CHAPTER 6

블루프린트 활용 III: 텔레포트 포탈과 더블 점프

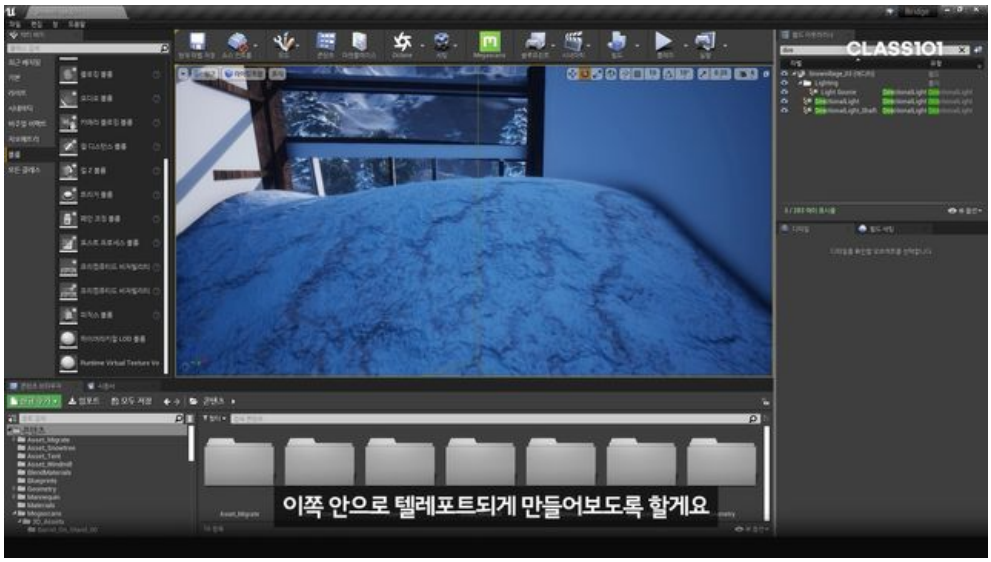
추가 팁: Play Sound 2D / Play Sound at Location 노드를 사용하시면 효과음도 넣어주실 수 있습니다! 수업 마지막 부분에서 추가해주시면 됩니다.

[수업 목표]

여러분 반갑습니다! 러셀입니다.

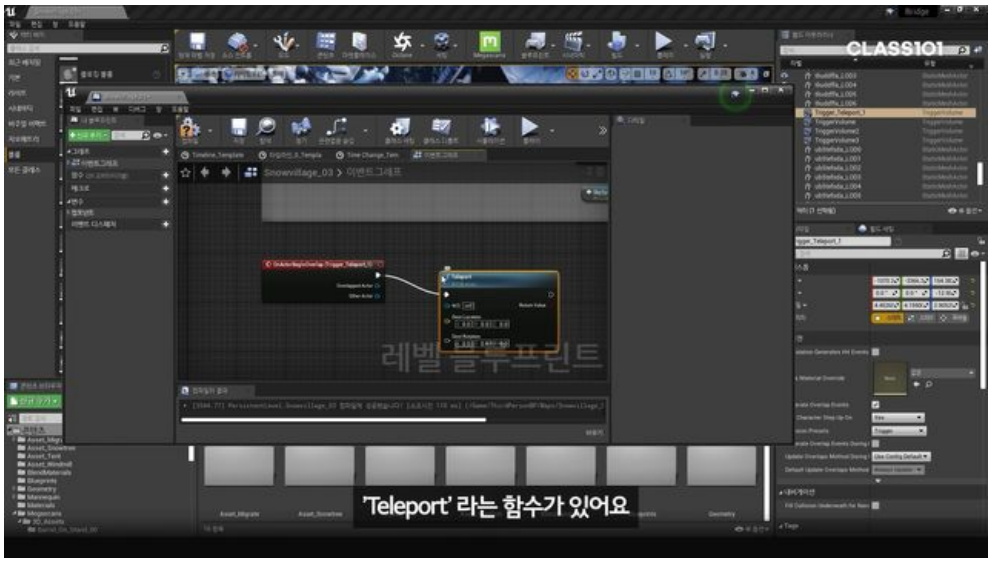
이번 시간에는 특정 구역에 들어가면 캐릭터가 다른 곳으로 이동되는 텔레포트 포탈과, 캐릭터 블루프린트를 수정해 공중에서 한 번 더 점프하는 더블 점프 기능을 배워보고 응용하는 방법까지 알아보도록 하겠습니다.

[수업 개요]



첫 번째로 만들 기능은 텔레포트 포탈로, 텐트 안으로 들어가면 헛간으로 이동되는 기능을 만들어보겠습니다.

