28_ UMG

<제목 차례>

인천대학교 컴퓨터공학부 박종승

1. 개요

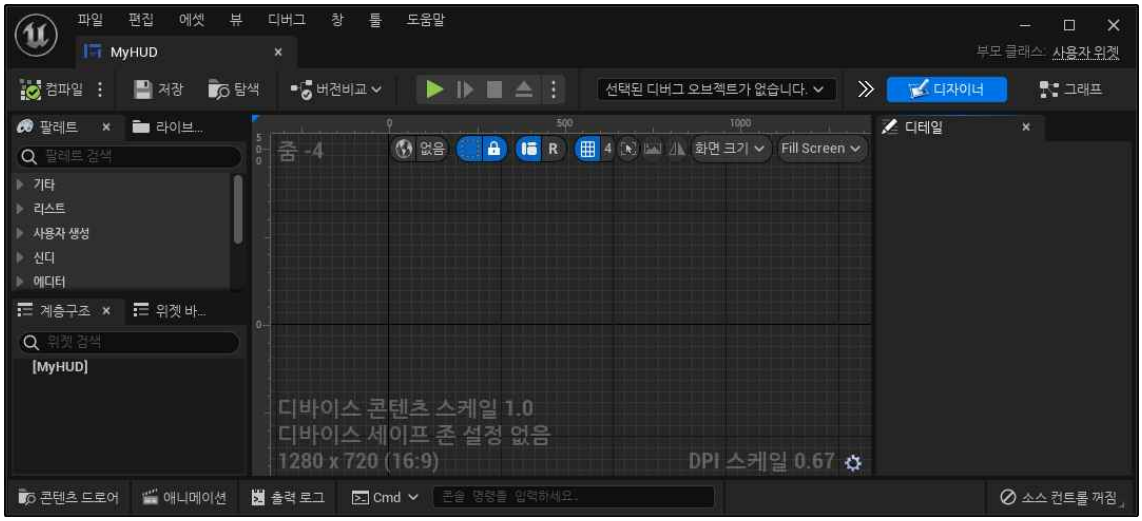
이 장에서는 UMG에 대해서 다룬다.

UMG(Unreal Motion Graphics UI Designer)는 시각적 UI 저작 도구이다. UMG를 사용하면 게임 내 HUD, 메뉴 등의 UI 요소를 쉽게 제작할 수 있다. UMG의 핵심은 위젯(Widget)이다. 위젯은 UI를 제작하는데 사용되는 버튼, 체크박스, 슬라이더, 진행바 등의 미리 제작된 부품들의 모음으로 구성된다.

<참고> UMG의 전체 문서에 대해서는 다음의 링크를 참조하자.
https://docs.unrealengine.com/umg-ui-designer-for-unreal-engine/

이 절에서 처음으로 UMG 디자인에 대해서 다루기 시작한다. 예제를 시작하기 전에 UMG 디자인에 대해서 기본 개념을 학습해보자.
UMG 디자인의 핵심은 위젯이다. 엔진이 제공하는 위젯을 적절히 사용하여 UI를 구성하는 것이다. 하나의 위젯 블루프린트는 하나의 UI 레이아웃을 작성한다. 하나의 UI 레이아웃이 표현하는 대상은 하나의 메뉴 화면일 수도 있고 HUD 화면일 수도 있고 화면의 일부 영역의 작은 표시일 수도 있다. 자신이 만들기 원하는 UI 레이아웃을 위해서 필요한 만큼의 위젯을 위젯 블루프린트에 배치하여 UI 화면을 구성하면 된다. 커스텀 위젯을 직접 생성하여 사용해도 되지만 대부분의 경우는 엔진 제공 위젯만으로도 충분하다.
위젯 작업은 UI 화면에서 위젯의 시각적인 모양과 배치를 다루는 부분과 위젯의 기능성에 대해서 스크립트를 작성하는 부분으로 이루어진다. 에디터에서는 시각적인 부분과 스크립트 부분에 대해서 별도로 구분된 편집 모드인 **디자이너** 모드와 **그래프** 모드를 두고 있다. **디자이너**(Designer) 모드에서는 마우스를 사용해 위젯을 격자탭에 끌어다 놓고 크기와 위치를 조절하고 각 위젯의 디테일 탭에서의 적절한 속성값을 지정하면 된다. **그래프**(Graph) 모드에서는 블루프린트 그래프를 작성하면 된다.

먼저 위젯 블루프린트 에디터에 대해서 알아보자.
위젯 블루프린트 애셋을 생성한 후에 에셋을 더블클릭하면 **위젯 블루프린트 에디터**가 열린다. UMG UI 구현의 모든 것은 이 에디터 내에서 이루어진다.
위젯 블루프린트 에디터를 열면 다음과 같은 모습의 창이 뜬다.



위젯 블루프린트 에디터의 레이아웃을 살펴보자.
상단에는 일반적인 메뉴바와 툴바가 있다.
상단 오른쪽에는 에디터 모드를 전환하는 버튼이 있다. 에디터 모드에는 **디자이너** 모드와 **그래프** 모드가 있다. 모드에 따라서 레이아웃이 달라진다. 먼저, **디자이너** 모드에서의 레이아웃을 살펴보자.
왼쪽에는 **팔레트** 탭이 있다. 이 팔레트 탭에는 사용 가능한 위젯들이 목록으로 표시된다. **디자이너** 모드에서 위젯을 드래그해서 격자탭에 배치하면 된다.
왼쪽 팔레트 탭 바로 아래에 **계층구조** 탭이 있다. 이것은 현재의 작성중인 위젯 블루프린트의 계층구조를 보여준다. **팔레트** 탭에서 배치할 위젯을 끌어다가 이 계층구조 탭에 드롭해도 된다.
중간의 격자맵이 **디자이너** 탭이다. 작성하고 있는 위젯 블루프린트의 시각적인 레이아웃을 표시해준다.
오른쪽에는 현재 선택된 위젯의 속성을 표시하는 **디테일** 탭이 있다.
하단바에는 **콘텐츠 드로어** 버튼과 **애니메이션** 버튼이 있다.
애니메이션 버튼을 클릭하면 위젯의 애니메이션 키프레임 작업을 할 수 있는 **애니메이션 트랙** 탭과 **타임라인** 탭이 나타난다.
한편, 상단 우측의 **그래프** 모드 버튼을 클릭하여 에디터 모드를 **그래프** 모드로 전환하면 화면의 레이아웃이 전체적으로 바뀐다. 레이아웃의 모양은 블루프린트 에디터와 거의 동일하다. 그래프 모드에서는 이벤트 그래프 탭에서 블루프린트 그래프를 작성하면 된다.

위젯 블루프린트를 작성하는 과정은 여러 위젯들을 추가하면서 시각적으로 편집하고 스크립팅하는 과정이다. 위젯 블루프린트에 추가되는 위젯은 하나의 계층구조를 이루도록 관리된다. 계층구조의 최상단에는 하나의 위젯이 배치된다. 그리고 위젯 아래에 다른 위젯들이 자식으로 추가된다. 위젯이 제공하는 기능은 위젯의 유형마다 다르다. 따라서 위젯의 사용에 앞서 위젯의 유형에 대해서 알고 있어야 한다. 엔진이 제공하는 위젯들은 위젯 블루프린트 에디터의 왼쪽 팔레트 탭에서 카테고리별로 분류되어 나열되어 있다.

<참고> 위젯 유형에 대해서는 다음의 링크를 참조하자.
https://docs.unrealengine.com/widget-type-reference-for-umg-ui-designer-in-unreal-engine/

위젯의 유형에 대해서 알아보자.
가장 최상위 위젯으로는 다른 위젯을 포함할 수 있는 위젯이 배치되어야 한다. 다른 위젯을 담을

수 있는 컨테이너 위젯을 패널 유형 위젯이라고 한다. 패널 위젯은 다른 위젯을 포함하면서 배치를 위한 기능을 제공한다. 패널 위젯에는 내부의 위젯을 임의의 위치에 배치할 수 있는 **Canvas Panel**이 있고, 격자 형태로 배치할 수 있는 **Grid Panel**이 있고, 수평 방향으로 배치할 수 있는 **Horizontal Box**가 있고, 수직 방향으로 배치할 수 있는 **Vertical Box**가 있다. 그 외에도 여러 형태의 **Panel**이 있다. 위젯 블루프린트를 생성하면 디폴트로 **Canvas Panel** 위젯을 최상단에 배치해준다. **Canvas Panel**에서는 내부에 위젯을 임의의 위치에 배치할 수 있고, 앵커를 설정할 수 있고, 캔버스의 다른 자손과 Z 순서를 조정할 수 있다. 따라서 내부에 임의의 원하는 위치에 배치하고 싶은 경우에는 **Canvas Panel**을 사용하면 된다. 그러나 좌우로 정렬 등의 배치 규칙에 따라 배치하고 싶은 경우에는 **Horizontal Box**와 같은 다른 패널을 사용하는 것이 좋다.

패널 위젯 내에서 자식 위젯이 차지하는 공간을 슬롯이라고 한다. 위젯을 배치할 때에 어떤 유형의 패널 내의 슬롯에 배치하는지에 따라서 배치 속성이 달라진다. **Canvas Panel**의 슬롯에 있는 위젯은 임의의 위치에 배치할 수 있으므로 앵커나 위치나 크기 등의 속성을 가지지만 **Horizontal Box**와 같은 패널의 슬롯에 있는 위젯은 정렬 방식 등의 다른 속성을 가진다.

2. UMG 프로젝트 준비

이 절에서 UMG 학습을 위한 프로젝트를 준비하는 과정을 학습한다.
이 절부터 시작하는 예제는 엔진의 튜토리얼 예제를 참조하였다.

<참고> UMG에 대한 퀵스타트 예제는 다음의 문서를 참조하였다.
https://docs.unrealengine.com/4.27/InteractiveExperiences/UMG/QuickStart/

이제부터 예제를 통해서 학습해보자

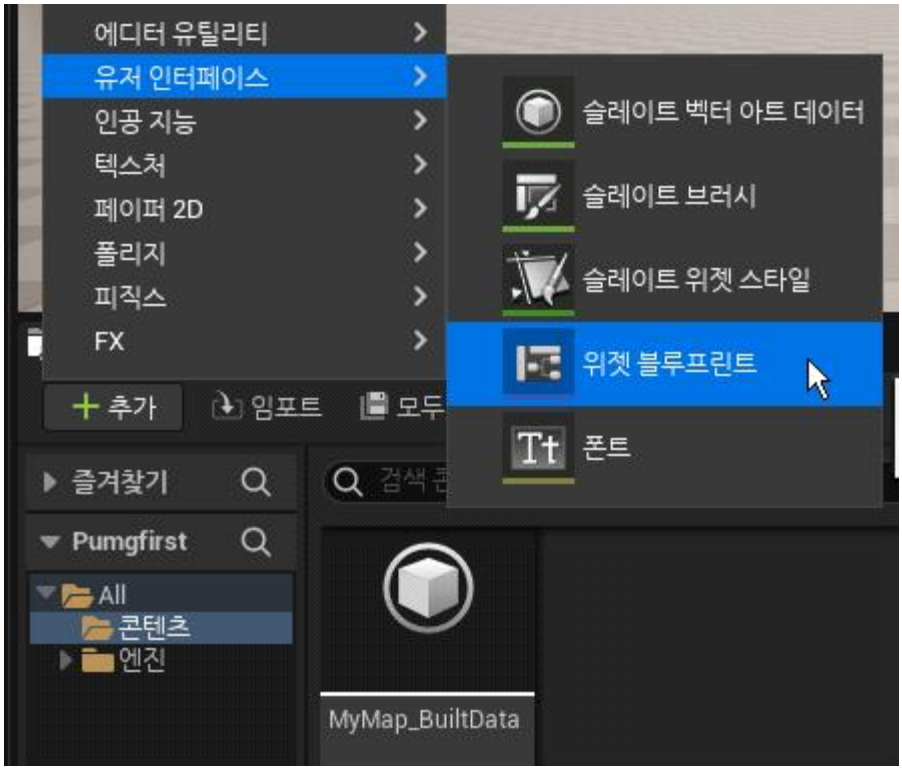
---

1. 새 프로젝트 **Pumgfirst**를 생성하자. 먼저, 언리얼 엔진을 실행하고 **언리얼 프로젝트 브라우저**에서 왼쪽의 **게임** 탭을 클릭하자. 오른쪽의 템플릿 목록에서 **기본** 템플릿을 선택하자. **프로젝트 이름**은 Pumgfirst로 입력하고 **생성** 버튼을 클릭하자. 프로젝트가 생성되고 언리얼 에디터 창이 뜰 것이다.
창이 뜨면, 메뉴바에서 **파일** » **새 레벨**을 선택하고 **Basic**을 선택하여 기본 템플릿 레벨을 생성하자.
그리고, 레벨 에디터 툴바의 저장 버튼을 클릭하여 현재의 레벨을 MyMap으로 저장하자.
그리고, **프로젝트 세팅** 창에서 **맵&모드** 탭에서의 **에디터 시작 맵** 속성값을 MyMap으로 수정하자.

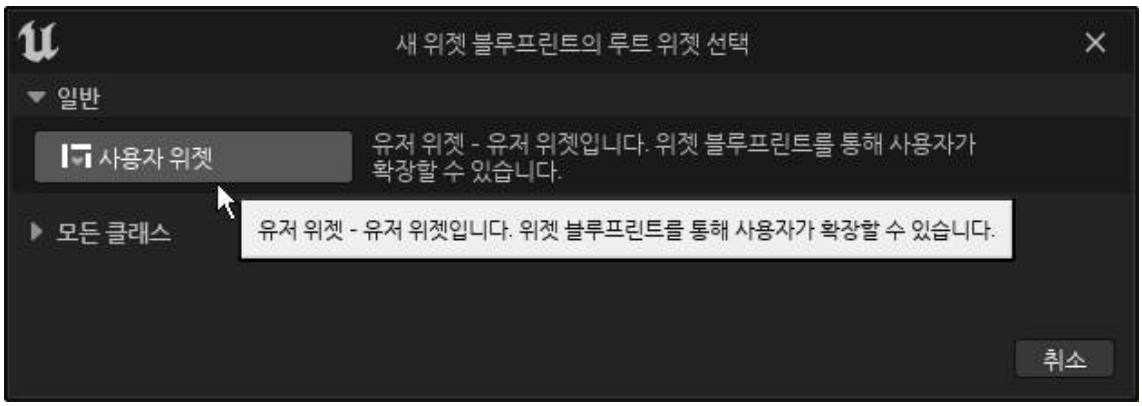
2. 지금부터, 프로젝트 준비 과정을 시작하자.
먼저, 이전 프로젝트에서의 작업을 가져오는 일을 진행하자.
첫 번째로, 이전 프로젝트해서 생성해둔 블루프린트 클래스를 가져오자. 윈도우 파일 탐색기에서 이전 Pinputevent 프로젝트의 **Content** 폴더로 이동하자. 그 아래에 있는 3개의 애셋 파일(BP_MyCharacter, BP_MyGameMode, BP_MyPlayerController)을 **Ctrl**+**C**로 복사하고 새 프로젝트의 **Content** 폴더 아래로 이동하여 **Ctrl**+**V**로 붙여넣자.
두 번째로, 이전 프로젝트에서 설정한 입력 매핑을 가져오자. 우선, 새 프로젝트를 종료하자. 그다음, 이전의 프로젝트 Pinputevent 프로젝트의 **Config** 폴더로 이동하자. 그 아래에 있는 DefaultInput.ini 파일을 **Ctrl**+**C**로 복사하고 새 프로젝트의 **Config** 폴더 아래로 이동하여 **Ctrl**+**V**로 붙여넣자. 그다음, 새 프로젝트를 다시 로드하자.
세 번째로, **MyMap** 레벨이 열린 상태에서 **월드 세팅** 탭으로 가자. 탭이 보이지 않으면 메뉴바의 **창** » **월드 세팅**을 선택하면 **디테일** 탭의 옆에 추가된다. **월드 세팅** 탭에서 **게임모드 오버라이드** 속성에 **BP_MyGameMode**를 지정하자.
이제, 이전 프로젝트에서의 모든 작업이 포함되도록 준비과정을 완료하였다.

3. 프로젝트에서 필요한 외부 애셋을 준비하자.
배경 사진으로 적합한 이미지를 하나 준비하자. 어떤 이미지라도 상관없다. 우리가 준비한 이미지 파일의 이름이 MyBackgroundImage.png라고 하자.
파일 탐색기에서 이 파일을 콘텐츠 브라우저의 콘텐츠 폴더 아래에 드래그해서 임포트하자. 콘텐츠 폴더 아래에 **MyBackgroundImage** 텍스처 애셋이 생길 것이다.
이제 필요한 모든 애셋이 준비되었다.

4. **위젯 블루프린트**는 모든 유저 인터페이스 요소를 저장할 수 있는 애셋이다. 위젯 블루프린트를 통해서 UI 요소의 시각적인 배치와 그 요소의 스크립트 작성을 수행한다.
이제부터, 위젯 블루프린트 애셋를 생성하자.
콘텐츠 브라우저에서 **+추가**를 누르고 **유저 인터페이스** » **위젯 블루프린트**를 선택하자.

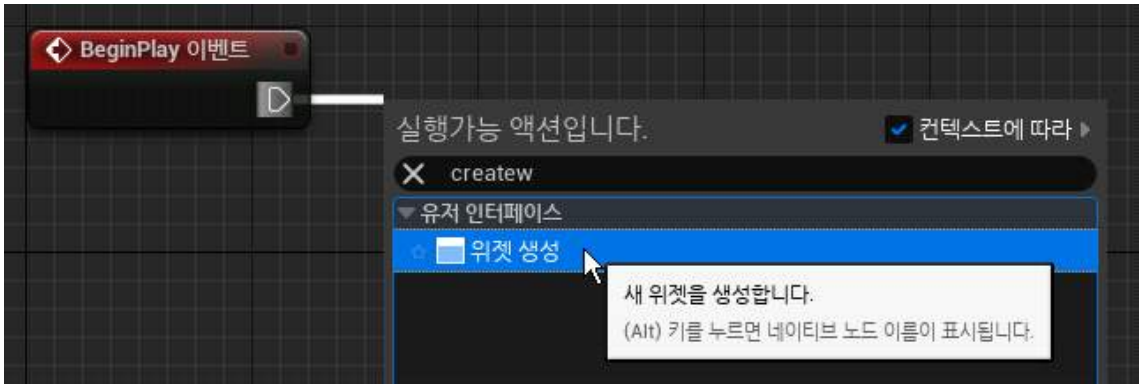


5. 다음 대화창에서 생성할 새 위젯 블루프린트의 루트 위젯을 선택한다. 사용자 위젯을 클릭하자.

