ЛЕКЦИЯ 13

Блюпринты в действии 4

ЦЕЛИ И ИТОГИ ЛЕКЦИИ

Goals

Цели этой лекции:

- Показать больше функций Blueprint
- Показать, как использовать функцию Add Child Actor Component
- Объяснить действие функции AI MoveTo
- Показать выполнение консольной команды в Blueprint

Outcomes

К концу этой лекции вы сможете:

- Определять Актора по тегу
- Наносить урон Актору
- Использовать функцию Add Child Actor Component
- Создавать простой AI который двигается по Уровню

ФУНКЦИЯ
MAP RANGE CLAMPED

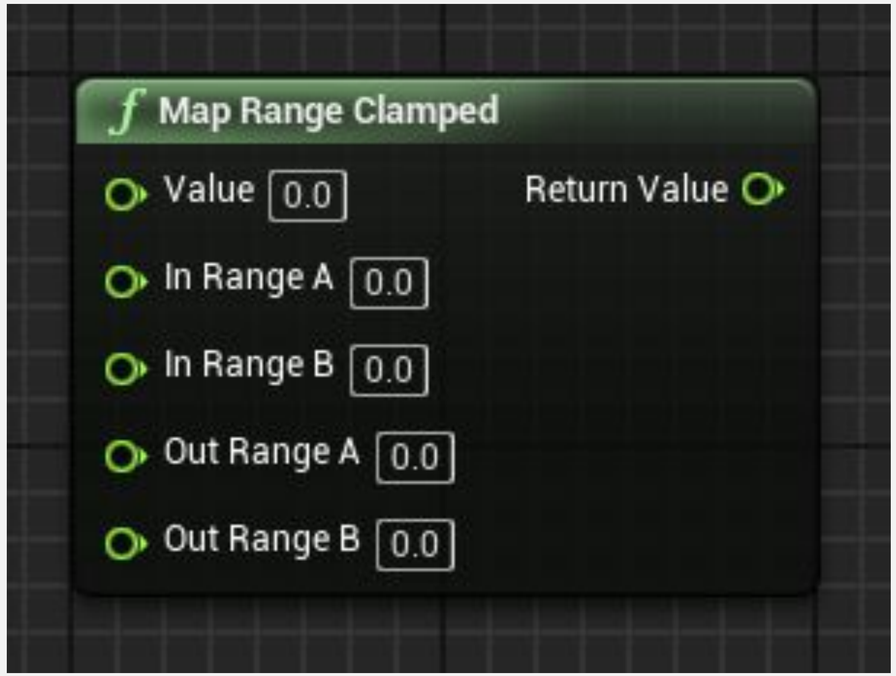
Функция **Map Range Clamped** преобразует значение из одного диапазона значений в соответствующее значение в другом диапазоне значений. Конечный результат всегда будет в диапазоне выходных значений.

Ввод

- **Value**: Исходное значение для преобразования.
- **In Range A**: Минимальное значение диапазона входных значений.
- **In Range B**: Максимальное значение диапазона входных значений.
- **Out Range A**: Минимальное значение диапазона выходных значений.
- **Out Range B**: Максимальное значение диапазона выходных значений.

Вывод

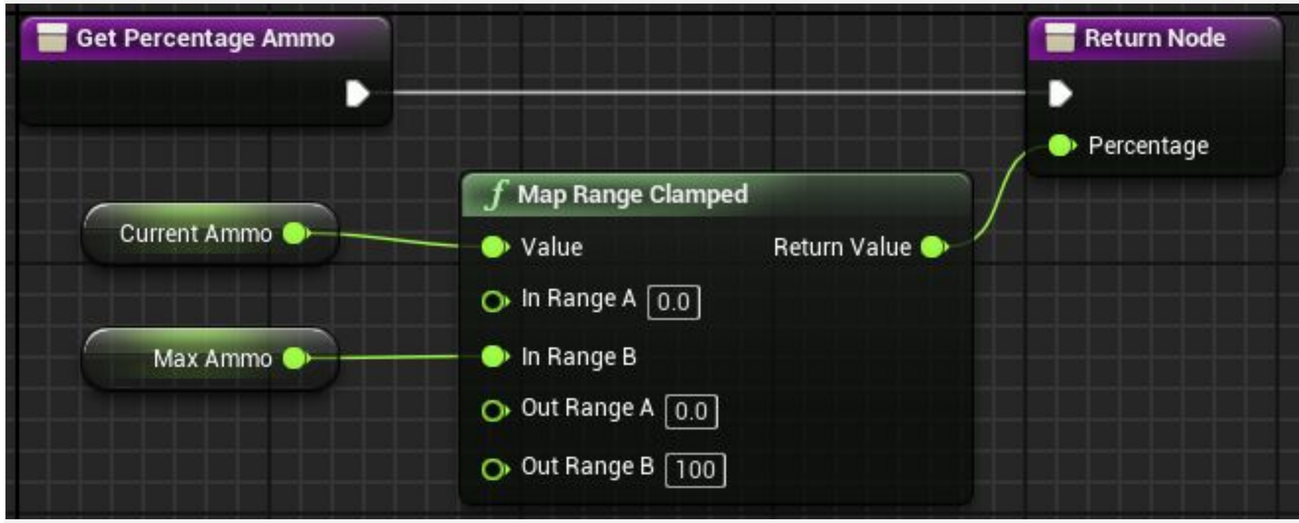
- **Return Value**: Преобразованное значение в диапазоне выходных значений.



