ЛЕКЦИЯ 8

Сообщение Блюпринтов

ЦЕЛИ И ВЫВОДЫ ЛЕКЦИИ

Goals

Цели этой лекции:

- Представить Сообщение Блюпринтов
- Показать, как использовать Прямое Сообщение Блюпринтов
- Объяснить отбор в Блюпринтах
- Показать, как ссылаться на Акторов в Level Blueprint
- Показать Диспетчеры Событий
- Представить интерфейс Блюпринтов

Outcomes

К концу этой лекции вы сможете

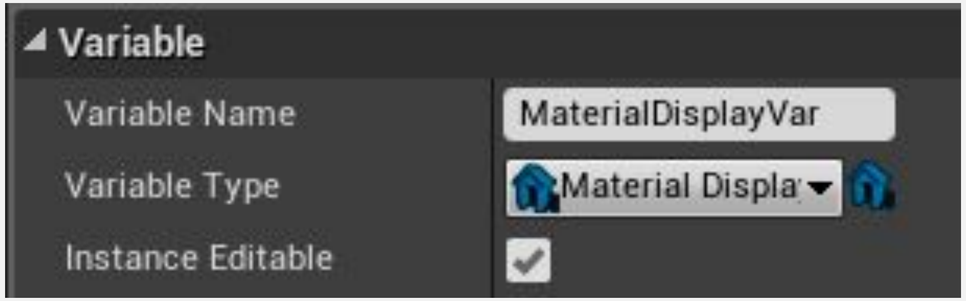
- Настроить взаимодействие между Блюпринтами
- Использовать отбор, чтобы получить ожидаемую ссылку на Блюпринт
- Создавать ссылки на акторов в Level Blueprint
- Создавать новые интерфейсы Блюпринтов

ПРЯМОЕ СООБЩЕНИЕ БЛЮПРИНТОВ

Direct Blueprint Communication - это простой метод связи между Blueprints. Его можно использовать, создав переменную, которая будет хранить ссылку на другой Blueprint.

На верхнем изображении справа показана переменная с именем «**MaterialDisplayVar**». Тип этой переменной - «**Material Display**», который является настраиваемым классом Blueprint.

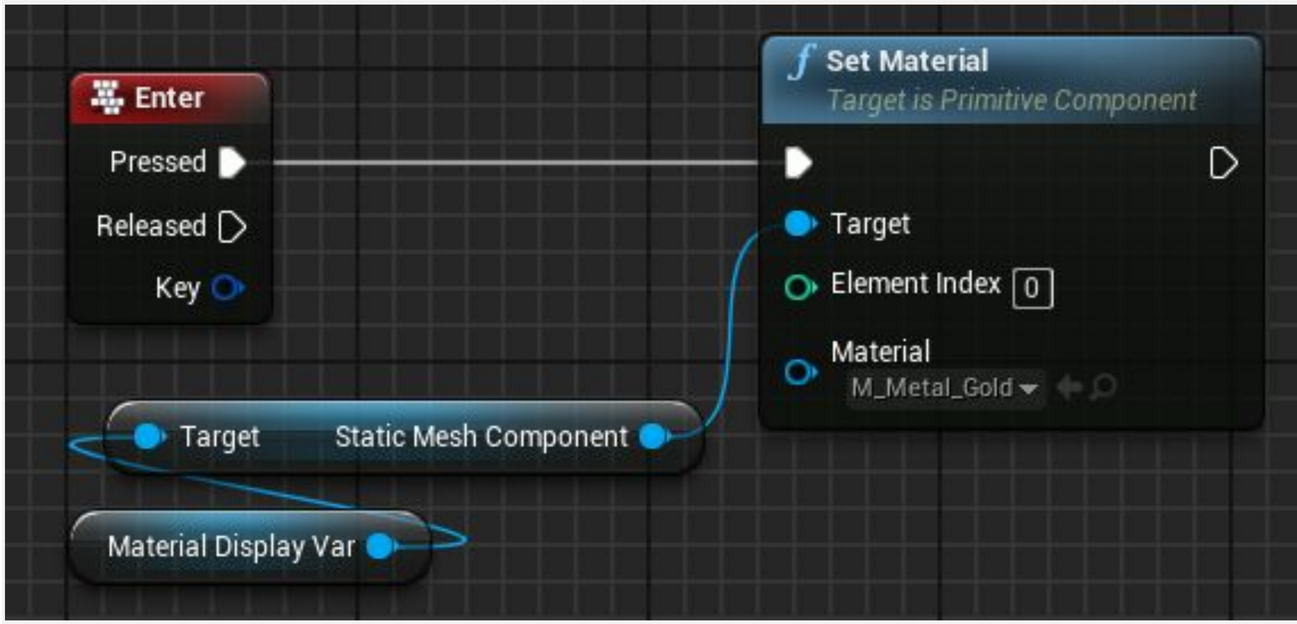
Поставлена галочка у свойства **Instance Editable**, поэтому эту переменную можно установить в Level Editor. На нижнем изображении показано раскрывающееся меню, в котором можно выбрать экземпляр класса **отображения материала** на уровне..



ПРЯМОЕ СООБЩЕНИЕ БЛЮПРИНТОВ: TARGET

На изображении справа показана функция **Set Material**Целью, используемой в функции, является компонент Static Mesh переменной **Material Display Var**.

Таким образом, Blueprint, который будет изменен функцией, является экземпляром **Material Display**, который был выбран в Level Editor.



