UNREAL
ENGINE

ЛЕКЦИЯ 7

Продвинутые основы Блюпринтов 2

ЦЕЛИ И ИТОГИ ЛЕКЦИИ

Goals

Цели этой лекции:

- Представить ноды управления потоком
- Показать, как перебирать массив с помощью ноды ForEachLoop
- Показать, как работать со строками
- Представить ноду Math Expression
- Объяснить функции случайных чисел

Outcomes

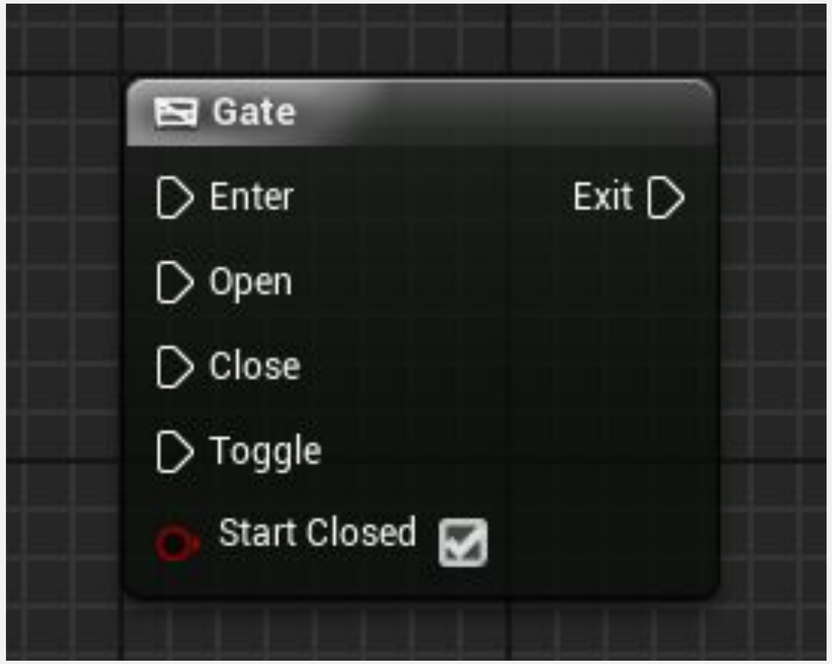
К концу этой лекции вы сможете

- Выбрать лучшую ноду управления потоком для каждой ситуации
- Иметь доступ к элементам массива с помощью узла ForEachLoop
- Форматировать и конвертировать строки
- Создавать выражения с помощью ноды Math Expression
- Использовать случайные числа и переменные Random Stream

УПРАВЛЕНИЕ ПОТОКОМ

НОДА GATE

Нода **Gate** - это нода управления потоком, которая может быть открытой или закрытой. Если она открыта, она позволяет выполнять действия, связанные с пином **Exit**.



НОДА GATE: ВВОД И ВЫВОД

Ввод

- **Enter**: Пин выполнения, который получает текущий поток выполнения.
- **Open**: Пин выполнения, который изменяет состояние Gate на «открыто».
- **Close**: Пин выполнения, который изменяет состояние Gate на «закрыто».
- **Toggle**: Пин выполнения, который переключает текущее состояние Gate.
- **Start Closed**: логическая переменная, которая определяет, должна ли нода **Gate** начать работу в «закрытом» состоянии.

Вывод

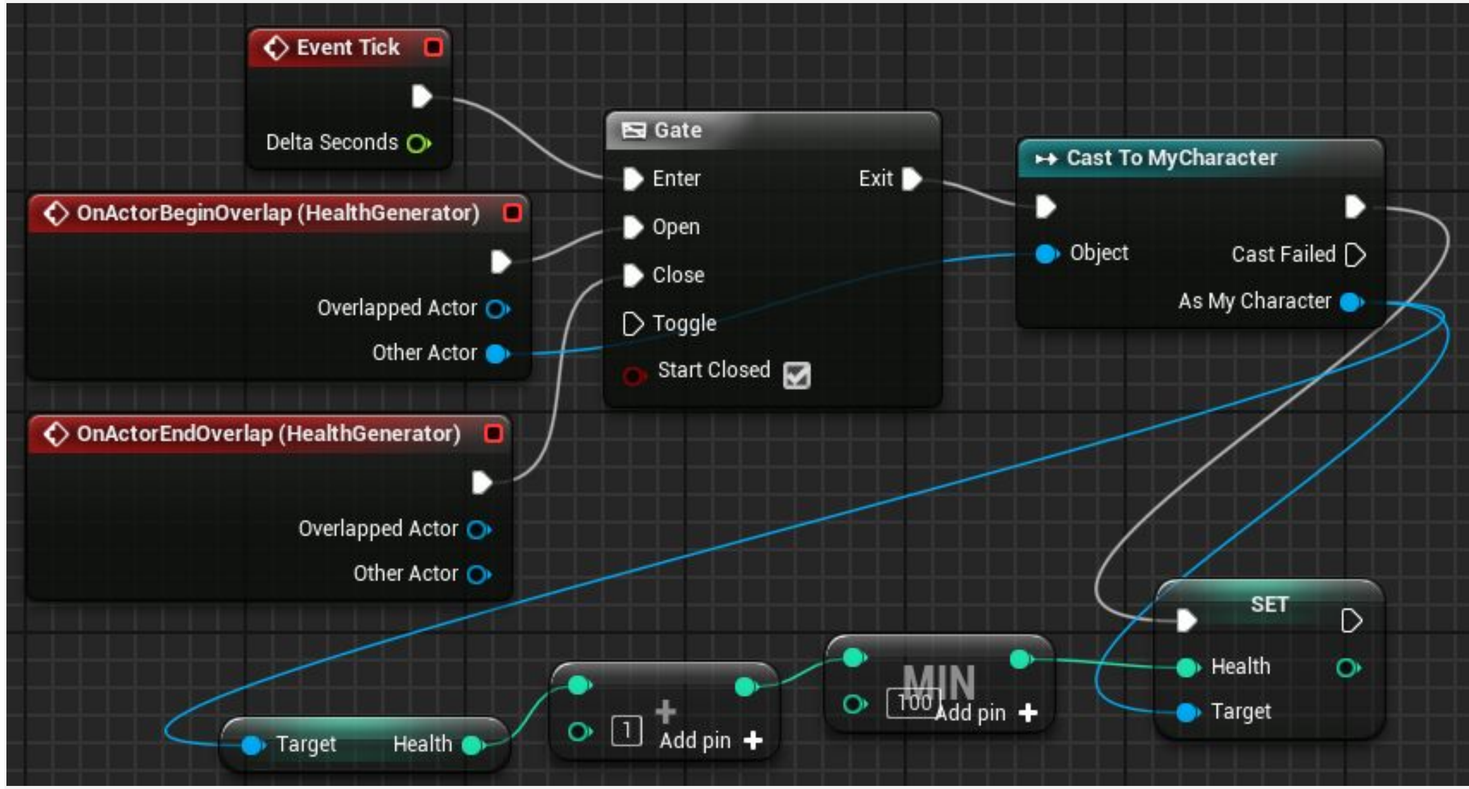
- **Exit**: Пин выполнения, который будет выполнен, если Gate открыта.

НОДА GATE: ПРИМЕР

В примере справа есть Актор с именем «**HealthGenerator**». Когда игрок сталкивается с этим Актором, его здоровье будет медленно восстанавливаться с каждым событием **Tick**.

Если игрок перестанет сталкиваться с **HealthGenerator**, Gate закроется действия, восстанавливающие здоровье, больше не будут выполняться.

Действие **Min** используется для того, чтобы значение переменной **Health** никогда не превышало «100».

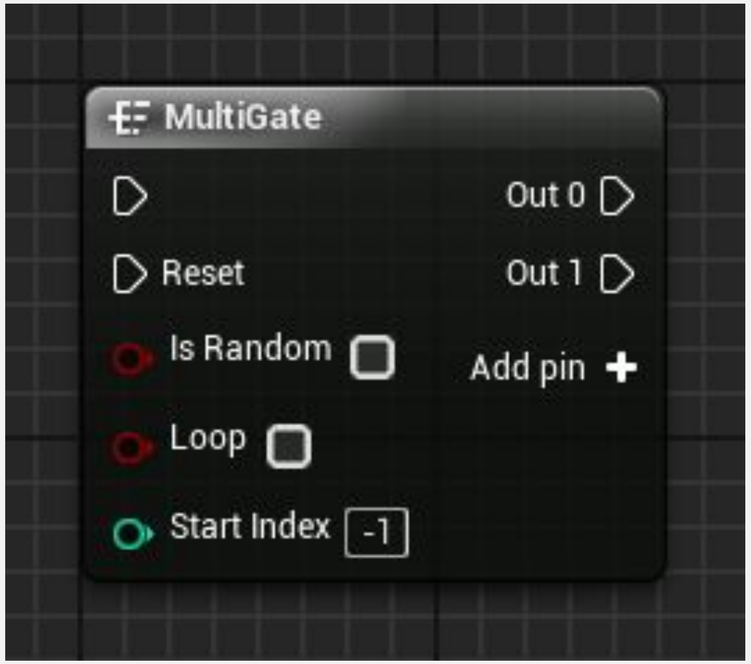


НОДА MULTIGATE

Нода **MultiGate** может иметь несколько выходных пинов. При каждом запуске **MultiGate** выполняется только один из выходных пинов. Порядок выполнения выходных пинов может быть последовательным или случайным.