ФУНКЦИЯ MAP RANGE CLAMPED: ПРИМЕР

В примере справа функция **Get Percentage Ammo** конвертирует текущее количество боеприпасов игрока в процентное значение, отображаемое на экране.

Расчет основан на переменных, которые содержат максимальное количество боеприпасов, которое может иметь игрок, и текущее количество боеприпасов.



ФУНКЦИЯ ACTOR HAS TAG

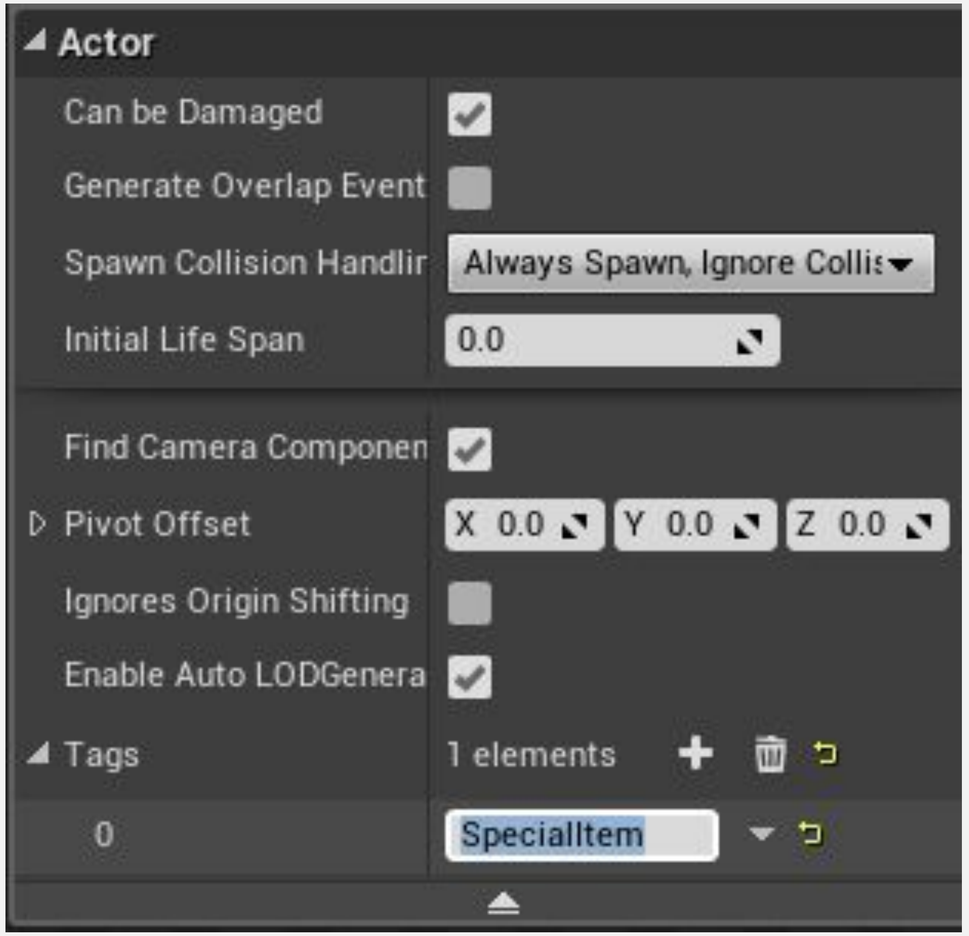
Функция **Actor Has Tag** проверяет, есть ли у Актора игры определенный тег. Использование тегов - простой способ различать Акторов на Уровне. Изображение справа показывает, что к Актору был добавлен тег под названием «**SpecialItem**».

Ввод

- **Target**: Ссылка на Актора, которая будет использоваться при проверке тега.
- **Tag**: Тег, который будет использоваться в тесте.

Вывод

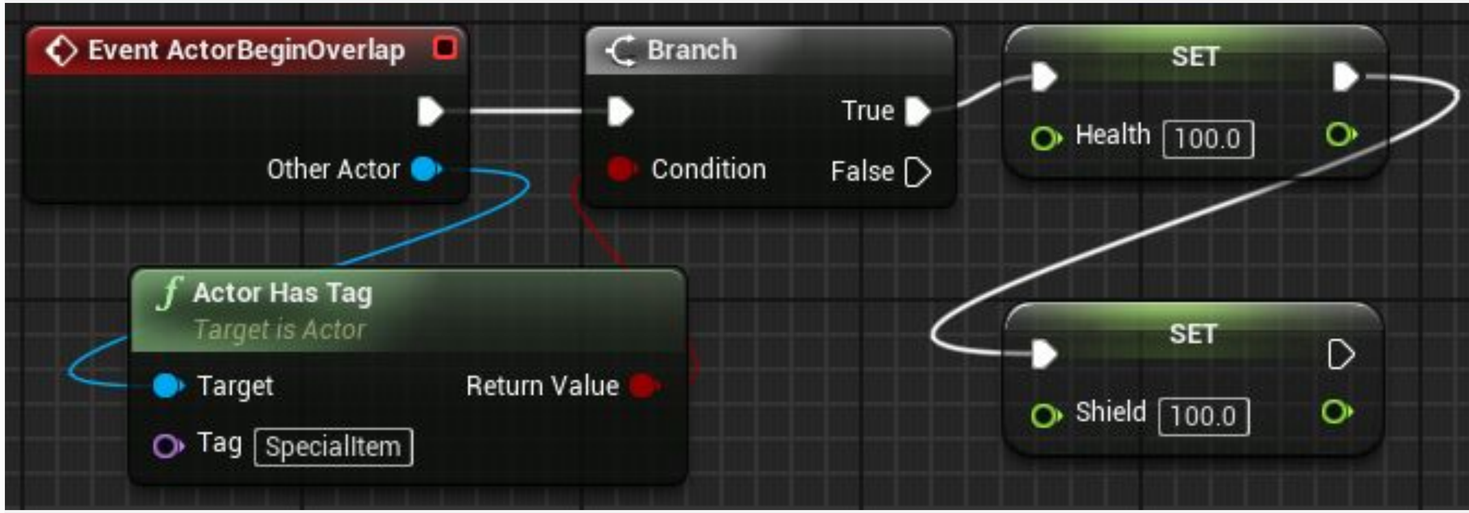
- **Return Value**: логическое значение. Если значение “**true**”, Актор имеет указанный тег.



