[유니티 기초] 18. 아이템 던지기

유니티·개발

2015. 11. 4. 22:05

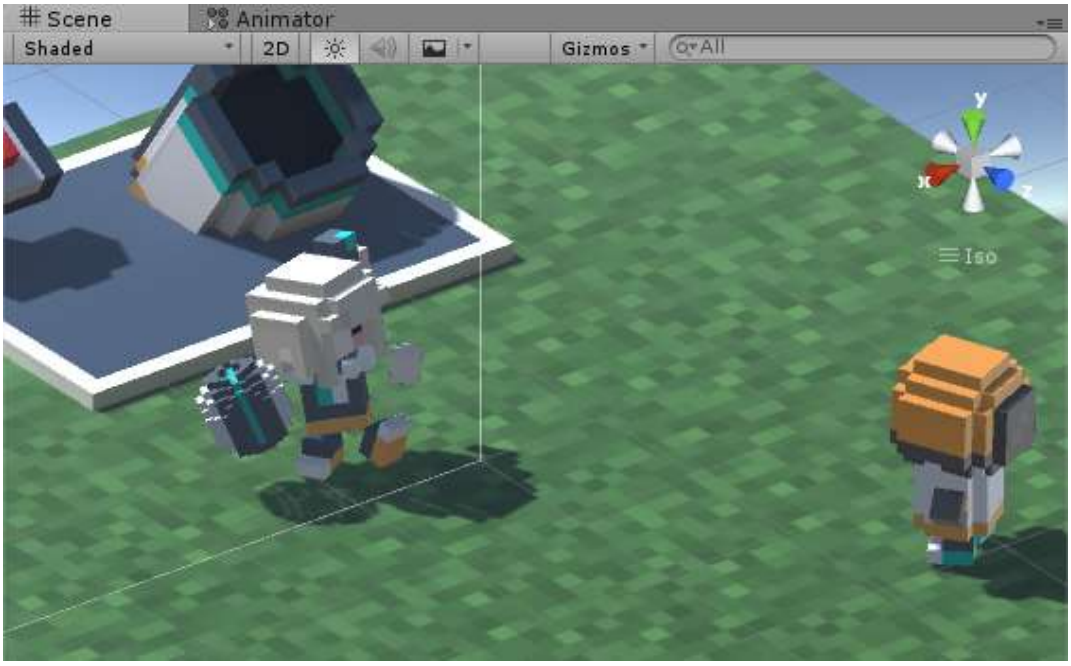
http://blog.naver.com/gold_metal/220529289273

안녕하세요
골드메탈입니다.

이번에는 유니티에서 광범위하게 쓰여지고 있는
Ray 시스템을 이용한 아이템 투척에 대해 구현해보고자 합니다.

17편에서 아이템을 집어들었다면
이번 편에서 그걸 던지는 모션을 구현할 것입니다.

마우스를 누르는 순간 조준 상태가 되고
누른 상태로 마우스를 움직여 캐릭터를 회전한 다음,
마우스를 떼면 그 방향으로 던지는 로직으로 가겠습니다.



일단 마우스를 누르면 조준 상태에 돌입하도록
애니메이션을 넣어주었습니다.