Когда все выходные пины выполнены и если опция **Loop** не выбрана, нода **MultiGate** прекратит выполнение выходных пинов. Чтобы **Multigate** снова запустила выходные пины, должен сработать пин **Reset**.

Выходные пины могут быть добавлены с помощью опции **Add pin +**. Чтобы удалить пин, щелкните его ПКМ и выберите“**Remove execution pin**”.



НОДА MULTIGATE: ВВОД

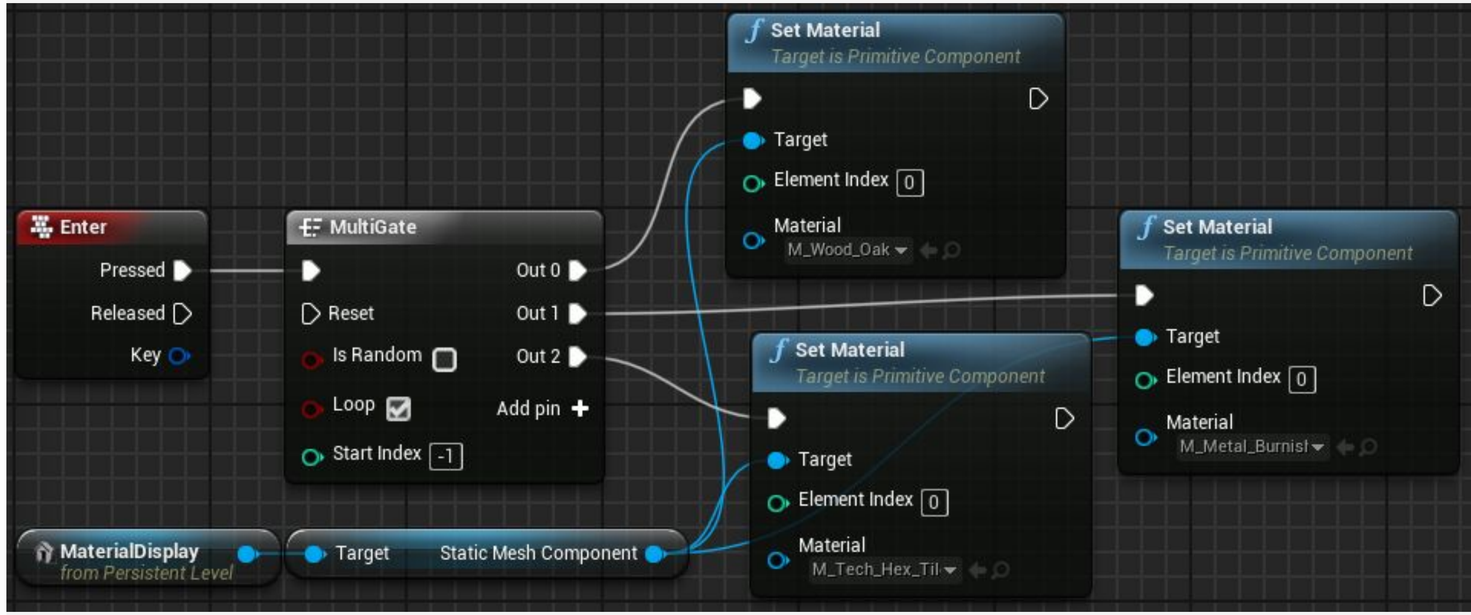
- **Reset**: Пин выполнения, используемый для сброса ноды **MultiGate** и разрешения новых запусков пинов вывода.
- **Is Random**: Логическая переменная. Если значение «**true**», порядок выполнения выходных пинов является случайным.
- **Loop**: Логическая переменная. Если значение равно «**true**», **MultiGate** продолжает выполнять выходные пины после выполнения последнего выходного пина.
- **Start Index**: Принимает целочисленное значение, указывающее на первый выходной пин, который должен быть выполнен.

НОДА MULTIGATE: ПРИМЕР

В примере справа на уровне есть объект под названием «**MaterialDisplay**», функция которого заключается в отображении различных Материалов для пользователя.

При нажатии клавиши **Enter** нода **MultiGate** используется для определения разных Материалов при каждом выполнении.

Поскольку параметр **Loop** проверяется, после выполнения последнего выходного пина **MultiGate** продолжит выполнение выходных пинов, начиная с первого выходного пина.



НОДА DO ONCE

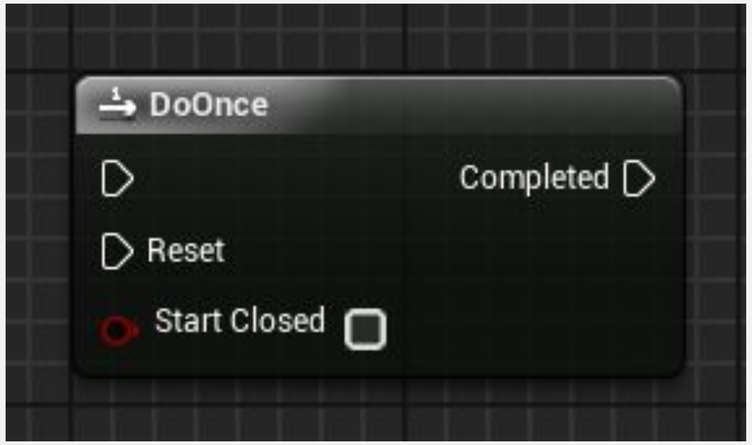
Нода **DoOnce** выполняет действия, связанные с выходным пином, но только один раз.

Если после первого запуска нода **DoOnce** вызывается снова, ее выходной пин не запускается.

Чтобы нода **DoOnce** могла снова выполнить выходной пин, необходимо активировать пин **Reset**.

Ввод

- **Reset**: Пин выполнения, позволяющий ноде **DoOnce** запускать пин вывода.
- **Start Closed**: Логическая переменная. Если значение равно «**true**», ноду **DoOnce** необходимо сбросить, чтобы разрешить первый запуск.

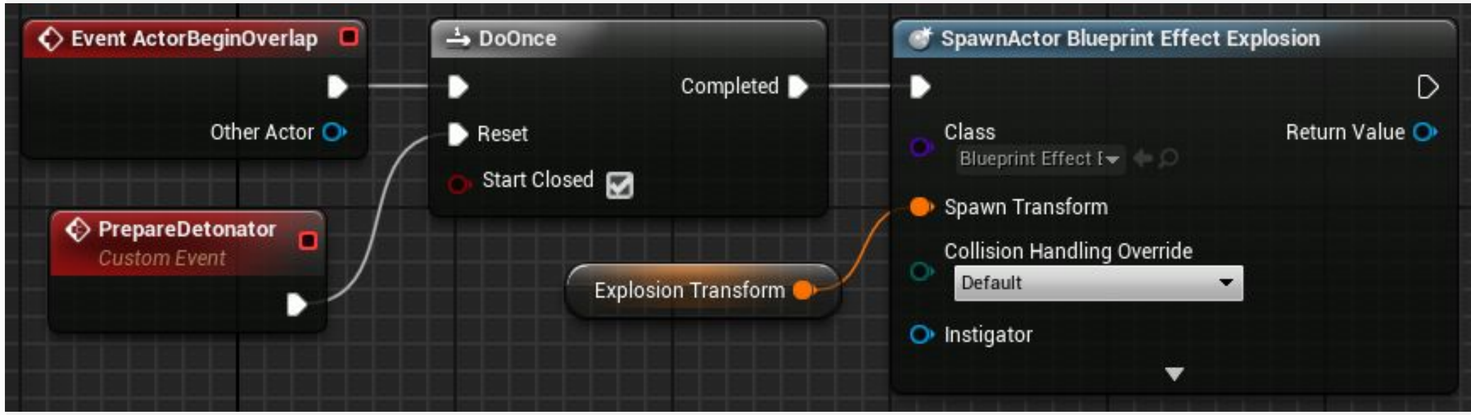


НОДА DO ONCE: ПРИМЕР

В примере справа есть детонатор, который производит взрыв, когда игрок сталкивается с ним. Этот детонатор использует действие **DoOnce**. Проверяется свойство **Start Closed**, что указывает на то, что детонатор запускается в отключенном состоянии.