ОТБОР В БЛЮПРИНТАХ

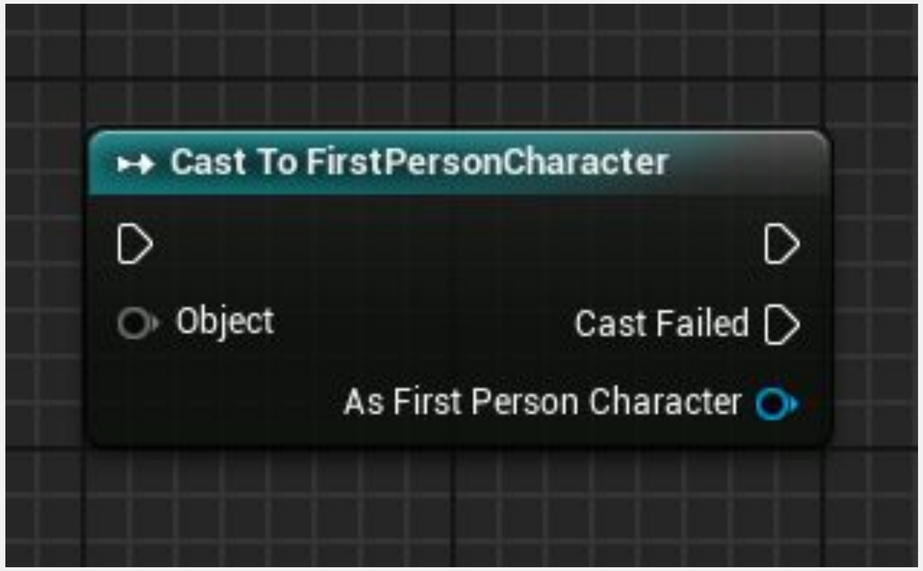
Нода **Cast To** преобразует тип ссылочной переменной в новый указанный тип. Это действие необходимо в некоторых ситуациях для доступа к переменным и функциям класса или Blueprint. На изображении справа показан проект **Cast To FirstPersonCharacter Blueprint**.

Ввод

- **Object**: Принимает ссылку на объект.

Вывод

- **Cast Failed**: Пин выполнения, который используется, если объект, на который указывает ссылка, не относится к типу, используемому в приведении.
- **As [new type]**: Выводит ссылку, используя новый тип, указанный в приведении.

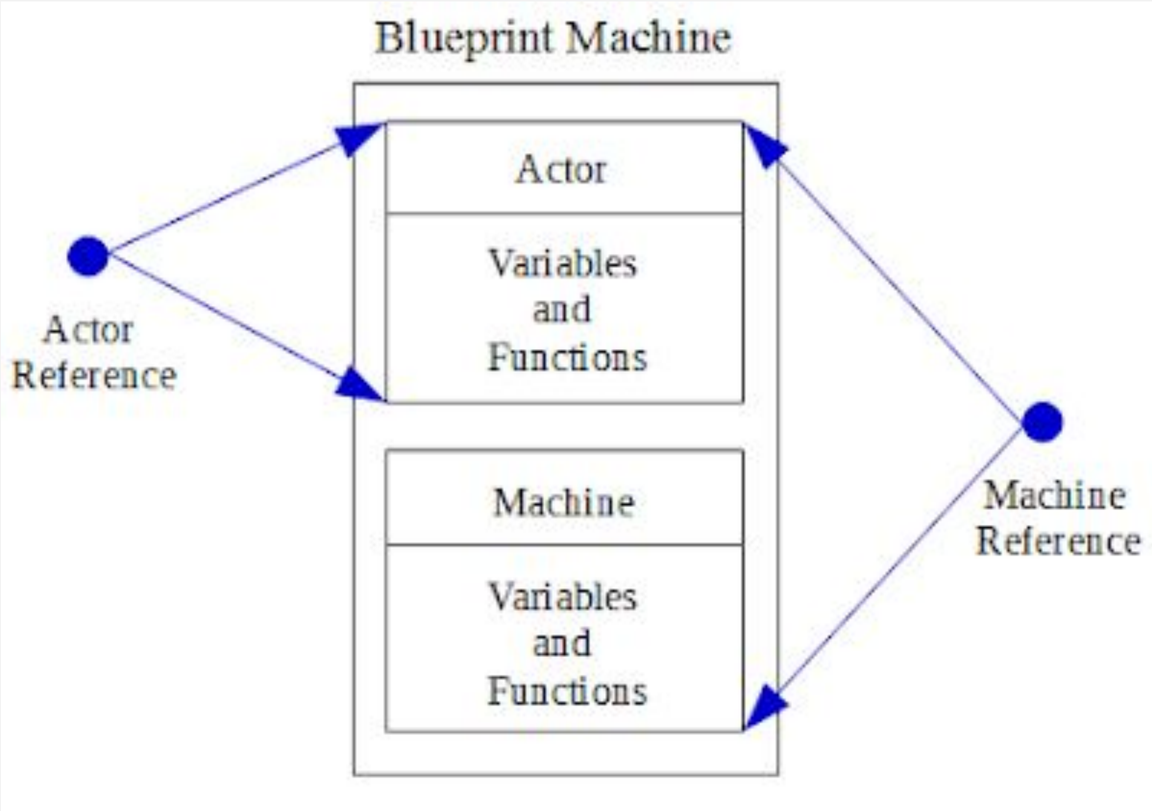


ОТБОР В БЛЮПРИНТАХ: ПРИМЕР 1/2

В этом примере был создан новый класс Blueprint под названием «**Machine**». Типа «**Actor**». В Blueprint есть функция под названием “**Recharge Battery**”.

Ссылка на класс **Actor** может использоваться для управления классом **Machine Blueprint**, но у него не будет доступа к функции **Recharge Battery**, поскольку он может получить доступ только к переменным и функциям, которые были созданы в типе **Actor**.