ФУНКЦИЯ ACTOR HAS TAG: ПРИМЕР

В примере справа функция **Actor Has Tag** используется для проверки того, есть ли у Актера, на который накладывается игрок, тег **SpecialItem**.

Если это так, то здоровье и щит игрока восстанавливаются путем установки их значений на «**100.0**».



ФУНКЦИЯ APPLY DAMAGE

Функция **Apply Damage** используется для нанесения урона Актору. Информация, связанная с повреждением, может быть передана через функцию. Актор, по которому попали, реагирует на урон с помощью события **AnyDamage**, но событие срабатывает только в том случае, если значение параметра **Base Damage** ноды **Apply Damage** отличается от «**0,0**».

Ввод

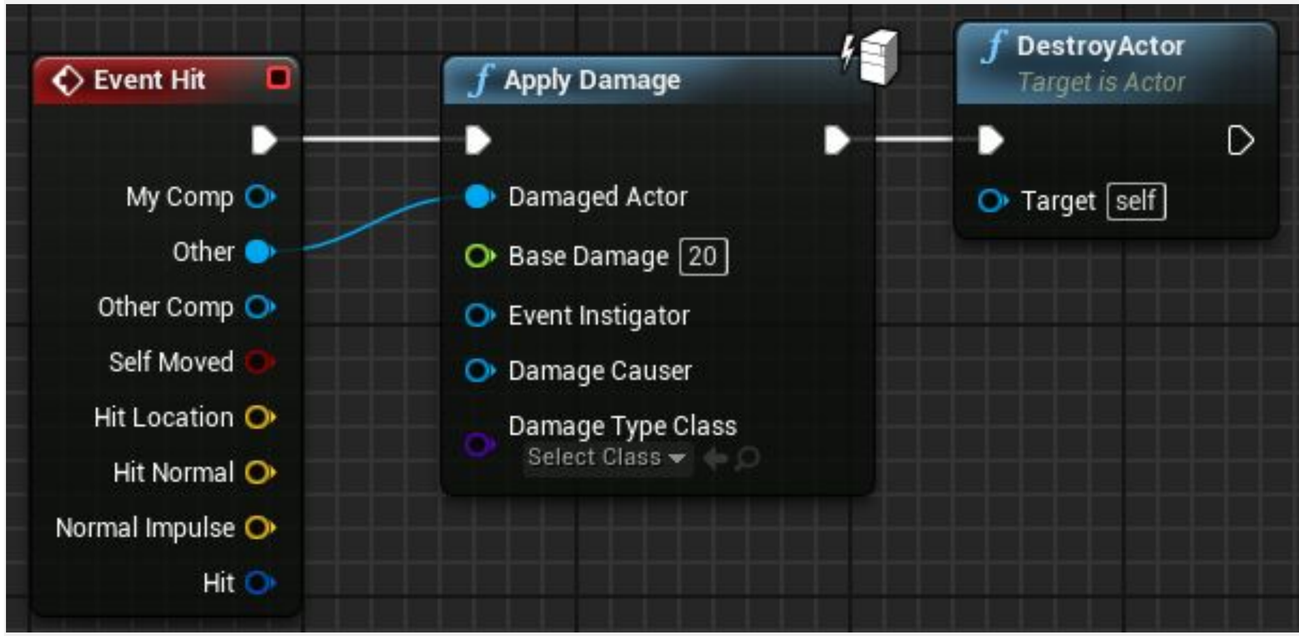
- **Damaged Actor**: Ссылка на Актор, который будет получать урон.
- **Base Damage**: Значение с плавающей запятой, представляющее урон.
- **Event Instigator**: Ссылка на контролер, ответственный за причинение урона. Использование этого параметра необязательно.
- **Damage Causer**: Ссылка на Актора, нанесшего ущерб. Использование этого параметра необязательно.
- **Damage Type Class**: Класс, представляющий тип нанесенного урона. Использование этого параметра необязательно.