Событие **PrepareDetonator** необходимо выполнить, чтобы активировать пин сброса (Reset) действия **DoOnce**.

После срабатывания пина **Reset** при столкновении с детонатором произойдет взрыв. Чтобы разрешить новый взрыв, событие **PrepareDetonator** необходимо выполнить снова.



НОДА DO N

Нода **Do N** похожа на ноду **DoOnce**, но вместо того, чтобы выполняться только один раз, действия, связанные с выходным пином, могут выполняться несколько раз.

После того, как заданное количество выполнений завершится, действия выходного пина будут выполнены снова, только если сработает пин сброса Reset.

Ввод

- **N**: Устанавливает, сколько раз действия выходного пина могут быть выполнены.
- **Reset**: Пин выполнения, используемый для сброса счетчика **Do N** и разрешения новых запусков вывода.

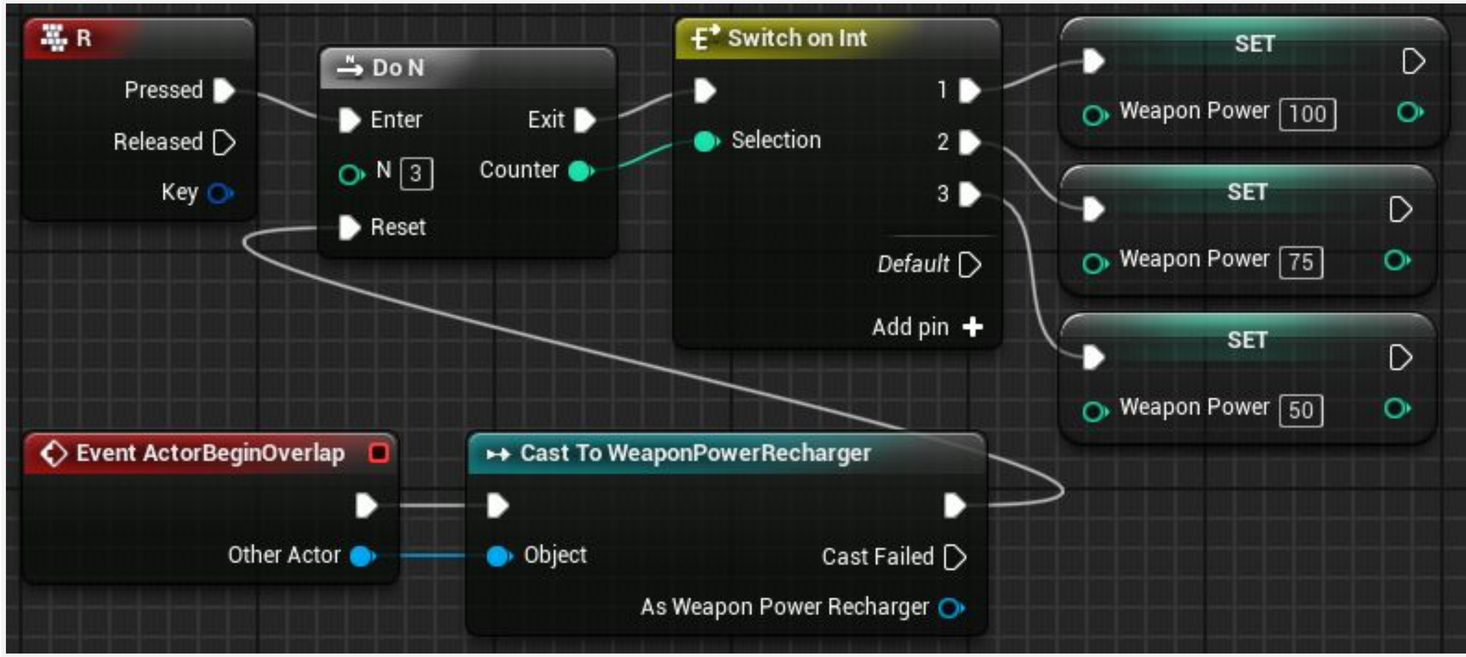
Вывод

- **Counter**: Выводит целочисленное значение, указывающее текущий номер выполнения.

НОДА DO N: ПРИМЕР

В примере справа у игрока есть особое оружие. Когда оружие полностью заряжено, его мощность составляет 100%. Игрок может нажать клавишу «**R**», чтобы перезарядить это оружие. Оружие можно перезарядить трижды.

Первая перезарядка оставляет оружие полностью заряженным; вторая перезарядка оставляет его на 75% емкости; а третья перезарядка восстанавливает только половину мощности оружия. Чтобы снова перезарядить оружие, игроку необходимо собрать предмет типа «**WeaponPowerRecharger**».



НОДА FLIP FLOP

Нода **FlipFlop** имеет два выходных пина, обозначенных как «**A**» и «**B**». Когда выполняется триггер, выполняется только один из выходных пинов. При следующем запуске будет выполнен только другой пин вывода.

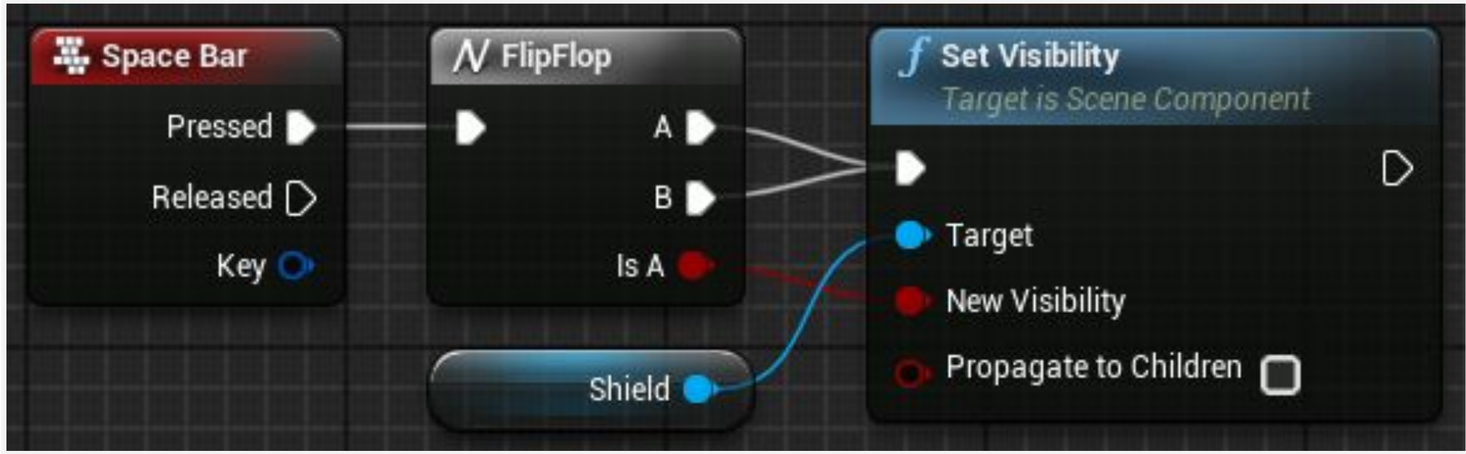
Вывод

- **A**: Пин выполнения, который будет выполняться, если значение пина **Is A** “**true**”.
- **B**: Пин выполнения, который будет выполняться, если значение вывода **Is A** равно “**false**”.
- **Is A**: Логическая переменная. Если значение «**true**», пин **A** работает. Если «**false**», пин **B** работает.

НОДА FLIP FLOP: ПРИМЕР

В примере справа нода **FlipFlop** используется для отображения или скрытия щита при нажатии клавиши **пробела**.

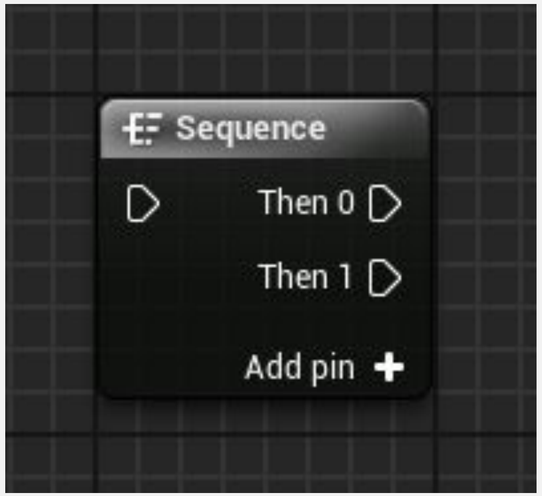
Значение выходного пина **Is A** используется для определения видимости щита.