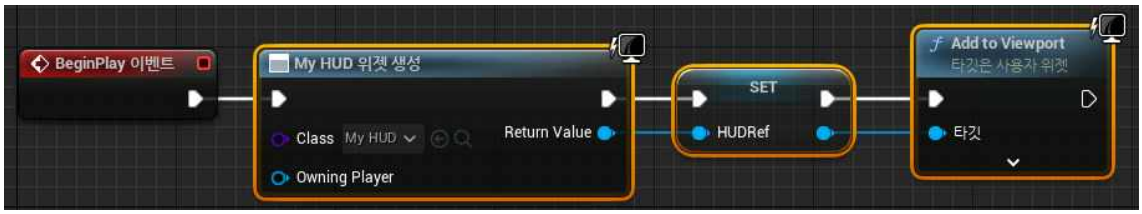


위젯 블루프린트가 생성된다. 디폴트 이름인 **NewWidgetBlueprint**을 **MyHUD**로 수정하자.

6. 이제 레벨이 플레이되면 위젯 인스턴스를 생성하고 화면에 보이도록 하자.
BP_MyCharacter의 이벤트 그래프 탭에서 **BeginPlay** 이벤트 노드를 당기고 **Create Widget** 노드를 배치하자.

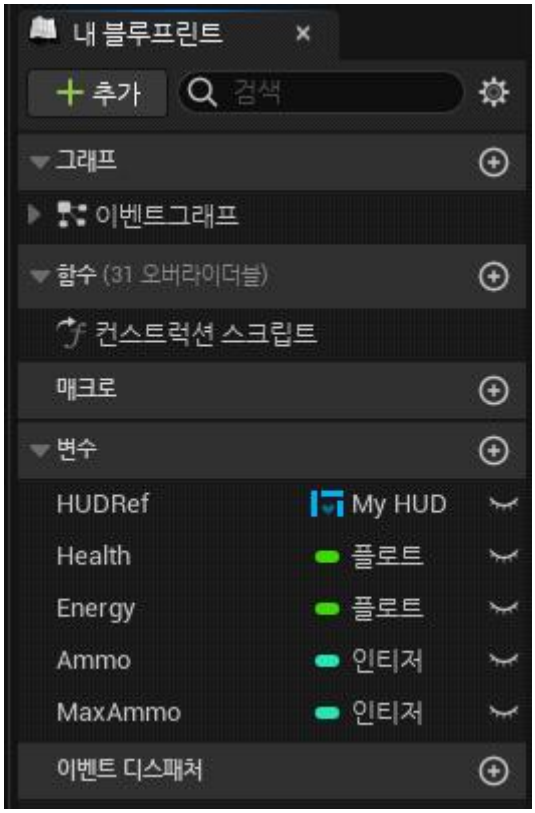


7. 위젯 생성 노드의 **Class** 입력핀을 선택하고 **MyHUD**를 선택하여 지정하자.
그다음, 위젯 생성 노드의 **Return Value** 출력핀을 당기고 **변수로 승격**을 선택하자. 생성된 변수의 이름을 **HUDRef**로 수정하자. 나중에 이 **HUDRef** 변수를 접근하여 위젯의 속성 설정이나 함수 호출을 편리하게 할 수 있다.
그다음, **Set** 노드의 출력 핀을 당기고 **Add to Viewport** 노드를 배치하자. 이 노드는 입력으로 명시된 위젯 블루프린트를 플레이어 뷰포트로의 **타깃**에 추가하여 화면에 표시되도록 한다. 이제 그래프가 다음과 같이 보일 것이다.



지금까지, 프로젝트 준비과정이 완료되었다.

8. **BP_MyCharacter** 블루프린트 에디터에서 계속 작업하자.
지금부터, HUD에 표시할 정보인 생명력, 에너지, 탄약수의 추가 과정을 진행하자.
나중에 HUD 위젯 블루프린트에서 표시되도록 할 플레이어 정보를 준비하자.
콘텐츠 브라우저에서 **BP_MyCharacter**를 더블클릭하여 블루프린트 에디터를 열자.
먼저, HUD에 표시할 플레이어 캐릭터의 생명력 정보를 추가하자. **내 블루프린트** 탭에서 **변수**를 추가하자. **플로트** 유형의 변수 **Health**를 추가하자. 컴파일하고 **디테일** 탭에서 **기본값**을 1로 설정하자.
다음으로, HUD에 표시할 플레이어 캐릭터의 에너지 정보를 추가하자. **플로트** 유형의 변수 **Energy**를 추가하자. 컴파일하고 디테일 탭에서 기본값을 1로 설정하자.
다음으로, 탄창 내의 탄약(애모, Ammo, ammunition) 개수 정보를 추가하자. **인티저** 유형의 변수 **Ammo**를 추가하자. 컴파일하고 **디테일** 탭에서 **기본값**을 25로 설정하자. 그리고, 최대 탄약 수를 나타내는 정보를 추가하자. **인티저** 유형의 변수 **MaxAmmo**를 추가하자. 컴파일하고 **디테일** 탭에서 **기본값**을 25로 설정하자.



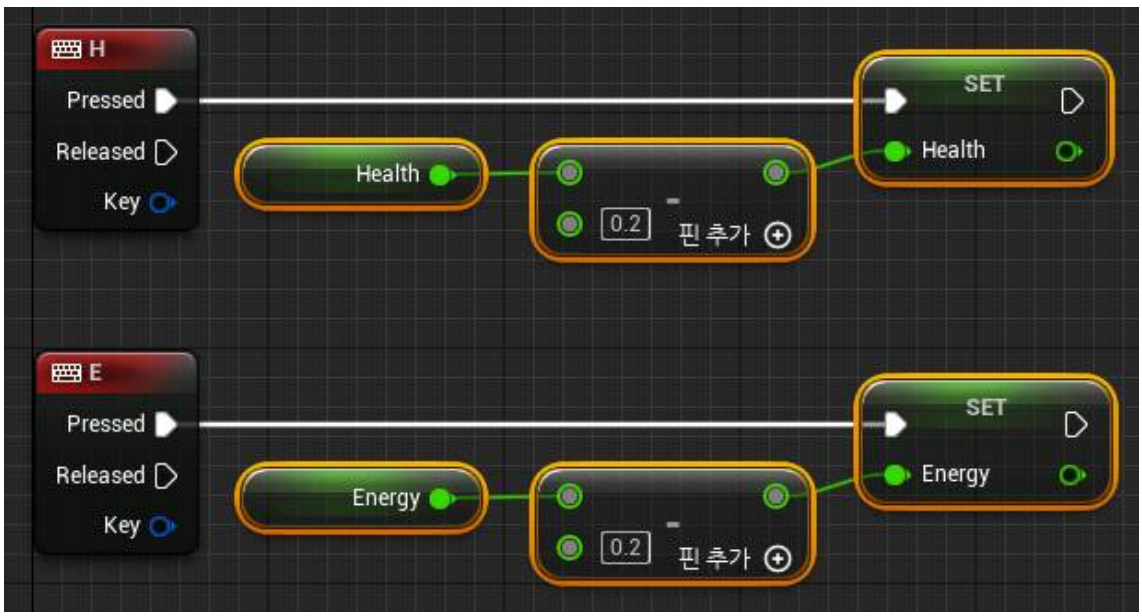
9. **BP_MyCharacter** 블루프린트 에디터에서 계속 작업하자.

이제 추가된 변수인 **Health**, **Energy**, **Ammo**가 적절하게 변경되도록 하자. 점프나, 사격이나, 격투 등의 이벤트에 대해서 값이 변경되도록 해주어야 하겠지만 우리는 테스트를 위해서 키 입력 이벤트에 따라 변경되도록 하자.

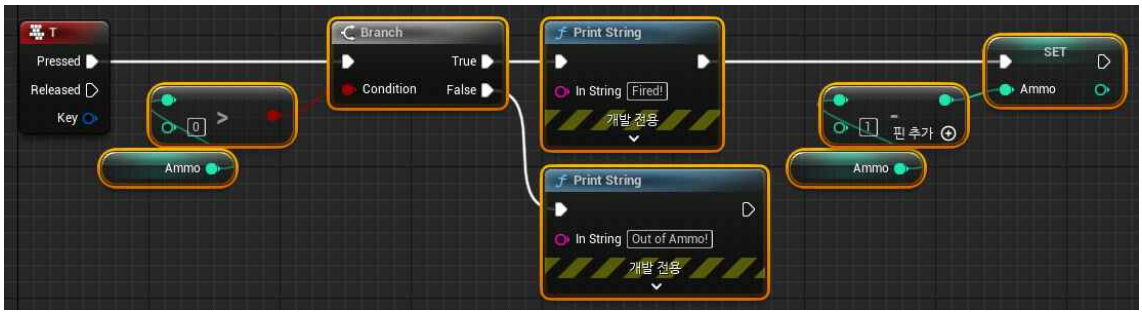
이벤트 그래프 탭에서 입력 이벤트 노드 그래프를 만들자.

먼저, **H** 키를 누를때마나 **Health**가 0.2씩 감소하도록 하자.

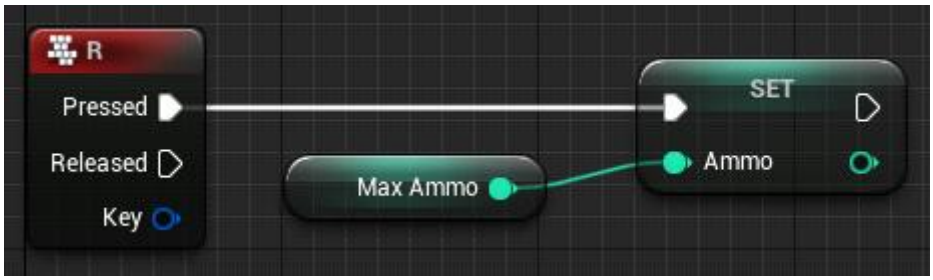
그다음, **E** 키를 누를때마나 **Energy**가 0.2씩 감소하도록 하자.



10. 그다음, **T** 키를 누를때마나 **Ammo**가 1씩 감소하도록 하자. 만약 0인 경우에는 발사가 불가능하도록 알려주고 발사되지 않도록 하자.



11. 그다음, **R** 키를 누르면 재장전이 되도록 하자. **Ammo**를 **MaxAmmo**로 지정하면 된다.



지금까지, HUD에 표시할 정보인 생명력, 에너지, 탄약수의 추가 과정을 진행하자.

이 절에서는 UMG 학습을 위한 프로젝트를 준비하는 과정을 학습하였다.

3. 캐릭터 정보 HUD 구현의 레이아웃 배치 단계

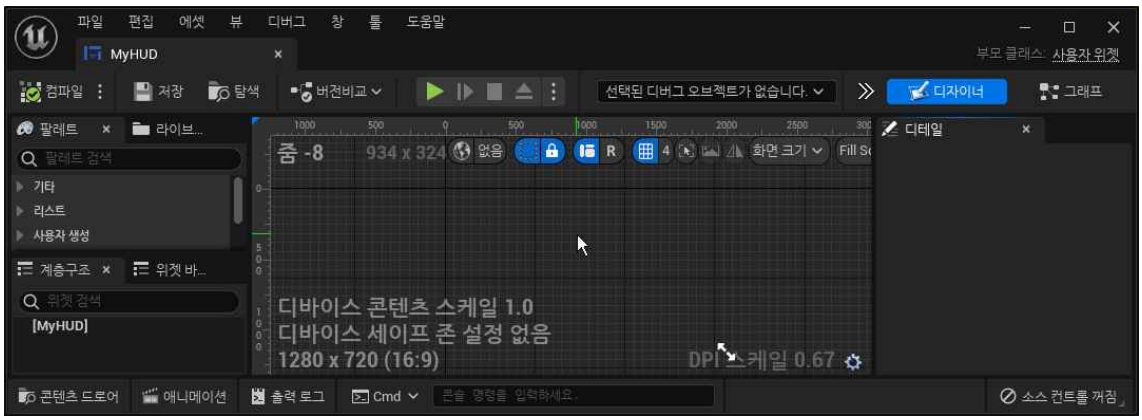
이 절에서는 게임 내에서 캐릭터 정보를 표시하는 HUD 구현에 대해서 학습한다.
먼저, 위젯을 적절히 배치하여 시각적인 레이아웃을 완성하는 단계를 진행한다. 그다음, 스크립팅 단계를 나누어서 진행할 것이다.

이제부터 예제를 통해서 학습해보자

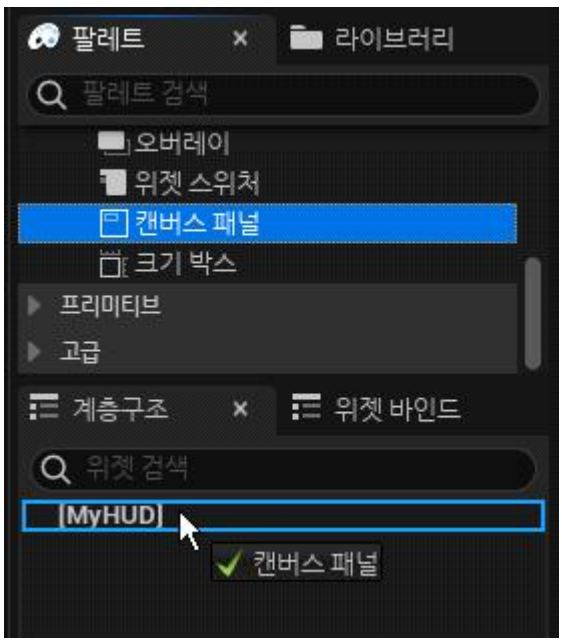
---

1. 이전 프로젝트 `Pumgfirst`에서 계속하자.

2. 이제부터, 위젯을 배치해서 시각적인 UI 레이아웃을 만들어보자.
콘텐츠 브라우저에서 **MyHUD**를 더블클릭하여 위젯 블루프린트 에디터를 열자.

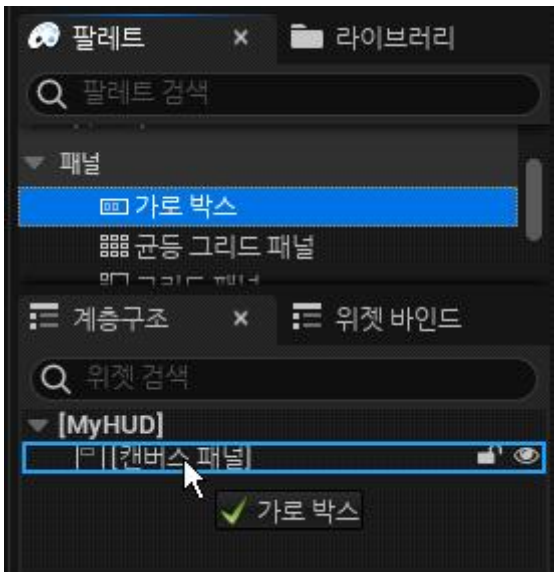


3. **팔레트** 탭에서 **패널** » **캔버스 패널(Canvas Panel)** 위젯을 끌어 바로 아래의 **계층구조** 탭에서 최상단 노드인 **[MyHUD]**에 드롭하자.

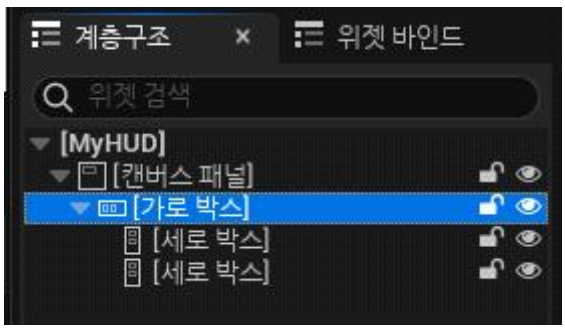


4. **팔레트** 탭에서 **패널** » **가로 박스(Horizontal Box)** 위젯을 끌어 바로 아래의 **계층구조** 탭에서 **[캔버스**

패널]에 드롭하자.

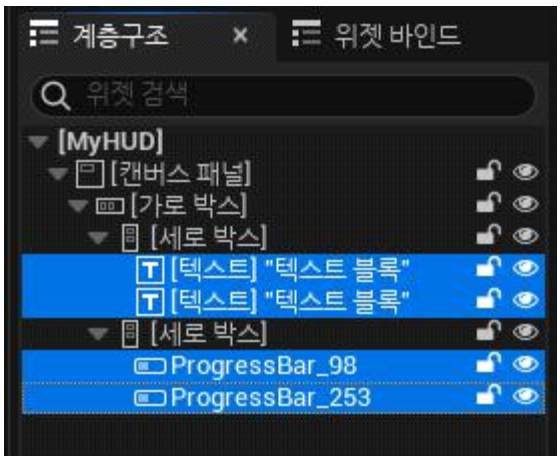


5. **팔레트** 탭에서 **패널** » **세로 박스**(**Vertical Box**) 위젯을 두 개를 끌어 **계층구조** 탭에서 **[가로 박스]**에 드롭하자. 세로 박스 두 개가 가로 박스 내에서 수평 방향으로 배치될 것이다.

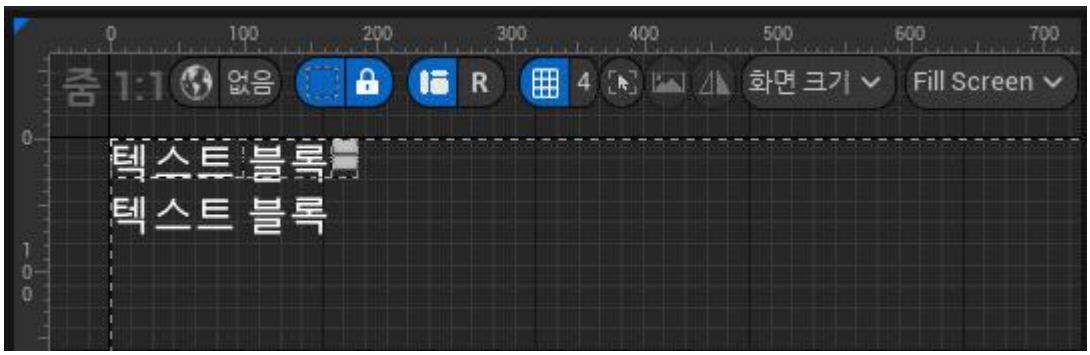


6. **팔레트** 탭에서 **일반** » **텍스트**(**Text**) 위젯을 두 개를 끌어 **계층구조** 탭에서 첫 번째 **[세로 박스]**에 드롭하자.