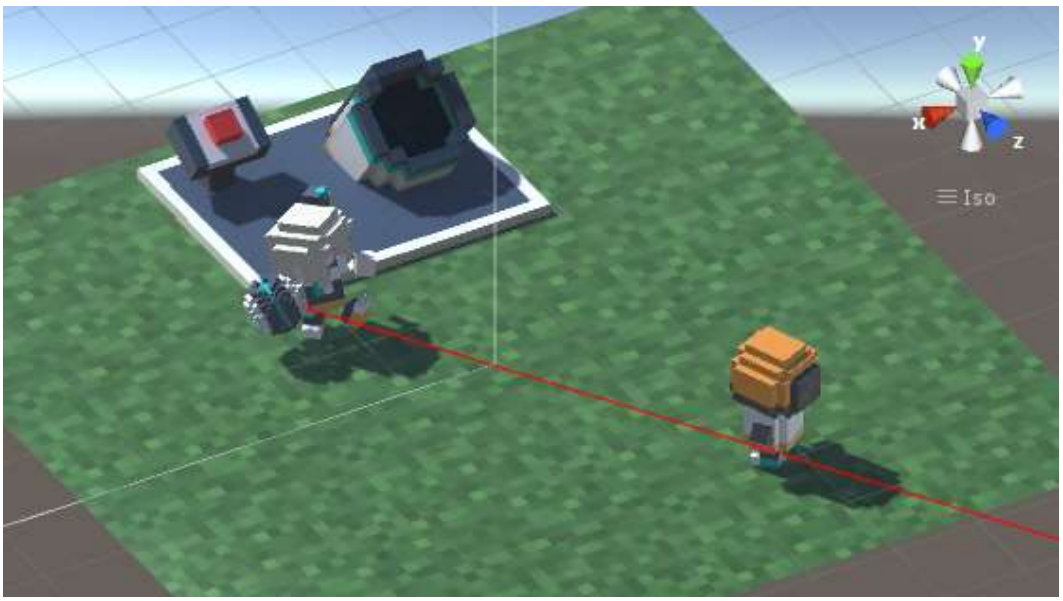
이제 이 정도는 직접 구현하실 수 있을거라 생각하고
본론으로 들어가지요.

Ray 는 무엇인가요?
일명 레이저 혹은 빔.
레이는 출발점과 도착점을 가지고 있으면서
일직선으로 나아가는 하나의 선입니다.

```
void Aim ()
{
    if(!isAimming)
        return;
    Debug.DrawRay(transform.position, Vector3.forward*1000f, Color.red);
}
```

출발점 도착점 색상

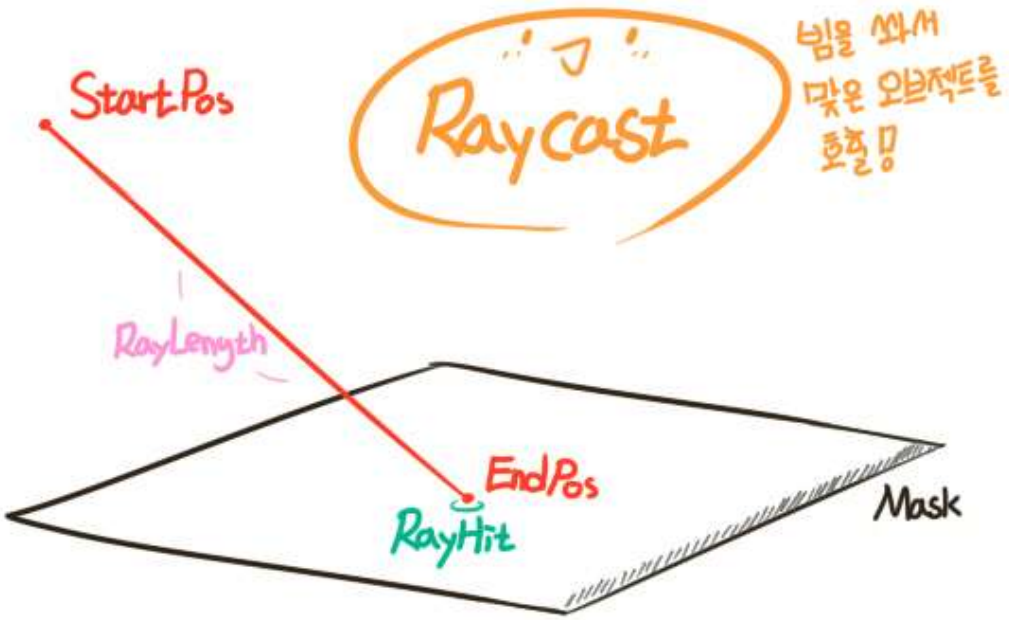
간단하게 **Debug.DrawRay()** 함수를 통하여
레이가 어떻게 쏘아지는지 확인할 수 있습니다.
(디버그 클래스이기 때문에 인 게임에서는 나타나지 않습니다.)



플레이어를 출발점으로, 플레이어 정면에서 멀리떨어진 지점을 도착점으로 정해놨더니
위 그림과 같이 레이가 쏘아지네요.

헌데 이걸 대체 어디에다 쓰는 것일까요?