НОДА SEQUENCE

Нода **Sequence** может использоваться для помощи в организации других действий Blueprint. При срабатывании триггера он выполняет все ноды, подключенные к выходным пинам, в последовательном порядке, то есть выполняет все действия пина **Then 0**, затем все действия пина **Then 1** и так далее.

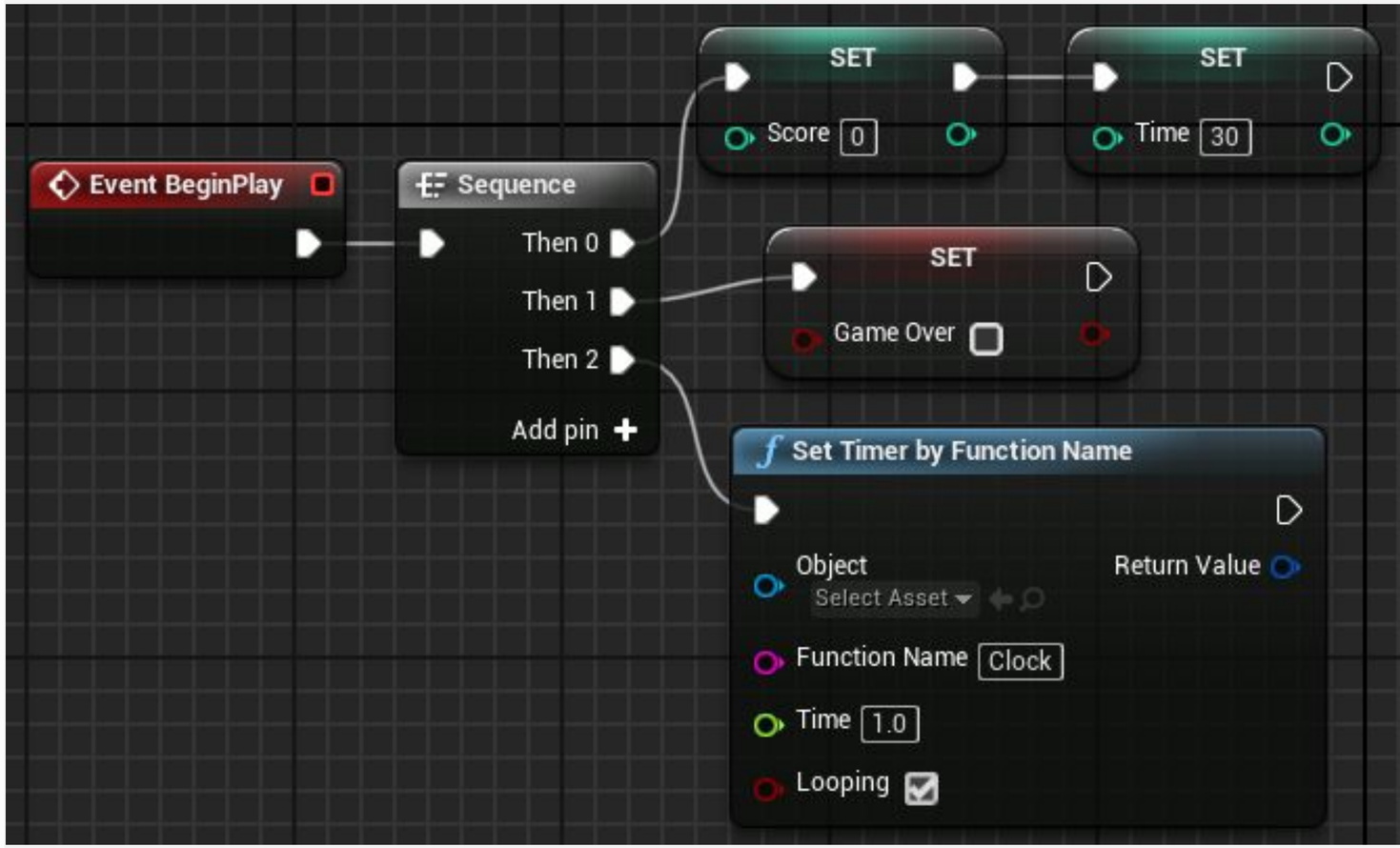
Выходные пины могут быть добавлены с помощью опции **Add pin +**. Чтобы удалить пин, щелкните по нему ПКМ и выберите параметр **Remove execution pin**.



НОДА SEQUENCE: ПРИМЕР

В примере справа нода **Sequence** используется для организации действий, которые будут выполняться после события **BeginPlay**.

Вместо использования одной строки выполнения для всех действий, нода **Sequence** используется для группировки действий по сходству.



НОДА FOR EACH LOOP

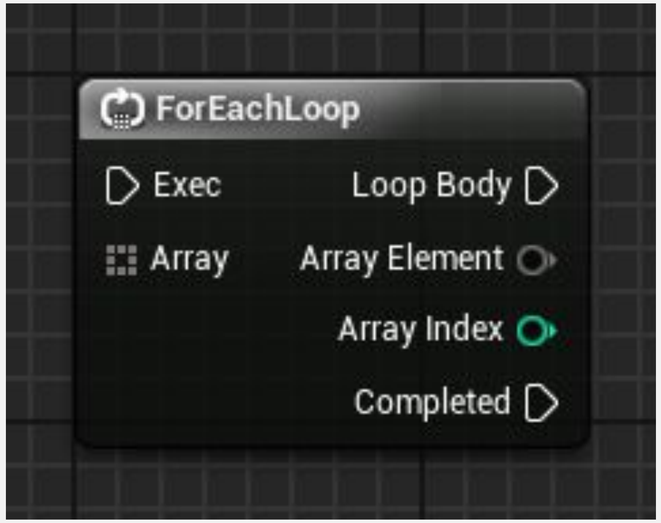
Нода **ForEachLoop** принимает массив в качестве входного параметра и выполняет набор действий, связанных с выходным пином **Loop Body** для каждого элемента массива, который может быть получен с выходного пина **Array Element**. После этого поток выполнения направляется на выходной пин **Completed**.

Ввод

- **Array**: Принимает массив, содержащий элементы, которые будут использоваться в цикле.

Вывод

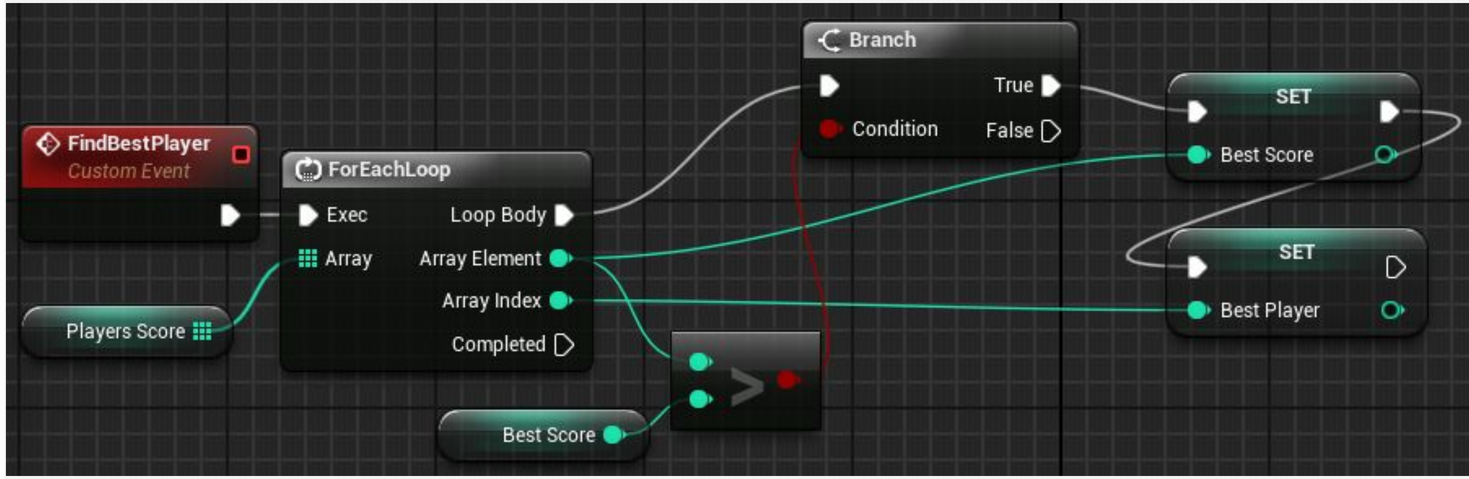
- **Array Element**: Выводит ссылку на текущий элемент массива.
- **Array Index**: Выводит индекс текущего элемента массива.



НОДА FOR EACH LOOP: ПРИМЕР

В примере справа нода **ForEachLoop** используется для итерации по массиву, содержащему оценки игроков.

