ЛЕКЦИЯ 11

Блюпринты в действии 2

ЦЕЛИ И ВЫВОДЫ ЛЕКЦИИ

Goals

Цели этой лекции:

- Объяснить что такое трассировка
- Показать, как делать разные типы трассировок
- Показать, как создавать и уничтожать звуки и частицы
- Представить различные типы анимаций, которые можно создать с помощью действий.

Outcomes

К концу этой лекции вы сможете:

- Использовать трассировку для проверки столкновения
- Управлять звуками и частицами
- Создавать простые анимации с помощью действий

ТРАССИРОВКИ

ТИПЫ ТРАССИРОВКИ

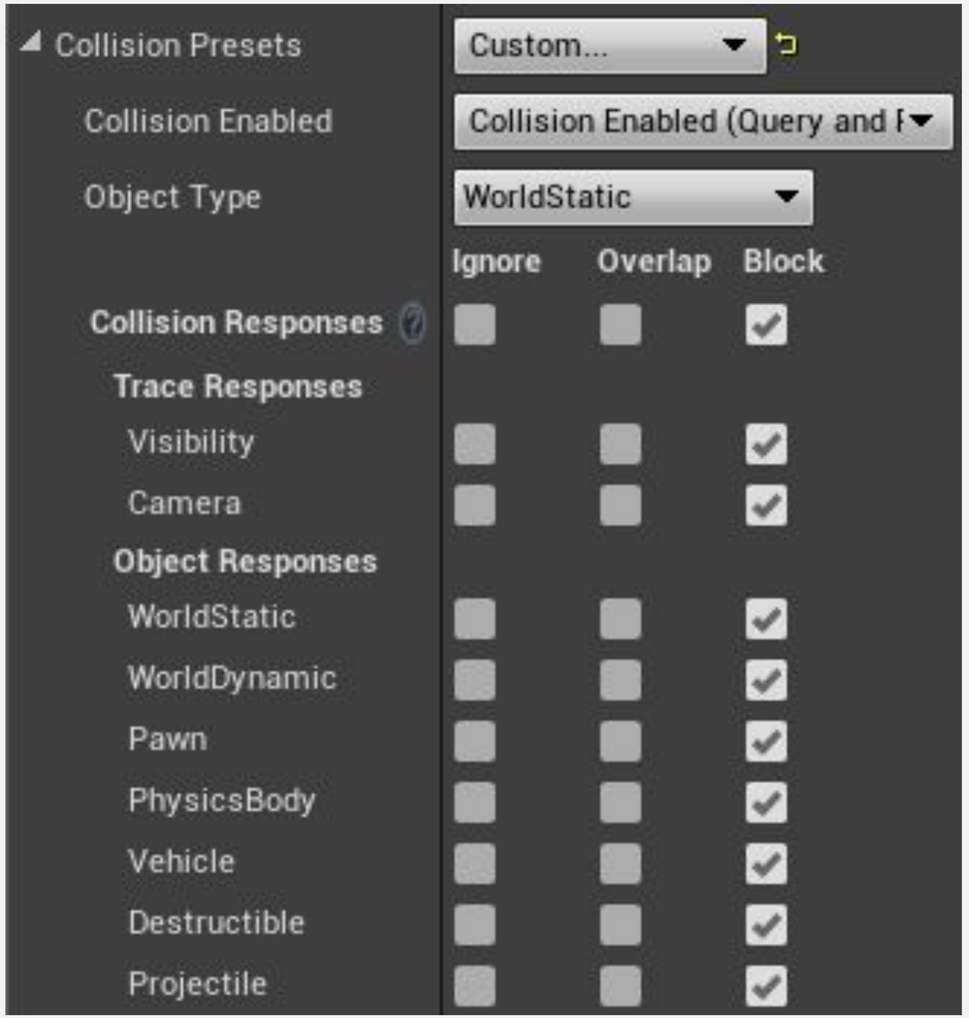
Трассировки используются для проверки наличия столкновений вдоль определенной линии и могут возвращать столкновение с первым или с несколькими объектами.

Трассировка может быть выполнена по **каналам** или по **типу объекта**.

Канал может быть “Visibility” или “Camera”. Типом объекта может быть “WorldStatic”, “WorldDynamic”, “Pawn”, “PhysicsBody”, “Vehicle”, “Destructible”, или “Projectile”.

На изображении справа показаны реакции Static Mesh Actor на столкновение.

Тип объекта определяется в раскрывающемся списке свойств **Object Type**, а ответы на отслеживание видимости и камеры определяются в разделе **Trace Responses** в таблице **Collision Responses**.



СТРУКТУРА HIT RESULT

Когда функция трассировки с чем-то сталкивается, она возвращает одну или несколько структур **Hit Result**. Нода **Break Hit Result** может использоваться для доступа к элементам **Hit Result**, как показано на изображении справа.

Вот некоторые элементы результата попадания:

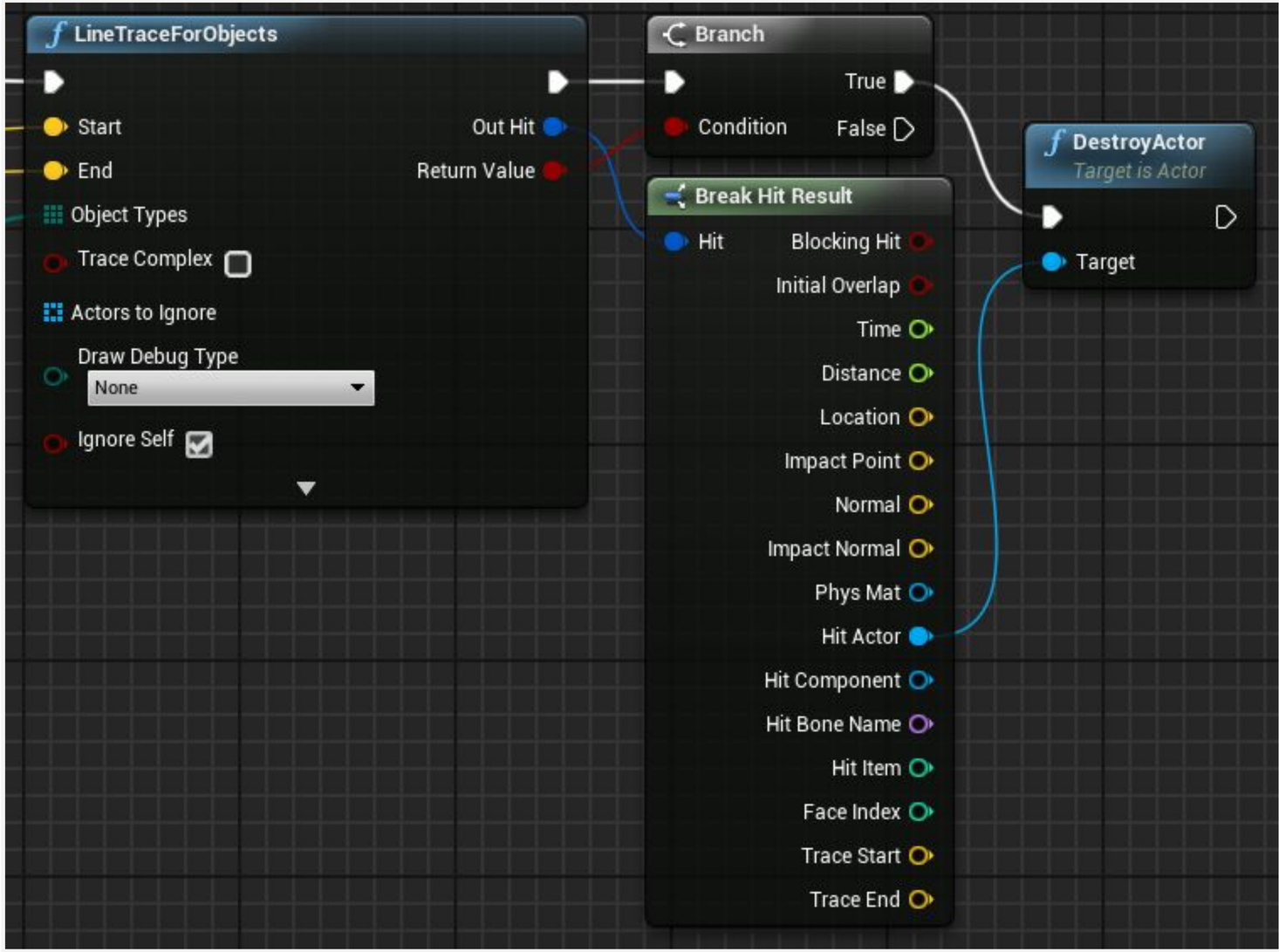
- **Blocking Hit**: Логическое значение, указывающее, было ли блокирующее попадание.
- **Location**: Местоположение попадания.
- **Normal**: Нормальный вектор попадания в мировом пространстве.
- **Hit Actor**: Отсылка к Актору, попавшему в трассировку.