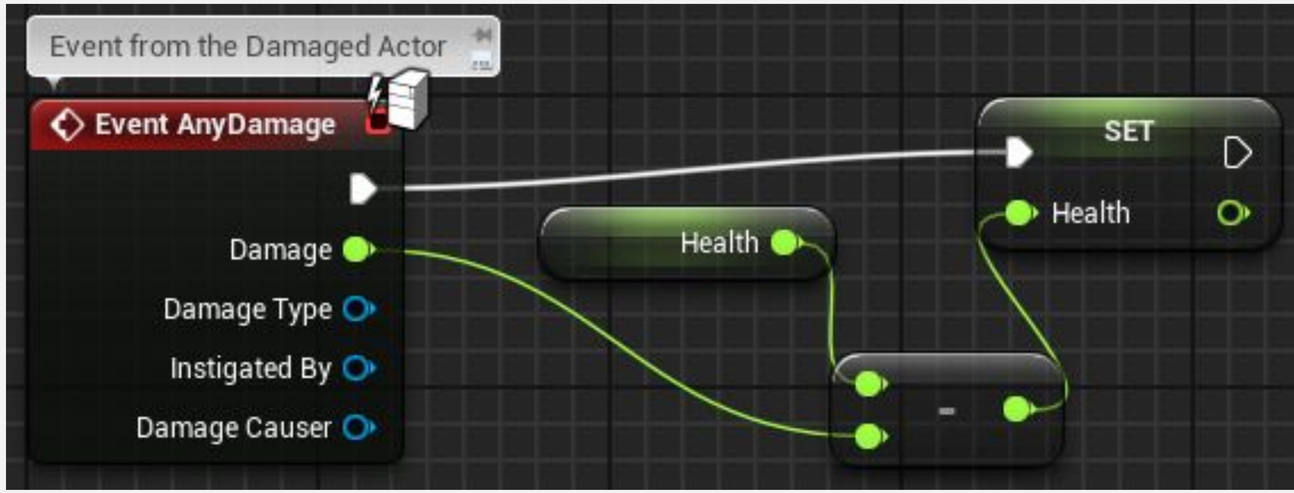
ФУНКЦИЯ APPLY DAMAGE: ПРИМЕР

Скрипт, показанный на верхнем изображении справа, можно использовать в Blueprint, который представляет собой пулю.

Когда пуля попадает в Актора, функция **Apply Damage** используется для нанесения урона со значением «**20**». После этого пуля уничтожается.

На нижнем изображении показано событие **AnyDamage** для поврежденного Актора. Это уменьшит значение переменной **Health** на значение **Damage**.





ФУНКЦИЯ GET OVERLAPPING ACTORS

Функция **Get Overlapping Actors** возвращает список Акторов, которые перекрывают Актор/компонент. Есть две версии этой функции: одна использует Актора в качестве цели, а другая использует компонент в качестве цели.

Ввод

- **Target**: Ссылка на Актор/компонент который будет использоваться в тесте на перекрытие.
- **Class Filter**: Указывает, какой класс или подклассы будут использоваться в тесте. Использование этого параметра необязательно.

Вывод

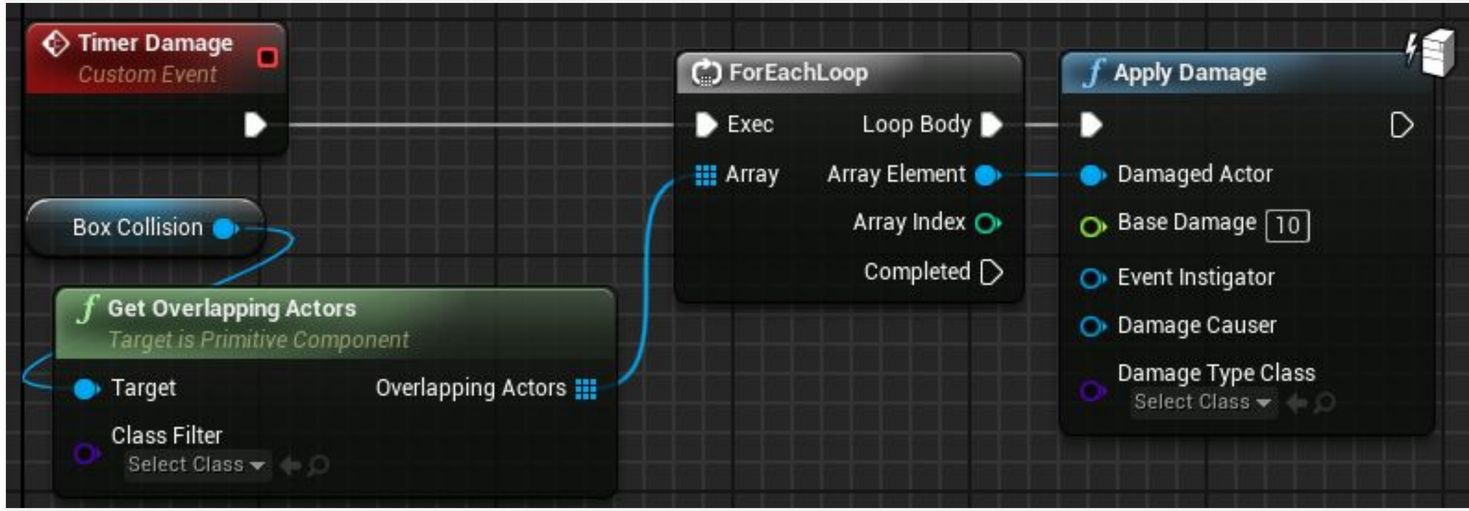
- **Overlapping Actors**: Массив, содержащий Акторов, которые перекрывают Актора/компонент.

ФУНКЦИЯ GET OVERLAPPING ACTORS: ПРИМЕР

Представьте себе область, которая наносит урон всем находящимся в ней Акторам.

В примере справа эта область представлена Blueprint с компонентом Box Collision, который определяет место, где Акторы получают урон.

Событие **Timer Damage** периодически вызывается, чтобы нанести урон всем Актерам, которые перекрывают Box Collision.