Для каждого значения проводится тест, чтобы проверить, соответствует ли оно наивысшему баллу. Если «**true**», значение сохраняется в переменной **Best Score**, а индекс игрока сохраняется в переменной **Best Player**.



НОДА SWITCH ON INT

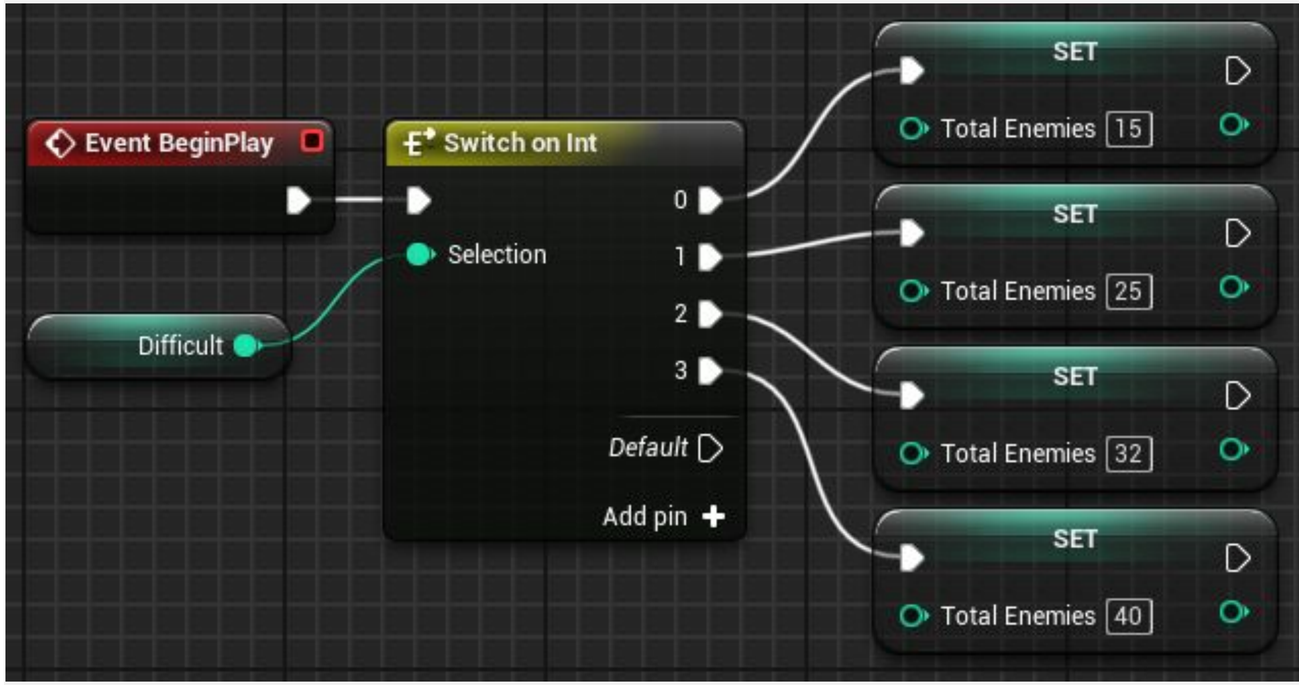
Нода **Switch on Int** определяет поток выполнения в соответствии с целочисленным входным значением. Выходные пины добавляются с помощью опции **Add pin +**.

Ввод

- **Selection**: Принимает целочисленное значение, определяющее, какой выходной пин будет выполняться. Если значение не найдено, будет выполнен вывод **Default**.

Пример

В примере справа сложность игры хранится в целочисленной переменной с именем “**Difficult**”, которая может принимать значения от «**0**» до «**3**». Общее количество врагов устанавливается в зависимости от сложности..



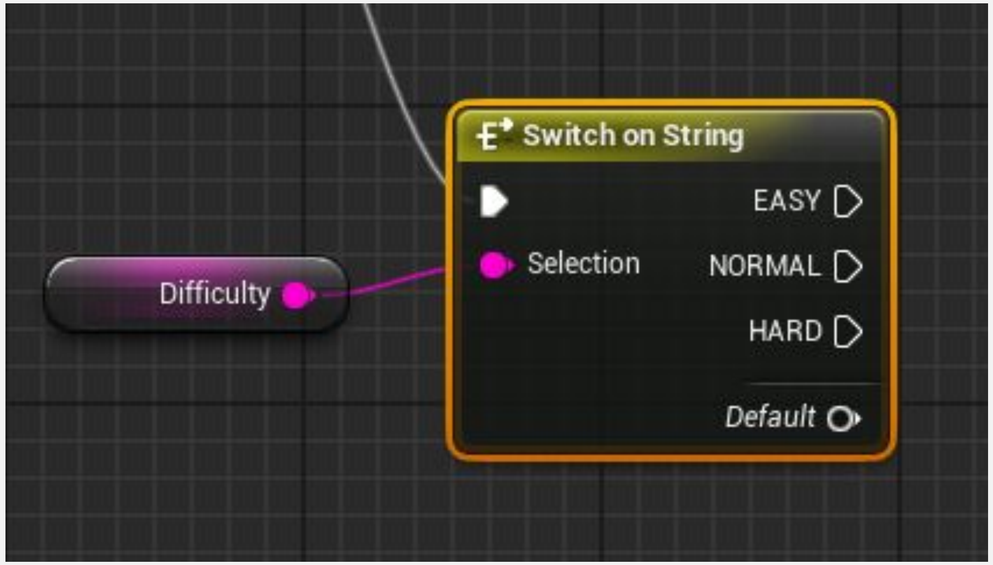
НОДА SWITCH ON STRING

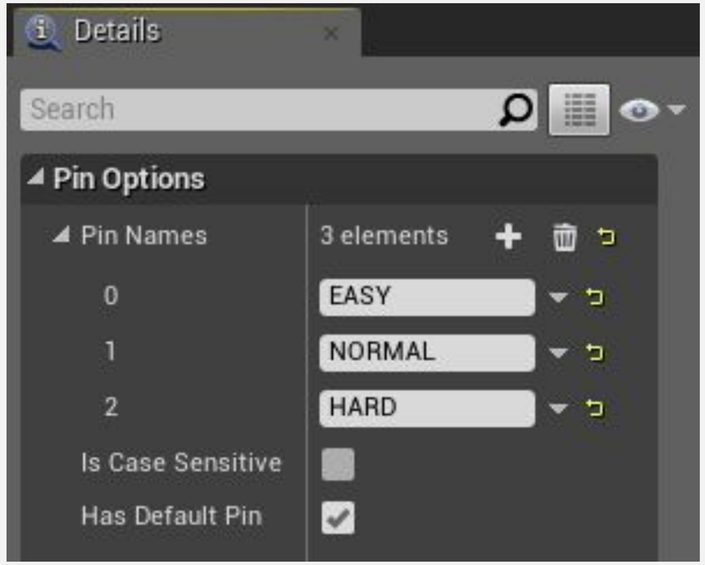
Нода **Switch on String** определяет поток выполнения в соответствии со входным значением строки. Входная строка сравнивается с каждым из имен выводов, и если они совпадают, вывод будет выполнен.

Выходные значения должны быть добавлены на панели Details для ноды **Switch on String** под “**Pin Options > Pin Names**”, как показано на нижней картинке справа.

Ввод

- **Selection**: Принимает строковое значение, определяющее вывод.



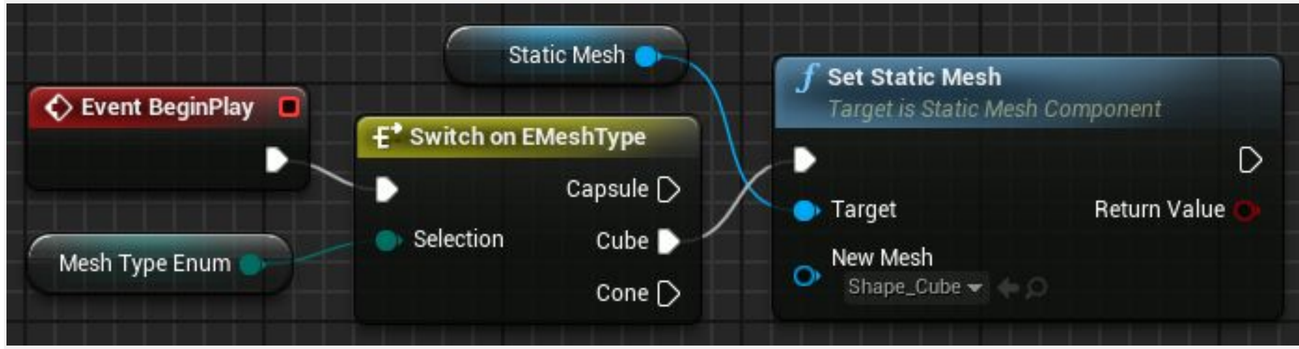


НОДА SWITCH ON ENUM

Нода **Switch on Enum** определяет поток выполнения в соответствии с входным значением перечисления. Для каждого перечисления существует эквивалент ноды переключения.