СТРУКТУРА HIT RESULT: ПРИМЕР

На примере справа используется нода **Break Hit Result**. Выход параметра **Out Hit** функции **LineTraceForObjects** возвращает структуру Hit Result если было столкновение.

Попадание Актора удаляется из игры с помощью функции **DestroyActor**, которая использует ссылку попадания Актора в качестве цели.

Функция **LineTraceForObjects** объясняется на следующем слайде.

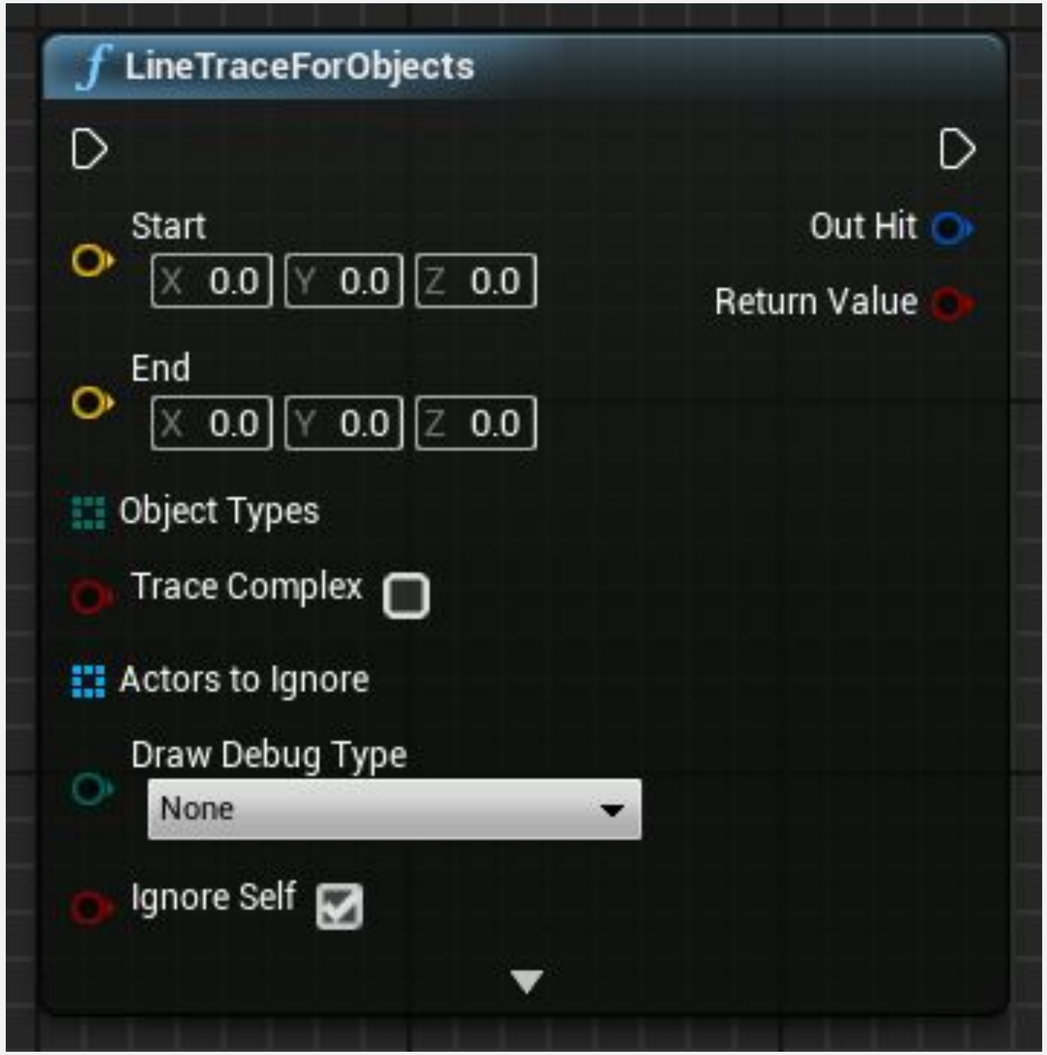


ФУНКЦИЯ
LINE TRACE FOR OBJECTS

Функция **LineTraceForObjects** проверяет наличие столкновений вдоль определенной линии и возвращает структуру Hit Result с данными для первого попадания Актора, которое соответствует одному из типов объектов, указанных в вызове функции.

Ввод

- **Start** и **End**: Векторы местоположения, которые определяют начало и конец линии, которая будет использоваться для теста столкновения.
- **Object Types**: Массив, содержащий типы объектов, которые будут использоваться в тесте на столкновение.
- **Trace Complex**: Логическое значение, указывающее, следует ли использовать сложные коллизии.
- **Actors to Ignore**: Массив Акторов уровня, которые следует игнорировать в тесте на столкновение.
- **Draw Debug Type**: Позволяет рисовать трехмерную линию, представляющую трассировку.



ФУНКЦИЯ MULTI LINE TRACE FOR OBJECTS

Функция **MultiLineTraceForObjects** имеет те же входные параметры, что и функция **LineTraceForObjects**.

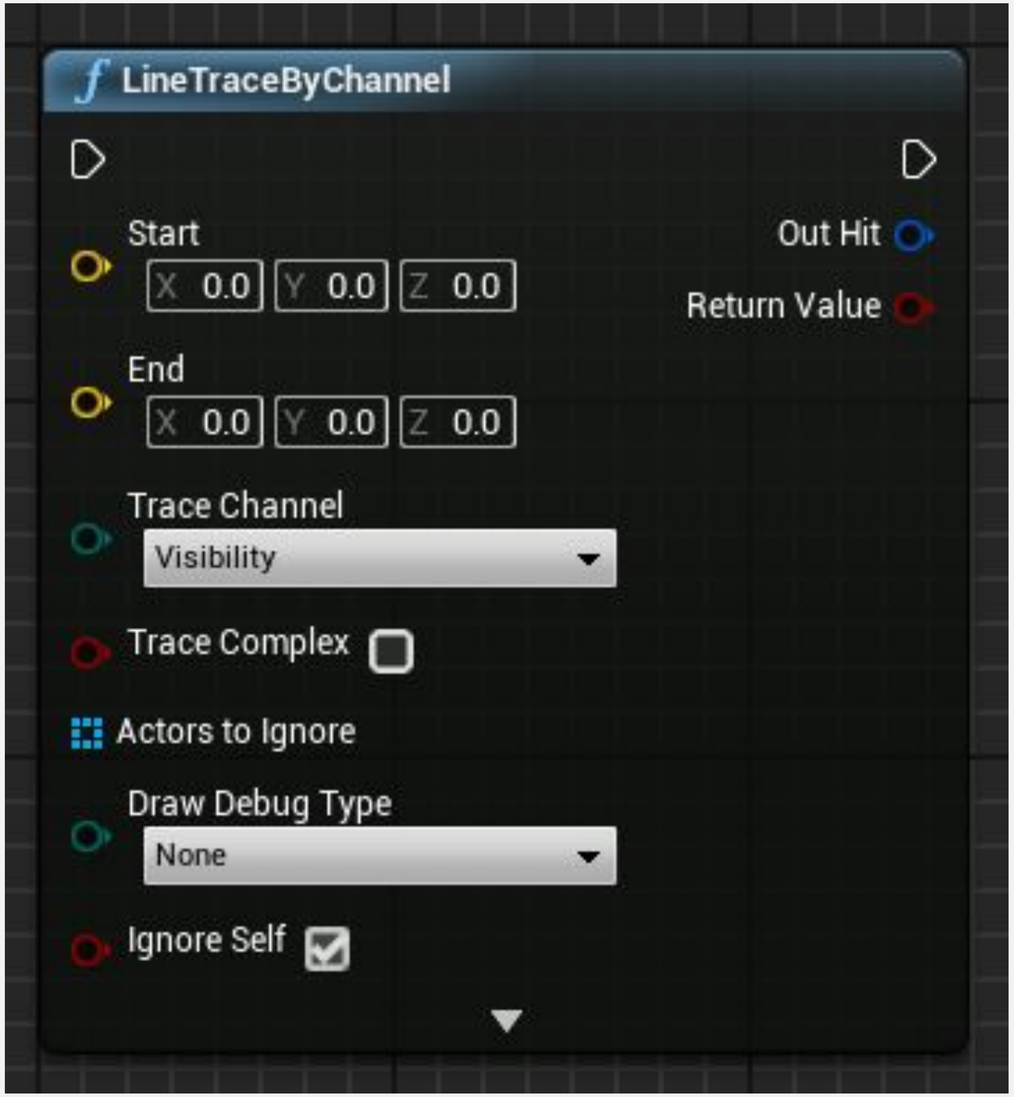
Разница между функциями заключается в том, что функция **MultiLineTraceForObjects** возвращает массив структур Hit Result, а не одну, что делает ее выполнение более дорогостоящим для просчета.