ФУНКЦИЯ ADD CHILD ACTOR COMPONENT

Функция **Add Child Actor Component** добавляет Актора как компонент другого Актора. Таким образом, компонент Актор следует преобразованиям родительского Актора. Когда родительский Актор уничтожается, компонент Актор также уничтожается. Новый класс актера должен быть указан на панели **Details** функции **Add Child Actor Component**.

Ввод

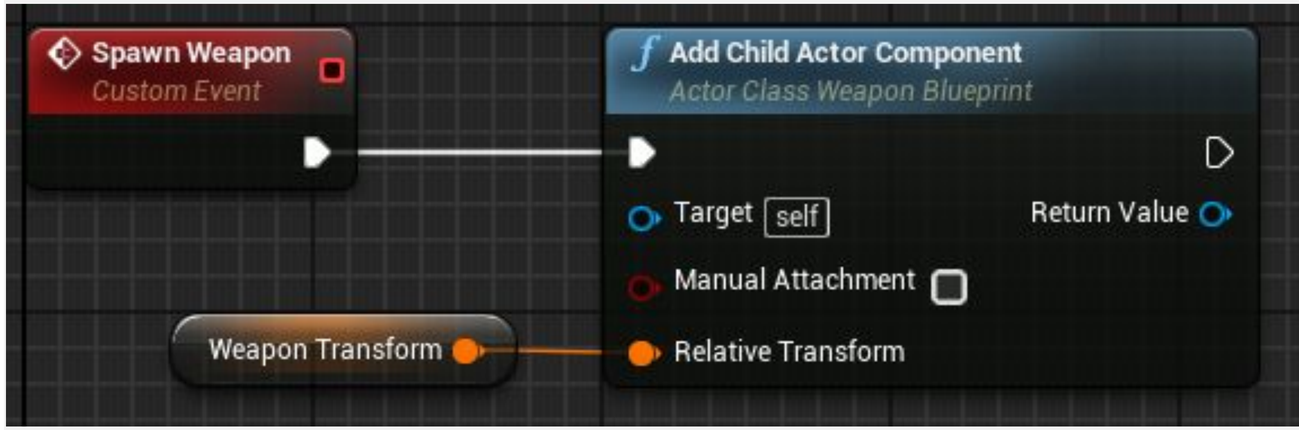
- **Target**: Ссылка на Актора, которому будет принадлежать новый компонент.
- **Manual Attachment**: Логическое значение. Если значение равно «false», новый Актор будет автоматически присоединен.
- **Relative Transform**: Преобразование, используемое новым компонентом.

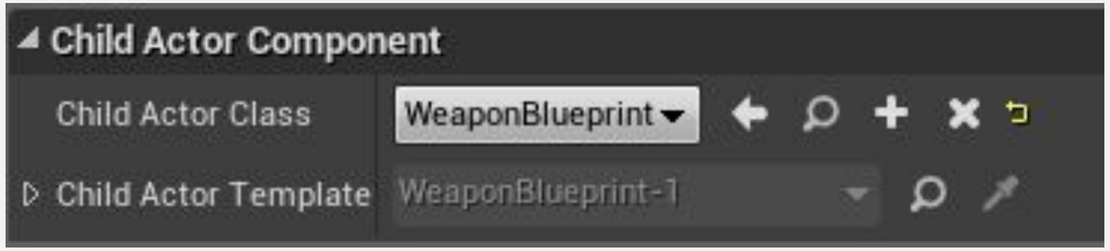
Вывод

ФУНКЦИЯ ADD CHILD ACTOR COMPONENT: ПРИМЕР

В примере справа есть Blueprint с именем «**WeaponBlueprint**», который представляет оружие. В другом Blueprint, который представляет персонажа в игре, есть событие, которое будет вызываться во время игры, чтобы добавить к персонажу оружие типа «**WeaponBlueprint**».

Нижнее изображение находится на панели **Details** функции **Add Child Actor Component**. Отображается при выборе функции. Класс, который будет использоваться новым Актором, должен быть указан в поле свойства **Child Actor Class**.