Пример

В примере справа Статик Меш выбирается на основе значения переменной перечисления..



СТРОКИ / ТЕКСТ

НОДА FORMAT TEXT

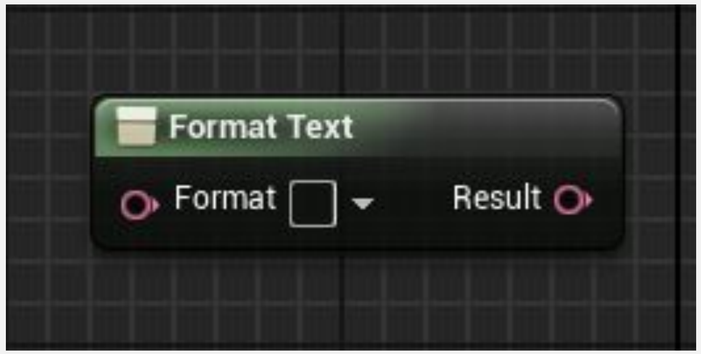
Нода **Format Text** создает текст на основе параметров, которые можно указать в параметре **Format**.

Ввод

- **Format**: Принимает текст, который будет частью окончательного результата. Чтобы установить параметры, просто поместите имя между разделителями {} для каждого параметра.
- **Parameters defined in “Format”**: Для каждой пары фигурных скобок создается новый входной параметр с именем, заключенным в фигурные скобки.

Вывод

- **Result**: Выводит окончательный текст, построенный со значениями параметра **Format** и других параметров.



НОДА FORMAT TEXT: ПРИМЕР

В примере справа в конце совпадения появится текст, отображающий результат. Этот текст содержит значения четырех переменных, связанных с именами и счетами двух игроков.

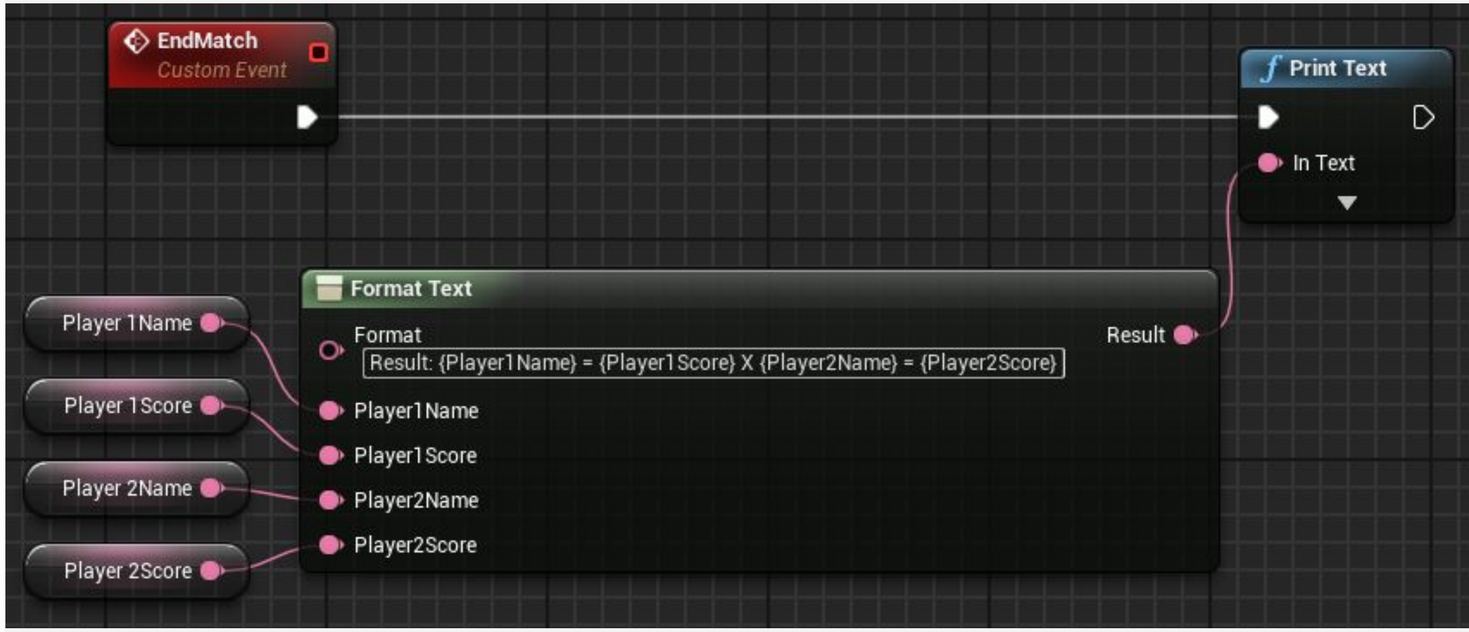
Текст, используемый в параметре **Format**, выглядит следующим образом:

Result: {Player1Name} = {Player1Score} X {Player2Name} = {Player2Score}

После размещения значения параметра **Format** редактор Blueprint сгенерирует другие входные параметры.

На верхнем изображении показана нода **Format Text**.

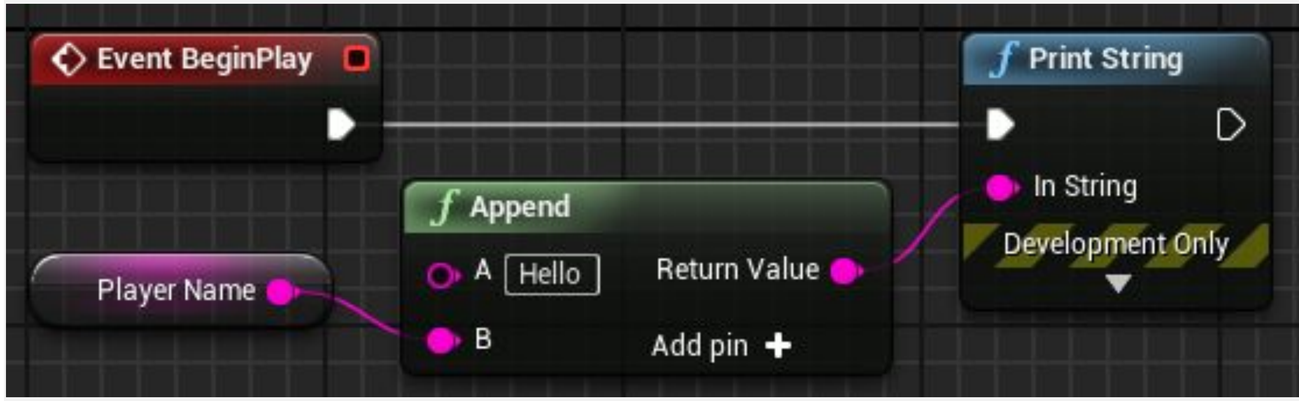
На нижнем изображении показан пример сгенерированного текста.



НОДА APPEND

Нода **Append** объединяет строки для создания новой строки. Дополнительные пины для строк можно добавить с помощью функции **Add pin +**.

В примере справа настраиваемое приветственное сообщение создается с использованием имени игрока, которое находится в переменной.