ФУНКЦИЯ
LINE TRACE BY CHANNEL

Функция **LineTraceByChannel** проверяет наличие столкновения вдоль определенной линии, используя канал трассировки, который может быть “Visibility” или “Camera”, возвращает структуру Hit Result с данными для первого попадания Актора в тесте на столкновение.

Ввод

- **Start** и **End**: Векторы местоположения, которые определяют начало и конец линии, которая будет использоваться для теста столкновения.
- **Trace Channel**: Канал, используемый для теста столкновения. Это может быть “Visibility” или “Camera”.
- **Trace Complex**, **Actors to Ignore**, и **Draw Debug Type**: Те же параметры, что и в функции **LineTraceForObjects** (см. Слайд 7)



ФУНКЦИЯ MULTI LINE TRACE BY CHANNEL

Функция **MultiLineTraceByChannel** имеет те же входные параметры, что и функция **LineTraceByChannel**.

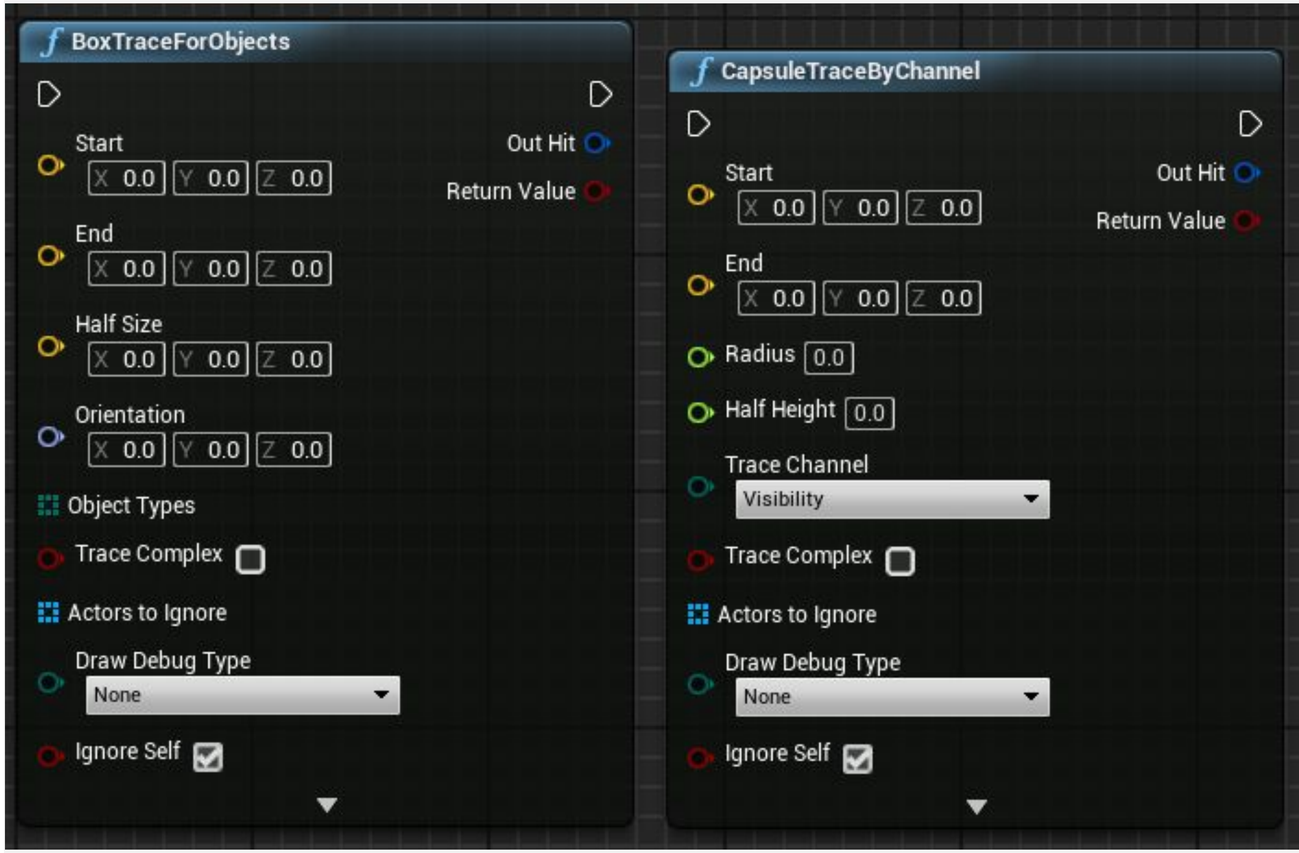
Разница между функциями заключается в том, что функция **MultiLineTraceByChannel** возвращает массив структур Hit Result, а не одну, что делает ее выполнение более дорогостоящим для просчета.

ТРАССИРОВКА В ВИДЕ ФИГУР

Трассировку также можно сделать с помощью фигур. Существуют функции трассировки для форм прямоугольника, капсулы и сферы, но выполнение этих функций дороже, чем трассировка линий.

Для всех этих форм есть функции для трассировки по каналам и по типу объекта. Также есть функции для одиночных или множественных попаданий.

На изображении справа показаны функции **BoxTraceForObjects** и **CapsuleTraceByChannel**.