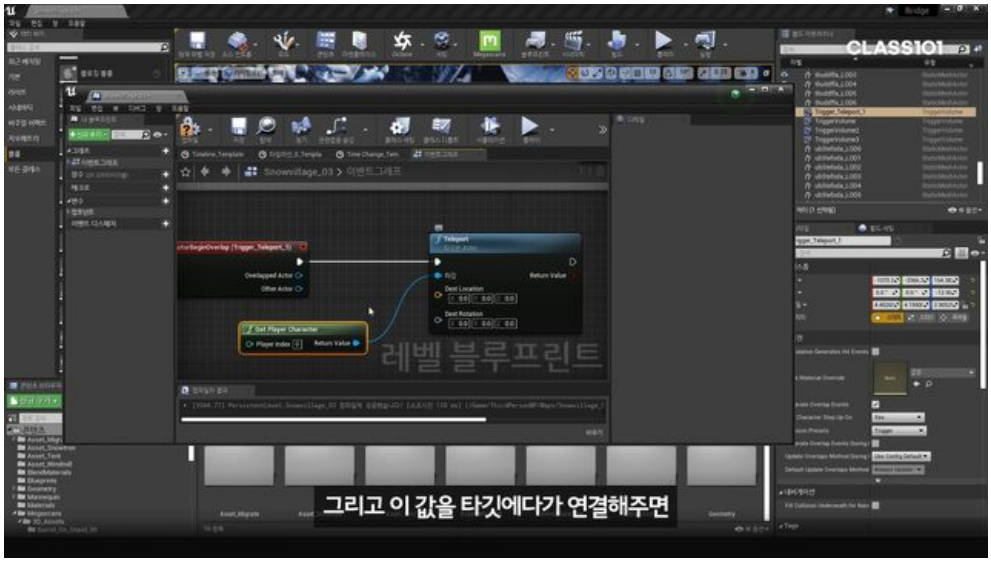
전 시간과 마찬가지로 특정 영역에 반응하는 것이기에 트리거 볼륨으로부터 시작합니다. 지금 환경에 설치된 트리거 볼륨이 많을텐데요, 월드 아웃라이너에서 트리거 볼륨의 이름을 식별하기 좋게 바꿔주셔도 됩니다.



1:39 텔레포트 블루프린트 작업 시작

트리거 볼륨으로부터 OnActorBeginOverlap을 꺼낸 뒤, 핀을 이어 Teleport라는 함수를 찾아 꺼내줍니다.

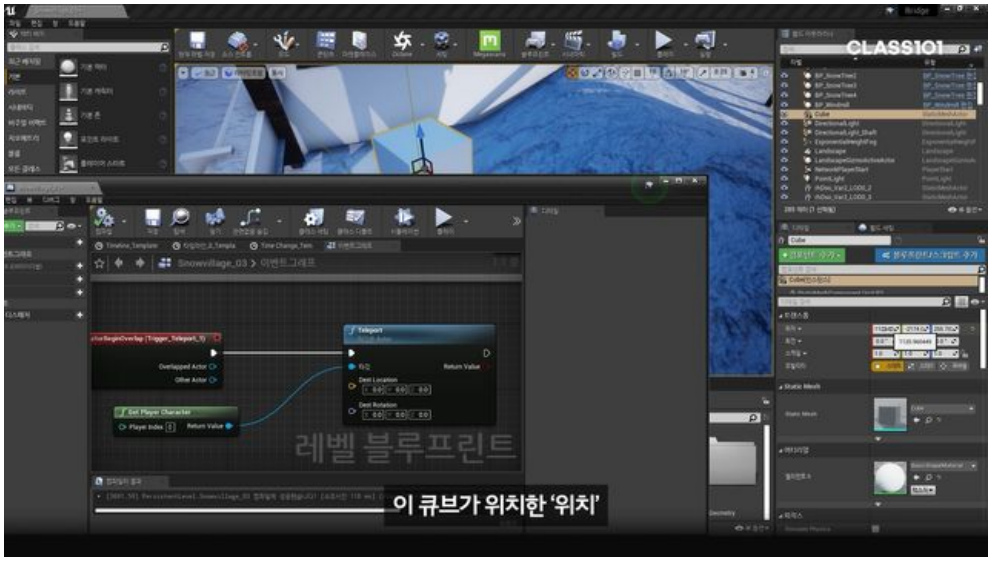
복잡한 과정을 거쳐야 텔레포트를 할 수 있는 것이 아니고, 간단히 이 함수만으로도 텔레포트를 사용할 수 있게 되어있습니다. 아주 편리하죠.



텔레포트 노드에 타깃이 기본으로 self로 되어있습니다.

하지만 여기는 레벨 블루프린트이죠. 레벨 블루프린트에서 self라고 하면 레벨을 지칭하는 것입니다.