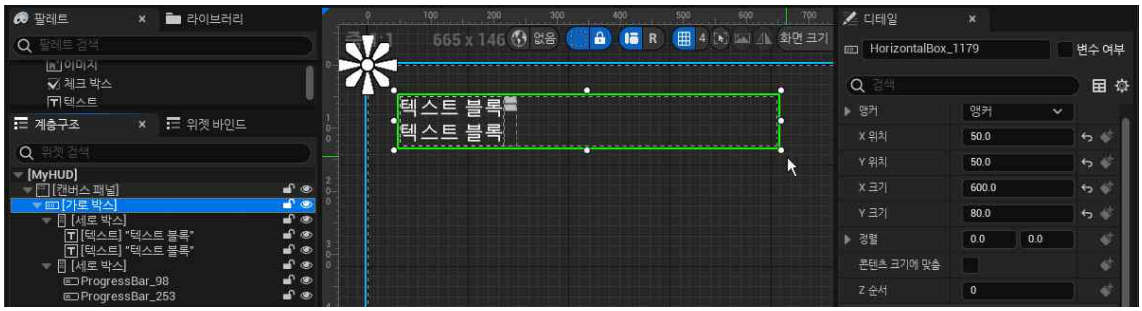
그다음, **일반** » **프로그레스 바**(**Progress Bar**) 위젯을 두 개를 끌어 두 번째 **[세로 박스]**에 드롭하자. 첫 번째 **[세로 박스]**에는 텍스트 블록 두 개가 수직 방향으로 배치될 것이고, 두 번째 **[세로 박스]**에는 진행바가 수직 방향으로 배치될 것이다.



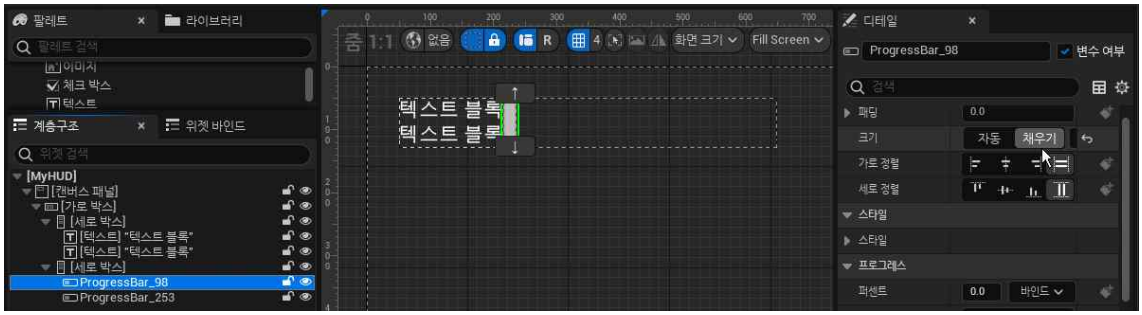
7. 이제, 디자이너 탭을 보자. 위젯들이 너무 작게 겹쳐서 왼쪽 위의 모서리에 표시될 것이다.



8. 디자이너 탭에서 위치와 크기가 되도록 배치를 조절해보자.
먼저, 계층구조 탭에서 **[가로 박스]**를 선택하자. 디자이너 탭에서 해당 위젯의 영역이 녹색 박스로 표시될 것이다. 박스를 드래그해서 위치와 크기를 조절해서 왼쪽 위의 부분에 배치되도록 하자. 디테일 탭에서 X 위치, Y 위치를 50,50으로 하고 X 크기, Y 크기를 600, 80으로 입력해도 된다.

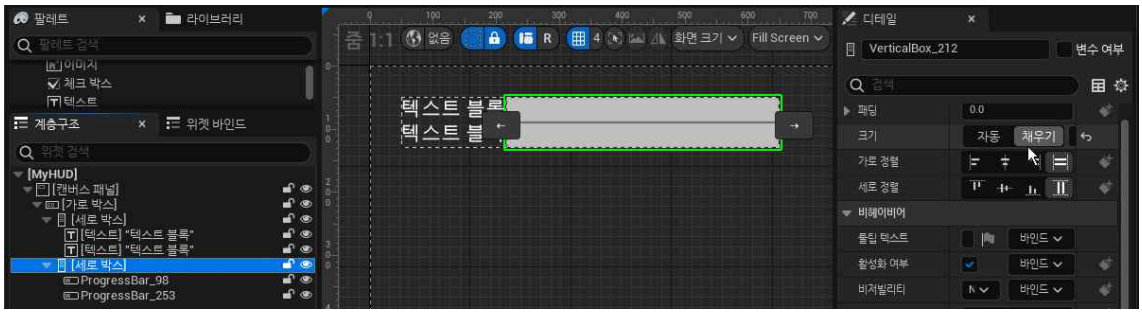


9. 그다음, 첫 번째 **Progress Bar**를 선택하고 배치를 살펴보자. 텍스트 오른쪽에 아주 작게 배치되어 있다.
Progress Bar의 **디테일** 탭에서 **크기(Size)** 속성을 찾아보자.
이 **크기(Size)** 속성에 대해서 알아보자. 속성값은 **자동** 또는 **채우기** 중의 하나이다. 속성값이 **자동**인 경우에는 필요한 만큼의 공간만 차지하도록 한다. **채우기**를 선택하면 차지할 수 있는 모든 공간을 차지하도록 한다. **채우기**일 경우에는 바로 오른쪽에 숫자 입력상자가 하나 더 표시된다. 숫자가 1일 경우에는 차지할 수 있는 모든 공간을 차지하지만 그 미만인 경우에는 비율만큼만 차지한다. 남은 공간은 다른 위젯이 차지할 수 있다. 예를 들어서 첫 번째 진행바를 1로 하고 다른 두 번째 진행바를 0.5로 하면 첫 번째 진행바가 2/3의 영역을 차지하고 두 번째 진행바가 1/3의 영역을 차지한다.
우리는 **크기**의 속성값을 디폴트인 **자동**에서 **채우기**로 바꾸자.



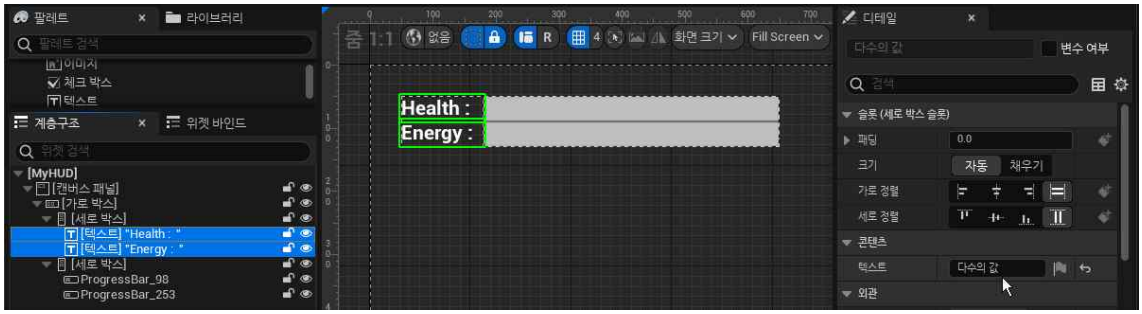
두번째 진행바에 대해서도 **크기**의 속성값을 디폴트인 **자동**에서 **채우기**로 바꾸자.
이제, 두 진행바가 수직 방향으로의 모든 공간을 절반씩 차지하도록 크기가 조정될 것이다.

10. 두 진행바가 들어있는 부모 위젯인 **[세로 박스]**를 살펴보자. 수직 방향으로는 전체를 차지하도록 수정하였으나, 수평 방향으로 보면 오른쪽에 빈 공간이 너무 많이 남는다.
[세로 박스]의 **크기(Size)** 속성도 **자동**에서 **채우기**로 바꾸자.

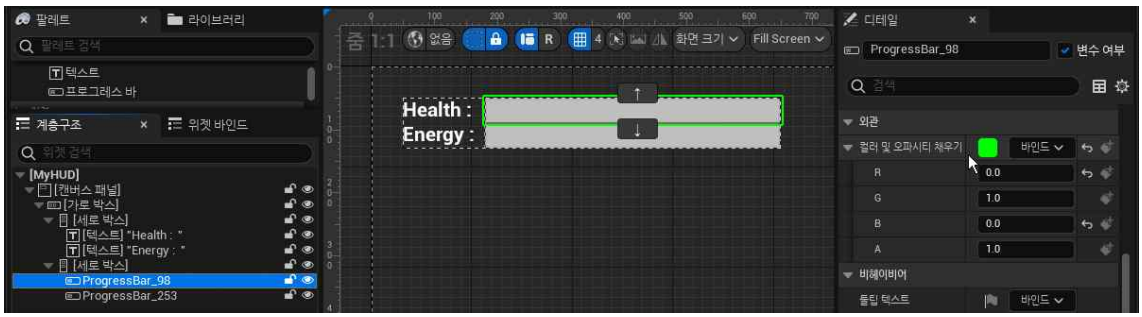


이제, **[세로 박스]**의 영역이 오른쪽의 남은 공간을 모두 포함하도록 커질 것이다.
[세로 박스]의 영역이 커지면 그 안에 포함된 두 진행바도 **채우기**로 되어 있으므로 자동으로 함께 커질 것이다.

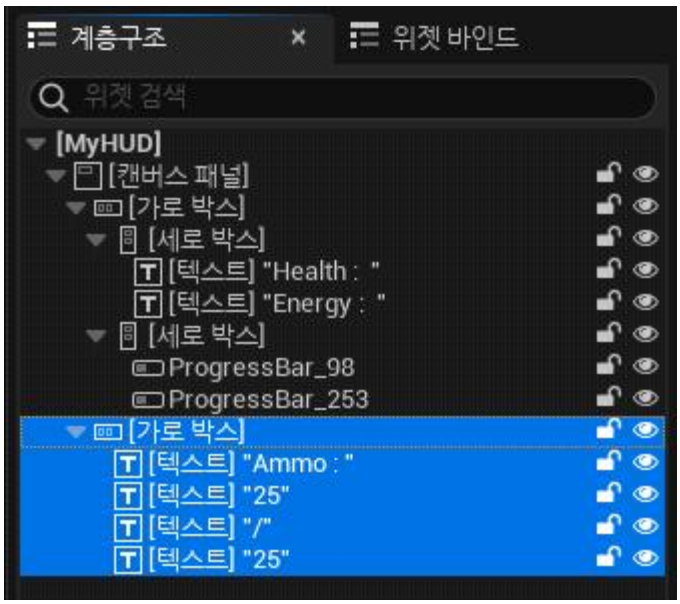
11. 텍스트 블록의 내부 문자열을 수정하자. 각 텍스트 위젯을 선택하고 디테일 탭에서 **콘텐츠** 영역의 **텍스트(Text)** 속성값을 디폴트인 '텍스트 블록'에서 '`Health :` '와 '`Energy :` '로 수정하자. 공백 문자를 사용해서 적당하게 띄워지도록 하였다.



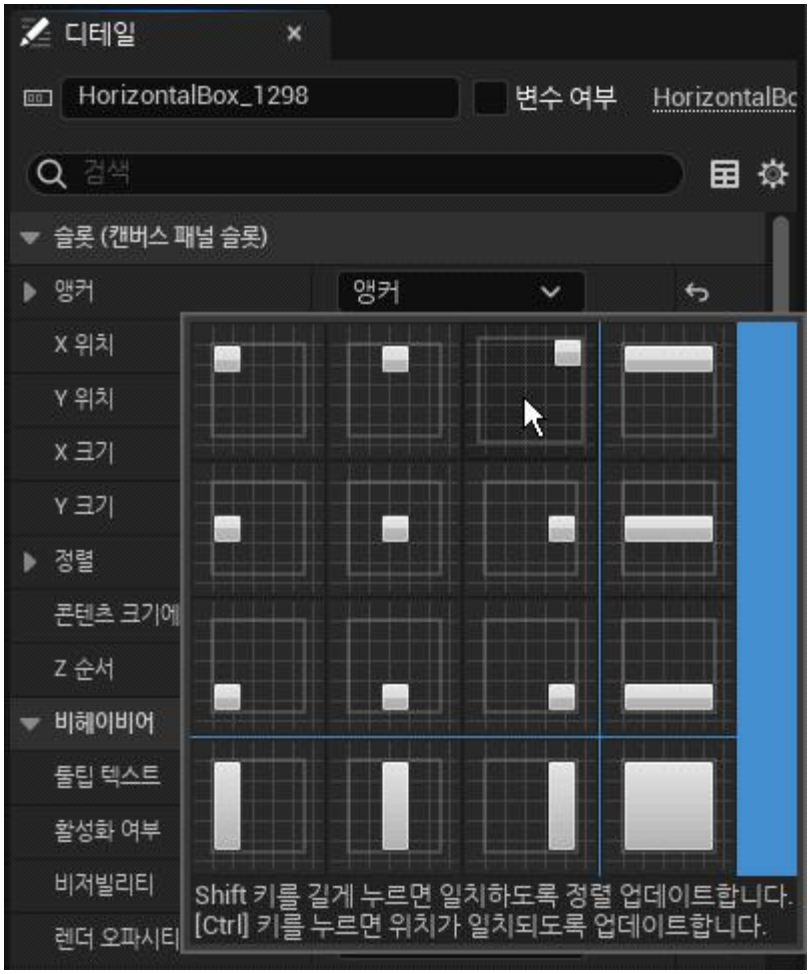
12. 진행바는 컬러가 디폴트로 파란색인 (0,0.5,1,1)인데 이것을 바꾸어보자.
Health 오른쪽의 첫 번째 **Progress Bar**를 선택하고 디테일 탭에서 외관 영역의 **컬러 및 오파시티 채우기(Fill Color and Opacity)** 속성값에서 RGBA에 디폴트인 (0, 0.5, 1, 1) 대신에 (0, 1, 0, 1)을 입력하여 녹색이 되도록 하자.
Energy 오른쪽의 두 번째 **Progress Bar**는 주황색이 되도록 (1, 0.5, 0, 1)을 입력하자.



13. 이제, 탄약수 정보인 **Ammo**도 표시해보자.
팔레트에서 새로운 **[가로 박스]**를 **계층구조** 탭의 **[캔버스 패널]**에 추가하자.
그다음, **Text** 위젯을 4개를 **[가로 박스]**에 추가하자. 그리고, 각각의 **Text** 속성값을 'Ammo : ', '25', '/', '25'로 수정하자.



14. 그다음, **[가로 박스]**의 디테일 탭에서 슬롯 영역의 앵커를 클릭하고 우상단 앵커를 선택하자. 선택한 후에는 앵커 표시 별표가 우상단 모서리로 바뀔 것이다.



<참고> 이제부터, 앵커(Anchor)에 기능에 대해서 알아보자.

Canvas Panel에 추가된 위젯은 앵커 속성을 가진다. 위젯이 선택되면 앵커의 위치를 잎이 8개인 꽃 모양의 아이콘으로 보여준다. 앵커를 사용하면 화면 크기와 상관없이 엥커의 위치를 기준으로 위젯의 표시 위치를 지정하도록 할 수 있다. 즉 화면의 크기가 바뀌어도 위젯을 앵커 위치에서 동일한 거리만큼 떨어진 지점에 표시된다.

앵커의 속성값 지정 방법에 대해서 알아보자. 앵커의 속성값은 Minimum(X,Y)과 Maximum(X,Y)의 형식으로 명시된다. 먼저, Min과 Max의 값을 동일하게 지정하는 방법을 알아보자. 둘 다 (0,0)으로 지정하면 왼쪽위 모서리를 기준으로 정렬하도록 한다. (1,1)은 오른쪽 하단 모서리를 기준으로 정렬한다. 즉 X가 0이면 왼쪽, 0.5이면 중간, 1이면 오른쪽이다. Y가 0이면 위쪽, 0.5이면 중간, 1이면 아래쪽이다.

그리고, 화면 해상도와 무관하게 **Canvas Panel**의 영역에서 위젯이 차지하는 영역의 비율이 유지되도록 자동으로 늘여서 맞추도록 지정할 수도 있다. 늘여서 맞추는 경우에는 앵커 표시가 한 지점이 아니라 두 지점 또는 네 지점에 다르게 표시된다. 수평방향으로 늘여서 비율이 맞도록 하려면 Min,Max의 X값을 0,1로 지정하면 된다. 수직방향으로 늘이려면 Min,Max의 Y값을 0,1로 지정하면 된다. 두 방향으로 모두 늘여서 비율이 맞도록 하려면 Min을 (0,0)으로 Max를 (1,1)로 하면 된다.

속성값 지정은 Min,Max에 값을 직접 입력해도 되지만 격자로 보여주는 아이콘에서 해당하는 형태를 선택해서 지정하면 편리하다.

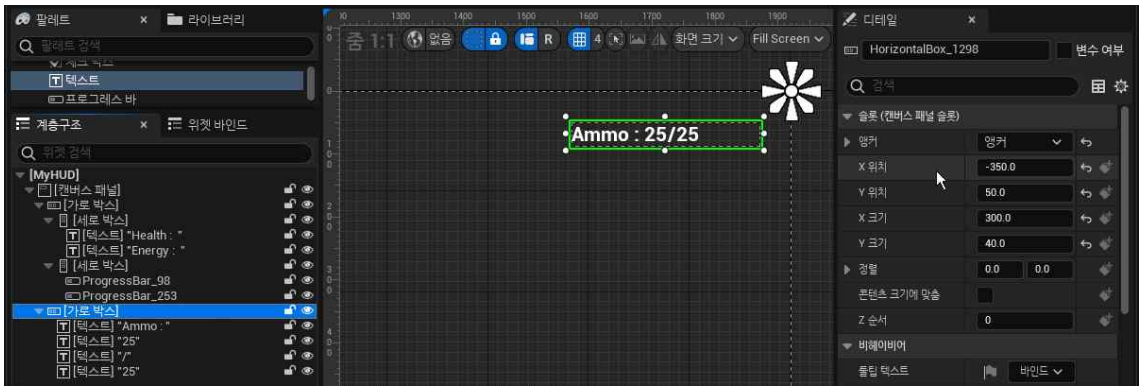
또한 표시된 앵커 아이콘을 드래그하여 이동시킬 수 있다. 앵커 아이콘의 중심을 드래그하면 위치를 이동시킬 수 있고 아이콘의 각 꽃잎을 드래그하면 늘이기 구간을 이동시킬 수 있다.

지금까지 앵커에 대해서 알아보았다.

앵커에 대한 자세한 내용은 다음의 문서를 참조하자.

https://docs.unrealengine.com/umg-anchors-in-unreal-engine-ui/

15. 그다음, 새로 추가된 **[가로 박스]**의 위치를 오른쪽 상단 모서리에 오도록 하고 크기를 약간 크게 하여 글씨가 모두 보이도록 한다.
디테일 탭에서 X 위치, Y 위치를 -350,50으로 하고 X 크기, Y 크기를 300, 40으로 입력해도 된다.



컴파일하고 저장하자.
이제 시각적인 레이아웃 배치를 완료하였다.