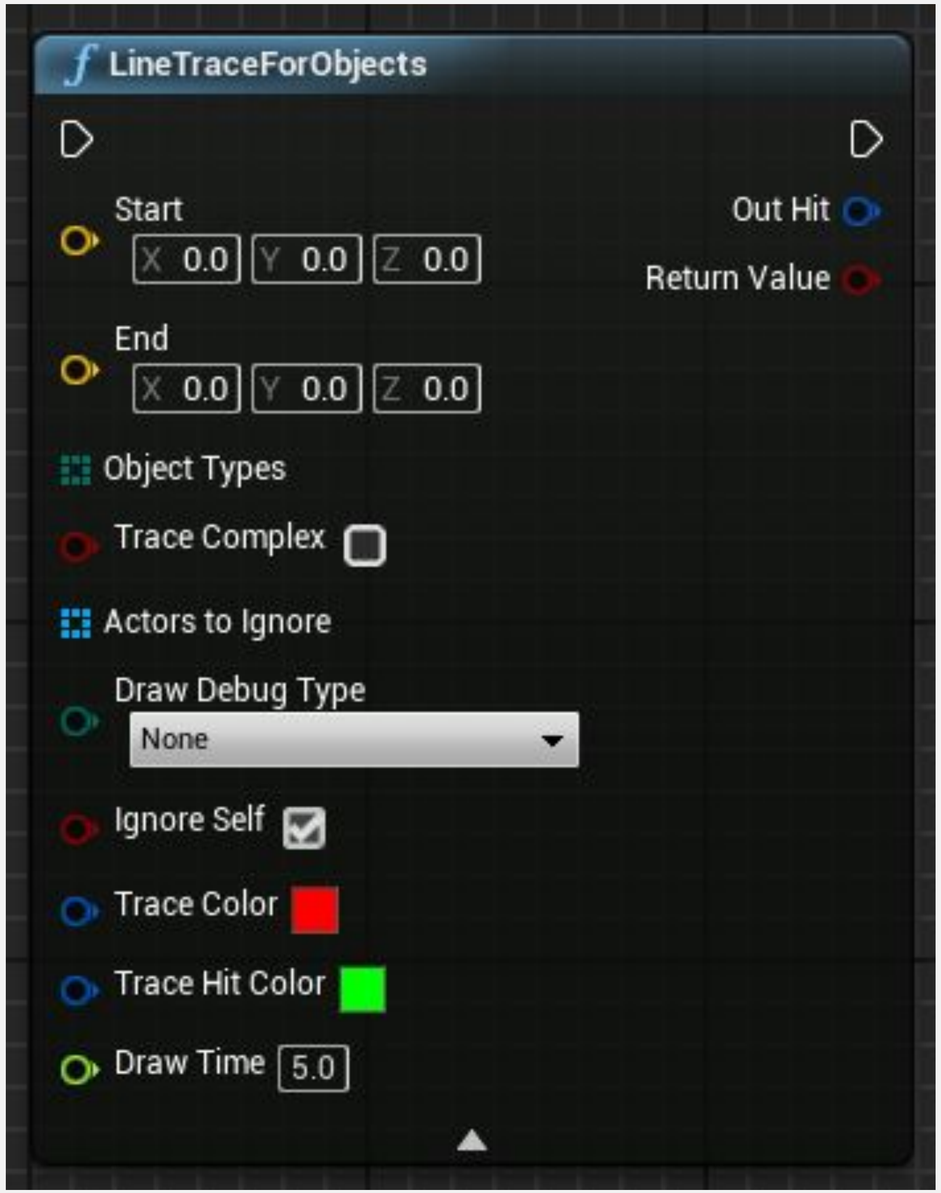
ОТЛАДКА

Функции трассировки имеют возможность рисовать линии отладки, которые помогают при тестировании трассировок.

Для параметра **Draw Debug Type** можно задать одно из следующих значений:

- **None**: Не рисовать линию.
- **For One Frame**: Линия появляется только на один кадр.
- **For Duration**: Линия остается в течение времени, указанного в параметре **Draw Time**.
- **Persistent**: Линия не исчезает.

Чтобы отобразить параметры **Trace Color**, **Trace Hit Color**, и **Draw Time** кликните по маленькой стрелке внизу функции.



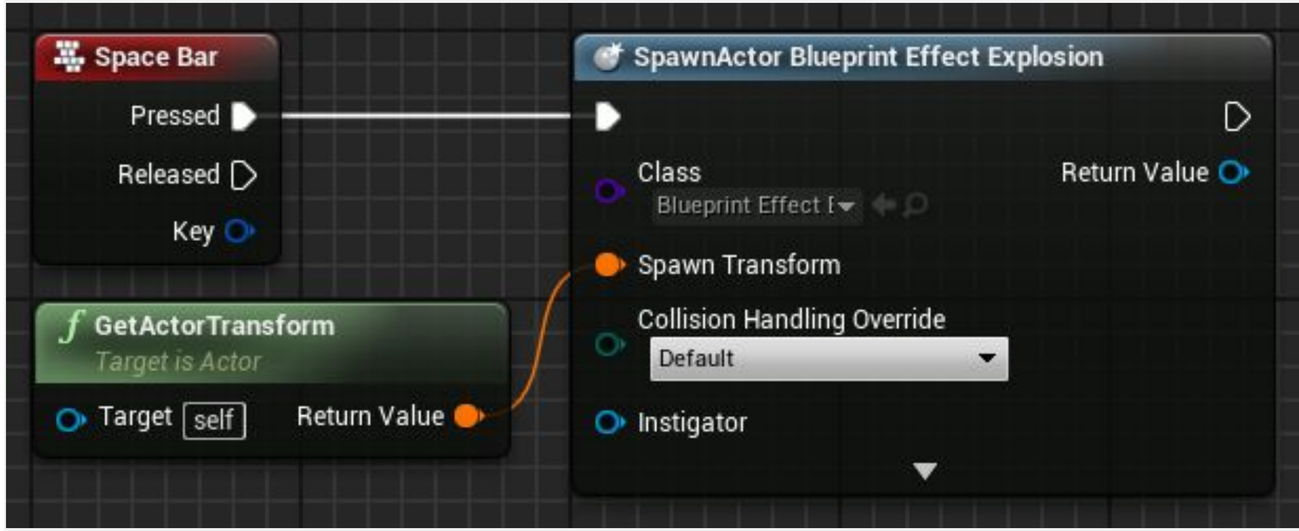
СОЗДАНИЕ И УНИЧТОЖЕНИЕ

СОЗДАНИЕ АКТОРОВ

Spawn Actor from Class - это функция, которая создает экземпляр Actor, используя указанный класс и преобразование.

Входные данные **Collision Handling Override** определяют, как обрабатывать столкновение во время создания. Выходной параметр **Return Value** - это ссылка на вновь созданный экземпляр.

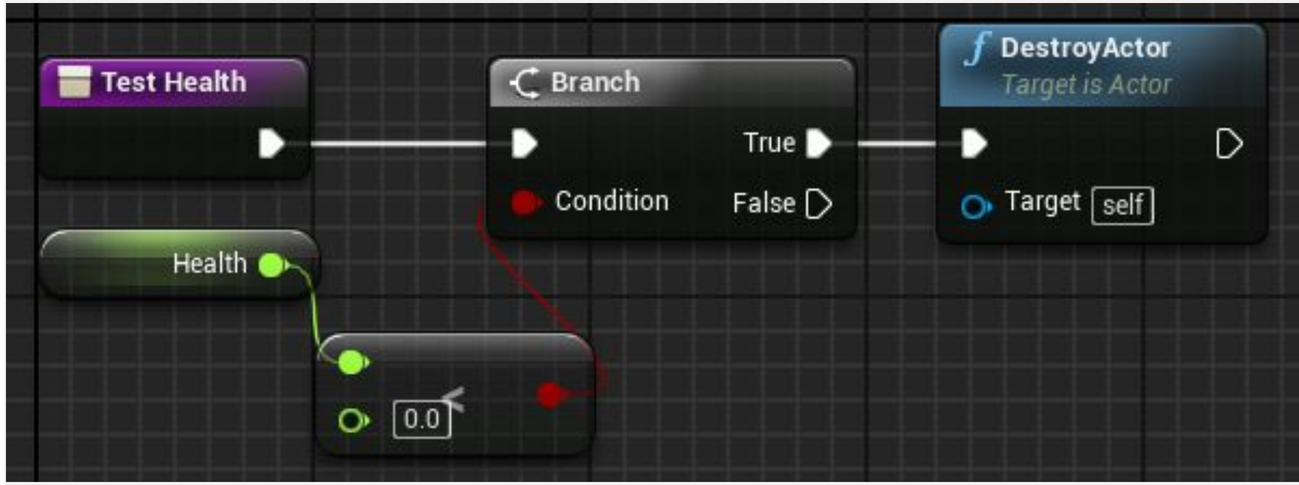
В примере справа, когда нажата клавиша **пробела**, создается экземпляр класса **Blueprint Effect Explosion** в том же месте (преобразовании) текущего Blueprint.



УНИЧТОЖЕНИЕ АКТОРОВ

Функция **DestroyActor** удаляет экземпляр Actor из уровня во время выполнения. Удаляемый экземпляр должен быть указан в параметре **Target.**

На изображении справа показана функция с именем «**Test Health**», которая проверяет, меньше ли значение переменной Health, чем ноль. Если «**true**», текущий экземпляр этого Blueprint, который представлен «**self**», будет уничтожен.