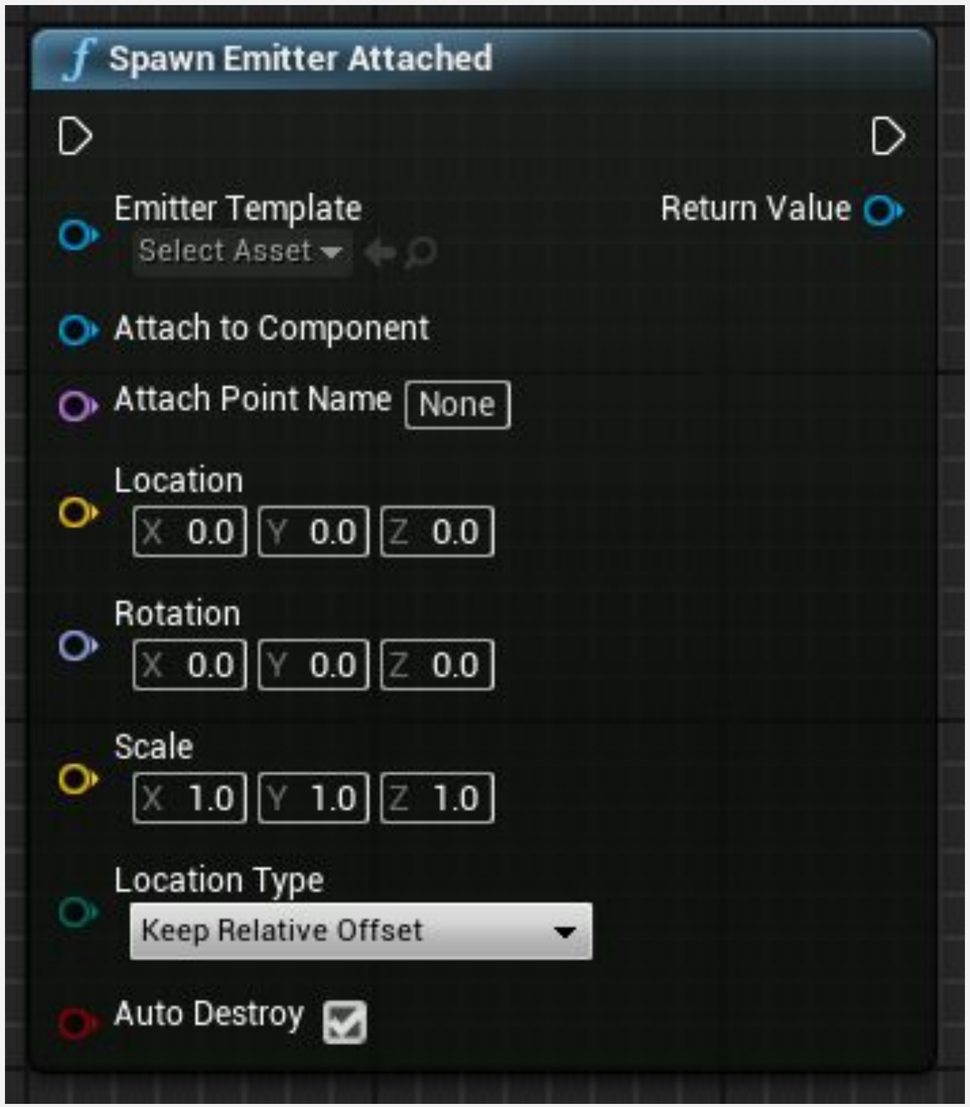


ФУНКЦИЯ SPAWN EMITTER ATTACHED

Функция **Spawn Emitter Attached** воспроизводит систему частиц и прикрепляет ее к компоненту.

Ввод

- **Emitter Template**: Шаблон системы частиц, который будет использоваться.
- **Attach to Component**: Ссылка на компонент.
- **Attach Point Name**: Имя сокета, к которому будет прикреплен эмиттер. Использование этого параметра необязательно.
- **Location, Rotation, Scale**: Мировое или относительное преобразование в зависимости от значения **Location Type**.
- **Location Type**: Мировое или относительное местоположение.
- **Auto Destroy**: Логическое значение. Если значение «true», система частиц будет уничтожена после завершения выполнения.

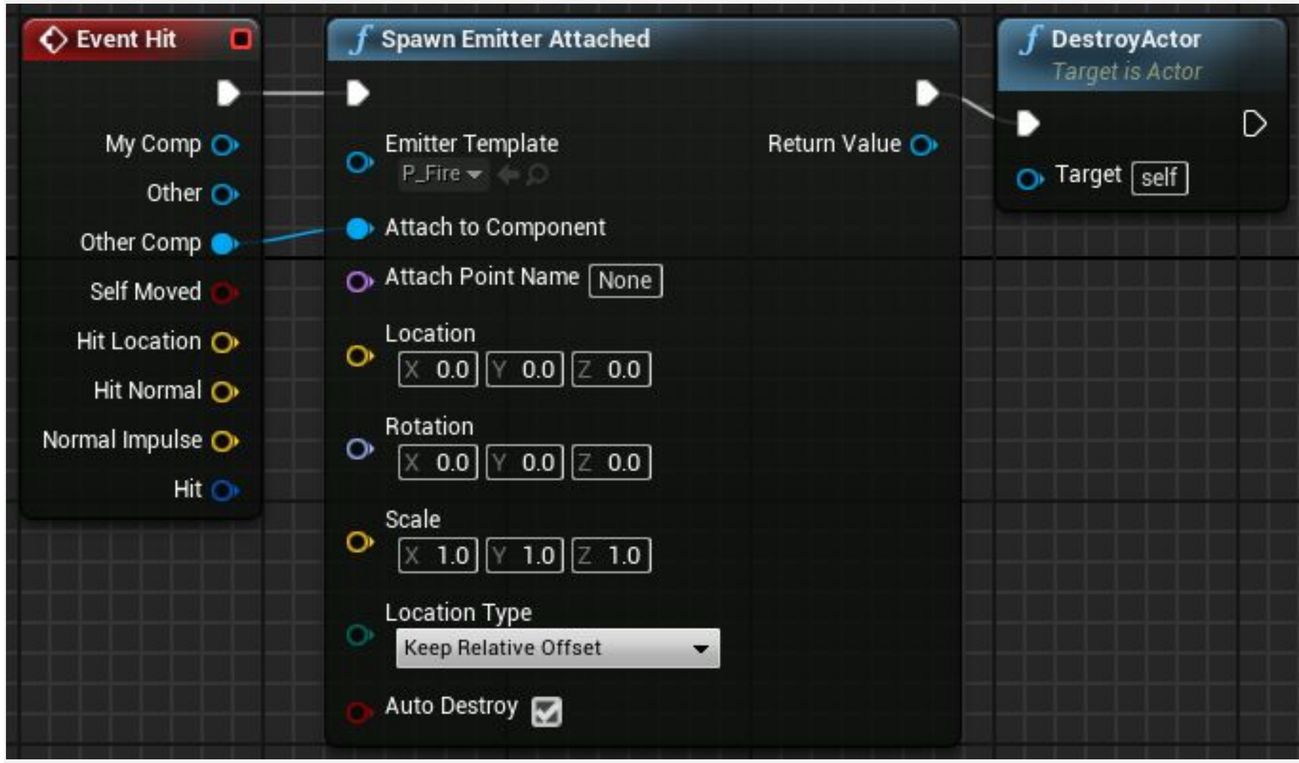


ФУНКЦИЯ SPAWN EMITTER ATTACHED: ПРИМЕР

Изображение справа взято из Blueprint, представляющего пулю.