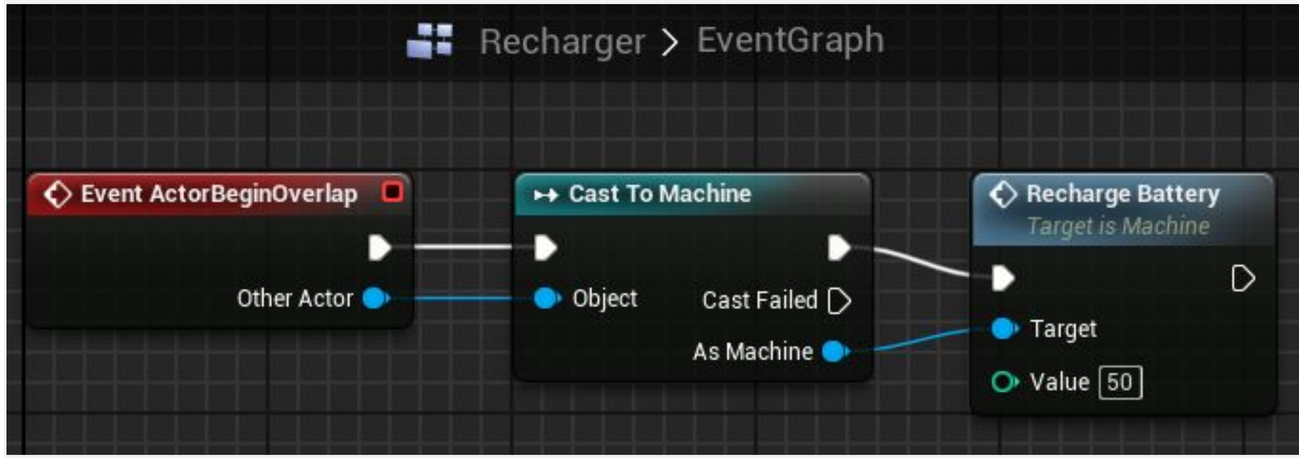
Для доступа к функции **Recharge Battery** необходима ссылка на класс **Machine**.



ОТБОР В БЛЮПРИНТАХ: ПРИМЕР 2/2

В другом классе Blueprint под названием «**Recharger**» есть событие Overlap.

Если перекрывающийся Актор относится к классу **Machine**, функция **Recharge Battery** вызывается с использованием ссылки на класс **Machine**, предоставленной в ноде **Cast To**.

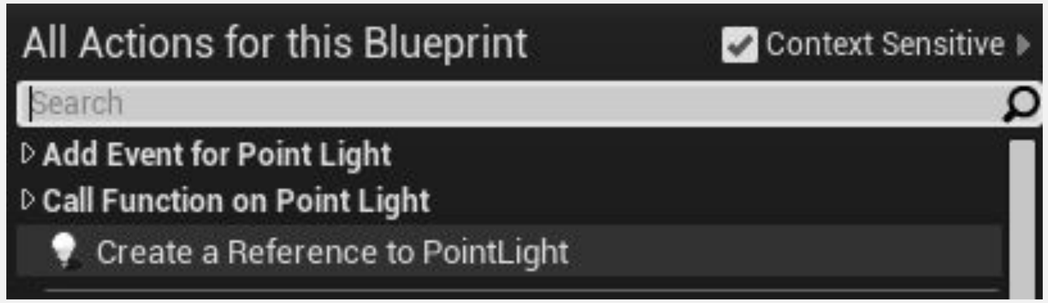


СООБЩЕНИЕ БЛЮПРИНТОВ УРОВНЕЙ 1/2

Легко добавлять ссылки на Level Actors в Level Blueprint, позволяя Level Blueprint напрямую взаимодействовать с ними.

Чтобы добавить ссылку, выполните следующие действия:

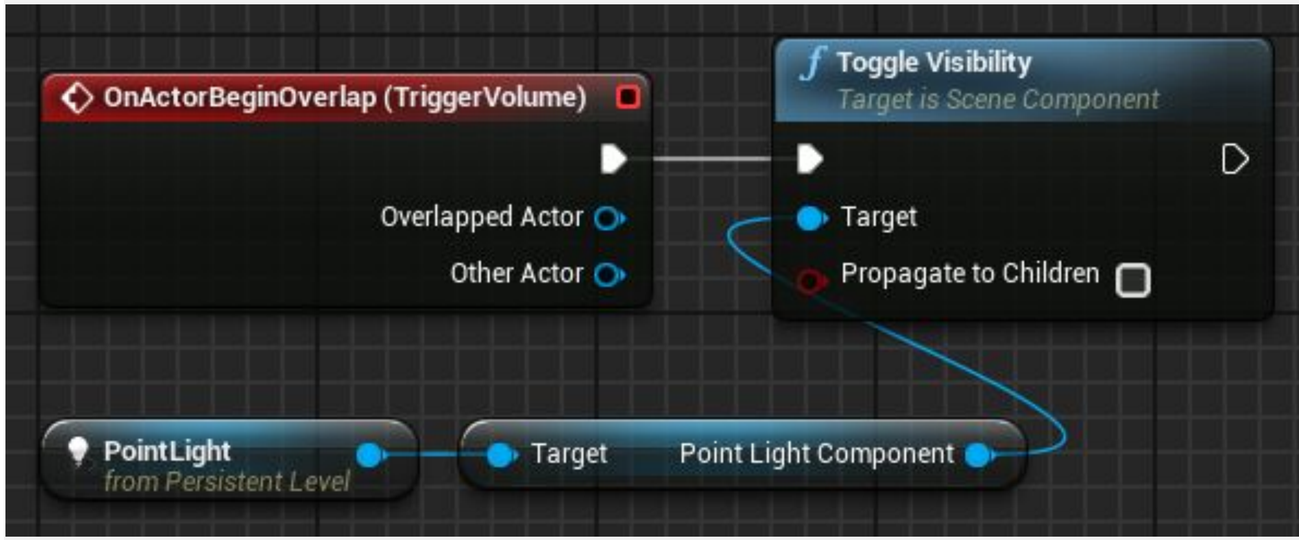
- Сначала выберите **Actor** в **Level Editor**.
- Откройте **Level Blueprint**.
- Нажмите ПКМ в **Event Graph** и выберите функцию “**Create a Reference to [Actor name]**”.