ПРИКРЕПЛЕНИЕ АКТОРА К КОМПОНЕНТУ

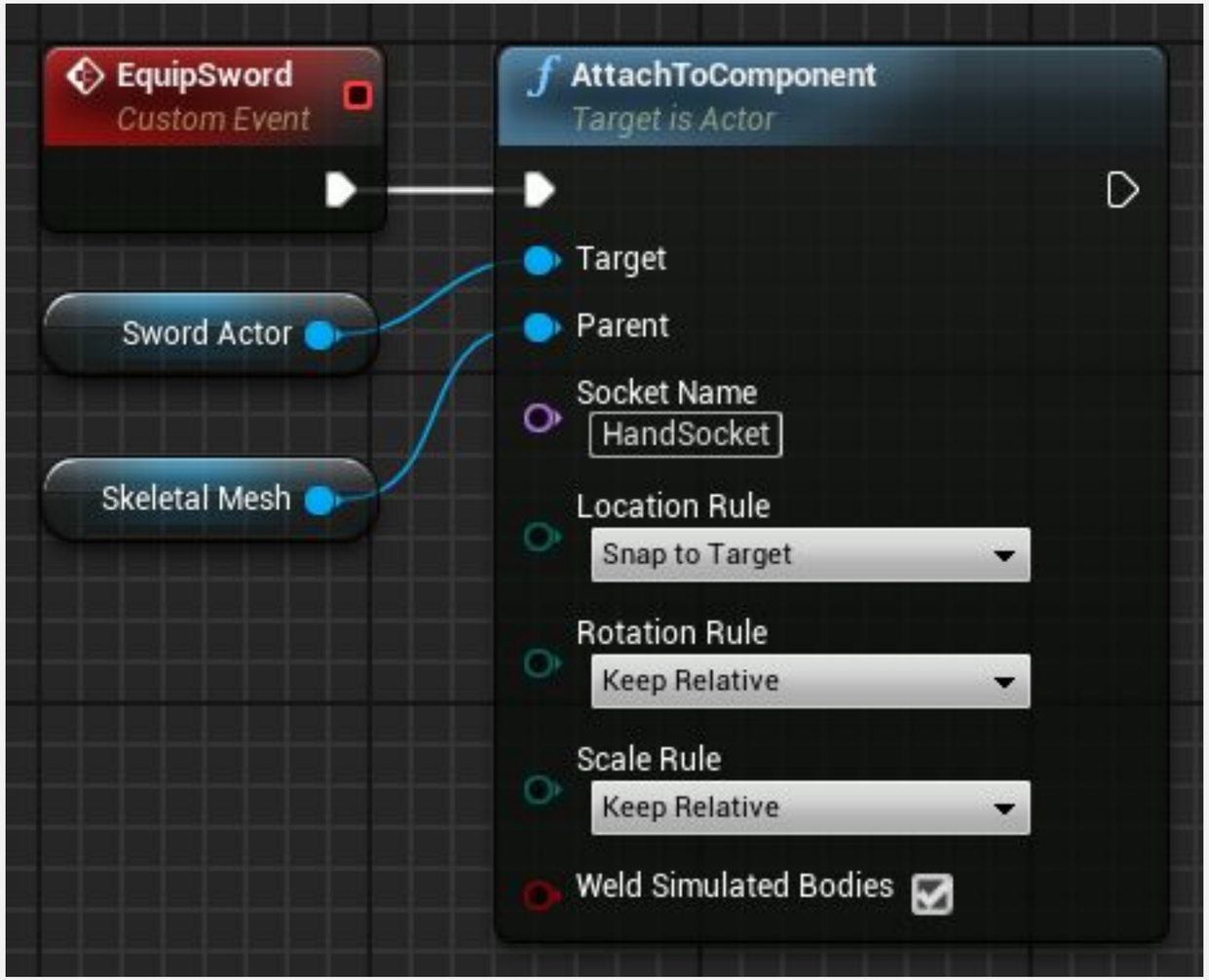
Функция **AttachToComponent** присоединяет Актор к компоненту, указанному во входном параметре **Parent**. Присоединенный Актор учитывает преобразования родительского компонента.

Ввод

- **Target**: Актор, которого нужно прикрепить.
- **Parent**: Компонент, который получает Актора.
- **Socket Name**: Имя сокета, к которому будет прикреплен Актор. Использование этого параметра необязательно.

Пример

В пользовательском событии справа Скелетал Меш игрока оснащен мечом, использующим гнездо с надписью «HandSocket», которое указывает, где меч должен оставаться в Скелетал Меше.



ЗВУКИ: ПРОИГРАТЬ ЗВУК НА ЛОКАЦИИ

В Blueprints есть простой способ воспроизвести звук. Просто используйте функцию **Play Sound at Location**.

Эта функция создает **Ambient Sound Actor** на уровне.

Используйте поле со списком, чтобы определить ресурс **Sound Cue** или **Sound Wave** для воспроизведения. Параметр **Location** используется для определения мировой позиции, откуда будет воспроизводиться звук.

Использование остальных параметров необязательно.

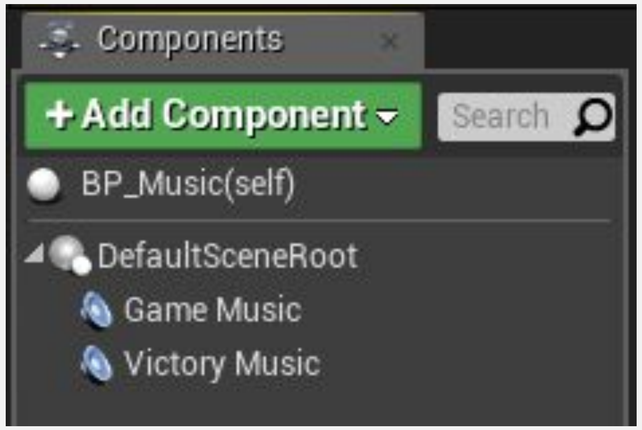
Когда Актор заканчивает играть, он уничтожается.

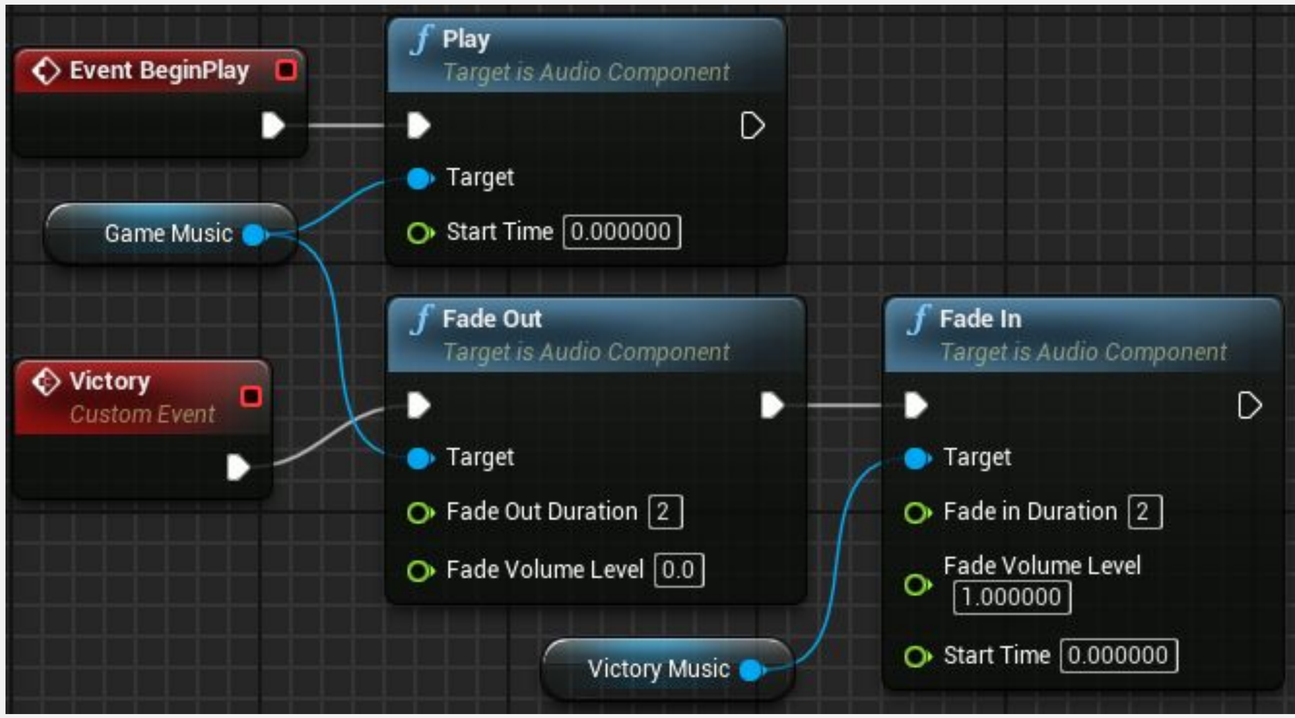
ЗВУКИ:
АУДИО КОМПОНЕНТ

Хороший способ контролировать звуки в Blueprints - использовать компонент **Audio**.