이 절에서는 게임 내에서 캐릭터 정보를 표시하는 HUD 구현의 시각적 레이아웃 배치 단계에 대해서 학습하였다.

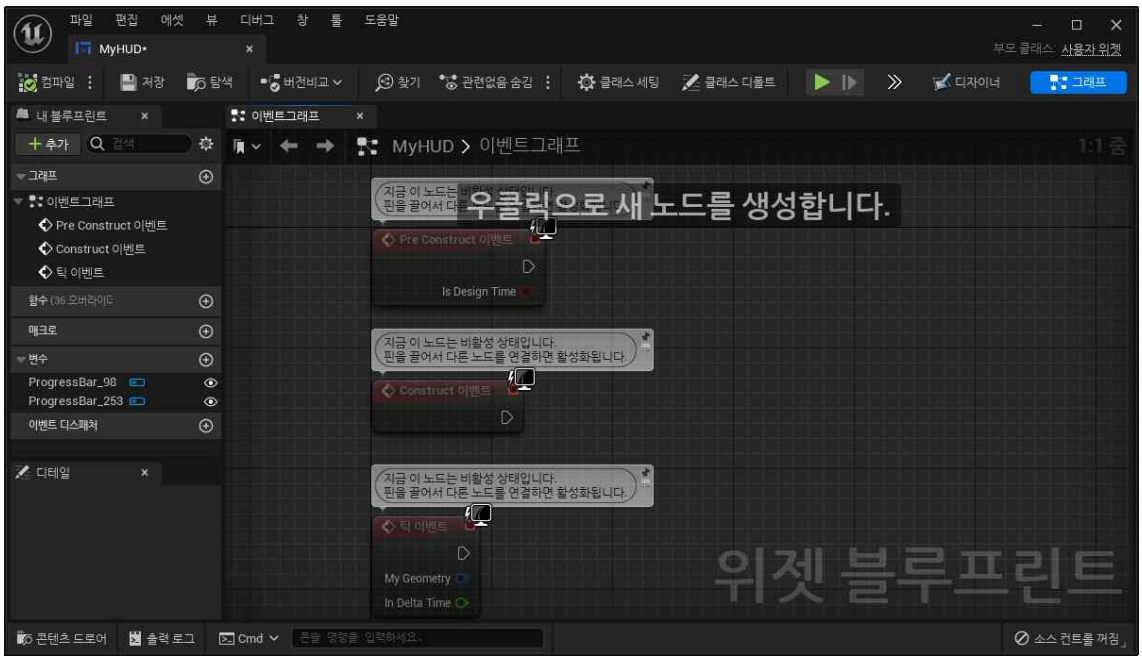
4. 캐릭터 정보 HUD 구현의 스크립팅 단계

이 절에서는 게임 내에서 캐릭터 정보를 표시하는 HUD 구현에 대해서 학습한다.
시각적 위젯 레이아웃이 배치된 후에 이들 위젯의 배후의 함수 기능을 제공하도록 스크립트를 구현한다.
이제부터 예제를 통해서 학습해보자

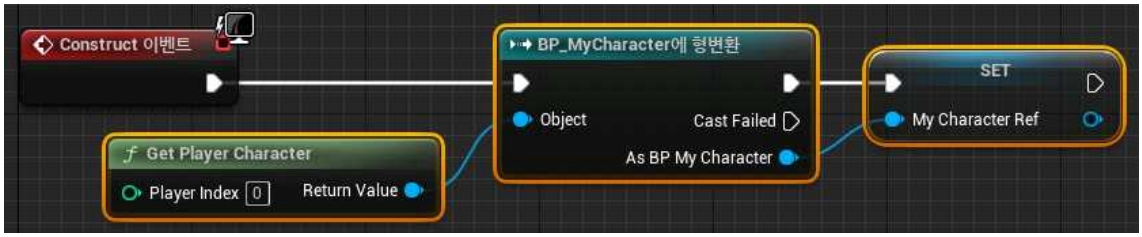
---

1. 이전 프로젝트 `Pumgfirst`에서 계속하자.

2. 콘텐트 브라우저에서 **MyHUD**를 더블클릭하여 위젯 블루프린트 에디터를 열자.
에디터의 우측 상단의 **그래프** 버튼을 클릭하여 **디자이너** 모드에서 **그래프** 모드로 바꾸자.
이벤트 그래프 격자판에서 **Pre Construct 이벤트** 노드, **Construct 이벤트** 노드, **Tick 이벤트** 노드의 세 개의 이벤트 노드가 이미 배치되어 있을 것이다.



3. 우리는, 이미 배치되어 있는 세 이벤트 노드 중에서 **Construct 이벤트** 노드의 그래프를 작성해보자.
이 노드는 HUD가 생성될 때에 처음에 한번 호출된다.
먼저, 빈 공간에서 우클릭하고 **GetPlayerCharacter** 노드를 추가하자.
그다음, 노드의 **Return Value** 출력핀을 드래그하고 **BP_MyCharacter에 형변환** 노드를 배치하자.
그다음, **As BP_MyCharacter** 출력핀을 드래그하고 변수로 승격을 선택하자. 변수의 이름을 **MyCharacter Ref**로 수정하자.
그다음, **Construct 이벤트** 노드의 실행핀을 **형변환** 노드의 실행핀에 연결하자.



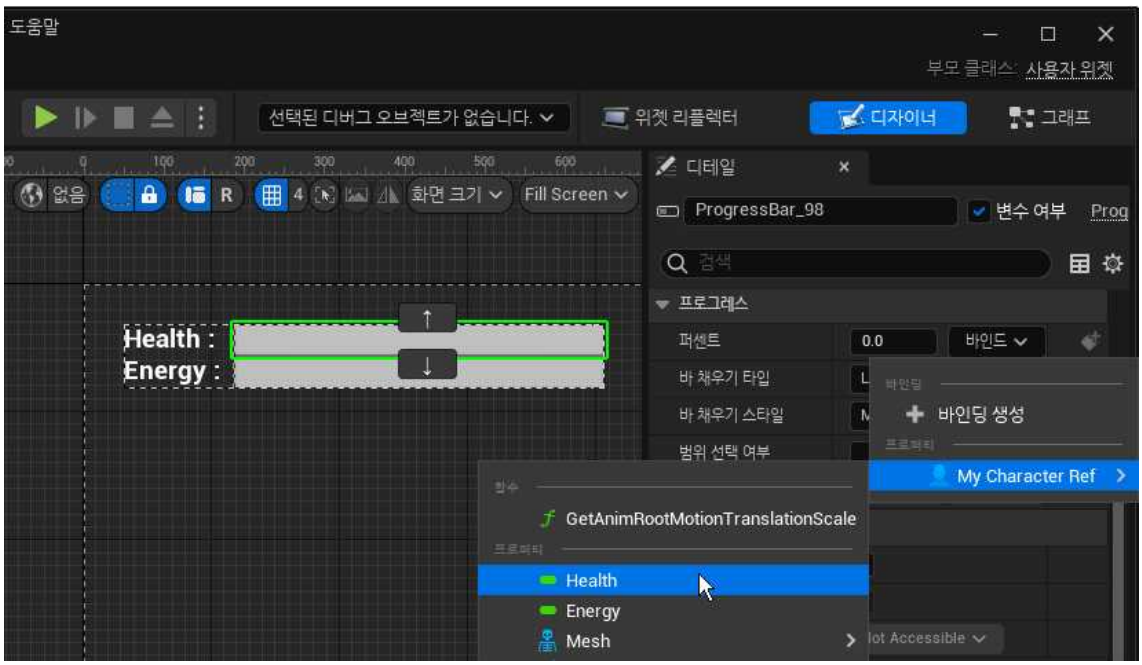
컴파일하고 저장하자.

우리는 플레이어 캐릭터의 레퍼런스를 가지는 변수 **MyCharacterRef**를 생성하였다. 이제부터 변수 **MyCharacterRef**의 멤버 변수나 멤버 컴포넌트에 대해서 접근할 수 있게 되었다.

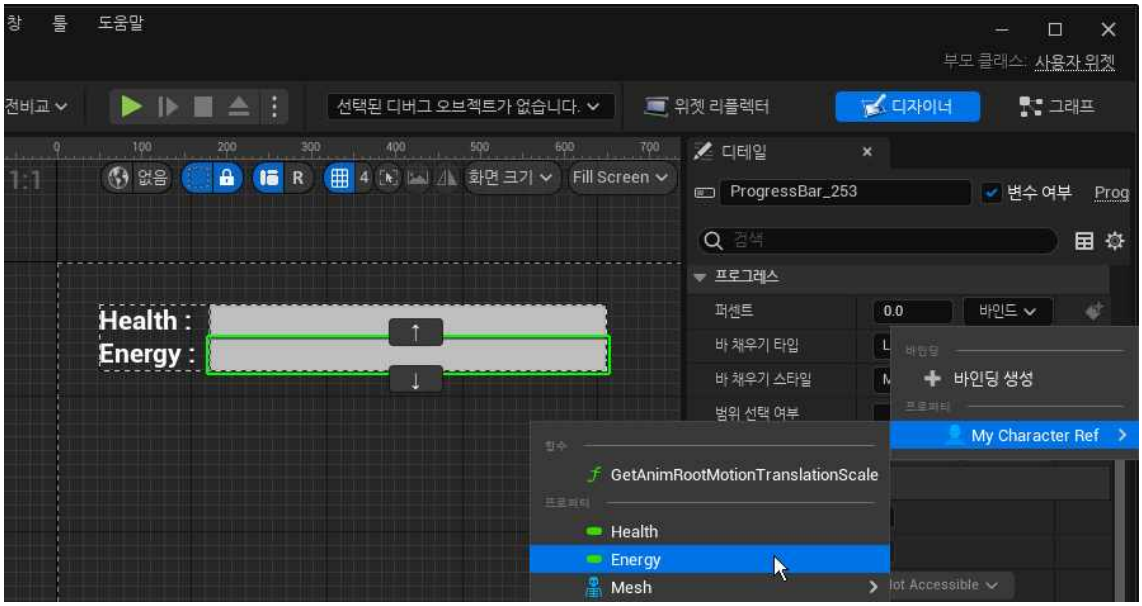
4. 에디터의 디자이너 탭을 클릭하여 디자이너 모드로 돌아가자.

그리고, **Health** 옆에 있는 첫 번째 **Progress Bar**를 선택하자.

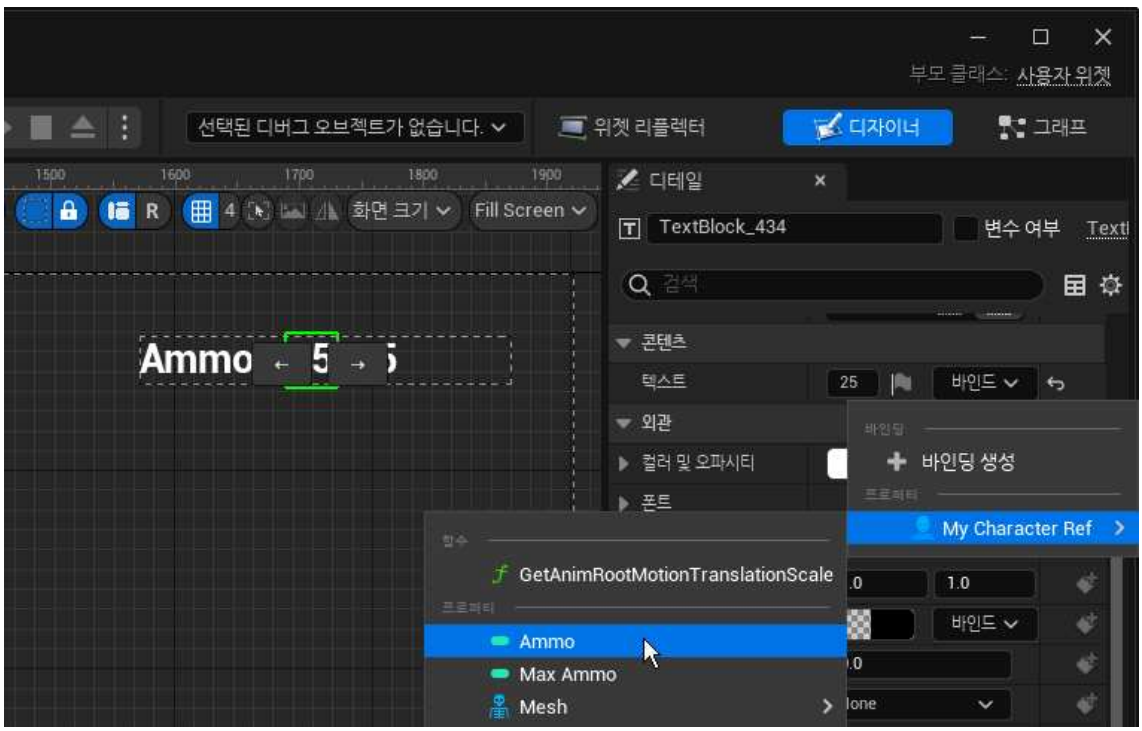
그다음, **디테일** 탭에서 **프로그레스** 영역의 **퍼센트** 속성 옆의 **바인드** 옵션을 선택하자. 그리고 드롭다운 메뉴에서 **MyCharacterRef** 아래의 **Health**를 선택하자. 이제 **Progress Bar**의 값이 선택된 변수의 값과 바인드된다. 따라서 캐릭터의 **Health** 값이 바뀌면 HUD에서의 값도 자동으로 업데이트된다.



5. 다음으로, **Energy** 옆에 있는 두 번째 **Progress Bar**에 대해서도 동일한 작업을 수행하자. 이 진행바는 **디테일** 탭에서 **Percent** 속성의 **바인드**를 클릭하고 **MyCharacterRef**의 **Energy**를 선택하자.



6. 다음으로, **Ammo**에 대해서도 바인드하자. '**Ammo**' 다음의 '**25**'에 해당하는 **Text**를 선택하자. 그다음, **디테일** 탭에서 **콘텐츠** 영역의 **텍스트(Text)** 속성을 찾고 그 옆의 **바인드**를 클릭하고 **MyCharacterRef**의 **Ammo**를 선택하자.

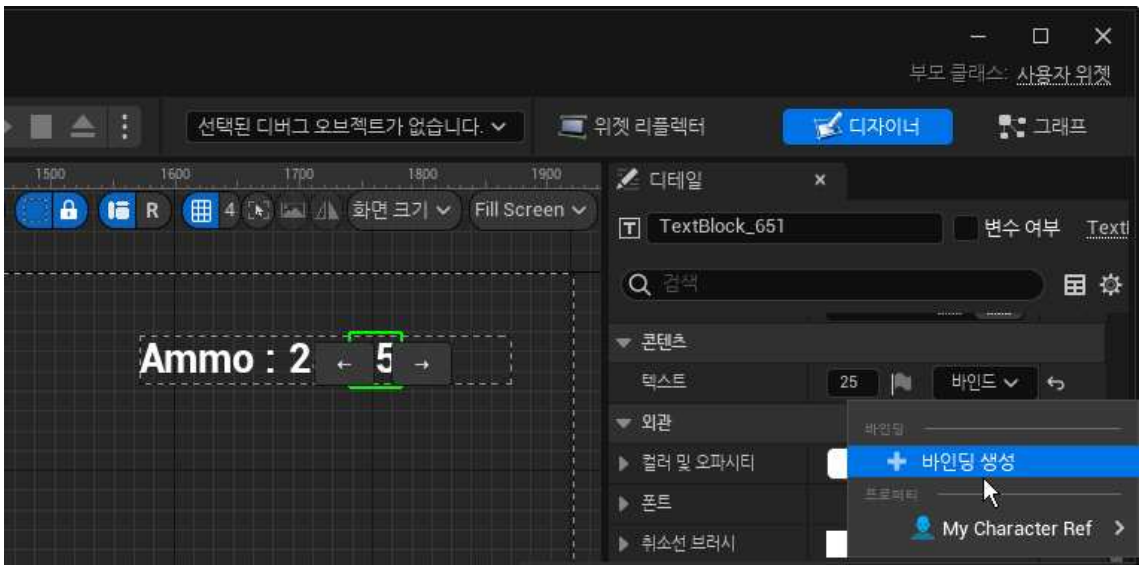


7. 다음으로, **Ammo**의 마지막 '**25**'에 대해서는 이전과 같은 방법으로 바인드를 클릭하고 **MyCharacter Ref**의 **MaxAmmo**를 선택하면 될 것이다. 만약 **MaxAmmo**가 바뀌지 않을 것이라면 바이드할 필요도 없을 것이다.

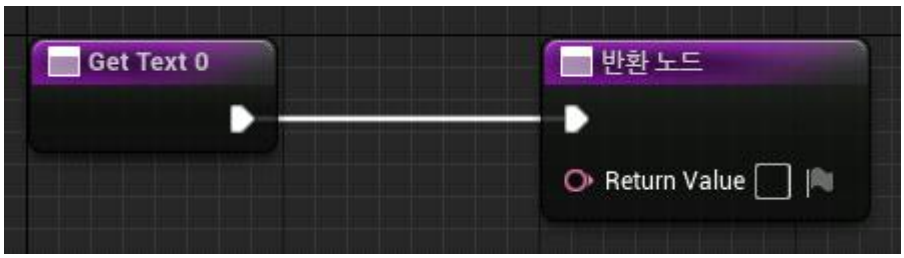
학습을 위해서 이번에는 다른 방식으로 바인딩을 해보자

마지막 '**25**'에 해당하는 **|텍스트|**를 선택하고 **콘텐츠** 영역의 **텍스트** 속성을 찾고 그 옆의 **바인드**를

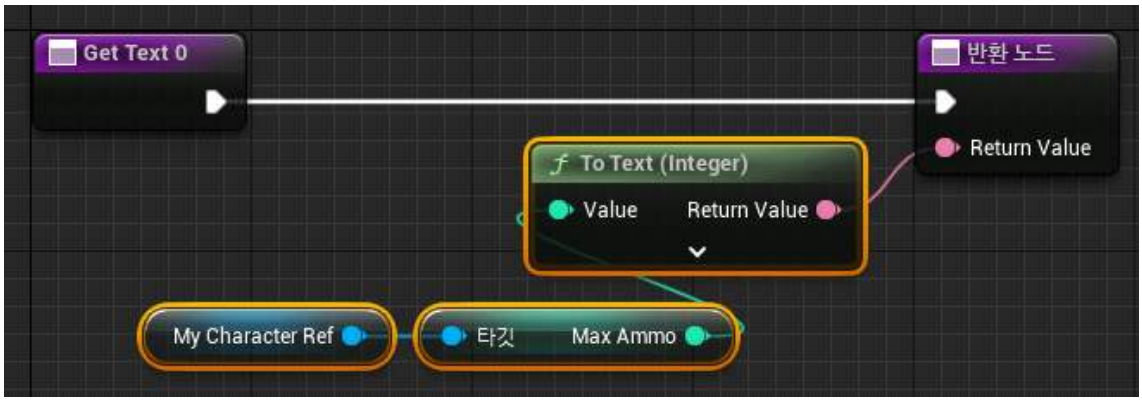
클릭하자. 그리고, 드롭다운 메뉴에서 **바인딩 생성**을 선택하자.