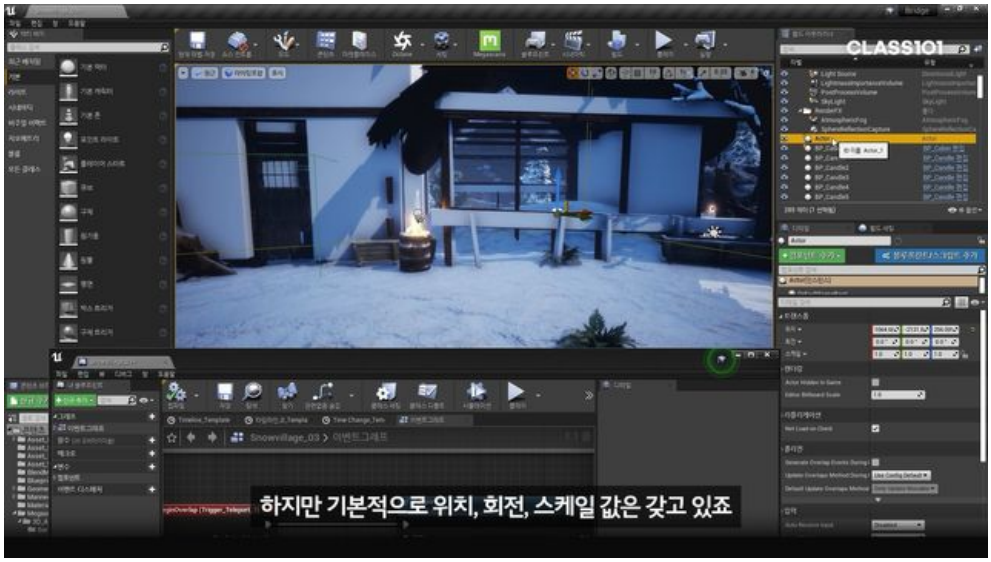
그러나 레벨을 텔레포트 시키는 것은 말이 안 되겠죠? 우리는 레벨이 아닌 캐릭터를 텔레포트 시킬 것이기 때문에 **Get Player Character** 라는 노드를 꺼내어 타깃에 연결해주도록 합시다.



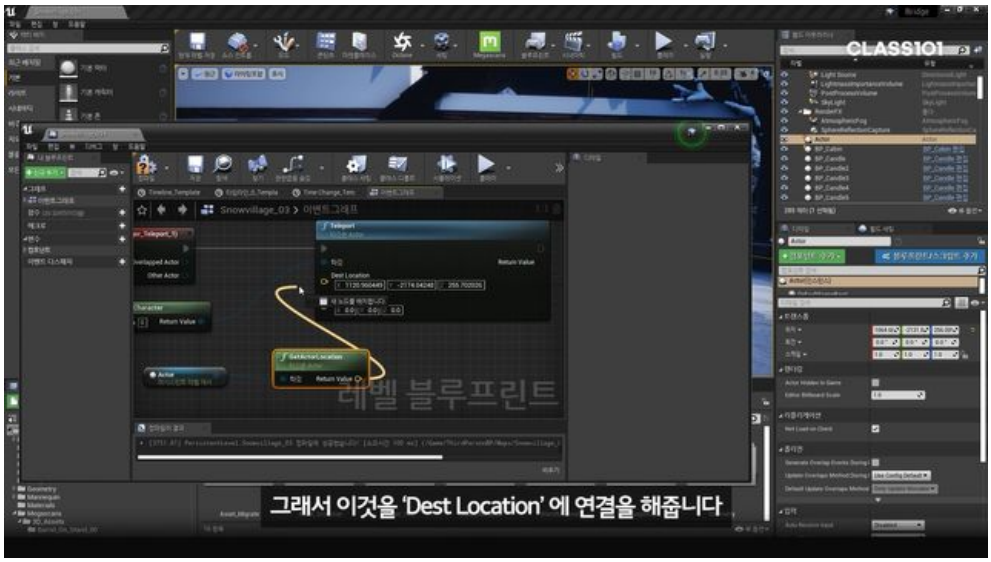
그리고 아래의 Dest Location은 Destination의 약자로 타깃이 텔레포트될 도착 위치를 뜻합니다. 그래서 이곳에 헛간 내부의 위치를 입력해주어야 하는데요,

헛간 내부의 위치를 알아내는 방법은 간단합니다. 액터 배치 패널에서 큐브를 꺼내와 헛간 내부에 임시로 배치한 뒤 큐브의 위치값을 그대로 텔레포트 노드에 입력해주면 됩니다.

하지만 이렇게 할 경우, 도착 위치를 바꾸고 싶을 때 다시 큐브를 배치하고 위치값을 직접 입력해주어야 합니다. 더군다나 이 값은 상수로 입력했기에 플레이 도중 바꿀 수도 없죠.



이럴 때 변수를 사용하면 효율적입니다. 액터 배치 패널에서 기본 액터라고 되어있는 것들 끌어서 헛간 내부로 가져갑니다. 이 때 G키가 눌려있다면 아무것도 보이지 않으니 G키를 비활성화 해주세요.



방금 설치한 기본 액터를 드래그하여 레벨 블루프린트로 오브젝트 변수 형태로 가져간 다음, GetActorLocation에 연결하여 기본 액터의 위치를 구하고 그 값을 Teleport의 Dest Location에 연결해줍니다.