Чтобы добавить аудио компонент, нажмите кнопку **Add Component** на панели **My Blueprint** и выберите “**Audio**”. На панели **Details** выберите в окне **Sound** звук или песню, которая будет использоваться и уберите галочку у свойства **Auto Activate** чтобы звук не начал проигрываться автоматически.

В примере справа есть два аудио компонента, один из которых назван «**Game Music**», а другой - «**Victory Music**». Ноды **Fade Out** и **Fade In** используются для постепенного изменения музыки в течение двух секунд..





ЧАСТИЦЫ: СОЗДАНИЕ ИЗЛУЧАТЕЛЯ НА ЛОКАЦИИ

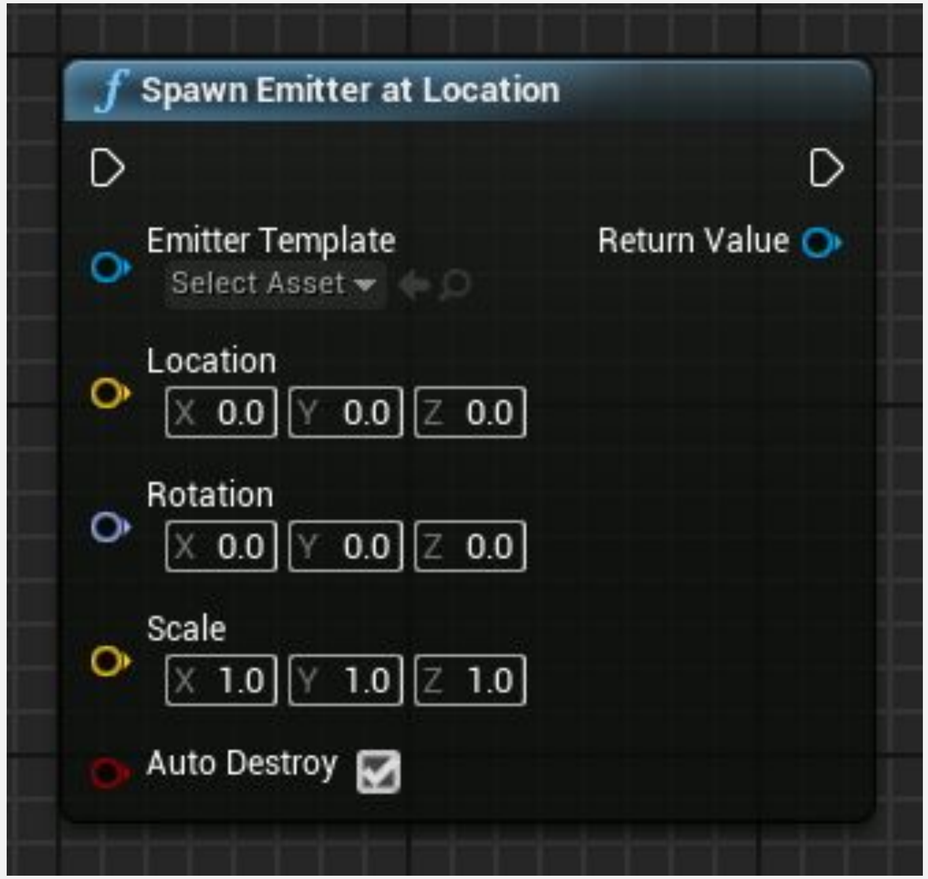
Функция **Spawn Emitter at Location** создает и воспроизводит компонент системы частиц в указанном месте.

Ввод

- **Emitter Template**: Ассет шаблона Системы Частиц, который будет использоваться
- **Location:** Место, в котором Система Частиц будет размещена.
- **Rotation: В**ращение, которое будет применено к Системе Частиц.
- **Scale**: Масштаб, который будет применен к Системе Частиц.
- **Auto Destroy**: Логическая переменная. Если значение равно «true», Система Частиц будет автоматически уничтожена после завершения выполнения.

Вывод

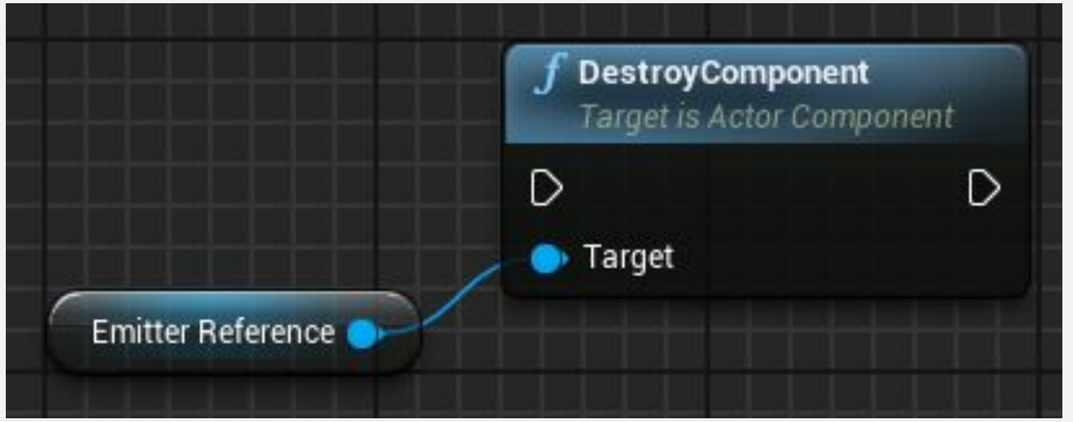
- **Return Value**: Ссылка на созданный компонент Системы Частиц.



ЧАСТИЦЫ: УНИЧТОЖЕНИЕ

Системы Частиц могут оставаться в цикле или быть деактивированы и сохранены в памяти для повторной активации позже.

Если необходимо уничтожить систему частиц, можно использовать функцию **DestroyComponent**, поскольку ссылка на систему частиц является компонентом системы частиц.



АНИМАЦИИ

ФУНКЦИЯ
SET VIEW TARGET WITH BLEND

Функция **Set View Target with Blend** из класса **Player Controller** очень полезна для переключения вида игры между разными камерами.

Ввод

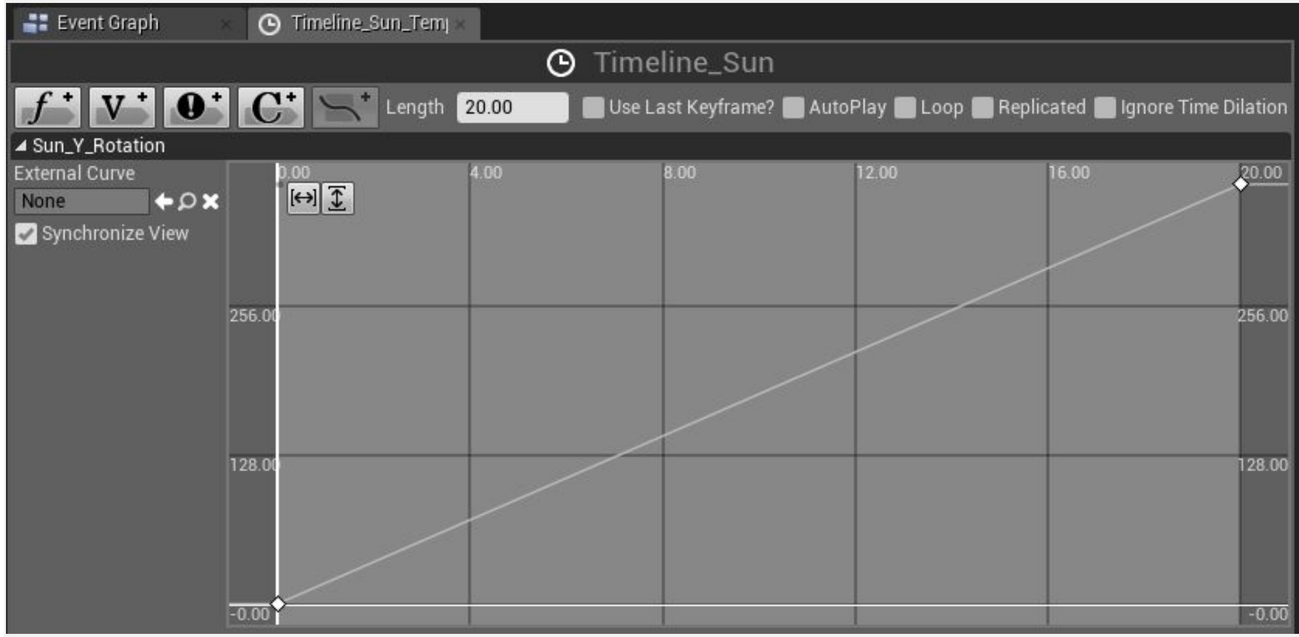
- **Target**: Ссылка на Player Controller.
- **New View Target**: Актор, который будет установлен в качестве цели просмотра. Обычно камера.
- **Blend Time**: Время, необходимое для завершения смешивания.
- **Blend Func**: Тип функции, используемой для смешивания.
- **Blend Exp**: значение экспоненты, которое контролирует форму кривой. Он используется некоторыми функциями наложения.
- **Lock Outgoing**: Логическая переменная. Если значение равно «true», исходящий объект просмотра будет привязан к положению камеры последнего кадра