8. 디폴트로 **GetText_0**이라는 함수가 생성되고 그래프 모드로 전환될 것이다. 생성된 함수에서 바인딩하는 스크립트를 작성하면 된다.



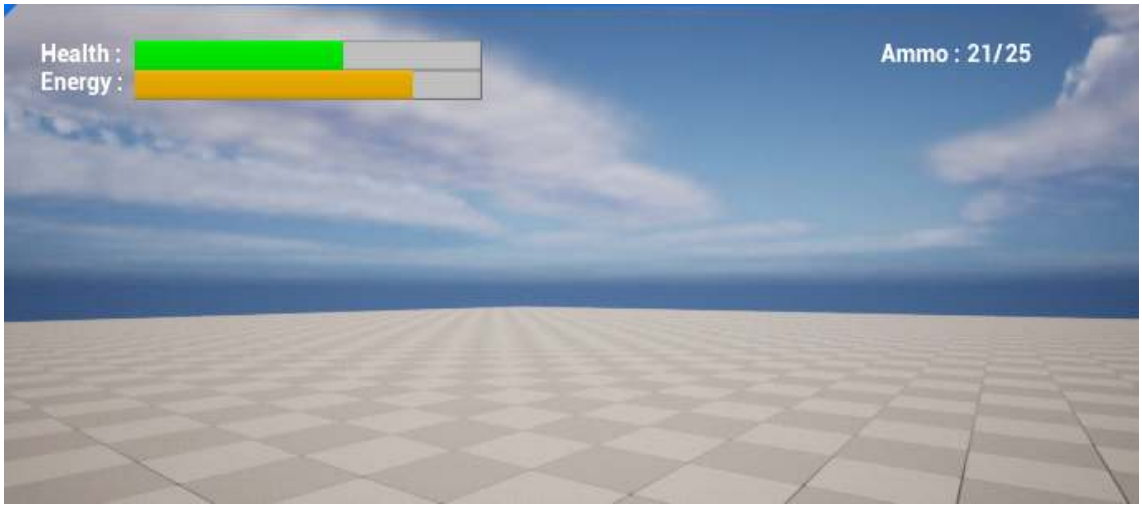
9. 이제 함수 **GetText_0**의 스크립트를 작성해보자.
먼저, 내 블루프린트 탭에서 **MyCharacterRef** 변수를 드래그하여 **Get** 노드를 배치하자.
그다음, **Get** 노드를 당기고 **Get MaxAmmo** 노드를 배치하자.
그다음, **MaxAmmo** 출력핀을 당기고 반환 노드의 **Return Value**에 연결하자. **인티저**를 **텍스트**로 변환하는 형변환 노드가 자동으로 배치될 것이다.



컴파일하고 저장하자.

10. 플레이해보자.

H 키와 **E** 키를 눌러보자. Health와 Energy의 진행바가 바뀔 것이다.
T 키를 눌러보자. 탄약수가 감소할 것이다.
R 키를 누르면 탄창이 재장전될 것이다.



이 절에서는 게임 내에서 캐릭터 정보를 표시하는 HUD 구현의 스크립팅 단계에 대해서 학습하였다.

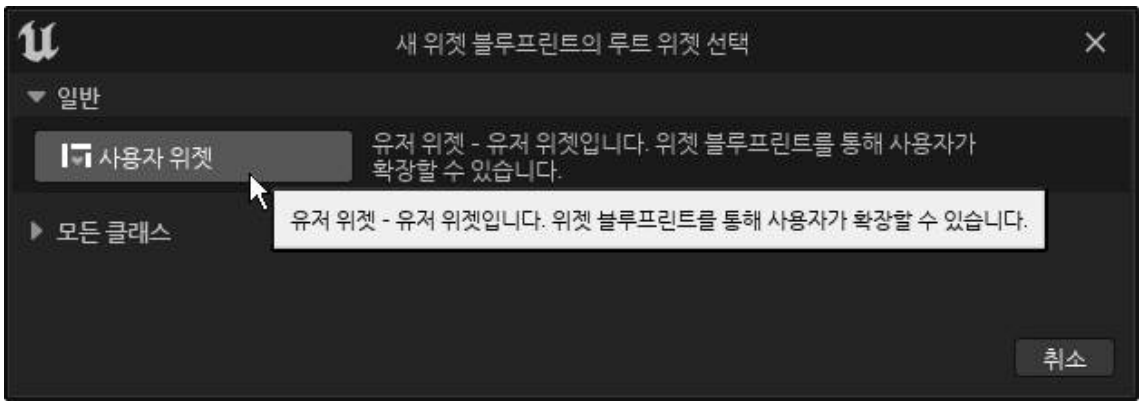
5. 메인 메뉴 구현의 레이아웃 배치 단계

이 절에서는 메인 메뉴 생성 방법에 대해서 학습한다.
메인 메뉴에서는 게임을 시작하고 정지할 수 있다. 또한 옵션 메뉴로 이동하여 해상도를 변경할 수도 있다.
메인 메뉴의 경우에도 레이아웃 배치 단계와 스크립팅 단계의 두 단계로 나누어서 진행해보자.
이제부터 예제를 통해서 학습해보자

---

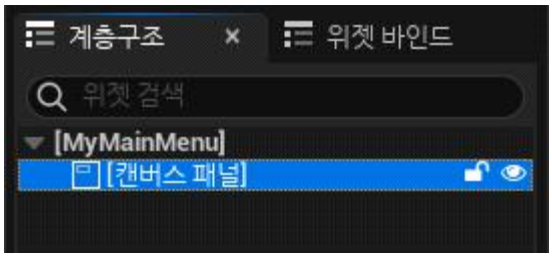
1. 이전 프로젝트 `Pumgfirst`에서 계속하자.

2. 메인 메뉴에 대한 위젯 블루프린트 애셋를 생성하자.
콘텐츠 브라우저에서 **+추가**를 누르고 **유저 인터페이스** » **위젯 블루프린트**를 선택하자.
그다음, **새 위젯 블루프린트의 루트 위젯 선택** 창에서 **일반** 아래의 **사용자 위젯**을 클릭하여 선택하자.
특별한 경우가 아니면 항상 **사용자 위젯**을 선택하면 된다.



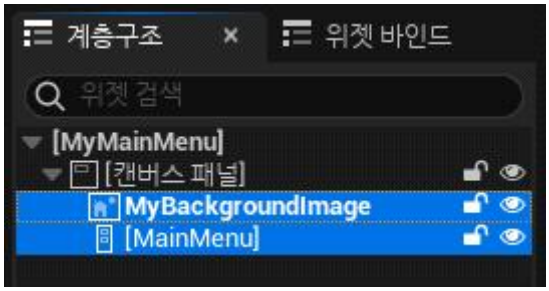
위젯이 생성될 것이다. 디폴트 이름인 **NewWidgetBlueprint**을 **MyMainMenu**로 수정하자.

3. 그다음, **MyMainMenu** 위젯 블루프린트를 더블클릭하여 에디터를 열자. 그리고, **팔레트** 탭에서 위젯을 찾아 **계층구조** 탭에 추가하는 작업을 시작하자.
먼저 **팔레트** 탭에서 **패널** 아래의 **캔버스 패널**을 드래그하여 아래의 **계층구조** 탭에서 루트 노드인 **[MyMainMenu]**에 드롭하여 추가하자.

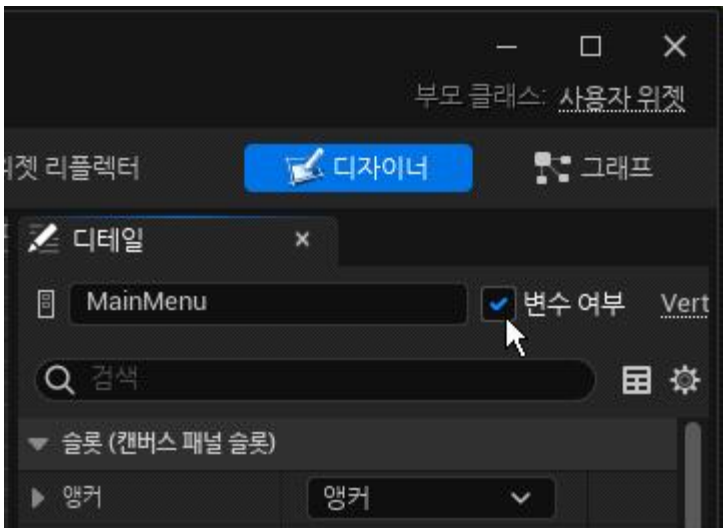


4. 그다음, **팔레트** 탭에서 **일반** 아래의 **이미지**(Image) 위젯을 드래그하여 **계층구조** 탭의 **[캔버스 패널]**에 추가하자.
추가된 이미지 위젯을 선택하고 **F2** 키를 눌러 이름을 **MyBackgroundImage**로 수정하자.
그다음, **팔레트** 탭에서 **패널** 아래의 **세로 박스**(Vertical Box) 위젯을 드래그하여 **계층구조** 탭의 **[캔버스 패널]**에 추가하자.

추가된 **[세로 박스]** 위젯을 선택하고 **F2** 키를 눌러 이름을 **MainMenu**로 수정하자. 이름을 명시해주면 이 위젯이 어떤 목적인지를 쉽게 구분할 수 있다.



5. **MainMenu**를 선택하고 디테일 탭을 보자.
MainMenu 이름 옆에 **변수 여부** 체크박스가 있다.
변수 여부 체크박스에 체크해서 변수로 사용할 수 있도록 설정하자. 변수로 설정해두면 나중에 블루프린트에서 이 위젯에 접근할 수 있다.



그리고, 디**테일** 탭에서 **슬롯(캔버스 패널 슬롯)** 영역 아래에 **Z 순서(ZOrder)** 속성이 있다.
이 **Z 순서** 속성값을 디폴트인 0에서 1로 수정하자. 이 속성은 위젯이 렌더링되는 순서를 나타낸다. 값이 높을수록 나중에 렌더링되고 따라서 가장 위에 나타난다.
참고로, Z값을 변경하여 보이는 순서를 조정해도 되지만 목록에서의 위젯의 나열 순서를 조정해도 된다. 위젯은 목록에 나열된 순서대로 렌더링되기 때문이다.

6. 계속해서 위젯을 추가하자.
그다음, **팔레트** 탭에서 **일반** 아래의 **버튼**(Button) 위젯을 드래그하여 **계층구조** 탭의 **MainMenu**에 추가하자. 총 세 개의 버튼을 추가하자.
그리고, **MainMenu**에 포함된 세 버튼의 이름을 순서대로 **PlayButton**, **OptionsButton**, **QuitButton**으로 수정하자.
그다음, **MainMenu**와 각 세 개의 **Button**에 대해서 하나씩의 **텍스트**(Text)를 추가하자. **텍스트** 위젯은 **팔레트** 탭에서 **일반** 영역 아래에 있다.
그 다음, **MainMenu**에 추가된 **텍스트**는 가장 마지막에 추가되는데, 이 **텍스트**를 드래그하여 **MainMenu**의 바로 아래에 배치되도록 하자.

그리고 **MainMenu**와 각 버튼에 달려있는 **텍스트** 위젯의 **텍스트**(Text) 속성값도 '`My Game`', '`Play Game`', '`Options`', '`Quit`'으로 수정하자.
이제 아래와 같은 모양이 될 것이다.

7. 그다음, 계층구조 탭에서 **MainMenu**를 선택하고 **Ctrl+C**를 누르자. 그다음, **[MyMainMenu]**를 누르고 **Ctrl+V**를 눌러 복제하자.
그리고, 복제된 마지막 **Button**인 **QuitButton_1**을 클릭하고 **Ctrl+C**를 누르고 **Ctrl+V**를 눌러 버튼 하나를 추가로 복제하자.

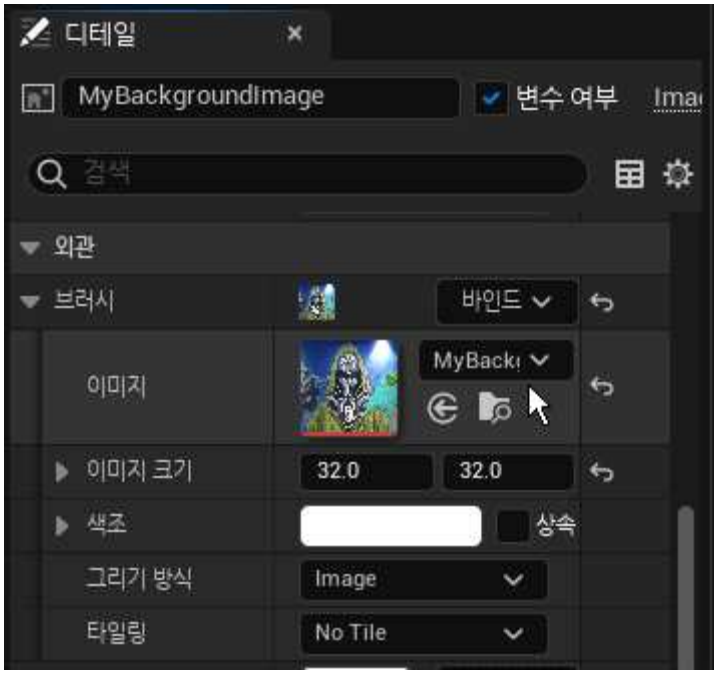
8. 이제 복제된 위젯들의 속성을 수정하자.

먼저, 복제된 **MainMenu_1**의 이름을 **OptionsMenu**로 수정하자.
그리고, **OptionsMenu**의 디테일 탭에서 **변수 여부**의 체크박스에 체크하자.
그리고, **OptionsMenu**에 포함된 네 버튼의 이름을 순서대로 **Setting1**, **Setting2**, **Setting3**, **ReturnButton**으로 수정하자. 그리고 **OptionsMenu**와 각 버튼에 달려있는 **Text** 위젯의 **Text** 속성값도 '`Resolution`', '`640x480`', '`1280x720`', '`1920x1080`', '`Return`'으로 수정하자.

9. 이제 버튼의 속성을 설정하자.
Ctrl 키를 누른 상태에서 모든 7개의 버튼을 클릭하여 선택하자.
그다음, 디테일 탭에서 **슬롯 (세로 박스 슬롯)** 영역의 **크기(Size)** 속성값을 **자동**에서 **채우기**로 수정하자. **채우기**로 지정하면 사용 가능한 모든 공간을 차지하게 된다.
그다음, **디테일** 탭에서 **외관** » **스타일(Style)** » **호버(Hovered)** 영역 아래의 속성값을 지정하자. 이 영역은 커서가 버튼 위에 올라갔을 때의 버튼의 외형을 설정한다. **색조(Tint)** 속성값을 디폴트인 어두운 회색 (0.7,0.7,0.7,1)에서 (1,1,0,1)로 지정하여 노란색이 되도록 하자.

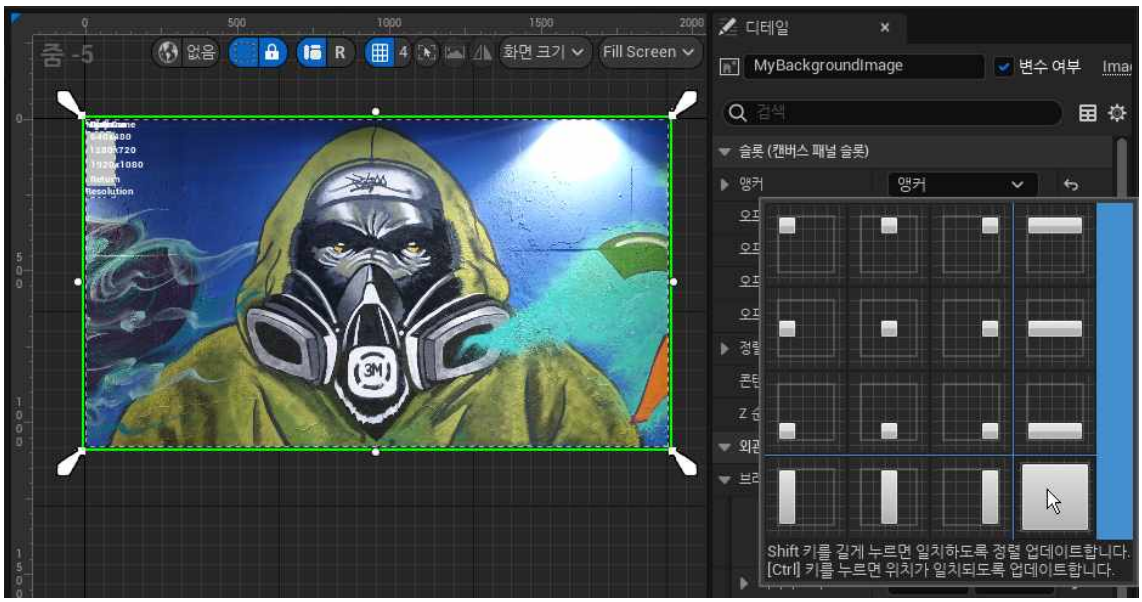
10. 이제 이미지를 설정하자.

계층구조 탭에서 **MyBackgroundImage**를 선택하고 **디테일** 탭에서 **외관** » **브러시(Brush)** 영역의 **이미지(Image)**에서 **없음**을 클릭하고 **MyBackgroundImage**를 선택하여 지정하자. 이 속성에는 **텍스처** 애셋 뿐만 아니라 **머티리얼** 애셋을 지정해도 된다.