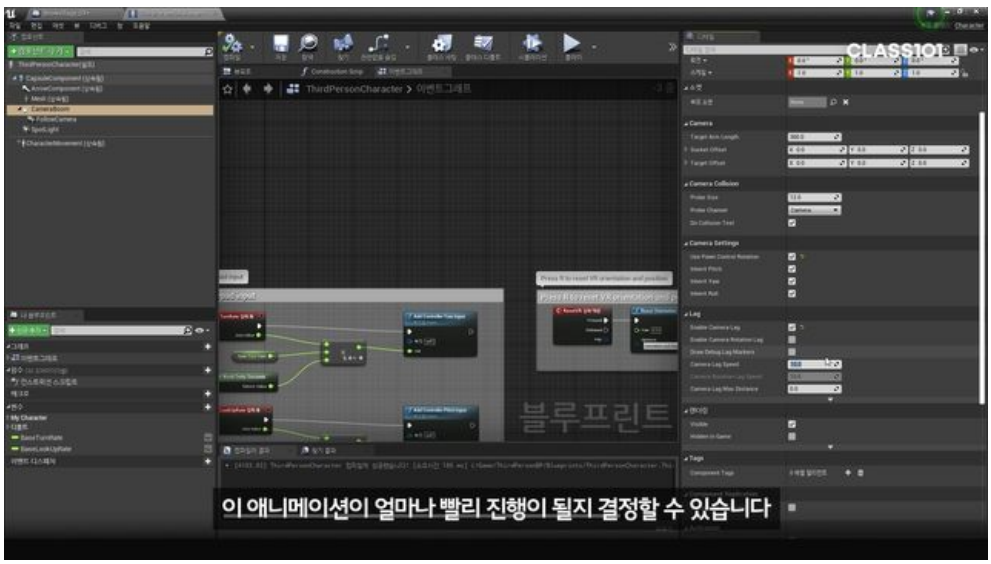
이렇게 하면 간단히 레벨에서 기본 액터의 위치를 옮겨주는 것으로 텔레포트될 위치를 변경할 수 있게 됩니다. 관리에 많은 도움이 되죠.

7:25 카메라 Lag을 활용한 부드러운 이동

위 과정까지 마무리했다면 텔레포트 기능이 정상적으로 작동할 것 입니다. 그러나 이동될 때 뚝 하며 바로 이동되어버려서 부자연스러운 감이 있죠.

캐릭터를 눌러 블루프린트 편집을 눌러주면 캐릭터 블루프린트가 나타납니다. 여기의 컴포넌트들 중 CameraBoom을 찾고, CameraBoom의 디테일 패널에서 Lag를 찾아봅시다.

그리고 Enable Camera Lag를 활성화해줍니다.



Camera Lag은 캐릭터가 이동했을 때, 따라가는 카메라의 딜레이를 조절하는 부분입니다.

Camera Lag Speed을 조정하여 카메라가 따라가는 속도를 바꿀 수 있습니다.

Camera Lag은 이렇게 텔레포트 될 때 뿐 아니라 평상시에도 적용되기 때문에 Camera Lag Speed가 너무 느리면 어지러움을 유발할 수 있습니다.

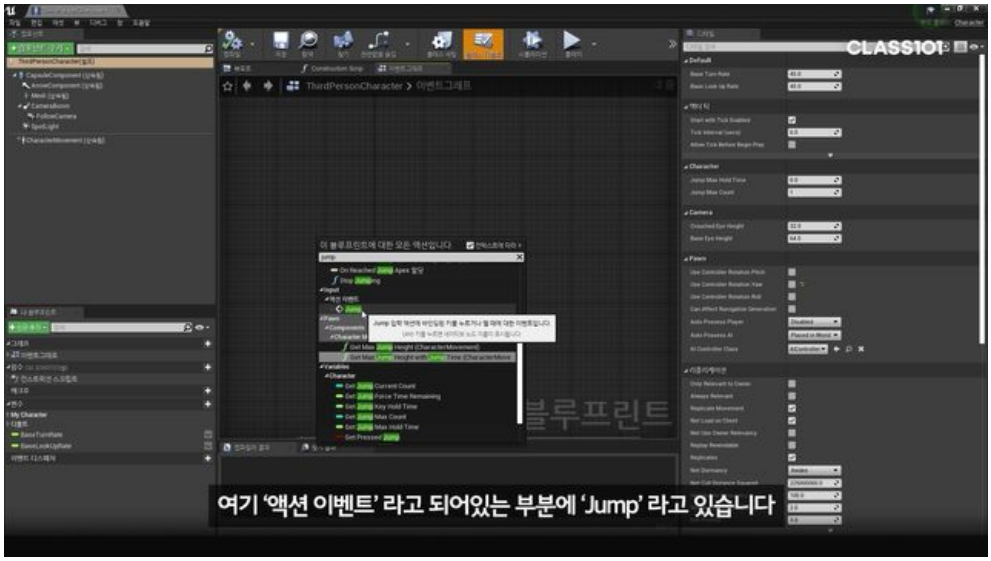
그래서 적당히 15~25 정도로 세팅하여 평상시엔 거의 없게 세팅해주고, 텔레포트같이 갑자기 이동될 때 자연스럽게 이동될 수 있도록 해주시면 됩니다.

(텔레포트 노드에서 캐릭터 블루프린트를 가져와 텔레포트 되는 순간에만 Camera Lag Speed를 5정도로 바꾸는 방법도 있겠네요! 한 번 시도해보셔도 재밌을 듯 합니다.)