Когда пуля сталкивается с чем-либо, система частиц будет создана с использованием шаблона **P_Fire** и будет прикреплена к компоненту, в который попал.

После этого пуля будет уничтожена.



НОДА AI MOVE TO

Нода **AI MoveTo** используется для перемещения Pawn. Пункт назначения может быть местом или другим Актором. Уровень должен иметь Nav Mesh Bounds Volume чтобы определить область, которую действие **AI MoveTo** может использовать. Это скрытое действие, поэтому оно выполняется параллельно с обычным потоком выполнения Blueprints.

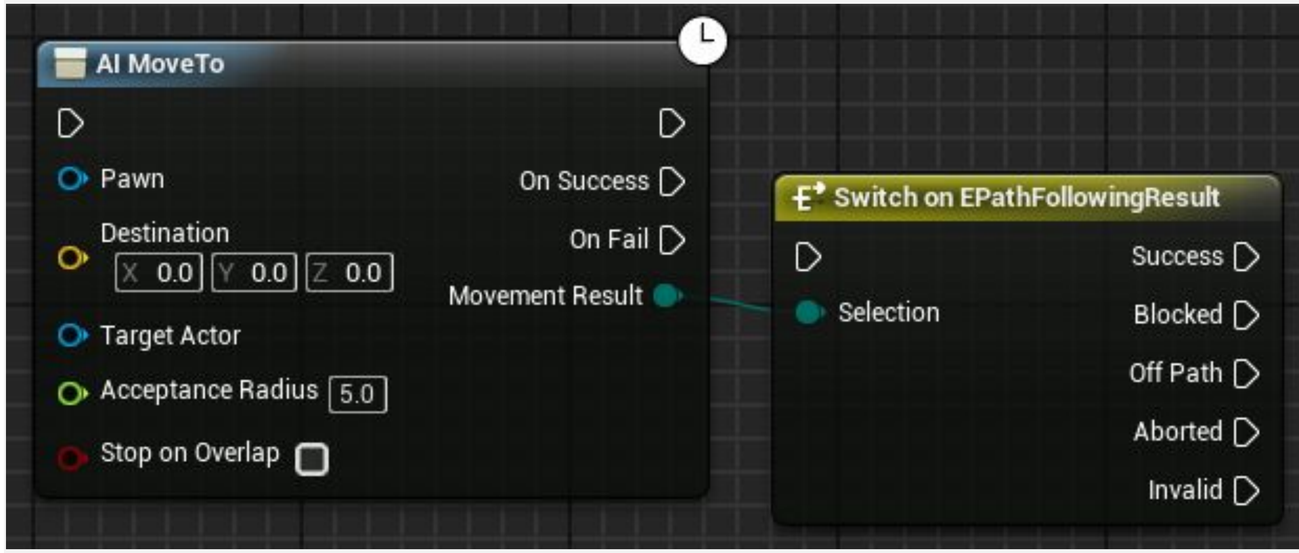
НОДА AI MOVE TO: ВВОД И ВЫВОД

Ввод

- **Pawn**: Ссылка на Pawn который будет перемещаться.
- **Destination**: Вектор, отображающий пункт назначения.
- **Target Actor**: Ссылка на Актор, который будет постоянно отслеживаться.
- **Acceptance Radius**: Максимальное расстояние до места назначения для завершения движения.
- **Stop on Overlap**: Логическое значение. Если значение «true», ход будет завершен, как только Pawn начнет перекрывать радиус приема.

Вывод

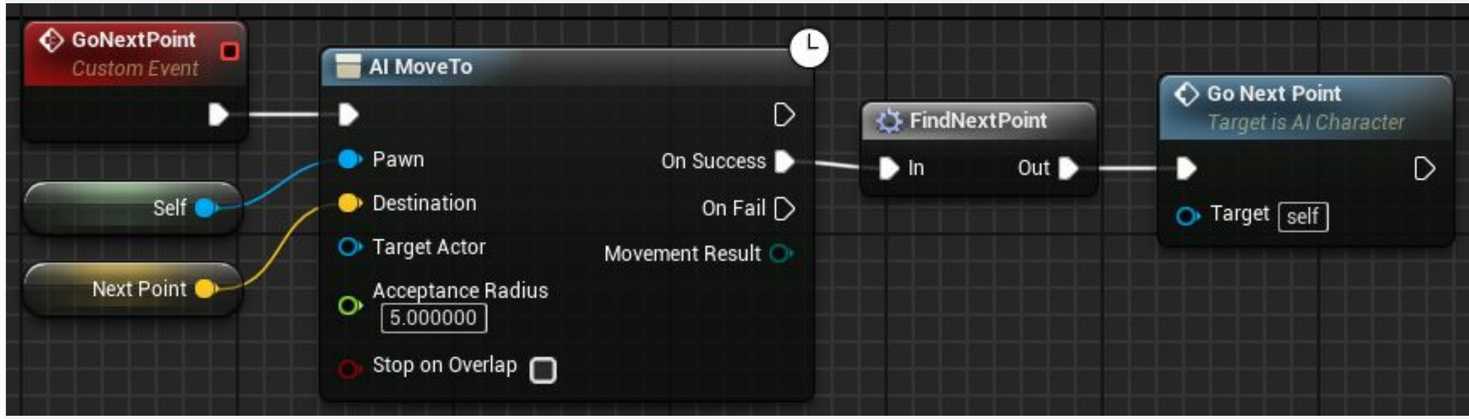
- **On Success**: Пин **Exec**, который будет выполнен, когда функция **AI MoveTo** достигнет своей цели.
- **On Fail**: Пин **Exec**, который будет выполнен, если функция **AI MoveTo** не может достичь своей цели.
- **Movement Result**: Перечисление со следующим возможным результатом: “Success”, “Blocked”, “Off Path”, “Aborted”, или “Invalid”.