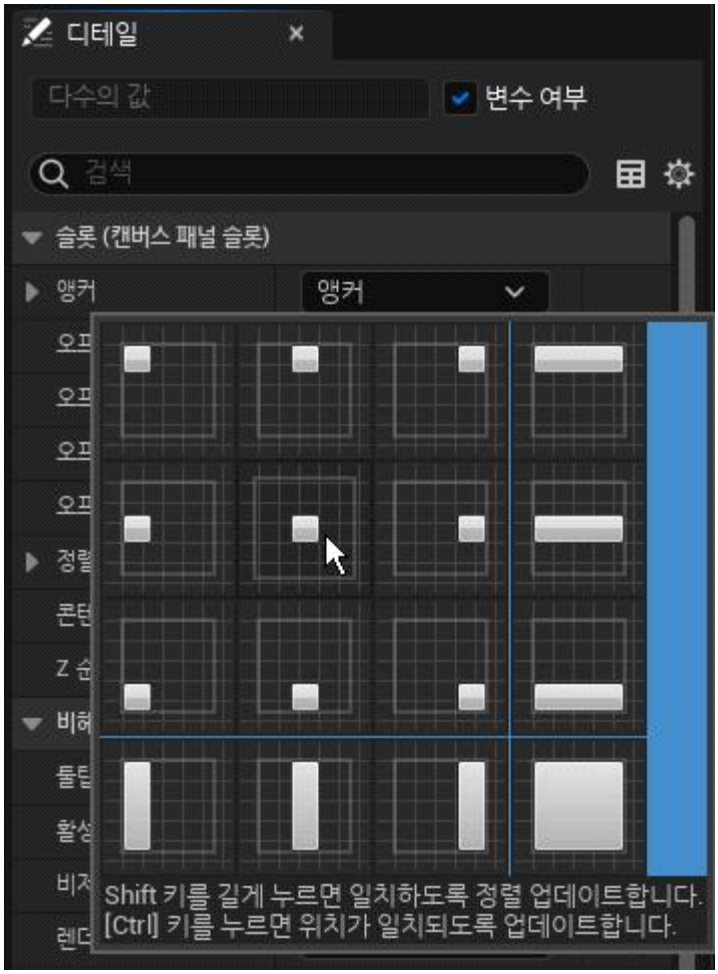


11. 그다음, 디자이너 모드의 격자탭에서 이미지가 전체 레이아웃에 맞도록 마우스로 드래그하여 크기를 크게 조절하자.

그다음, **Image** 위젯의 디테일 탭에서 슬롯 영역의 앵커 속성을 클릭하고 우측하단의 화면 채우기 옵션을 선택하자. **Scale Box**를 사용하여 이미지를 담으면, 종횡비에 맞게 스케일과 크기가 자동 조절되도록 할 수 있다.



12. 그다음, 계층구조에서 **MainMenu**와 **OptionsMenu**의 두 개의 **Vertical Box**를 모두 선택하자. 그리고 디테일 탭에서 **슬롯 (캔버스 패널 슬롯)** 영역에 있는 **앵커** 위치를 수직 수평 방향 모두 중앙으로 설정하자. 왼쪽에서 두 번째이고 위에서 두 번째의 격자를 클릭하면 된다.

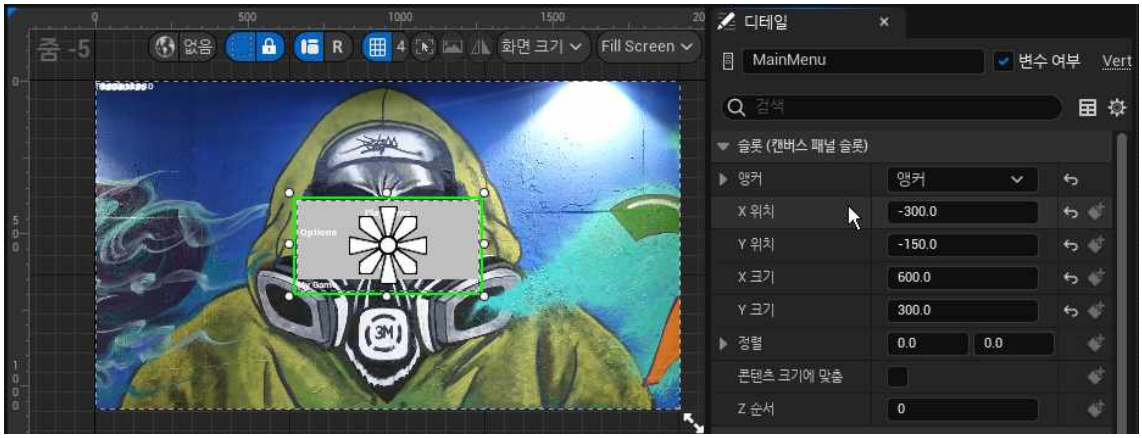


13. 그다음, **계층구조**에서 **MenuMenu**를 선택하고 디자이너 탭에서 적당한 크기와 위치로 배치하자. 그리고, **OptionsMenu**도 크기와 위치를 조정하자.

MainMenu와 **OptionsMenu**의 두 개 모두에 대해서, 디테일 탭에서 X,Y 위치에 각각 -300,-150을 입력하고 X,Y 크기에 600,300을 입력해도 좋다.

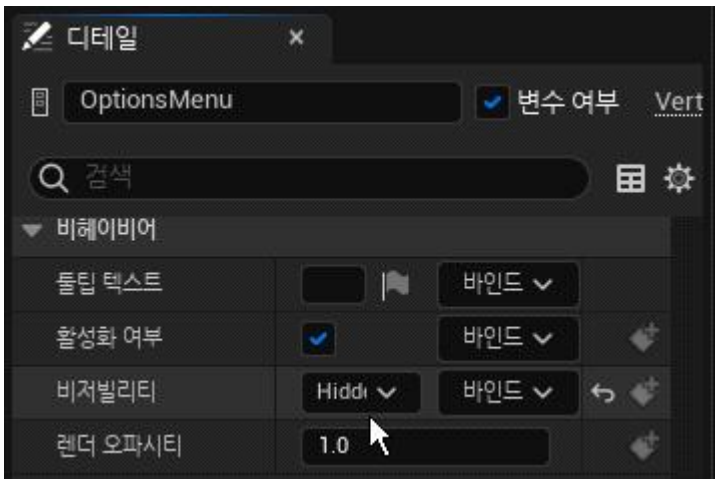
이 두 **Vertical Box** 위젯은 화면 중간에 동일한 위치에 크게 잘 보이게 배치하면 된다. 플레이 시에는 한번에 하나만 보일 것이므로 겹치더라도 중간에 잘 보이게 두는 것이 좋다.

한편, 배치할 때에 여러 위젯이 겹쳐서 특정 위젯을 선택해서 드래그하는 것이 어려울 때에는 계층구조 탭



에서 자물쇠 아이콘 기능과 눈 아이콘 기능을 활용하면 된다.

14. 그다음, **MainMenu**를 선택하고 **비헤이비어** 영역의 **비저빌리티(Visibility)** 속성이 **Visible**이 되도록 두자. 그리고 **OptionsMenu**를 선택하고 **비헤이비어** 영역의 **비저빌리티** 속성이 **Hidden**이 되도록 수정하자. **Visible**이면 화면에 보이도록 표시되고 **Hidden**이면 화면에서 보이지 않도록 숨긴다. 즉, 디폴트로 메인 메뉴는 기본적으로 보이도록 하고, 옵션 메뉴는 보이지 않도록 숨긴다. 이들 속성값은 스크립트를 실행하여 바뀌게 할 수 있다.



15. 그다음, **MainMenu**와 **OptionsMenu**의 문자열이 '`My Game`'과 '`Resolution`'으로 되어 있는 두 **Text** 위젯을 선택하자. 그리고 디테일 탭에서 설정하자.

먼저, **슬롯 (세로 박스 슬롯)** 영역에서 **가로 정렬(Horizontal Alignment)**를 두 번째 옵션인 중간 정렬로 바꾸자.

그다음, **외관** 영역에서 **폰트** 영역에서 **크기(Size)**를 디폴트인 24에서 36으로 바꾸자.

그다음, **외관** 영역에서 **컬러 및 오파시티**를 디폴트인 (1,1,1,1)에서 (1,0,0,1)으로 바꾸자.

16. 그다음, 버튼 아래의 모든 7개의 **Text** 위젯을 선택하자. 그리고 디테일 탭에서 설정하자.
먼저, **슬롯 (세로 박스 슬롯)** 영역에서 **가로 정렬(Horizontal Alignment)**를 두 번째 옵션인 중간 정렬로 바꾸자.
그다음, **외관** 영역에서 **컬러 및 오파시티**를 디폴트인 (1,1,1,1)에서 (0,0,1,1)으로 바꾸자.

이제 메인 메뉴를 위한 레이아웃 배치 단계가 완료되었다.

이 절에서는 메인 메뉴 구현의 레이아웃 배치 단계에 대해서 학습하였다.

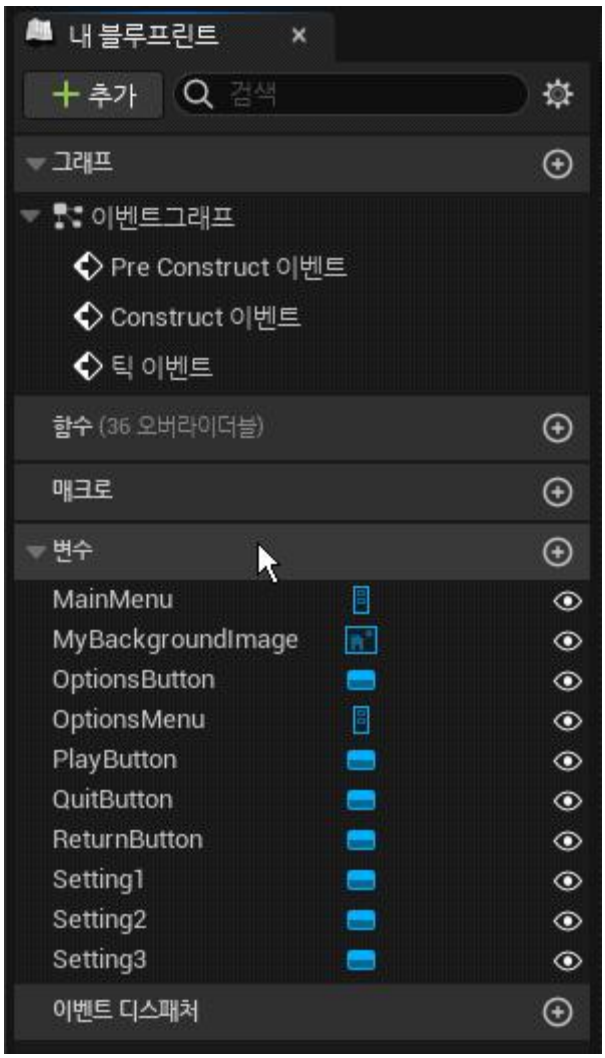
6. 메인 메뉴 구현의 스크립팅 단계

이 절에서는 메인 메뉴 생성 방법에 대해서 학습한다.
메인 메뉴의 레이아웃 배치 단계에 이어서 스크립팅 단계를 진행해보자.
이제부터 예제를 통해서 학습해보자

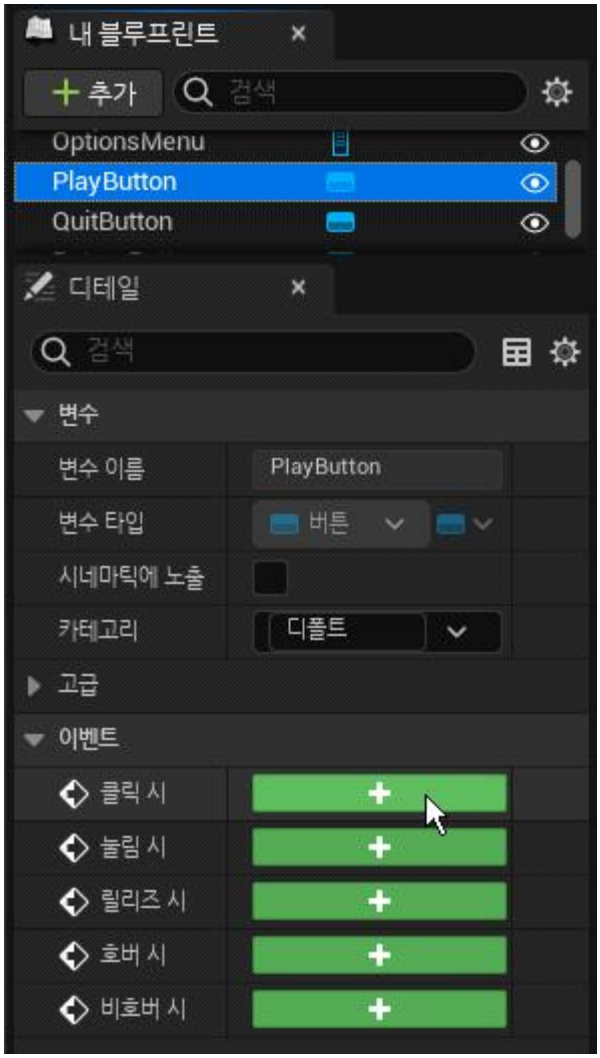
---

1. 이전 프로젝트 `Pumgfirst`에서 계속하자.

2. 콘텐트 브라우저에서 **MyMainMenu** 위젯 블루프린트를 더블클릭하여 에디터를 열자. 그리고, 에디터의 우측상단의 **그래프** 탭을 클릭하여 그래프 모드로 변경하자.
지금부터 각 버튼을 클릭할 때에 실행할 스크립트에 대한 이벤트 그래프를 작성하자.
먼저, **PlayButton**의 그래프를 작성하자.
우리가 이전에 **디자이너** 모드에서 위젯의 **디테일** 탭에서의 **변수 여부** 속성의 체크박스에 체크한 위젯들이 있다. 이들 위젯은 스크립트에서 참조할 수 있도록 변수로 추가된다. **내 블루프린트** 탭에서의 **변수** 영역을 살펴보자.



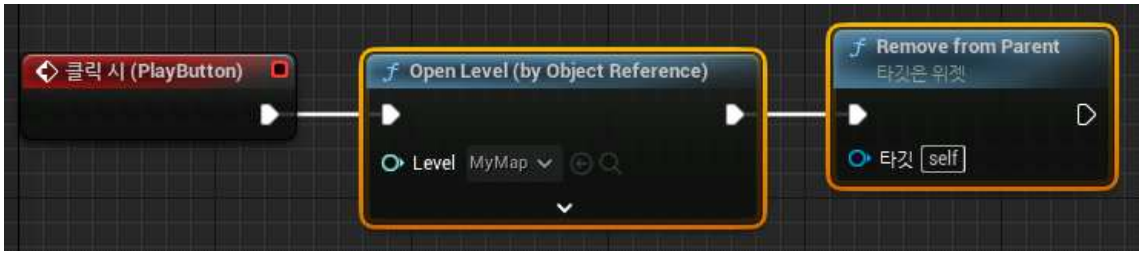
3. 먼저, **변수** 영역에서 **PlayButton**을 선택하자. 그다음, 디테일 탭에서 이벤트 영역의 **클릭 시(OnClicked)**를 선택하여 버튼이 클릭될 때에 실행되는 이벤트 노드를 추가하자.



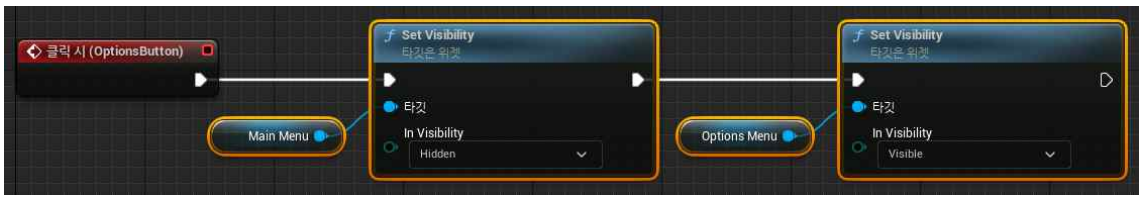
4. 그다음, 배치된 **OnClicked (PlayButton)** 노드를 당기고 **Open Level (by Object Reference)** 노드를 배치하자. 그리고 **Level** 입력핀에 **MyMap**을 선택하여 지정하자.

여기서, **Open Level** 함수는 새로운 레벨을 여는 함수이다. 레벨의 이름을 문자열로 명시하는 **Open Level (by Name)** 노드가 있고 선택 가능한 레벨 애셋들을 나열하고 목록에서 선택해서 지정하는 **Open Level (by Object Reference)** 노드가 있다. 어느 것을 사용하든 상관없다.

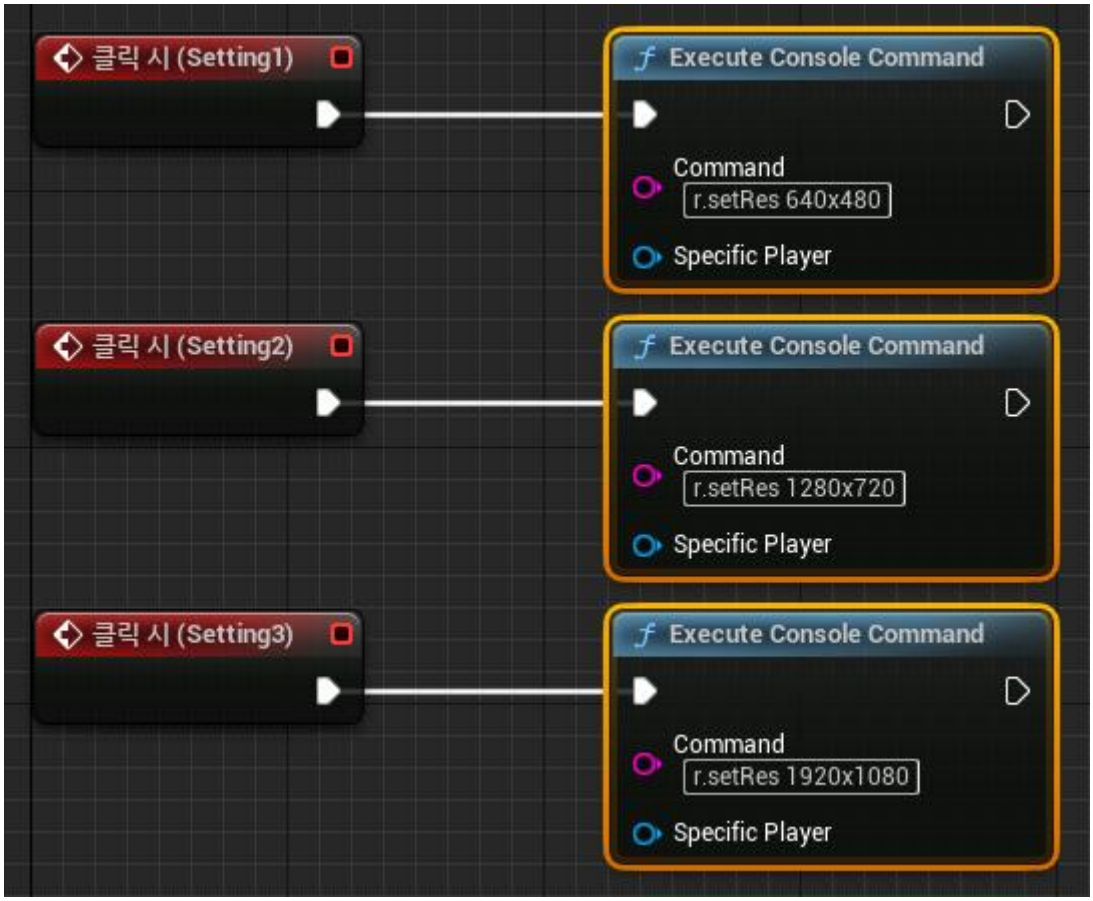
그다음, 레벨이 로드된 이후에는 뷰에서 메인 메뉴를 제거하자. **Open Level** 노드를 당기고 **Remove from Parent** 노드를 배치하자. 이 노드는 뷰포트에서 인자로 주어진 타깃 위젯 블루프린트를 제거한다. 타깃이 셀프로 지정되저 있으므로 현재 애셋인 **MyMainMenu**를 의미한다.