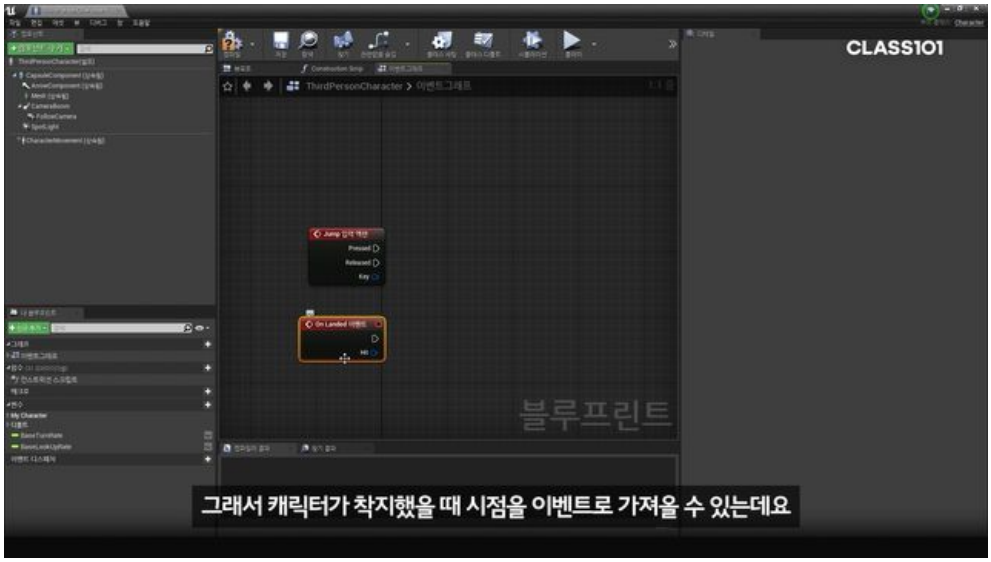
트리거 볼륨은 플레이 도중 시각적으로 보이지 않기 때문에, 전 시간의 밤 낮 경계와 마찬가지로 플레이어에게 '이 곳에 들어가면 무슨 일이 벌어진다'는 것을 시각적으로 알려줄 수 있는 액터를 배치해주시면 좋습니다. 그럴 때 저는 파티클 애셋을 자주 사용하는 편 입니다.

11:30 캐릭터 블루프린트에서 더블 점프 작업 시작

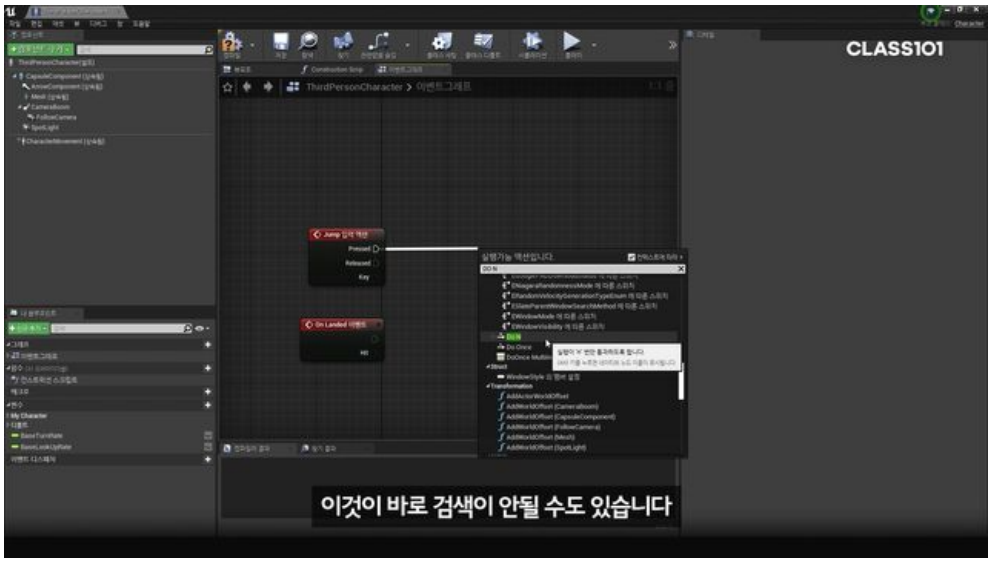
캐릭터 블루프린트는 조금 전처럼 캐릭터를 누른 후 블루프린트 편집에서 열 수 있습니다.



다른 노드들이 많은데, 빈 공간에 자리를 잡아주시고 우클릭해서 Jump라고 검색해줍니다. 그러면 이벤트 Jump가 있는데 그것을 꺼내주세요. 이것은 Jump 키를 눌렀을 때 발생하는 이벤트입니다.



그리고 땅에 다시 닿았을 때의 시점이 필요합니다. 그것은 On Landed라는 이벤트로 할 수 있습니다. 마찬가지로 검색해서 On Landed 이벤트를 꺼내주세요.



이후 Jump에 연결될 노드로 **Do N** 이라는 노드를 꺼내줍니다.