СООБЩЕНИЕ БЛЮПРИНТОВ УРОВНЕЙ 2/2

На изображении справа событие перекрытия **Trigger Volume** было помещено в Level Blueprint.

Когда происходит перекрытие, функция **Toggle Visibility** вызывается для компонента **Point Light Actor**.



ДИСПЕТЧЕР СОБЫТИЙ

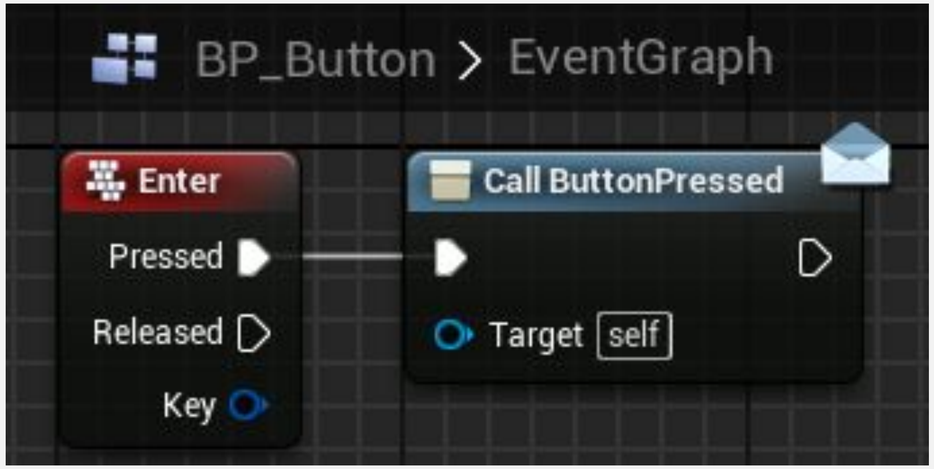
Диспетчер событий обеспечивает связь между классами Blueprint и Level Blueprint. Он создается в классе Blueprint и может быть реализован в Level Blueprint.

Диспетчеры событий создаются на панели My Blueprint. На изображении справа показан диспетчер событий, созданный с именем «**ButtonPressed**»

ДИСПЕТЧЕР СОБЫТИЙ: ПРИМЕР 1/3

В этом примере класс Blueprint с именем «**BP Button**» представляет кнопку, которую можно нажимать. Единственная цель этого Blueprint - общаться при нажатии кнопки. В результате кнопку можно использовать в нескольких различных ситуациях. Действия, которые будут происходить при нажатии кнопки, будут установлены в Level Blueprint.

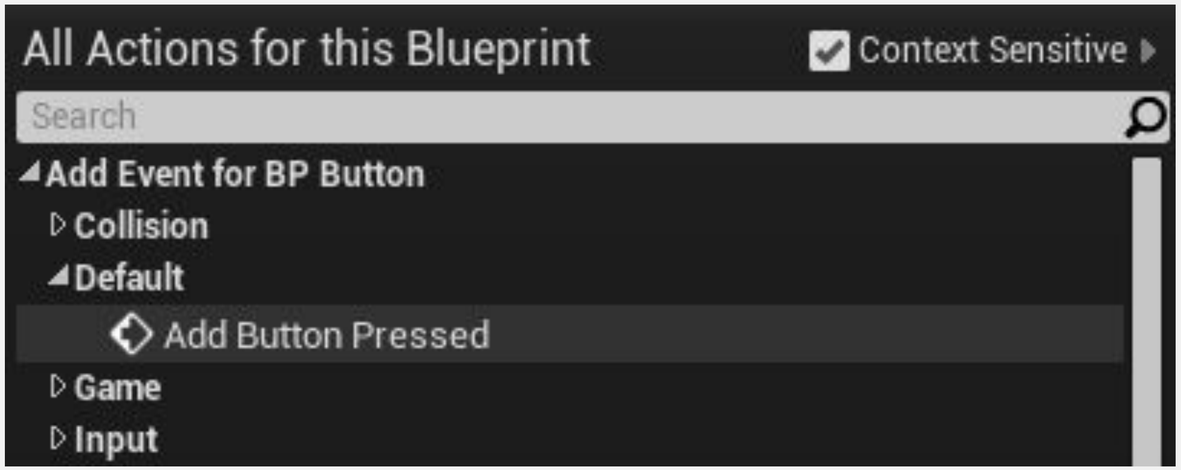
BP_Button Blueprint содержит событие клавиатуры **Enter**. При нажатии клавиши **Enter** вызывается диспетчер событий **ButtonPressed**, как показано на рисунке справа.