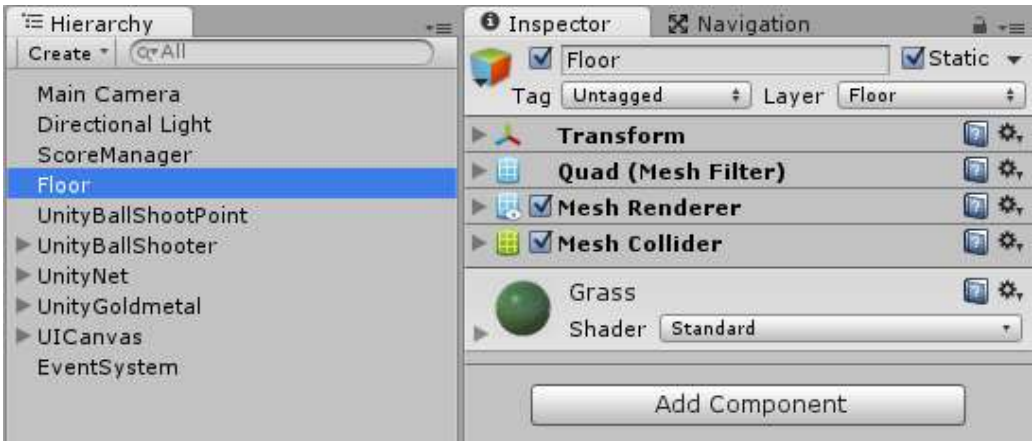
이제부터 심도있게 차근차근 설명해드리겠습니다.



레이를 쏘는 목적!
일직선의 빔을 쏘아 맞춘 오브젝트의 정보를 얻고자 할 때입니다.
그리고 이렇게 오브젝트를 검색하는 행동을 **Raycast** 라고 하죠.

이렇게 레이로 오브젝트를 맞추는 것, 이것을 **RaycastHit** 라고 합니다.
요 녀석은 맞춘 위치 정보와 더불어 오브젝트의 정보를 제공해준답니다.

검색 대상을 딱 지정하고프면 (캐릭터라던가, 지형이라던가)
Mask 를 걸들여주면 됩니다.



Mask가 되는 **Layer(레이어)**는
속성창 위쪽에 태그 옆에 있답니다.

태그와 마찬가지로 원하는 이름을 생성해서 지정할 수 있습니다.

```
void Aim ()
{
    if(!isAimming)
        return;

    Ray ray = Camera.main.ScreenPointToRay (Input.mousePosition);
    RaycastHit rayHit;
    float rayLength = 500f;
    int floorMask = LayerMask.GetMask ("Floor");

    if (Physics.Raycast(ray, out rayHit, rayLength, floorMask)) {

    }
}
```

이 것을 코드로 작성하면 이와 같습니다.

Camera.main.ScreenPointToRay(출발점, 시작점, 색상)
이 카메라 함수는 실제 게임이 아닌 게임을 보여주는 화면(스크린)의
좌표에다가 레이를 쏴주는 기능을 합니다.

특히 마우스 커서 위치에 레이를 쏘아서
마우스 커서에 닿은 오브젝트를 찾을 수 있게 되는 것이죠!

레이를 만들었다면 **Physics.Raycast()** 함수를 사용해서
레이에 맞은 오브젝트를 찾아냅니다.

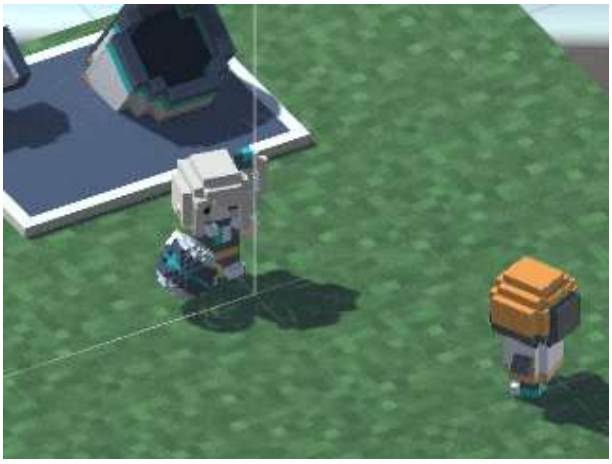
RaycastHit 변수를 초기화하지 않은 이유는
이 함수를 통해서 결과값이 변수로 저장되기 때문이지요.

