4강 플레이어 점프 구현하기

4-1 물리력에 의한 점프

Update()문에 키보드 눌리는지 액션을 체크한다. 스페이스바 눌리면 위로 AddForce()

올라가는 속도와 내려오는 속도가 조금 다를 수 있다. 올라갈때는 순간힘을 받아 빠르게 올라가다가 lineardrag와 gravity에 의해 속력이 감소되는데 이때 마이너스로 속력값이 바뀔때는 똑같이 lineardrag와 gravity의 영향을 받지만 이때는 gravity는 마이너스 방향 그대로지만 lineardrag는 다시 반대로 플러스방향으로 작용하기 때문

자연스러워질때까지 중력이나 매스 혹은 중력스캐일 등을 조절하며 바꾼다.

4-2 점프-착지 애니메이션

아틀라스에서 점프관련 텍스쳐를 끌어와 애니메이션 클립을 따고 walking과 같은 방법으로 트랜지션과 파라미터를 추가하여 애니메이션 상태를 정의한다. 이때 워킹과 달리 점프는 반복적으로 이륙 착륙 모션이 공중에서 계속 반복되면 안되므로 애니메이션 클립에 루핑을 끈다.

그리고 물리점프코드에 setbool()을 추가하여 애니메이션 재생되게 한다.

4-3 레이캐스트

앞서 만든 애니메이션은 점프가 끝나고나서 상태변경의 조건이 정의가 안되었다. 이 조건을 검사하는게 레이캐스트

Debug.DrawRay()

//에디터 상에서만 ray를 그려주는 함수이다.

레이캐스트가 무엇인지 알아보기 좋은 함수이다.

//Landing Platform

Debug.DrawRay(rigid.position, Vector3.down, new Color(0, 1, 0));

//에디터 상에서만 ray를 그려주는 함수이다.

RaycastHit2D rayHit = Physics2D.Raycast(rigid.position, Vector3.down, 1);

//레이캐스트2d를 담을 변수를 선언하고 physics2d.raycast()함수를 이용하여 생성한다.

//Physics2D.Raycast(레이캐스트 시작점, 방향, 크기);

//레이캐스트함수를 이용해 쏜 정보가 rayhit에 담긴 형태이다.

if (rayHit.collider != null) //충돌된 콜라이더를 검사해서 null이 아니라면

{

Debug.Log(rayHit.collider.name);//충돌된 정보의 이름을 출력하는데

//raycast가 쏜 정보는 콜라이더 하나밖에 안담긴다. 플레이어 중심부에서

//레이캐스트를 쏘면 플레이어의 충돌체를 담아오기때문에 바닥감지가 안된다.

}

이때 물리효과의 적용 범위나 이런것들을 제어하는 게 레이어가 있다. 레이어를 설정하고 조건을 걸면 이 현상이 해결될 수 있다.

4-4 무한 점프 막기

점프키가 입력되었는지 검사하는 문장으로 넘어가서 지금 점프중인지 아닌지를 체크하여 점프가 아닐때만 애드포스하도록 바꾸면 그만이다.