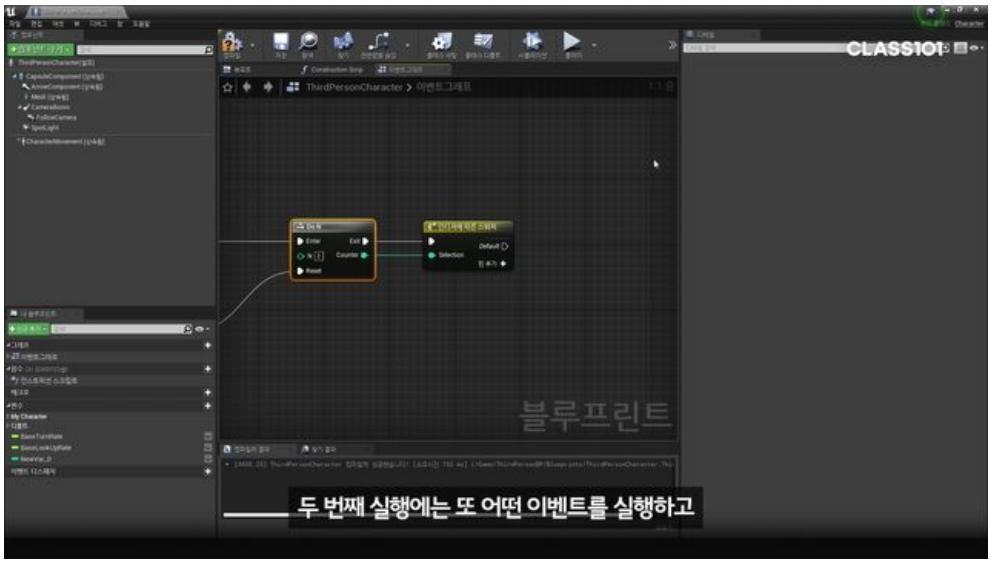
이 노드가 Do N이라고 검색했을 때 바로 나타나지 않을 수도 있습니다. 스크롤을 내려 찾아보면 저렇게 생긴 노드가 있어요. 같은 것을 잘 찾아주세요!

Do N 노드는 실행이 N번만 통과되게 하는 노드입니다.

Jump에 이 노드를 연결했다면, Jump키를 여러 번 눌러도 점프는 N번만 실행됩니다. 하지만 On Landed, 즉 지면에 다시 닿았을 때 Do N을 초기화합니다.

고로 공중에서 점프를 몇 번까지 허용할 것인지를 결정하는 것이 바로 Do N입니다.

더블 점프를 할 것이기 때문에 N에 2를 입력해줍니다.



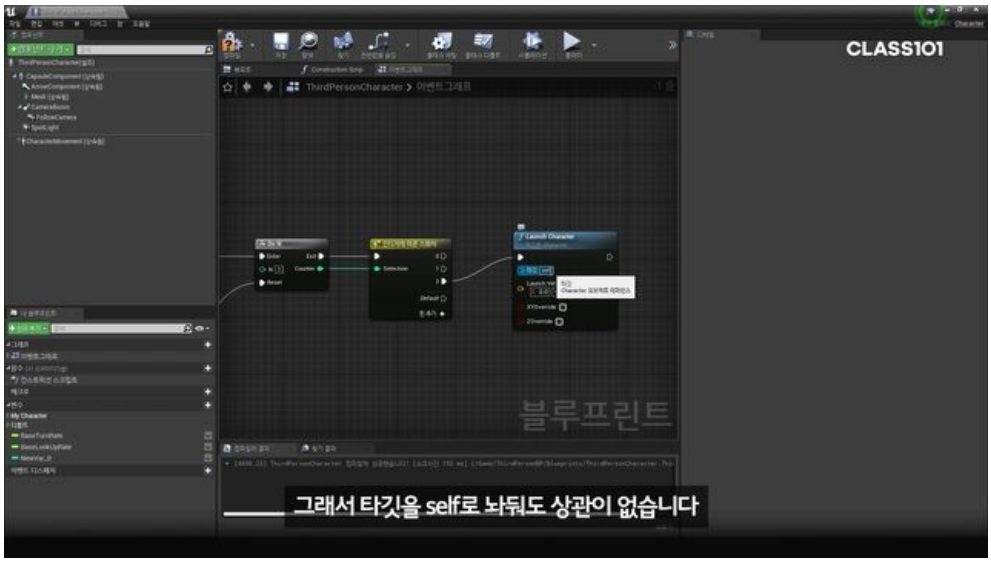
그 다음으로 Do N에 연결해줄 노드는 **인티저에 따른 스위치** 라는 노드입니다.

이 노드는 Do N에서 받아온 각각 N번째 실행에 어떤 함수를 실행시킬지 결정하는 노드입니다.

예를 들어, 점프를 1번째 했을 땐 어떤 액션을, 2번째 했을 땐 어떤 액션을 시킬지 갈림길을 만들어주는 역할을 하죠.

Do N의 counter를 인티저에 따른 스위치의 Selection에 연결해줍니다.

그리고 인티저에 따른 스위치의 핀 추가를 눌러 0, 1, 2번째를 만들어주세요.