ДИСПЕТЧЕР СОБЫТИЙ: ПРИМЕР 2/3

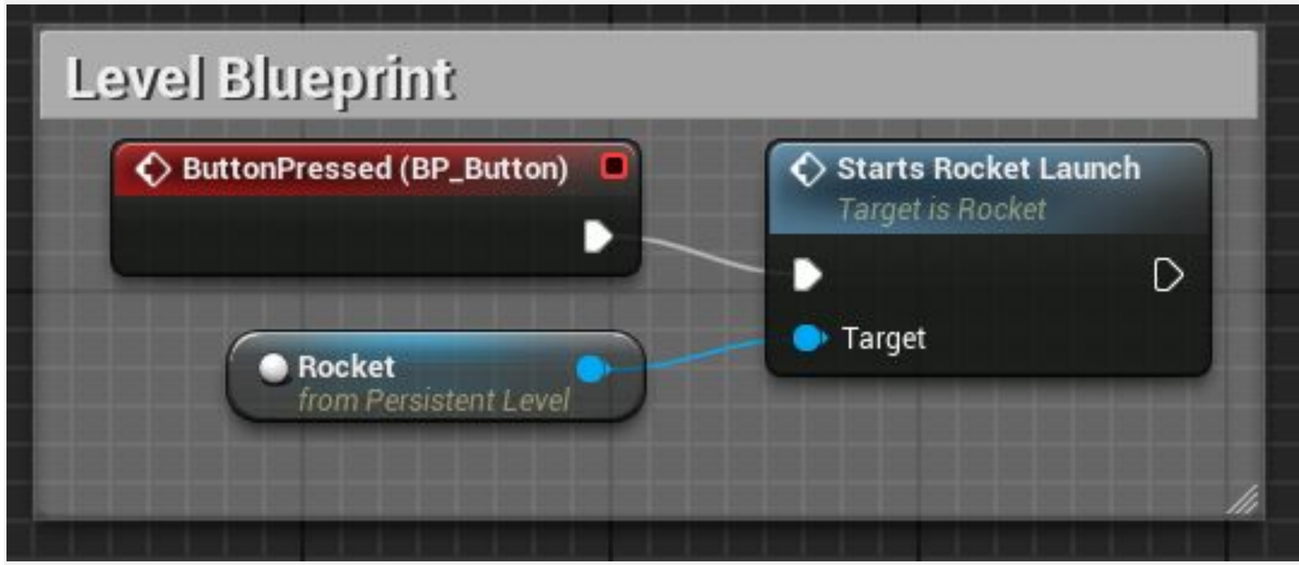
Добавьте Актор **BP_Button** на уровень и выберите его.

Откройте **Level Blueprint** и нажмите ПКМ по **Event Graph** чтобы добавить событие **ButtonPressed** связанное с созданным диспетчером событий.



ДИСПЕТЧЕР СОБЫТИЙ: ПРИМЕР 3/3

Предположим, что на уровне есть чертеж под названием “**Rocket**”, который содержит функцию под названием “**Starts Rocket Launch**”. Level Blueprint отвечает за вызов функции **Starts Rocket Launch** при нажатии указанной кнопки .

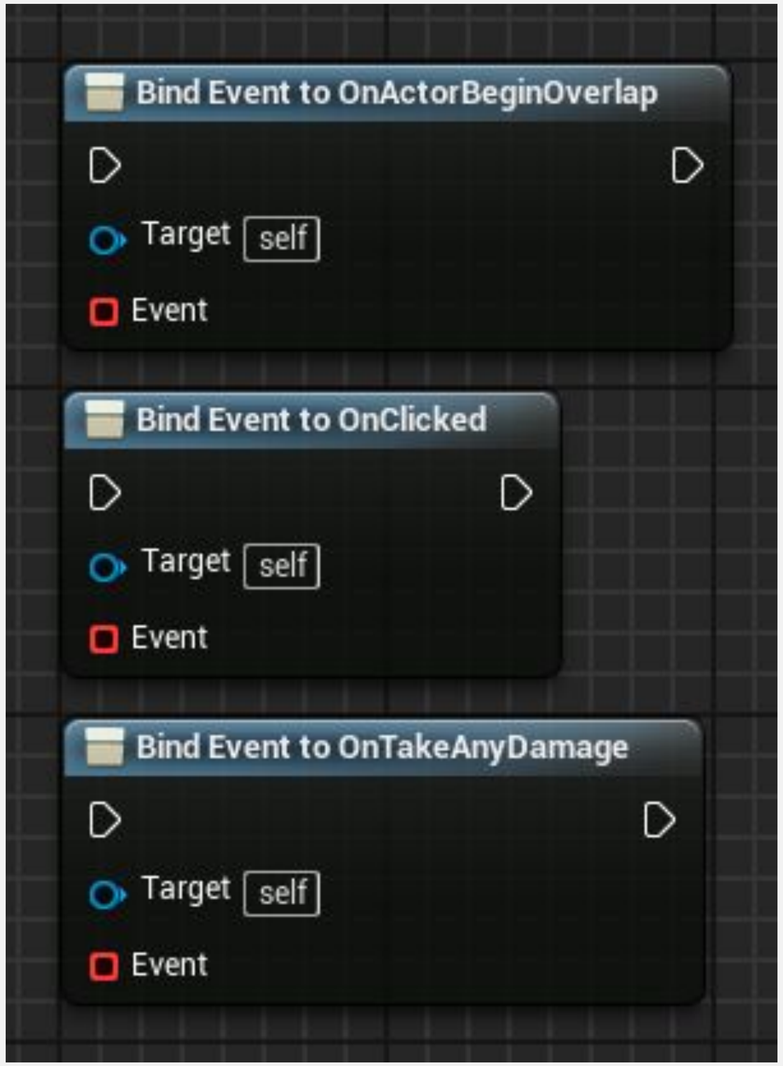


НОДА BIND EVENT

Нода **Bind Event** связывает одно событие с другим событием или с диспетчером событий, который может находиться в другом Blueprint. Когда вызывается событие, также вызываются все другие связанные с ним события. Привязка выполняется с помощью event delegate (красный пин на ноде события).

