Day 17. 캐릭터 움직임 수정

17-1. 캐릭터 조작 변경

회피스킬의 조작과 여러 가지 스킬들의 조작을 더 쉽게 하기위해 이리저리 시도해보던 도중에 기존 캐릭터의 움직임에 대한 수정이 필요함을 깨달았습니다. 기존 캐릭터의 조작에 대한 방식은 어떠한 상하좌우, 대각 키를 누르든 카메라가 바라보는 방향으로 캐릭터가 항상 앞을 향하고 있는 방식으로 이동하는 것이었고 이러한 조작 방식은 외국에서는 8ways movement라고 불렸습니다.

하지만 이러한 조작방식은 주로 FPS게임에서 사용하거나 어떠한 화면을 조준하고있는 상황, 혹은 막기를 하고있는 상황등에서 주로 사용되었고 액션게임에서는 그다지 어울리지 않았습니다. 또한 8방향의 움직임을 자연스럽게 구현하기 위해서는 BlendSpace에서 움직임을 묶기 위해 8방향에 대한 애니메이션이 필요했는데 여기에 대한 소스도 없었습니다. 그래서 조작을 바꾸기 위해 여러 레퍼런스들을 찾아보고 또 직접 세키로, 원신등과 같은 게임을 설치해 어느 정도 캐릭터의 움직임을 조작해본 결과 대부분의 액션게임에서는 1. 카메라는 특정하게 고정되지 않은 채로 조작이 가능하고 2. 누른 방향키의 방향으로 캐릭터가 회전해서 앞으로가는 방식을 채택하고 있음을 알게 되었습니다.

다만 알게 된 것 까지는 좋았는데 해당 움직임을 구현하는데 어려움이 발생했습니다. 문제는 내가 생각보다 Roll,Pitch,Yaw의 개념에 대해서 잘못 알고 있었다는 것이었습니다. 기존에는 단순히 x,y,z가 각각 Roll,Pitch,Yaw에 대응한다고 알고 있었는데 이것은 매우 잘못된 지식이었습니다. 잘못된 지식으로 회전값을 넣다보니 캐릭터가 무작위로 움직였고 제대로된 캐릭터의 조작을 구현할 수 없었습니다. 그래서 Roll,Pitch,Yaw에 대해 다시 공부했습니다. 전체적인 내용은 길어서 다 말할 수는 없겠지만 어쨌든 핵심은 x,y,z가 단순하게 대응하는 개념이 아니라 회전값이 대응한다는 개념이며, 언리얼은 왼손 좌표계를 사용하기 때문에 +z축이 화면 안쪽을 향하고 각 축에 대해 시계방향이라는 것이었습니다.

우여곡절 끝에 공부한 것을 토대로 다시 코드를 만들었습니다. 코드의 내용은 왼손 좌표계에 따르면 z의 회전값이 캐릭터의 좌우 회전 값을 뜻하기 때문에 Yaw를 주어야하고 따라서 GetController에서 값을 받아 Yaw값만 추출하고 FRotationMatrix에 Yaw값을 넣어 회전 매트릭스를 생성했습니다. 위아래 움직임은 x축의 단위 벡터로 좌우 움직임은 y축의 단위벡터로 addMovementInput에 값을 넣어 움직이는 방식으로 구현했습니다.