여튼 레이에 맞은 오브젝트가 있다면
true 값을 반환해주게 됩니다.

```
if (Physics.Raycast(ray, out rayHit, rayLength, floorMask)) {
    Debug.DrawRay(playerEquipPoint.transform.position, rayHit.point, Color.red);

    Vector3 playerToMouse = rayHit.point - playerEquipPoint.transform.position;
    playerToMouse.y = 0f;

    Quaternion newRotation = Quaternion.LookRotation(playerToMouse);
    rigidbody.MoveRotation(newRotation);
}
```

오브젝트를 찾았으면 요리를 해야겠지요?
if문을 사용하여 계획했던 로직을 작성해줍니다.

필자는 바닥에 맞은 지점의 위치값으로
캐릭터를 회전시키겠습니다.



Debug.DrawRay()를 통해 프로젝트 상에서
레이가 어떻게 쏴지는지 확인해봅시다.

정확히 바닥 안에서 마우스 커서에 닿은 지점의 방향으로
플레이어가 회전하게 되는 모습이군요.

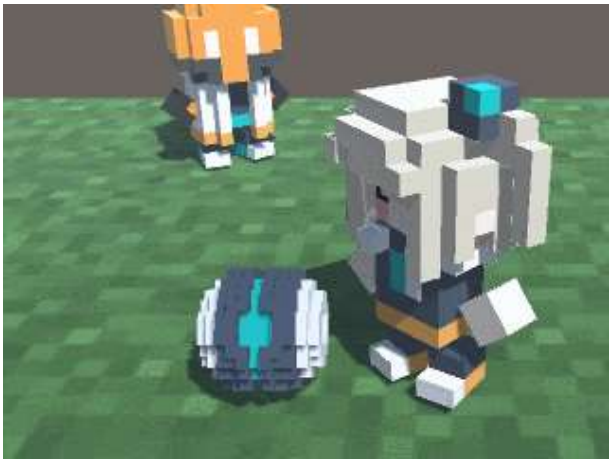
이제 원하는 방향으로
아이템을 던질 수 있게 해보겠습니다.

```
void Drop ()
{
    GameObject item = playerEquipPoint.GetComponentInChildren<Rigidbody> ().gameObject;
    Rigidbody rigidbody = item.GetComponent<Rigidbody> ();

    //Detach Item
    SetEquip(item, false);
    playerEquipPoint.transform.DetachChildren();

    //RaycastHit For Throwing Angle
    Ray ray = Camera.main.ScreenPointToRay (Input.mousePosition);
    RaycastHit rayHit;
    float rayLength = 500f;
    int floorMask = LayerMask.GetMask ("Floor");
    Vector3 throwAngle;

    //ThrowAngle : RaycastHit
    if (Physics.Raycast(ray, out rayHit, rayLength, floorMask)){
        throwAngle = rayHit.point - playerEquipPoint.transform.position;
    }
    //ThrowAngle : Not RaycastHit
    else {
        throwAngle = transform.forward * 50f;
    }

    //Throw Item
    throwAngle.y = 25f;
    rigidbody.AddForce (throwAngle * throwPower, ForceMode.Impulse);

    animator.SetTrigger("doThrow");
    isAimming = false;
    isPicking = false;
}
```

아이템 호출
종속 관계 해제
Ray 쏘기
RayHit 방향
정면 방향
던지기
애니메이션 플래그 정리

17편의 버리기 함수에서 Raycast 로직을 추가하여
마우스 커서 방향으로 아이템을 던지게 구현하였습니다.

마우스 커서가 바닥에서 벗어나면 Raycast 가 안되니까
이 때에는 그냥 앞으로 던지도록 예외 처리도 해둡시다.



16, 17, 18편의 조합으로
공을 던져 적을 맞추어 점수를 얻는 장면을 만들어냈습니다.

이제 정말 하나의 게임이 됐네요~ 멋집니다!

이상으로 열여덟번째 강좌를 마칩니다.
다음에는 이미지, 사운드를 추가하는 것에 대해 다뤄보겠습니다.
그럼.