Ввод

- **Target**: Объект, имеющий событие, которое получает привязку.
- **Event**: Ссылка на событие, которое будет связано с другим событием, которое будет вызвано позже.

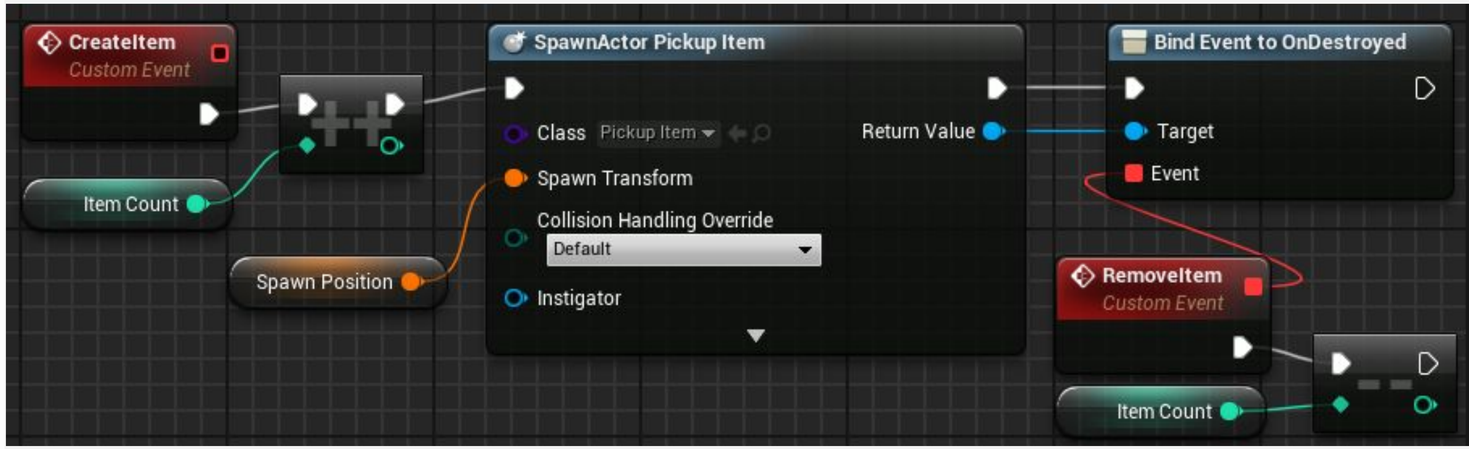


НОДА BIND EVENT: ПРИМЕР

В этом примере есть класс Blueprint под названием «**PickupManager**», который отвечает за создание Акторов **Pickup Item** и содержит переменную под названием “**Item Count**” , которая отслеживает количество Акторов **Pickup Item**, которые существуют на уровне.

Настраиваемое событие под названием «**RemoveItem**» уменьшает на «**1**» значение переменной «**Item Count**» всякий раз, когда порожденный «**Pickup Item Actor**» больше не существует на уровне.

Событие **RemoveItem** привязано к событию **OnDestroyed** созданного объекта **Pickup Item**. Таким образом, когда в игре уничтожается Актор **Pickup Item**, вызывается событие **RemoveItem**.

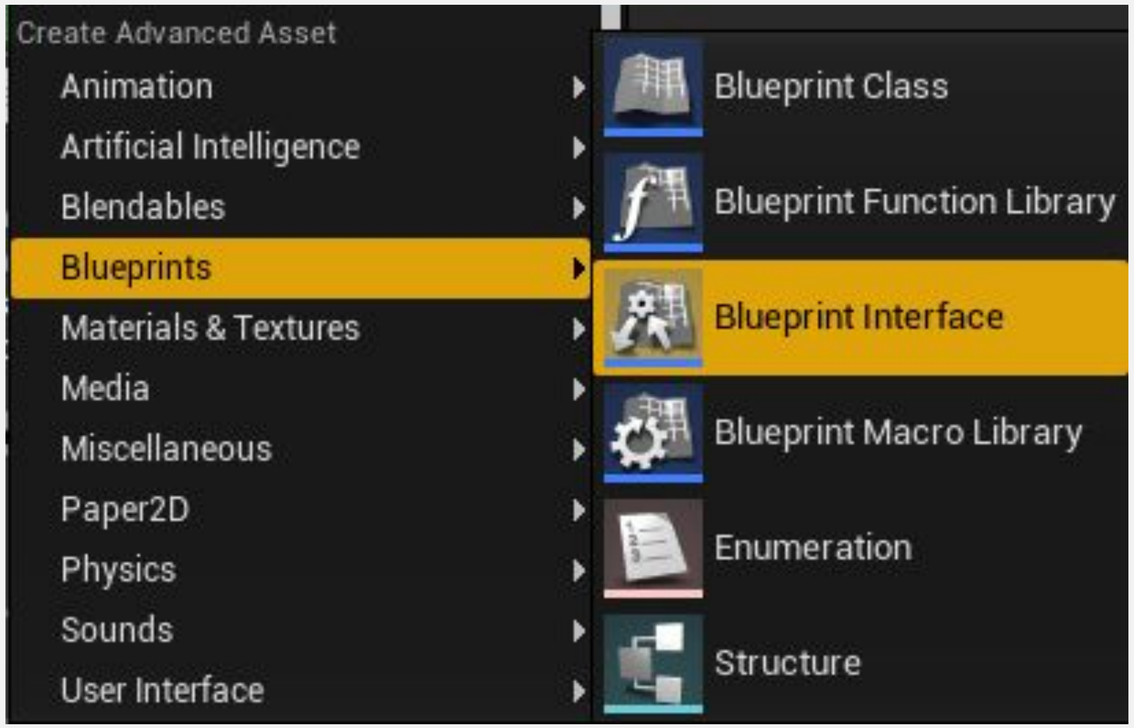


ИНТЕРФЕЙС БЛЮПРИНТ

Blueprint Interface (**BPI**) содержит только определения функций, без реализации.