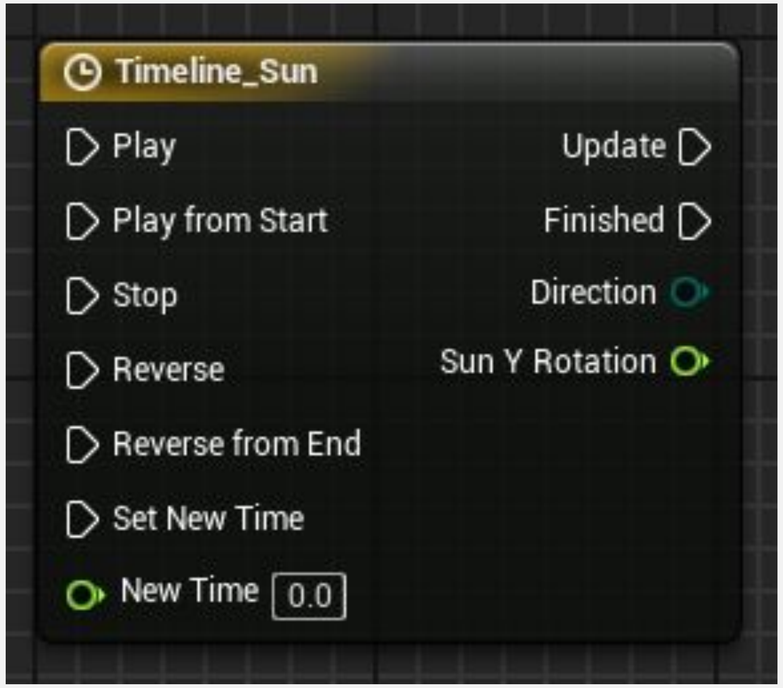
ВРЕМЕННЫЕ ШКАЛЫ

Временные шкалы позволяют создавать простые временные анимации внутри Blueprints. После того, как шкала времени была добавлена в график событий, ее можно редактировать в редакторе схем, дважды щелкнув по ней.

Переменные, добавленные на временную шкалу, отображаются как выходные параметры, чтобы можно было получить доступ к их значениям. На верхнем изображении справа показан редактор временной шкалы с дорожкой под названием «**Sun_Y_Rotation**».Во время работы шкалы времени кнопка обновления вызывается постоянно.

Временную шкалу можно воспроизводить вперед или назад.

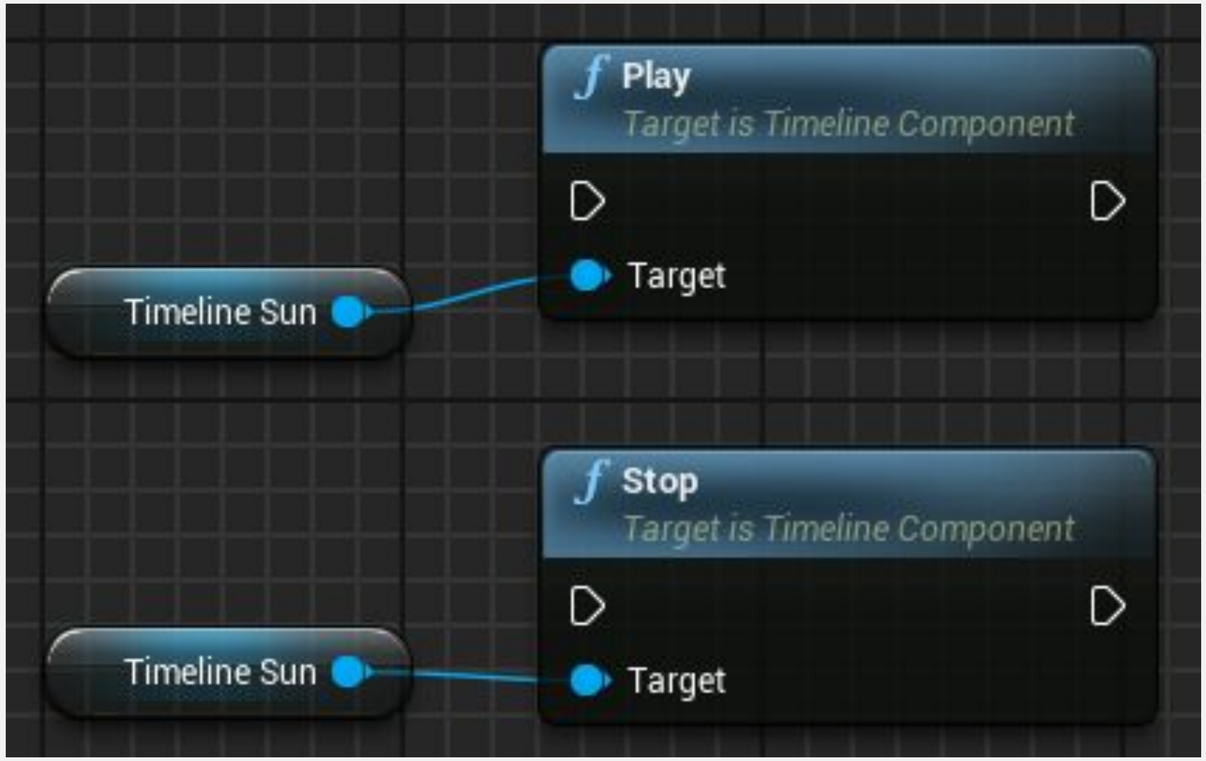




ВРЕМЕННЫЕ ШКАЛЫ КАК ПЕРЕМЕННЫЕ

После создания временные шкалы также могут быть доступны как переменные, поэтому можно вызывать функции временной шкалы из любого места на графике.

Нода **Get** у шкалы может быть получена из панели **My Blueprint** через **Variables > Components**.



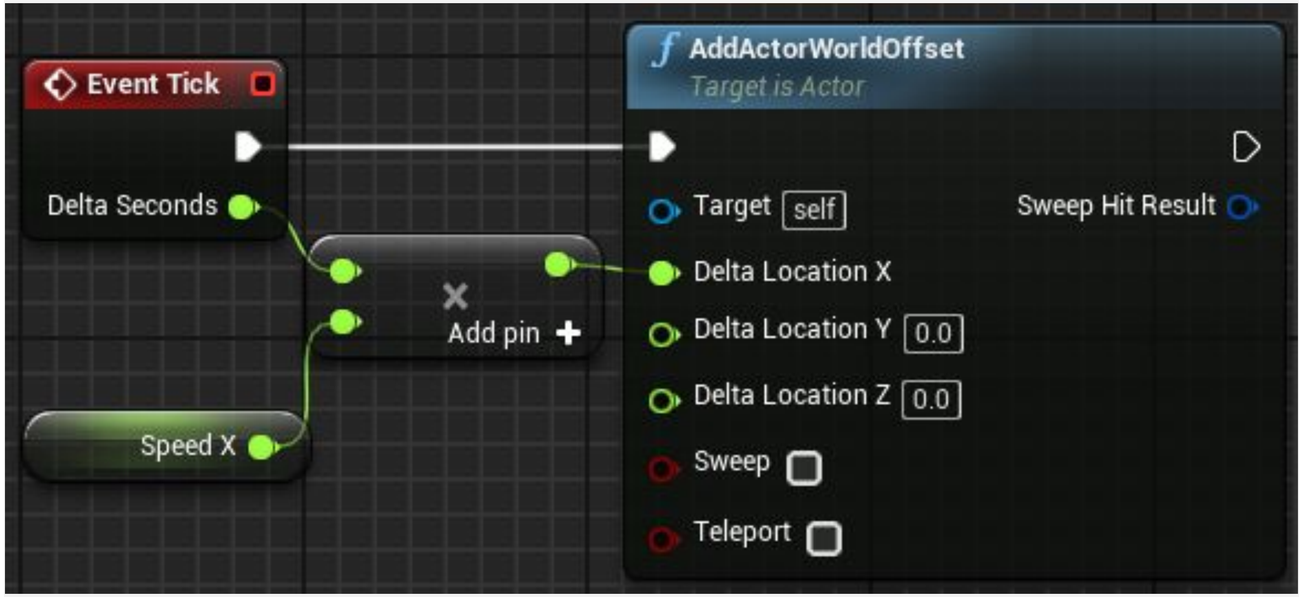
СОБЫТИЕ TICK И DELTA TIME

Есть событие под названием **Tick**, которое вызывается в каждом кадре игры. Например, в игре, которая работает со скоростью 60 кадров в секунду, событие **Tick** вызывается 60 раз в секунду.