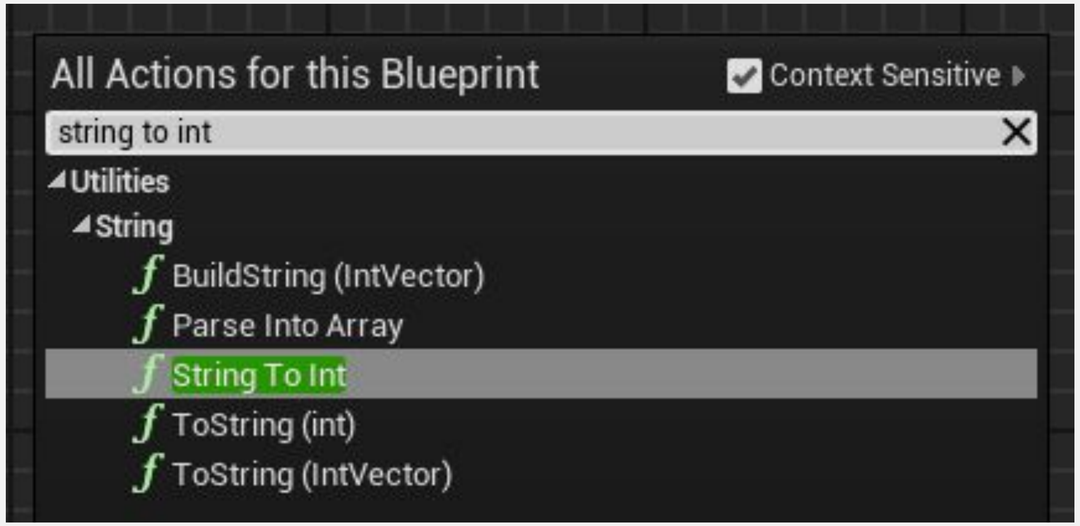


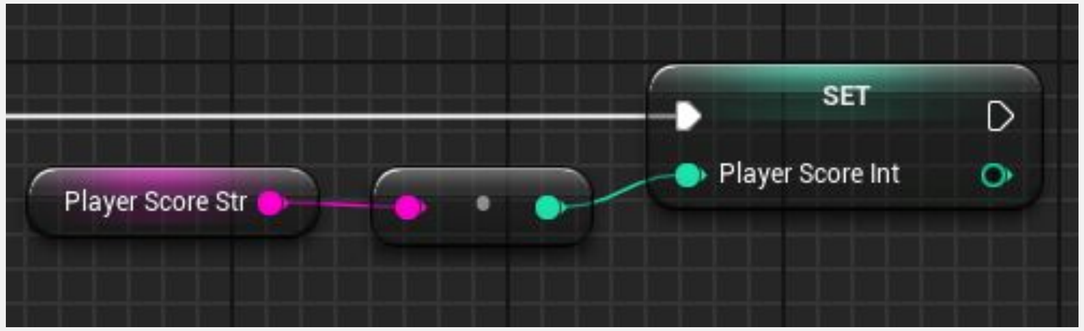
НОДА STRING TO INT

Иногда одна и та же информация, содержащаяся в строковой переменной для отображения, необходима для использования в качестве целого числа или значения с плавающей запятой для вычислений.

Чтобы преобразовать строковое значение в целочисленное значение, вы можете использовать ноду функции с именем “**String To Int**”.

Это преобразование также можно выполнить, просто соединив пин ноды **Get String** с пином целочисленного входа. Редактор автоматически создаст ноду **String To Int**, как показано в нижнем примере справа.





МАТЕМАТИКА

НОДА MATH EXPRESSION

Нода **Math Expression** - это нода особого типа, которая генерирует подграф с использованием указанного математического выражения.

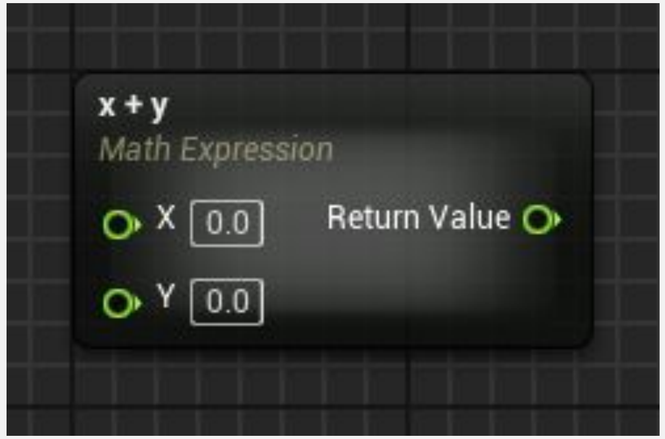
Чтобы использовать ее, выберите “**Add Math Expression…**” в **context menu**. На картинке справа показан простой пример выражения “**x + y**”.

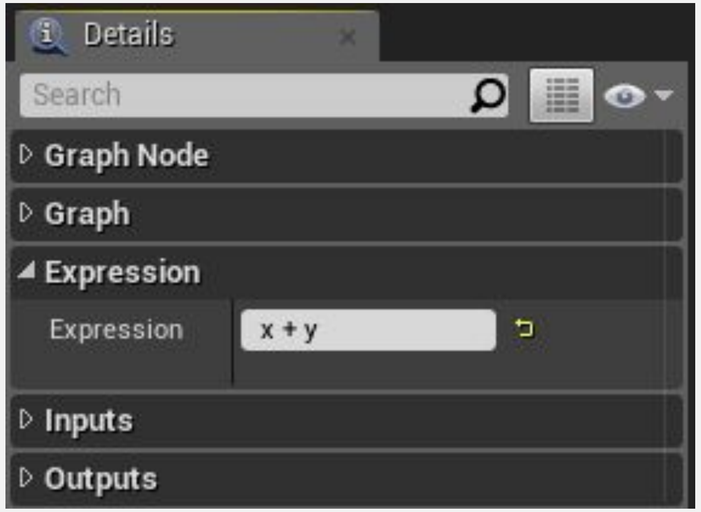
Ввод

- **Expression**: Выражение, которое будет анализироваться.
- **Parameters defined in “Expression”**: Для каждого имени переменной в выражении будет создаваться новый входной параметр.

Вывод

- **Return Value**: Выдает результат выражения.



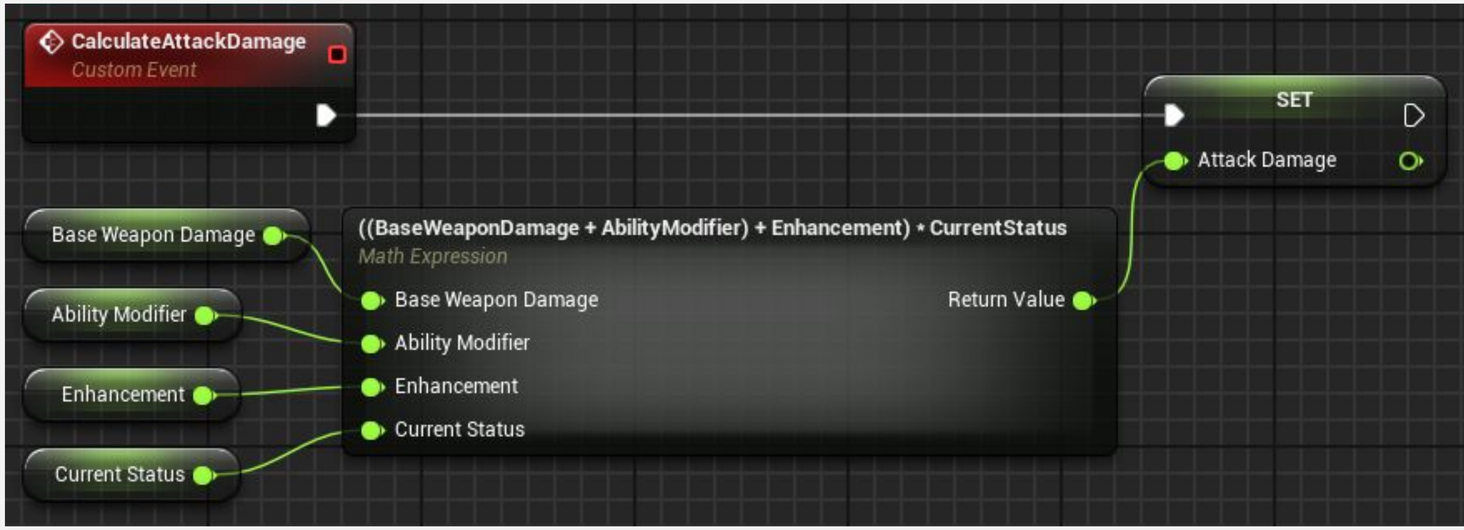


НОДА
MATH EXPRESSION : ПРИМЕР

На примере справа, событие **CalculateAttackDamage** считает урон атаки и хранит результат в переменной. Оно использует ноду **Math Expression** со следующим выражением:

((BaseWeaponDamage + AbilityModifier) + Enhancement) * CurrentStatus

Входные параметры были сгенерированы на основе этого выражения.



НОДА LERP

“**Lerp**” это сокращение “**linear interpolation**” (линейная интерполяция). Эта нода функции генерирует значение в диапазоне двух указанных значений на основе значения параметра Alpha.

Ввод

- **A**: Принимает значение с плавающей запятой, представляющее наименьшее значение, которое может быть возвращено.
- **B**: Принимает значение с плавающей запятой, представляющее наибольшее значение, которое может быть возвращено.
- **Alpha**: Принимает значение с плавающей запятой от «0» до «1». Если значение равно «0», возвращается наименьшее значение; если значение равно «1», возвращается максимальное значение.