Если класс Blueprint реализует BPI, он использует предоставленное определение, а затем реализует свою собственную логику для этой функции.

Чтобы создать новый Blueprint Interface, кликните по зеленой кнопке **Add New** в **Content Browser** и в подменю **Blueprints** выберите “**Blueprint Interface**”.

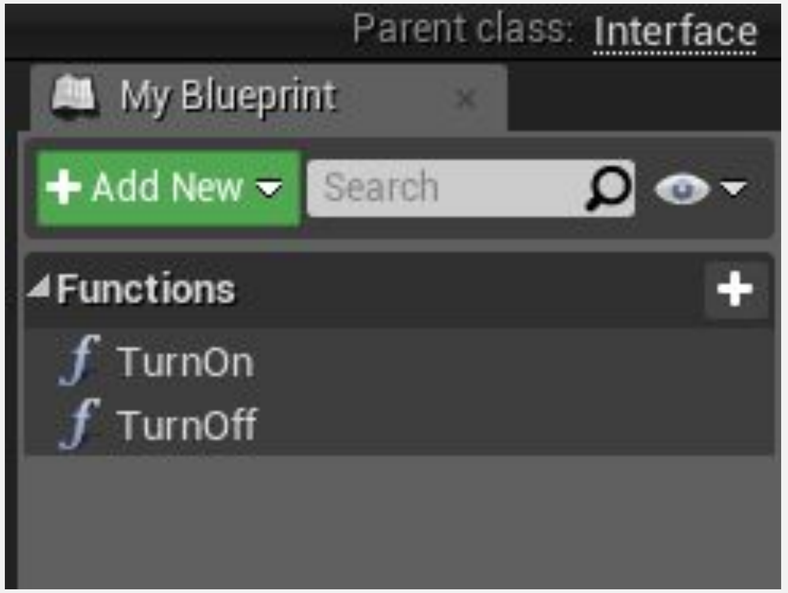


ИНТЕРФЕЙС БЛЮПРИНТ: СОЗДАНИЕ ФУНКЦИИ

В этом примере был создан интерфейс Blueprint с именем “**BP_Interface_TurnOnOff**”.

При редактировании BPI можно называть функции и создавать параметры ввода и вывода, но невозможно реализовать логику в интерфейсе.

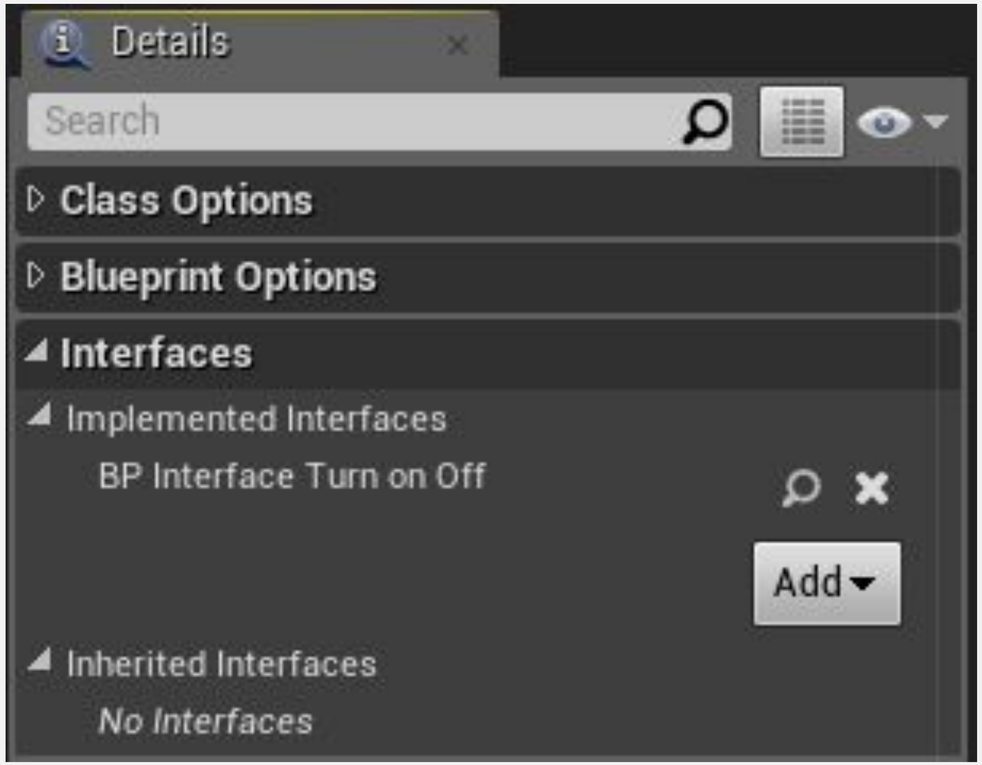
Этот BPI имеет две функции: **TurnOn** и **TurnOff**, как показано на изображении справа.