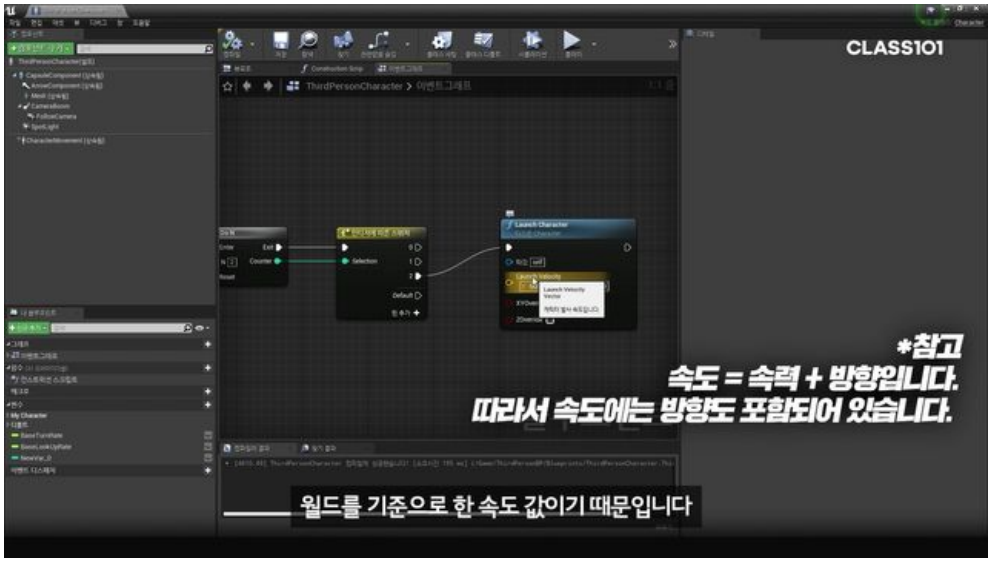


다음으로 연결해줄 노드는 **Launch Character** 입니다.

Launch Character는 캐릭터를 특정 값만큼 발사하는 함수입니다.

이것을 인티저에 따른 스위치의 2번에서 연결해줍니다. 왜냐하면 0번째와 1번째는 그냥 점프여야 하기 때문이죠.

Launch Velocity에 모두 600을 입력하고 실행해봅시다.



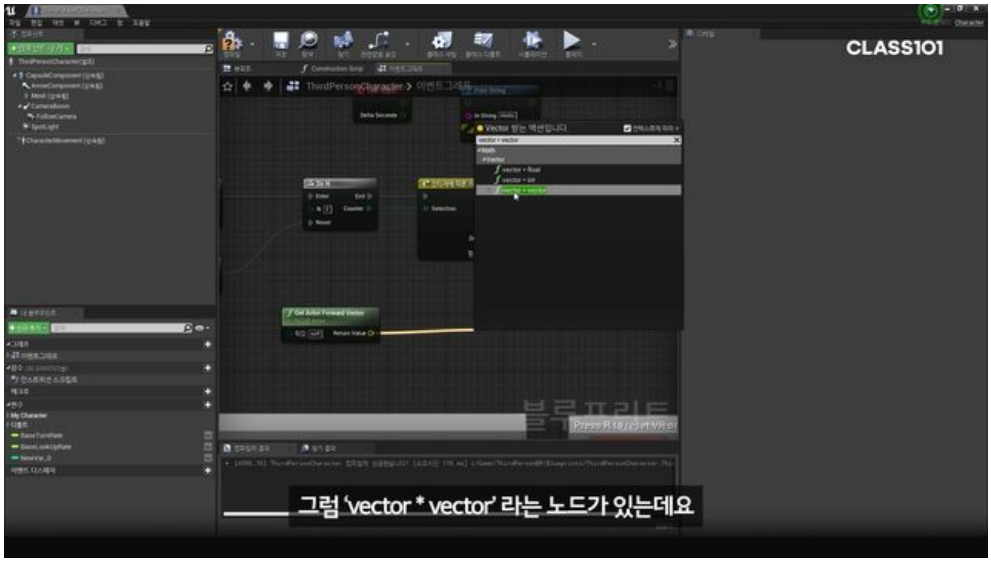
16:40 더블 점프의 방향 오류 해결

그런데 점프를 해보니 뭔가 이상합니다. 캐릭터가 보고 있는 방향에 따라 점프되는 강도가 다른데, 그것은 Launch Velocity가 월드 기준의 값으로 캐릭터를 발사시키기 때문입니다.

곧, 어떤 방향을 보고 있어도 월드 기준으로 +600 시켜버리는 것이죠. 그렇기에 반대 방향을 보고 있다면 이상하게 점프하는 것입니다.

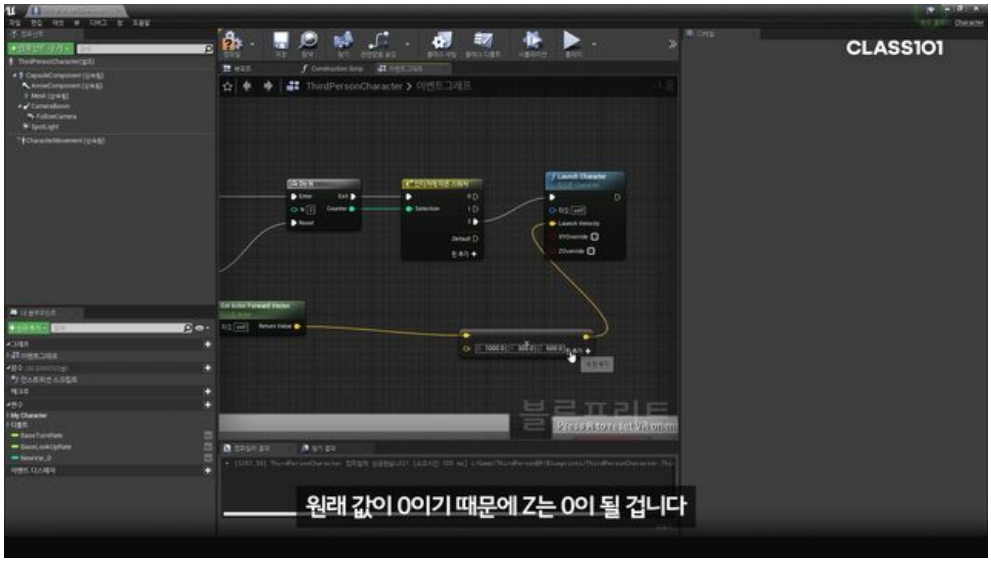
이럴 땐 캐릭터의 현재 방향을 구해서 Launch Velocity에 곱하는 식으로 해결할 수 있습니다.