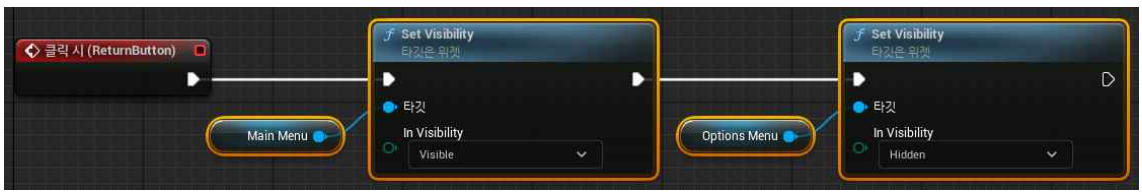
5. 다음으로, **OptionsButton**이 클릭되면 실행할 그래프를 작성하자.
이 옵션 버튼을 클릭하면 메인 메뉴를 끄고 옵션 메뉴를 켜도록 그래프를 작성하자.
변수 **OptionsButton**를 선택하고 디테일 탭에서 **클릭 시**(**OnClicked**) 이벤트 노드를 추가하자.
그다음, **MainMenu** 변수의 **Get** 노드를 배치하자. 그리고 **Get** 노드를 드래그하고 **Set Visibility** 노드를 추가하자. 그리고 노드의 **In Visibility** 입력핀을 **Visible**에서 **Hidden**으로 수정하자.
그다음, 변수 **OptionsMenu**에 대해서도 동일한 방법으로 **Get** 노드를 배치하고 **Set Visibility** 노드를 추가하자. 이번에는 노드의 **In Visibility** 입력핀을 **Visible**로 두자.
그다음, 실행핀을 모두 연결하자.



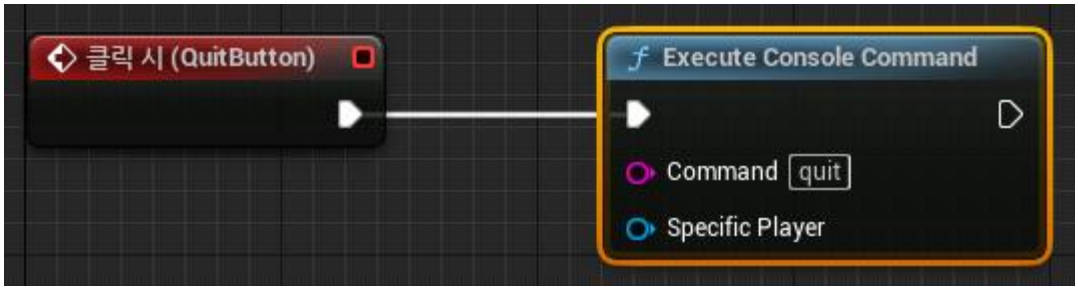
6. 다음으로, **Setting1**, **Setting2**, **Setting3** 버튼이 클릭되면 실행할 그래프를 작성하자. 이 버튼이 클릭되면 콘솔 명령어를 실행하여 해상도를 바꾸도록 하자.
먼저, 변수 **Setting1**, **Setting2**, **Setting3**를 하나씩 선택하고 디테일 탭에서 **클릭 시**(**OnClicked**) 이벤트 노드를 추가하자.
그다음, 각 이벤트 노드를 드래그하고 콘솔 명령어를 실행하는 함수인 **Execute Console Command** 노드를 배치하자. 노드의 **Command** 입력핀에 해상도를 수정하는 명령어 문자열을 입력하자. 해상도를 수정하는 콘솔 명령어는 '`r.setRes 640x480`', '`r.setRes 1280x720`', '`r.setRes 1920x1080`' 으로 입력하면 된다.



7. 다음으로, **ReturnButton**이 클릭되면 실행할 그래프를 작성하자. 이 옵션 버튼을 클릭하면 메인 메뉴를 켜고 옵션 메뉴를 끄도록 그래프를 작성하자. **OptionsButton** 버튼의 그래프에서와 유사하게 작성하면 된다.



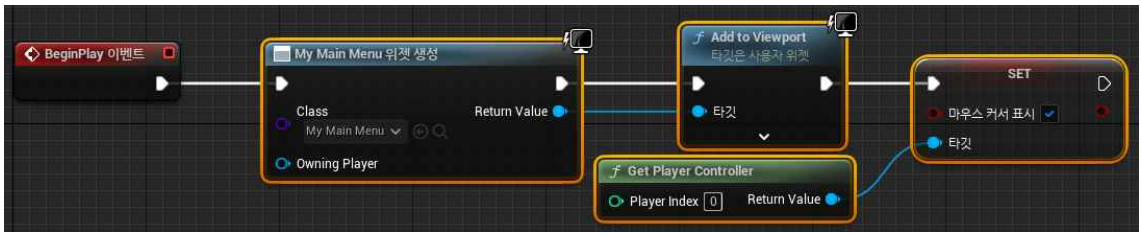
8. 다음으로, **QuitButton**이 클릭되면 실행할 그래프를 작성하자. 이 버튼이 클릭되면 콘솔 명령어를 실행하여 게임을 종료하도록 하자.
먼저, 변수 **QuitButton**를 선택하고 디테일 탭에서 **OnClicked** 이벤트 노드를 추가하자.
그다음, 이벤트 노드를 드래그하고 콘솔 명령어를 실행하는 함수인 **Execute Console Command** 노드를 배치하자. 노드의 **Command** 입력핀에 게임을 종료하는 콘솔 명령어인 '`quit`'을 입력하자.



이제 모든 이벤트 그래프를 작성하였다.
컴파일하고 저장하자.

9. 메인 메뉴는 게임 플레이 레벨이 아니라 별도의 다른 레벨에서 보여지도록 할 것이다. 메인 메뉴를 나타나게 할 새 레벨을 만들자.
레벨 에디터로 가자. 메뉴바에서 **파일** » **새 레벨**을 클릭하고 템플릿으로 **빈 레벨**을 선택하여 새 레벨을 만들자. 저장 버튼을 클릭하고 레벨의 이름은 **MyMainMenuMap**로 지정하자.

10. 그리고 레벨 에디터에서 툴바의 **블루프린트** » **레벨 블루프린트 열기**를 선택하여 **MyMainMenuMap**의 레벨 블루프린트 에디터를 열자.
BeginPlay 이벤트 노드를 당기고 **Create Widget**를 검색하여 **위젯 생성** 노드를 배치하자.
그다음, 배치된 노드의 **Class** 입력핀을 클릭하고 위젯 블루프린트 목록에서 **MyMainMenu**를 선택하자.
그다음, **위젯 생성** 노드의 **Return Value** 출력핀을 당기고 **Add to Viewport** 노드를 배치하자.
그다음, 빈 곳에 **GetPlayerController** 노드를 추가하자. 그리고 노드의 **Return Value** 출력핀을 당기고 **Set ShowMouseCursor** 노드를 배치하자. 그리고 **마우스 커서 표시**(**ShowMouseCursor**) 입력핀의 체크박스에 체크하자.
그다음, 앞의 **Add to Viewport** 노드와 실행핀을 연결하자.



컴파일하고 저장하자.

11. 이제 메인 메뉴 표시를 위한 새 레벨을 완성하였다.
새 레벨에는 게임 모드를 따로 설정하지 않았으므로 우리의 커스텀 게임 모드가 아니라 엔진의 디폴트 게임 모드가 적용된다.
따라서 메인 메뉴 레벨이 생성될 때에는 우리의 커스텀 플레이어 캐릭터가 생성되지 않는다.
컴파일하고 저장하자.

12. 메인 메뉴 레벨이 로드된 상태에서 플레이해보자. 메인 메뉴가 나타날 것이다. 모든 메뉴가 잘 동작되는지 확인해보자.

13. 입력 모드에 대해서 알아보자.

입력 모드의 종류에는 **Game Only**, **UI Only**, **Game And UI**가 있다.

게임 전용으로 설정하면 플레이어 입력이나 플레이어 컨트롤러만 사용자 입력에 반응하도록 허용한다. 이렇게 하면 UI를 위한 커서가 보이지 않게 되고 UI 위젯은 사용자 입력에 반응하지 않는다.

UI 전용으로 설정하면 입력을 플레이어 컨트롤러나 플레이어 입력 시스템으로 보내지 않는다. 그리고 마우스 커서를 보여준다.

실제 게임플레이 레벨에서는 메뉴가 사용자 입력에 반응하지 않도록 해야 한다.

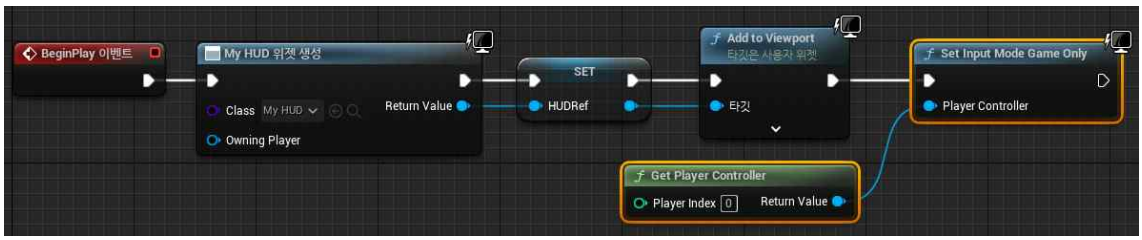
우리는 플레이어 액터에 대해서 입력 모드를 게임 전용으로 설정하자.

먼저, 콘텐츠 브라우저에서 **BP_MyCharacter**를 더블클릭하여 블루프린트 에디터를 열자.

그리고, **BeginPlay** 이벤트 노드의 마지막에서 빈 곳에서 우클릭하고 **Get Player Controller** 노드를 추가하자.

그다음, 추가된 노드의 **Return Value** 출력핀을 당기고 **Set Input Mode Game Only** 노드를 추가하자.

그다음, 기존의 이벤트 그래프의 마지막 노드와 실행핀을 연결하자.



14. 이제 **MyMainMenuMap** 레벨과 **MyMainMenu** 위젯 블루프린트를 사용한 메인 메뉴 구성이 모두 완료되었다.

이 절에서는 메인 메뉴 구현의 스크립팅 단계에 대해서 학습하였다.

7. 일시정지 메뉴 구현의 레이아웃 배치와 스크립팅

이 절에서는 일시정지 메뉴 생성 방법에 대해서 학습한다.

이전 절에서 게임 시작 시에 보이는 메인 메뉴는 완료하였다. 그러나 게임 플레이가 시작된 후에 뜨는 메뉴가 없다. 이제부터 게임중 플레이어가 게임을 일시정지하거나 게임을 종료하거나 메인 메뉴로 갈 수 있는 일시정지 메뉴를 추가해보자.

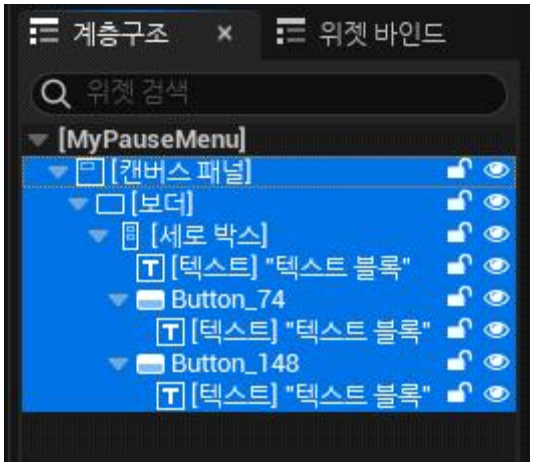
일시정지 메뉴의 레이아웃 배치 단계와 스크립팅 단계를 진행해보자.
이제부터 예제를 통해서 학습해보자

---

1. 이전 프로젝트 Pumgfirst에서 계속하자.

2. 이제부터, 일시정지 메뉴의 레이아웃 배치에 대해서 진행해보자.
먼저 일시정지 메뉴에 대한 위젯 블루프린트 애셋를 생성하자.
콘텐츠 브라우저에서 **+추가**를 누르고 **유저 인터페이스** » **위젯 블루프린트**를 선택하자.
그다음, **새 위젯 블루프린트의 루트 위젯 선택** 창에서 **일반** 아래의 **사용자 위젯**을 클릭하여 선택하자.
생성된 위젯의 이름을 **MyPauseMenu**로 수정하자.

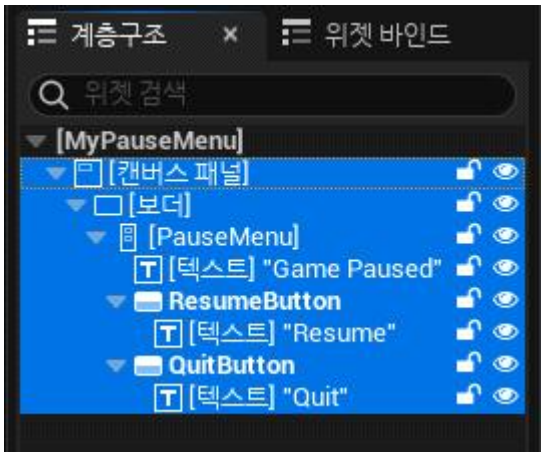
3. 그다음, **MyPauseMenu** 위젯 블루프린트를 더블클릭하여 에디터를 열자. 그리고, **팔레트** 탭에서 위젯을 찾아 **계층구조** 탭에 추가하는 작업을 시작하자.
먼저, **패널** 아래의 **캔버스 패널(Canvas Panel)** 위젯을 **계층구조** 탭의 **[MyPauseMenu]**에 추가하자.
그다음, **일반** 아래의 **보더(Border)** 위젯 하나를 **[캔버스 패널]**에 추가하자. **Border**는 테두리를 표시하면서 내부에 하나의 자식 위젯을 포함하는 위젯이다.
그다음, **패널** 아래의 **세로 박스(Vertical Box)** 위젯을 **[보더]**에 추가하자.
그다음, **일반** 아래의 **버튼(Button)** 위젯을 **[세로 박스]**에 추가하자. 또 하나를 더 추가하여 총 2개의 버튼을 추가하자.
그리고, **일반** 아래의 **텍스트(Text)** 위젯을 **[세로 박스]**와 두 개의 버튼에 각각 추가하자.
[세로 박스]에 추가된 텍스트는 그 위치를 **[세로 박스]**의 바로 아래에 있도록 드래그하여 위치를 수정하자.
이제 **계층구조**는 다음과 같은 모습이 될 것이다.



4. 위젯의 이름을 적절히 수정하자.

계층구조 탭에서 **Vertical Box** 위젯을 선택하고 **F2** 키를 눌러 이름을 **PauseMenu**로 수정하자. 그리고, 그 아래의 **Text** 위젯의 디테일 탭에서의 **콘텐츠** 영역 아래의 **Text** 속성값을 '`Game Paused`'로 지정하자. 그다음, 두 버튼 위젯의 이름을 각각 **ResumeButton**과 **QuitButton**으로 수정하자.

그다음, 두 버튼 아래에 있는 각 **Text** 위젯의 **디테일** 탭에서의 **콘텐츠** 영역 아래의 **Text** 속성값을 '`Resume`'과 '`Quit`'으로 지정하자.



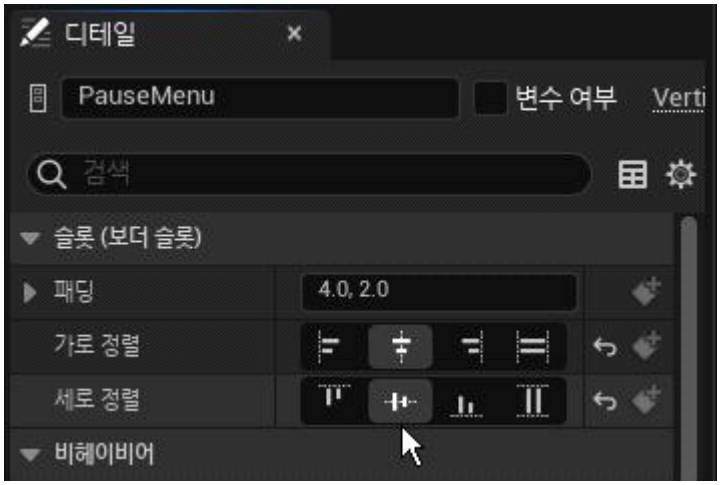
5. 디자이너 탭에서 **Border** 위젯의 크기를 조절하자.

먼저, 디테일 탭에서 **슬롯 (캔버스 패널 슬롯)** 영역의 **앵커**를 클릭하고 전체 화면을 채우도록 가장 우측 하단의 아이콘을 클릭하자. 이제 스크린 해상도와 상관없이 **Border** 위젯이 전체 화면을 가득 채우게 된다.

그다음, 점선으로 표시된 **Canvas Panel** 위젯의 전체 영역을 채우도록 크기를 조절하자. 또는 디테일 탭에서 **슬롯 (캔버스 패널 슬롯)** 영역의 오프셋 좌측,상단,우측,하단에 모두 0,0,0,0을 입력해도 된다.

그다음, 디테일 탭에서 외관 영역의 **브러시 컬러(Brush Color)**를 클릭하고 밝은 청록색이 되도록 RGB A를 디폴트 (1,1,1,1)에서 (0.5,1,1,0.5)로 지정하자. 여기서 컬러의 알파값을 0.5로 설정하여 약간 반투명하게 지정하였다. 이렇게 반투명하게 설정하면 게임 도중에 일시정지 메뉴가 열리더라도 메뉴가 반투명하게 보이면서 뒤의 게임 진행 모습을 계속 확인할 수 있게 된다.