НОДА AI MOVE TO: ПРИМЕР

В примере справа нода **AI MoveTo** используется, чтобы направить Pawn по пути, представленному набором предопределенных точек на уровне. Эти точки, представляющие путь, могут храниться в массиве. Макрос **FindNextPoint** отвечает за выбор следующей точки на пути и сохранение ее в переменной **Next Point**.

Значок часов указывает на то, что **AI MoveTo** является скрытым действием. Событие **GoNextPoint** инициирует действие **AI MoveTo**, но действия, связанные с выводом **On Success**, будут выполнены только тогда, когда Pawn достигнет своего пункта назначения. После того, как макрос **FindNextPoint** обновляет переменную **Next Point**, событие **GoNextPoint** вызывается снова.



ФУНКЦИЯ EXECUTE CONSOLE COMMAND

Функция **Execute Console Command** позволяет выполнять консольную команду.

Консольные команды - это текстовые команды, которые можно выполнять в редакторе или в игре.

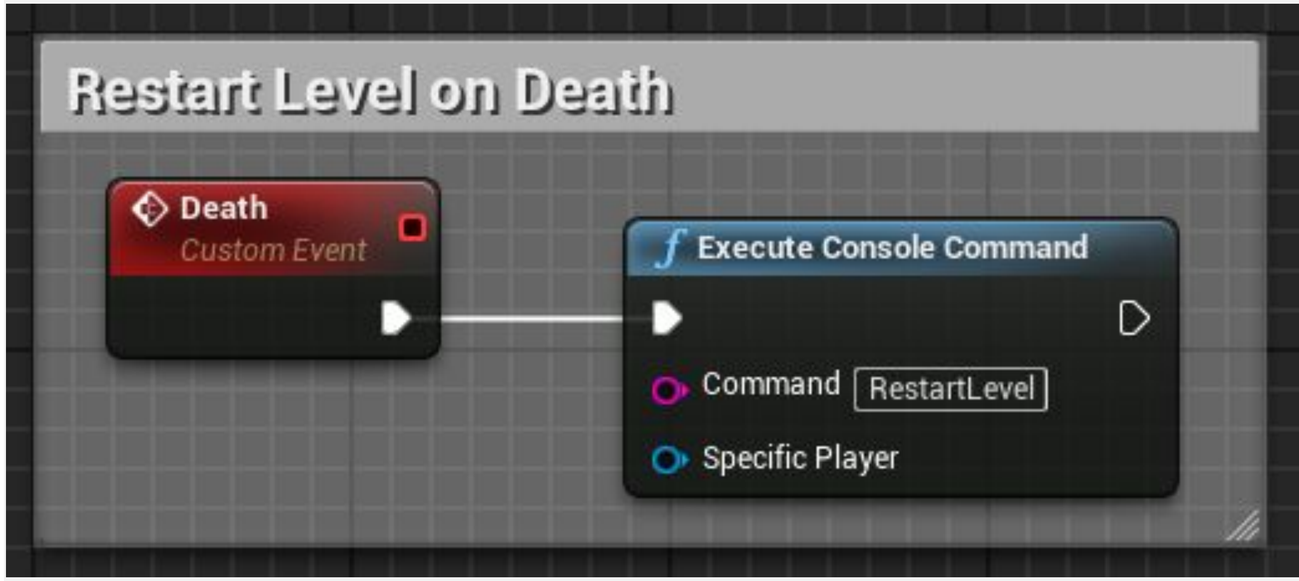
Ввод

- **Command**: Консольная команда, которая будет выполняться.
- **Specific Player**: Ссылка на Player Controller, который будет выполнять команду. Использование этого параметра необязательно.

ФУНКЦИЯ EXECUTE CONSOLE COMMAND: ПРИМЕР

В простой игре уровень будет перезапущен после смерти игрока. В примере справа это делается с помощью настраиваемого события с именем “**Death**” которое выполняет консольную команду “**RestartLevel**”.

Чтобы получить полный список консольных команд, откройте окно журнала вывода (**Window > Developer Tools > Output Log**). Строка внизу окна предназначена для ввода консольных команд. Введите команду «**DumpConsoleCommands**» и нажмите клавишу **Enter**, чтобы получить список команд.



ИТОГ

В этой лекции было представлено больше функций Blueprint. Также было показано, как использовать компонент Add Child Actor и функции AI MoveTo, а также как выполнять консольную команду в Blueprint.