Get Actor Forward Vector는 현재 캐릭터가 보고 있는 방향을 구합니다. 마치 나침반같은 역할을 하지요.



GetActorForwardVector를 Launch Velocity에 곱하기 위해 vector * vector라고 검색해서 벡터 형태의 변수끼리 곱할 수 있는 노드를 꺼냅니다.

그리고 이 값에 600, 600, 600을 넣어준 뒤 Launch Character의 Launch Velocity에 연결합니다.



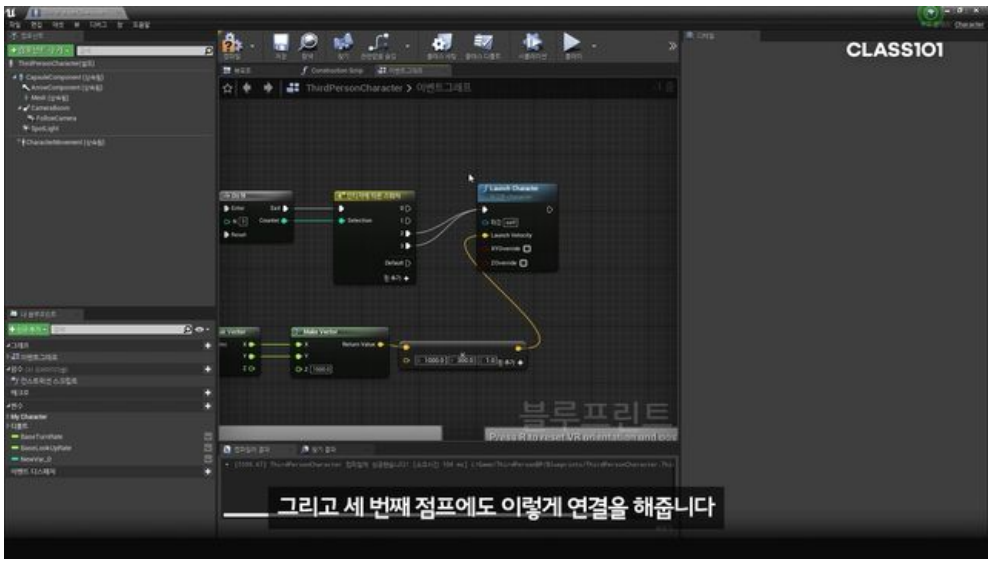
그리고 실행해보면 잘 되지만 Z값에 이슈가 있습니다.

GetActorForwardVector는 Z값을 항상 0으로 반환하는 특징이 있습니다. 때문에 여기에 600을 곱해도 Z 값은 항상 0으로 고정됩니다.

이를 해결하기 위해 GetActorForwardVector의 Z값을 버리고, Z값에만 우리가 직접 원하는 값으로 설정해 주려 합니다.

그래서 GetActorForwardVector에서 출력하는 Vector 변수를 각각 X, Y, Z의 Float값으로 분해한 뒤, 다시 Make Vector로 연결할 때 Z값은 연결하지 않고 직접 600을 입력합니다.

그렇게 만들어진 새 Vector값을 기존 곱하기 노드에 연결하면, 이제 Z는 항상 600으로 고정되고 X, Y값은 현재 캐릭터가 보고 있는 방향에 따라 유동적으로 변화합니다.



24:45 N번 연속 점프로의 응용

Do N의 N값과 인티저에 따른 스위치의 핀 추가를 한 후, 그것을 연결해주는 식으로 3번 연속, 4번 연속 점프 등으로 응용할 수도 있습니다.

마지막으로 공중 연속 점프를 실행할 때 이펙트를 출력하는 방법에 대해 알아봅니다. Launch Character 다음에 **Spawn Emitter at Location** 이라는 노드를 연결합니다.

(참고로, Spawn Emitter Attached는 이펙트가 캐릭터에 붙어있게 됩니다.)

Emitter Template에서 원하는 파티클 애셋을 선택해주세요.

그리고 Location에 GetActorLocation으로 현재 캐릭터의 위치를 구해 연결합니다.

더블 점프 기능이 완성되었습니다!

(추가 팁: Play Sound 2D / Play Sound at Location 노드를 사용하시면 효과음도 넣어주실 수 있습니다!)

[다음 수업 예고]

다음 시간엔 마우스로 버튼을 눌러 이벤트를 실행시키는 UI 제작 방법에 대해 배워보겠습니다.

감사합니다! 러셀이었습니다 :)