    if(nav.destination != transform.position)
        animator.SetBool("isWalking",true);
    else
        animator.SetBool("isWalking",false);
}
```

위에서 보이는 내비 속성을 사용하여
간단하게 애니메이션을 작동시켜보죠.

필자는 **NavMeshAgent** 클래스의 **destination** 값을 가져와서
목적지에 도달하기 전까지 걷는 애니메이션이 실행되도록 하였습니다.



이렇게해서 인공지능이 자연스럽게 따라오는
장면이 완성되었습니다.

이 방법으로 플레이어를 따라오는
몬스터 혹은 NPC를 구현할 수 있습니다.

열다섯번째 강좌는 여기까지입니다.

다음 강좌에서는 Static을 이용한
점수 시스템 구현에 대해 다뤄보겠습니다.

그럼.