Событие **Tick** имеет параметр, известный как **Delta Seconds**, который содержит количество времени, прошедшее с момента последнего кадра.

В событии **Tick**, показанном справа, значение **Delta Seconds** умножается на скорость, указанную в переменной **Speed X**, чтобы определить скорость (в см / с), с которой Актор должен перемещаться по оси **X** для каждого кадра.

Действия с отметками могут быть очень дорогими на просчет, поэтому по возможности следует рассмотреть альтернативы, такие как временные шкалы или таймеры.



ФУНКЦИИ INTERP TO

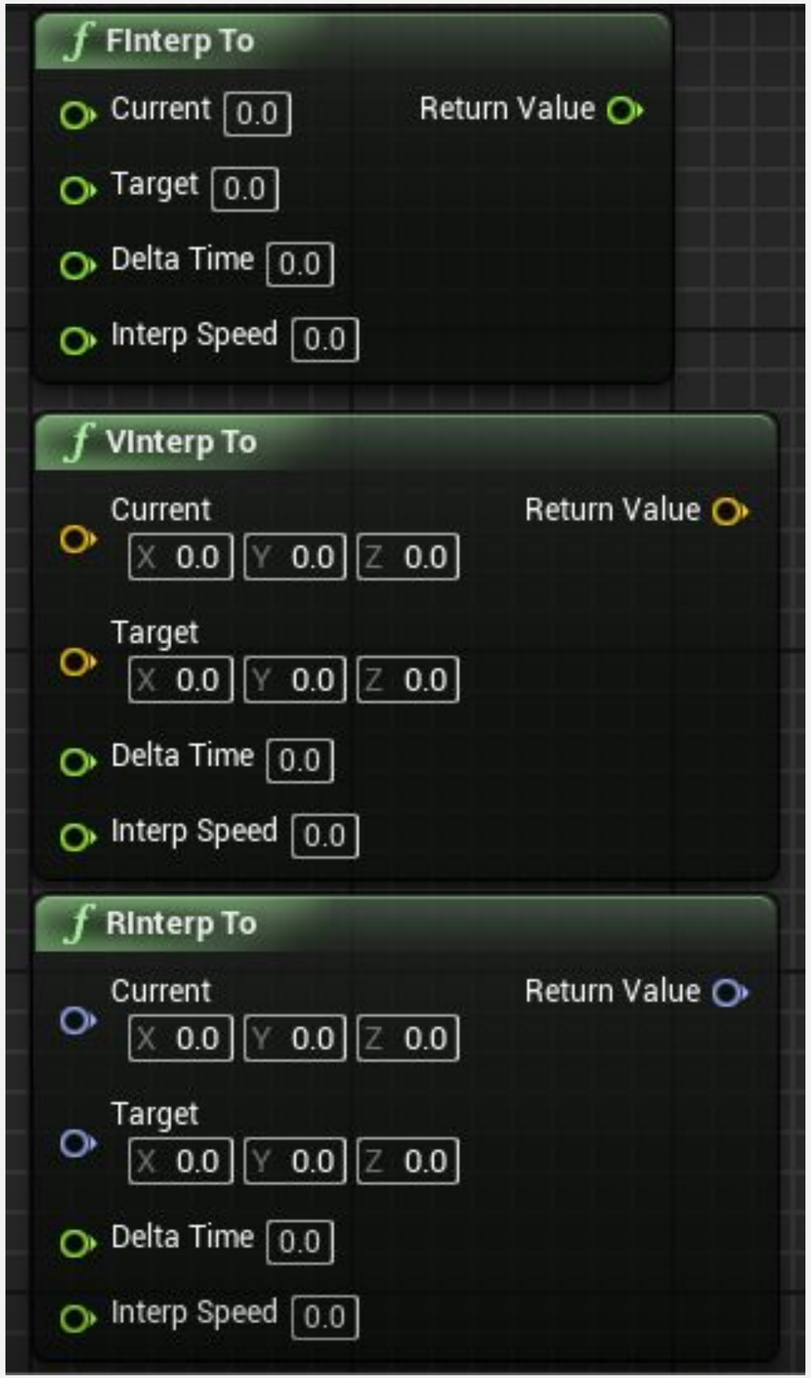
Функции **Interp To** используются для плавного изменения значения, пока оно не достигнет заданного целевого значения. Некоторые примеры включают функцию **FInterp To** для значений с плавающей запятой, **VInterp To** для векторов и функцию **RInterp To** для ротаторов.

Ввод

- **Current**: Текущее значение.
- **Target**: Целевое значение, которого необходимо достичь.
- **Delta Time**: Временной интервал, прошедший с момента последнего выполнения.
- **Interp Speed**: Скорость интерполяции.

Вывод

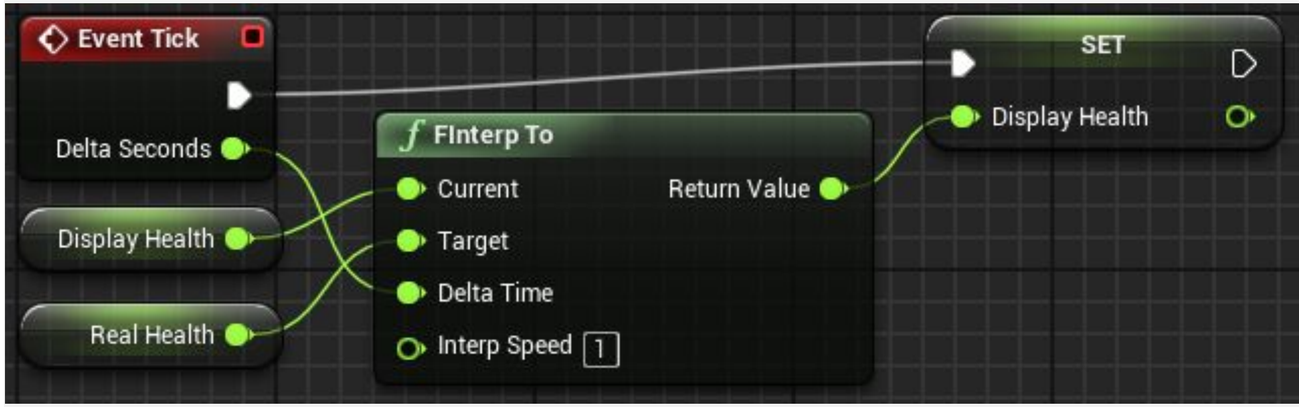
- **Return Value**: Новое значение ближе к целевому значению.



ФУНКЦИЯ INTERP TO: ПРИМЕР

Пример справа содержит две переменные. Переменная **Real Health** хранит текущее здоровье игрока. Переменная **Display Health** используется для отображения на экране шкалы работоспособности.

Когда игрок получает урон, значение **Real Health** немедленно изменяется, но значение **Display Health** изменяется с помощью функции **FInterp To**, так что полоса здоровья плавно уменьшается до тех пор, пока значение **Display Health** не станет равным значению **Real Health**.



ИТОГ

В этой лекции объясняется концепция трассировки. Она показала, как создавать и уничтожать звуки и частицы, а также как создавать различные типы анимации с помощью действий.