ИНТЕРФЕЙС БЛЮПРИНТ: ДОБАВЛЕНИЕ В БЛЮПРИНТ

Чтобы добавить интерфейс Blueprint к классу Blueprint, нажмите кнопку **Class Settings** на панели инструментов **Toolbar** в редакторе **Blueprint Editor**. В разделе **Interfaces** на панели **Detail** нажмите кнопку **Add** и выберите класс Blueprint Interface.

На изображении справа был добавлен **BP Interface Turn On Off**.

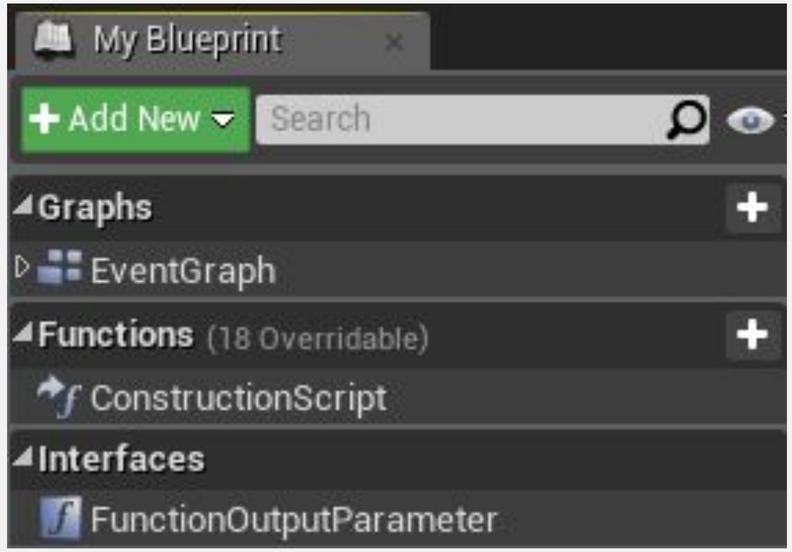


ИНТЕРФЕЙС БЛЮПРИНТ: РЕАЛИЗАЦИЯ ФУНКЦИЙ

Функции интерфейса Blueprint, не имеющие выходных параметров, появляются как события в Blueprint, реализующем **BPI**.

Функции интерфейса Blueprint, у которых есть выходной параметр, отображаются на панели **My Blueprint** в разделе **Interfaces**.

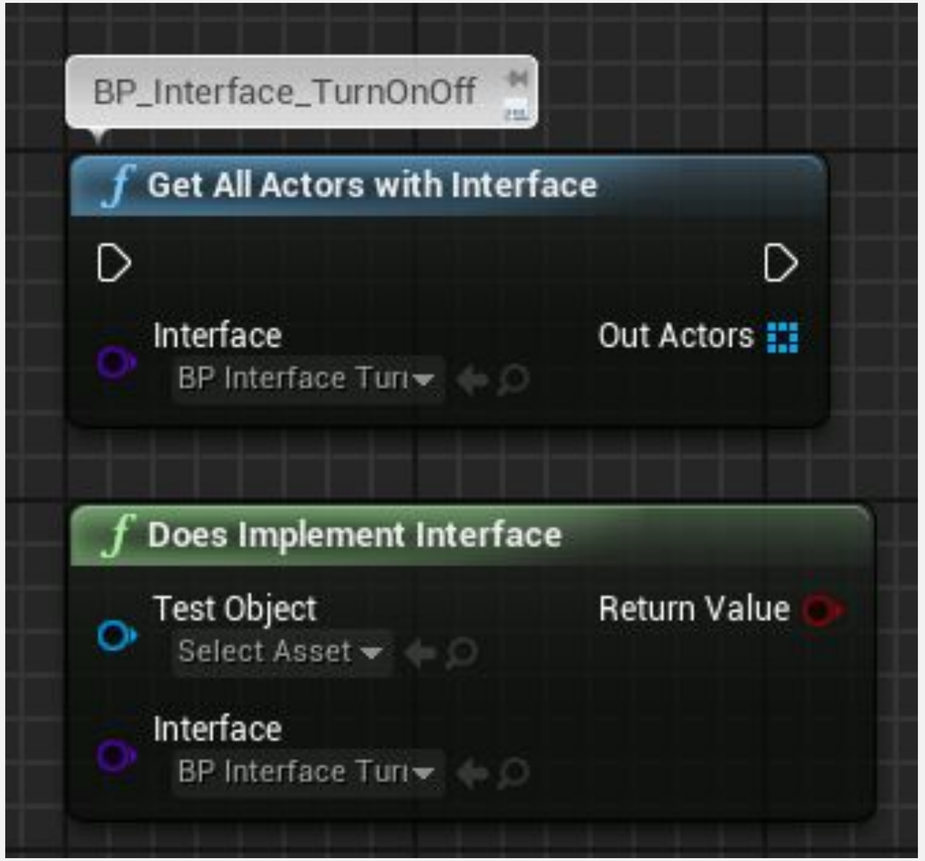
На нижнем изображении справа показана функция с именем «**FunctionOutputParameter**». Функция реализуется двойным щелчком по ней, чтобы открыть функцию для редактирования.



ИНТЕРФЕЙС БЛЮПРИНТ: СЛУЖЕБНЫЕ ФУНКЦИИ

Есть некоторые служебные функции, связанные с интерфейсами Blueprint. Ниже приведены два примера:

- **Get All Actors with Interface**: Находит всех Акторов на текущем уровне, которые реализуют указанный BPI.
- **Does Implement Interface**: Проверяет, реализует ли один конкретный объект BPI.



ИТОГ

Эта лекция представила Blueprint Communication и показала, как использовать Direct Blueprint Communication, интерфейсы Blueprint и диспетчеры событий.

Она также объяснила, как приводить, связывать события и ссылаться на Акторов в Level Blueprint