3강 플레이어 이동 구현하기

3-1 물리력에 의한 이동

코드 PlayerMove.cs를 참고한다.

3-2 저항 수치 설정

콜라이더 컴포넌트의 속성 파일인 콜라이더 머티리얼을 생성하고 거기에 속성값을 적용하여 마찰을 조절할 수 있다.

friction: 마찰력

bounciness: 탄성력

리지드바디 2d 컴포넌트를 건드려본다.

위에서 2d 머티리얼의 마찰력을 0으로 만든 머티리얼을 생성한 뒤에 바닥타일에 적용한다.

그렇게 되면 플레이어를 움직일때 빙판길마냥 멈추질 못하고 계속 미끄러지게 되는데 이것을 공기저항으로 잡을 수도 있다.

플레이어의 리지드바디 2d 컴포넌트의 linear Drag, Angular Drag가 공기저항 옵션이라고 (이동감쇠 속성) 생각한다. linear는 선형 이동속도에 영향을 주고 Angular Drag는 회전속도에 영향을 준다.

3-3 애니메이션 순환

애니메이션 좌우 반전: 스프라이트의 플립을 이용해서 방향전환 한다.

if (Input.GetButtonDown("Horizontal"))

spriteRenderer.flipX = Input.GetAxisRaw("Horizontal") == -1;

//오른쪽 키를 누르면 +1이고 이것은 거짓이 되므로 flipX엔 거짓 할당되어 체크해제

//왼쪽 키를 누르면 -1이고 참이되어 flopX엔 참 할당되어 체크함. 따라서 플립이 일어남.

애니메이터 창으로 돌아와서 Idle에서 바꾸고 싶은 상태가 있으면 트랜지션을 연결해주고 거기에서 사용할 매개변수인 파라미터를 설정해야된다.

2d 이다 보니까 겹쳐서 부드럽게 전환되는 구간을 없앤다. 그리고 애니메이션 재생이 끝나면 빠져나오는 has exit time도 체크해제한다.

//객체의 속도를 체크해서 속도가 있을땐 워킹 애니메이션 재생

if (rigid.velocity.normalized.x == 0)

animator.SetBool("isWalking", false);

else

animator.SetBool("isWalking", true);

Mathf.Abs() : 절대값을 만들어주는 유니티의 함수.