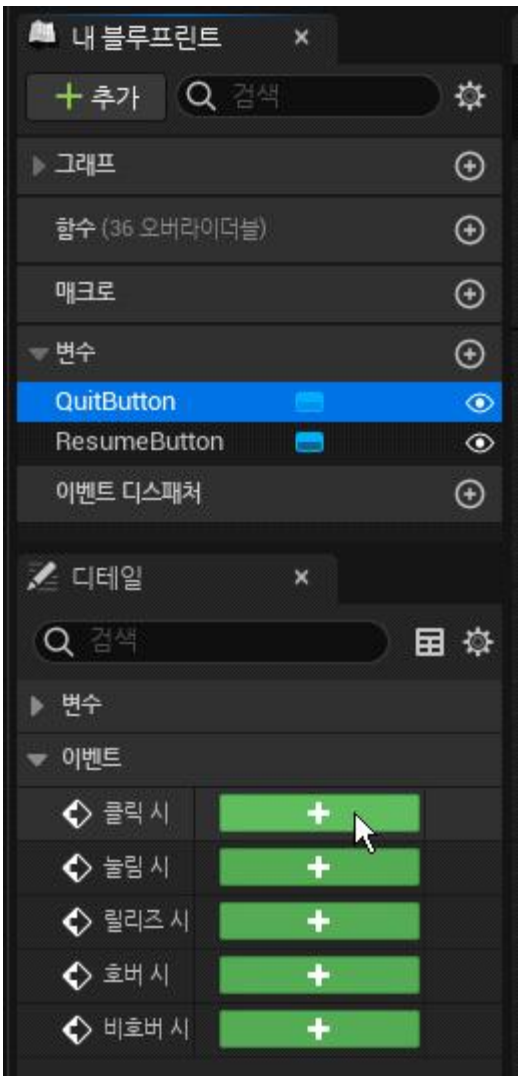
6. 이제 계층구조 탭에서 **[PauseMenu]** 위젯을 선택하자.
디테일 탭에서 **슬롯** 영역의 **가로 정렬**(**Horizontal Alignment**)과 **세로 정렬**(**Vertical Alignment**)을 모두 중앙 정렬이 되도록 왼쪽에서 두 번째 아이콘을 선택하자.



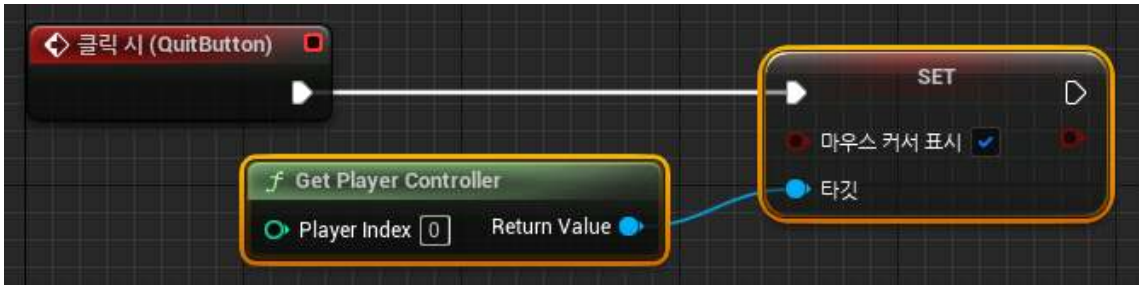
7. 두 버튼 위젯을 모두 선택하자.
디테일 탭에서 **외관** » **스타일** » **호버(Hovered)** 아래의 **색조(Tint)**에 노란색이 되도록 RGBA를 (1,1,0,1)로 지정하자. 이것은 커서가 버튼 위에 위치할 때의 버튼 컬러이다.

컴파일하고 저장하자.
이제 일시정지 메뉴의 레이아웃 배치 단계를 모두 완료하였다.

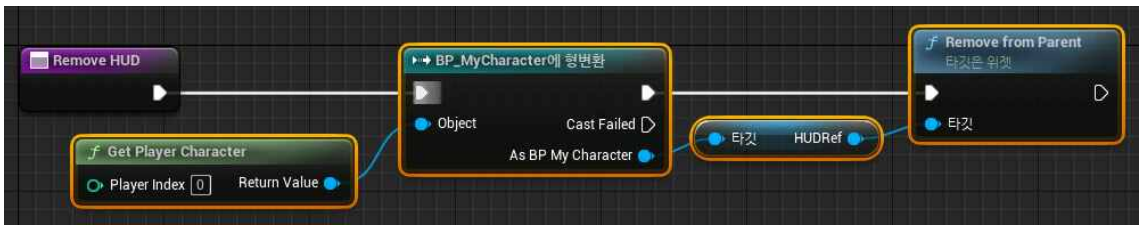
8. 이제부터 일시정지 메뉴의 스크립팅 단계를 진행해보자.
먼저, 위젯 블루프린트 에디터의 우측 상단의 그래프 버튼을 클릭하여 **그래프** 모드로 전환하자.
그다음, **내 블루프린트** 탭에서 변수 영역의 **QuitButton**을 선택하고 디테일 탭에서 이벤트 영역의 **클릭 시(OnClicked)**를 클릭하자. **OnClicked (QuitButton)** 이벤트 노드가 추가될 것이다.



9. 이제, **클릭 시(OnClicked) (QuitButton)** 이벤트 그래프를 작성하자.
먼저, 격자맵의 빈 곳에서 우클릭하고 **GetPlayerController** 노드를 추가하자.
그다음, **GetPlayerController** 노드의 **Return Value**를 드래그하고 **Set Show Mouse Cursor** 노드를 배치하자. **ShowMouseCursor** 입력핀은 **true**가 되도록 체크하자. 메인메뉴가 표시되는 레벨로 이동할 것이므로 메인 메뉴를 조작할 수 있도록 커서가 표시되도록 한다.
실행핀을 연결하자. 이제 아래과 같이 보일 것이다.



10. 계속해서 **MyPauseMenu** 위젯 블루프린트 에디터의 에벤트 그래프 탭에서 작업하자.
더 진행하기 전에 함수 하나를 새로 만들자.
이 함수는 플레이어 뷰에서 게임 HUD를 제거하는 기능을 수행하도록 구현할 것이다.
먼저, **내 블루프린트** 탭에서 **함수** 영역의 **+**를 클릭하자. 생성된 함수 이름을 **RemoveHUD**로 수정하자.
그다음, 함수 그래프에서, 격자맵의 빈 곳에서 **GetPlayerCharacter** 노드를 추가하자.
그다음, **GetPlayerCharacter** 노드의 **Return Value**를 당기고 **BP_MyCharacter에 형변환** 노드를 배치하자.
이제 캐릭터 블루프린트에 접근할 수 있다.
그다음, 형변환 노드의 **As BP_MyCharacter** 출력핀을 당기고 **Get HUDRef** 노드를 배치하자. 이제 캐릭터 블루프린트 내에 HUD 위젯 블루프린트의 레퍼런스에 접근할 수 있다.
그다음, **HUDRef**의 출력핀을 당기고 **Remove from Parent** 노드를 배치하자. **Target**이 플레이어 캐릭터에 사용중인 HUD 위젯 블루프린트이므로 뷰포트에서 이 HUD를 제거한다. 함수 노드의 실행핀을 이후의 노드들에 연결하자. 이제 함수가 다음과 같은 모습으로 완성되었다. 컴파일하고 저장하자.



참고로, **MyHUD** 위젯 블루프린트는 **BP_MyCharacter**의 **BeginPlay** 이벤트 그래프에서 생성된다. 따라서 디폴트 폰이 **BP_MyCharacter**인 레벨이 시작될 때마다 플레이어 캐릭터와 더불어 함께 생성된다.

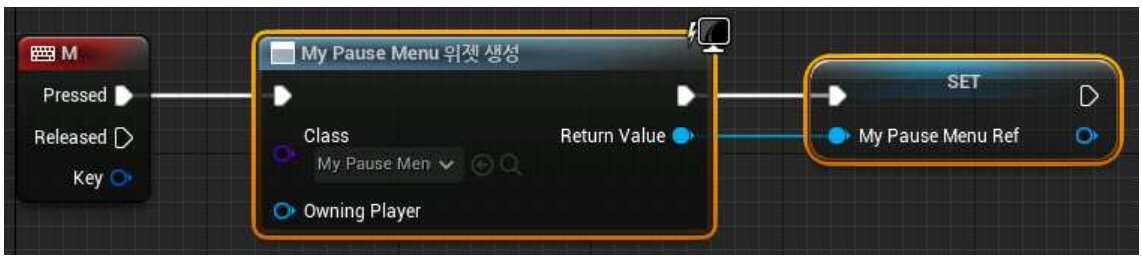
11. 계속해서 **클릭 시(OnClicked) (QuitButton)** 이벤트 그래프를 작성하자.
Set 노드 뒤에 위에서 만든 **RemoveHUD** 함수 노드를 추가하자.
그다음, **Open Level (by Object Reference)** 노드를 추가하자. 노드의 **Level** 입력핀을 클릭하고 목록에서 **MyMainMenuMap**을 선택하여 지정하자. 이제 메인 메뉴가 표시되는 레벨을 연다.
그다음, **Remove from Parent** 노드를 배치하자. 이 노드는 **Target**이 **self**이므로 뷰포트에서 자기 자신인 **MyPauseMenu** 위젯 블루프린트를 제거한다.
그다음, **Set GamePaused** 노드를 배치하자. **Paused** 입력핀을 **false**로 두어 더이상 일시정지가 아님을 표시하자.

12. 이제부터 **ResumeButton**의 그래프를 작성해보자.
이 버튼을 클릭하면 일시정지 메뉴를 종료하고 원래의 레벨로 돌아간다.

먼저, 내 블루프린트 탭에서 변수 영역의 **ResumeButton**을 선택하고 디테일 탭에서 **이벤트** 영역의 **클릭 시(OnClicked)**를 클릭하자. **클릭 시(OnClicked) (ResumeButton)** 이벤트 노드가 추가될 것이다.
그다음, 격자맵의 빈 곳에서 우클릭하고 **GetPlayerController** 노드를 추가하자.
그다음, **GetPlayerController** 노드의 **Return Value**를 드래그하고 **Set Input Mode Game Only** 노드를 배치하자. 이 노드는 입력 모드를 게임 전용으로 설정한다.
그다음, 다시 **GetPlayerController** 노드의 **Return Value**를 드래그하고 **Set ShowMouseCursor** 노드를 배치하자. **ShowMouseCursor** 입력핀은 **false**가 되도록 체크해제로 두자. 이 노드는 일시정지 메뉴가 활성화될 때에 표시되기 시작하였던 커서가 다시 제거되도록 한다.
그다음, **Remove from Parent** 노드를 배치하고 이어서 **Set Game Paused** 노드를 배치하자. 마지막 두 노드는 **OnClicked (QuitButton)** 이벤트 그래프의 마지막 두 노드와 동일하다.
모든 노드의 실행핀을 순서대로 연결해주자.

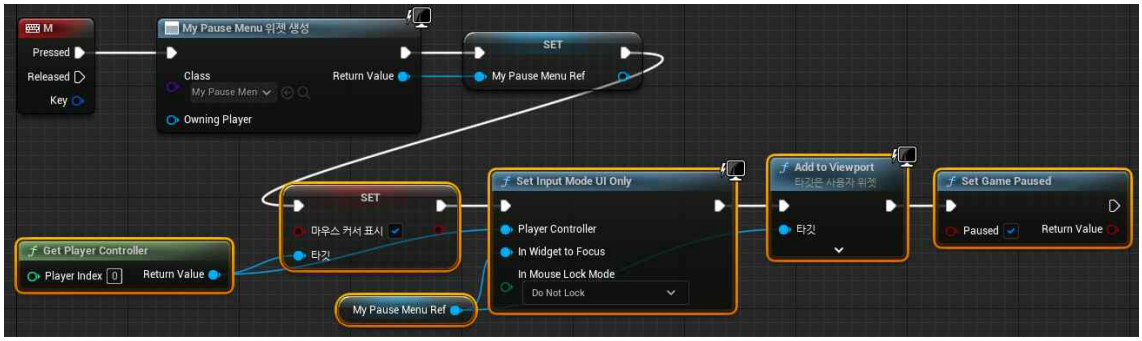
이제 두 버튼의 그래프를 모두 작성하였다.
컴파일하고 저장하자.

13. 이제 게임 플레이중에 플레이어가 특정 키를 누르면 게임이 일시정지되도록 하는 일이 남았다.
콘텐츠 브라우저에서 **BP_MyCharacter**를 더블클릭하여 블루프린트 에디터를 열자.
그다음, 격자맵의 가장 아래에서, 빈 곳에서 우클릭하고 '키 m'을 검색하여 **M** 키보드 이벤트 노드를 배치하자.
그다음, 노드의 **Pressed** 실행핀을 당기고 **Create Widget**을 검색하여 **위젯 생성** 노드를 배치하자. 노드의 **Class** 입력핀을 클릭하고 **MyPauseMenu**를 선택하여 지정하자.
그다음, **위젯 생성** 노드의 **Return Value** 출력핀을 당기고 변수로 승격을 선택하자. 생성된 변수 이름을 **NewVar_0**에서 **MyPauseMenuRef**로 수정하자. 이제 생성된 **MyPauseMenu** 위젯 블루프린트의 레퍼런스를 변수로 저장해두었다.

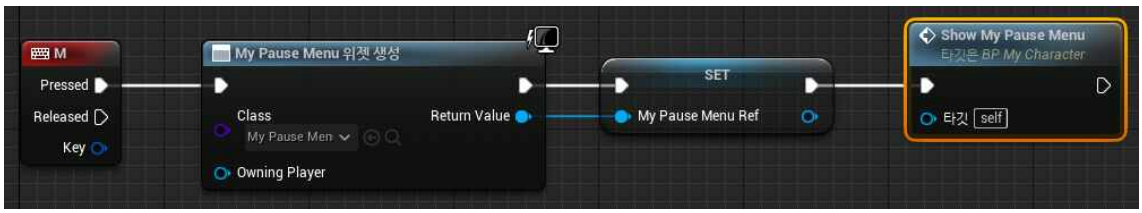


14. 이제, 격자판의 빈 곳에 **GetPlayerController** 노드를 배치하고 **Return Value** 출력핀을 당겨서 **Set ShowMouseCursor** 노드를 배치하자.
그리고, **ShowMouseCursor** 입력핀을 체크하여 **true**로 수정하자. 이제 마우스 커서가 보이도록 하여 메뉴를 클릭할 수 있도록 한다.
그다음, 변수 **MyPauseMenuRef**의 **Get** 노드를 배치하고 출력핀을 당겨서 **Set Input Mode UI Only** 노드를 배치하자. 그리고 **GetPlayerController** 노드의 출력핀을 **Set Input Mode UI Only** 노드의 **PlayerController** 입력핀에 연결하자. 이제 **MyPauseMenu**의 UI 위젯만 사용자 입력에 반응하도록 입력 모드를 구성하

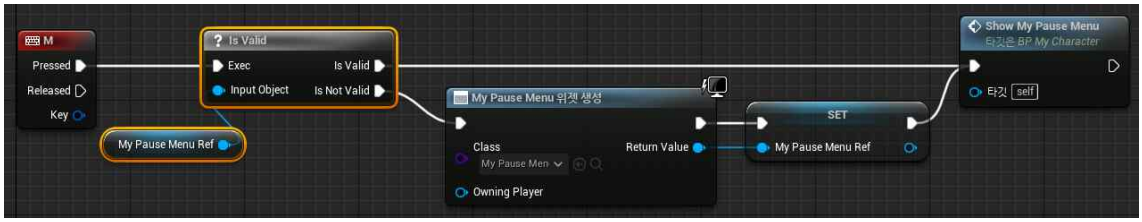
였다.
그다음, 이전의 **Set MyPauseMenuRef** 노드의 출력 실행핀을 **Set ShowMouseCursor** 노드에 연결하고 또 **Set Input Mode UI Only** 노드에 연결하자.
그다음, **Get MyPauseMenuRef** 노드의 출력핀을 당기고 **Add to Viewport** 노드를 배치하자. 그리고 이전의 **Set Input Mode UI Only** 노드의 실행핀을 연결하자.
그다음, **Add to Viewport** 노드의 실행핀을 당기고 **Set Game Paused** 노드를 배치하자. 그리고 **Paused** 입력핀에 체크하여 **true**가 되도록 하자.
이제 일시정지를 처음 실행하는 경우에 대하여 처리를 완료하였다.



15. 이제, 그래프에서 **Set MyPauseMenuRef** 노드 다음의 모든 노드들을 선택하자. 그다음, 우클릭하고 **함수로 접기**를 실행하자. 생성되는 함수의 이름을 **ShowMyPauseMenu**라고 수정하자.



16. 위의 이벤트 그래프를 살펴보자.
만약 **M** 키를 눌러 일시정지 메뉴로 갔다가 재개 버튼을 클릭하여 다시 돌아오는 것을 반복하는 경우를 생각해보자. 처음 **M** 키가 눌러지는 경우에 **MyPauseMenu** 위젯 블루프린트가 생성된다. 그다음의 **M** 키가 눌러지는 경우에는 또 위젯을 생성하지 않도록 해야 한다. 이것을 구현해보자.
먼저, **MyPauseMenuRef** 변수의 **Get** 노드를 배치하자. 그리고 **Get** 노드의 출력핀을 당기고 **IsValid** 노드를 배치하자. 여기서, **Boolean** 값을 리턴하는 **유효 여부(IsValid)** 함수 노드를 배치하는 것이 아니라, 유효 여부에 따라 실행핀으로 펄스를 내보내는 **IsValid** 노드를 배치해야 한다.
그다음, **IsValid** 노드의 **IsNotValid** 출력 실행핀을 당기고 이전의 **위젯 생성** 노드로 연결하자. 그리고 **M** 이벤트 노드의 실행핀을 **IsValid** 노드로 연결하자.
한편, 일시정지를 다시 실행하는 경우에는 위젯 생성 노드와 **Set MyPauseMenu** 노드를 건너뛰면 된다. 즉, **IsValid** 의 **IsValid** 출력 실행핀을 **Set ShowMouseCursor** 노드에 연결하자. 이제 **MyPauseMenuRef** 변수의 값이 유효한 경우에는 일시정지 메뉴를 다시 생성하지 않고 바로 변수에 접근한다.
이제 다음과 같은 모습이 될 것이다.



컴파일하고 저장하자.

17. 플레이해보자. 게임 플레이 중에 **M** 키를 눌러보자. 게임이 일시정지되고 일시정지 메뉴가 뜰 것이다. 메뉴에서 **Resume** 버튼을 누르면 게임플레이를 재개하고 **Quit** 버튼을 누르면 메인 메뉴로 빠져나간다.

<참고> 만약 **Quit** 버튼을 누를 때에 메인 메뉴로 가지 않는다면 메인 메뉴 레벨인 **MyMainMenuMap**을 잘못 만들었을 가능성이 있다. 이 경우에는 **MyMainMenuMap**를 삭제한 후에 다시 만들고 실행해보자.

이제 모든 UI 구현이 완료되었다.

이 절에서는 일시정지 메뉴 구현의 레이아웃 배치 단계와 스크립팅 단계에 대해서 학습하였다.

□