Вывод

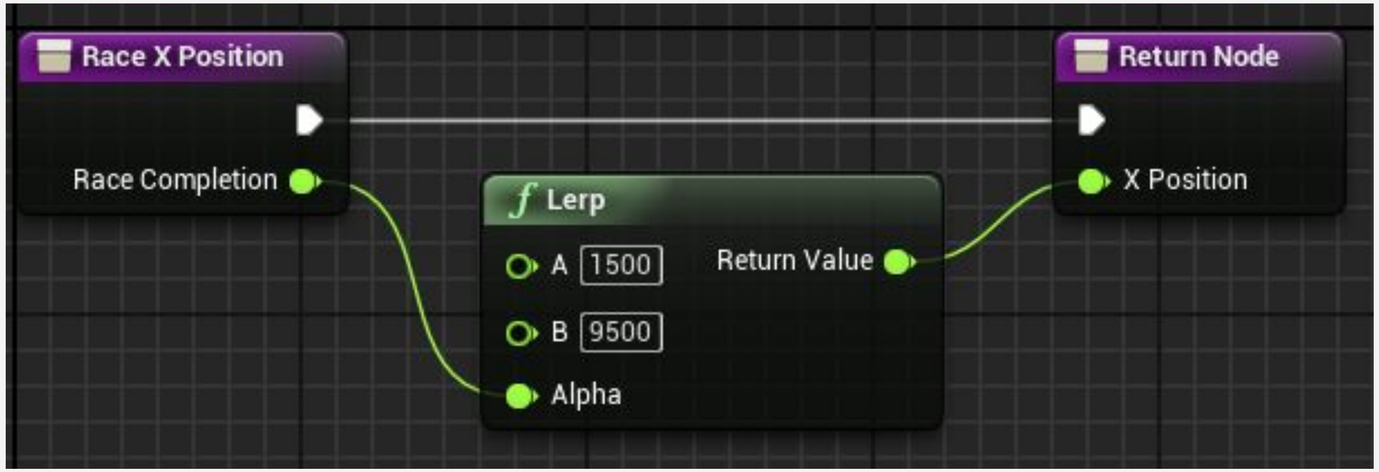
- **Return Value**: Выводит значение с плавающей запятой между значениями параметров A и B, которое зависит от значения параметра Alpha.

НОДА LERP: ПРИМЕР

В примере справа есть отрезок, который проходит вдоль оси X. Отрезок начинается в позиции X = 1500 и заканчивается в позиции X = 9500.

Нода **Lerp** используется со значением параметра **A**, установленным на «**1500**», и параметром **B**, установленным на «**9500**». Параметр **Alpha** принимает значение от «**0**» до «**1**», которое указывает, какая часть отрезка была завершена, и возвращает соответствующее значение для позиции X.

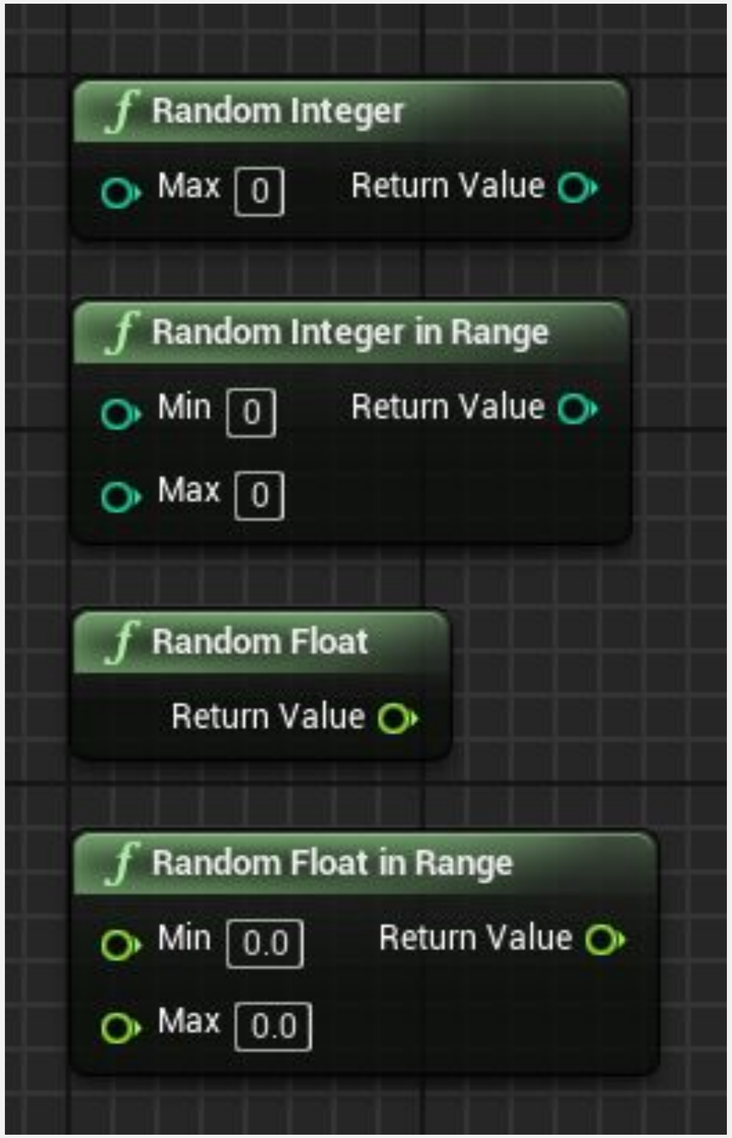
Если значение параметра **Alpha** установлено на «**0,5**», возвращаемое значение будет «**5500**», что соответствует середине отрезка.



СЛУЧАЙНЫЕ ЧИСЛА

Существуют функции случайных чисел, которые возвращают случайное значение. Основные из них перечислены ниже:

- **Random Integer**: возвращает целое число между “0” и “Max – 1”.
- **Random Integer in Range**: Возвращает целок число между “Min” и “Max”.
- **Random Float**: Возвращает значение с плавающей запятой между “0” и “1”.
- **Random Float in Range**: Возвращает значение с плавающей запятой между “Min” и “Max”.



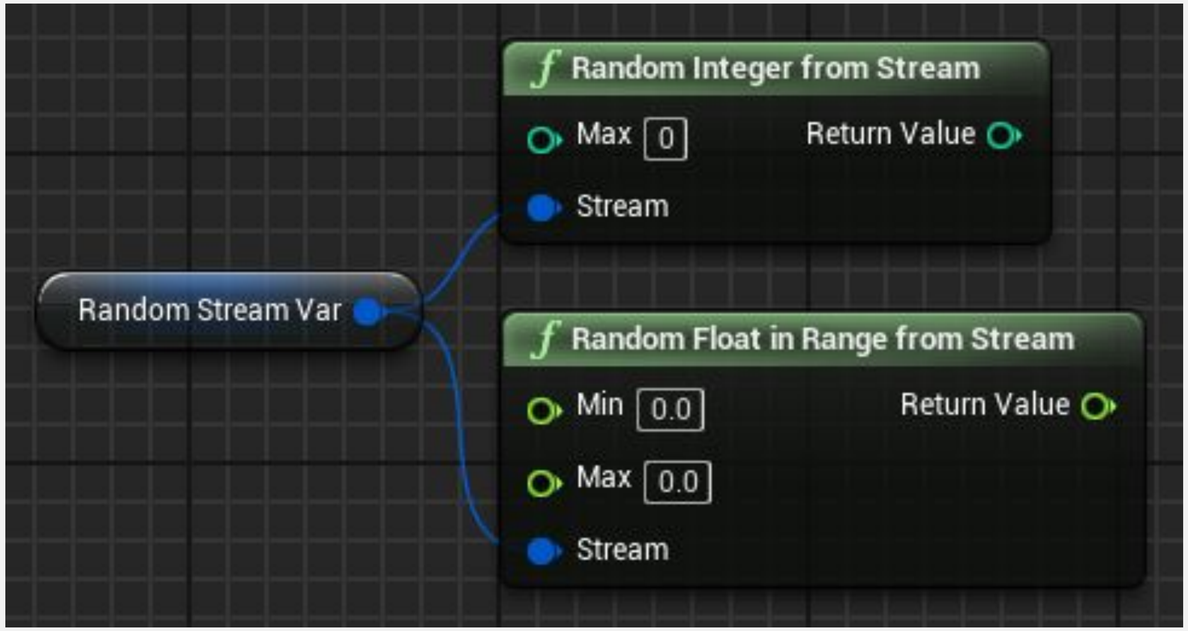
СЛУЧАЙНЫЕ ЧИСЛА ИЗ ПОТОКОВ

Можно создать последовательность повторяющихся случайных чисел, используя переменную **Random Stream**.

Для этого создайте переменную типа “**Random Stream**”. Установите свойство **Initial Seed** в разделе **Default Value** на панели **Details** для переменной.

Значение свойства **Initial Seed** будет определять последовательность случайных чисел.

На изображении справа показаны некоторые функции случайных чисел, которые используют переменную случайного потока.



ИТОГ

В этой лекции было представлено множество нод управления потоком. Он показал, как использовать ForEachLoop и переключать ноды.

В ней объяснялось, как работать со строковыми функциями, нодой математического выражения и функциями случайных чисел.