
# ЛЕКЦИЯ 6

## Продвинутые основы Блюпринтов 1

# ЦЕЛИ И ИТОГИ ЛЕКЦИИ

## Goals

Цели этой лекции:

- Показать, как использовать конструкции
- Представить контейнеры массива, набора и карты
- Показать, как использовать перечисления
- Представить таблицы данных
- Показать скрытые функции
- Объяснить функции, используемые для сохранения и загрузки данных

## Outcomes

К концу этой лекции вы сможете

- Использовать структуры и перечисления
- Использовать контейнеры для организации данных
- Использовать скрытые функции
- Внедрять базовую систему сохранения / загрузки

# МАССИВЫ

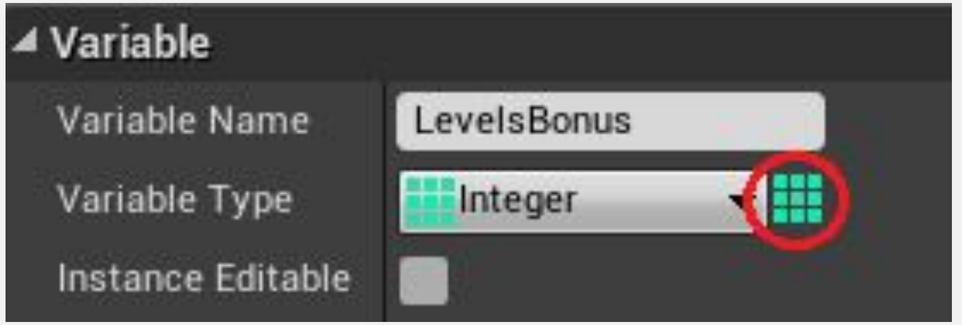
**Массив** - это упорядоченный список, и вы устанавливаете и получаете его элементы с помощью целочисленного индекса.

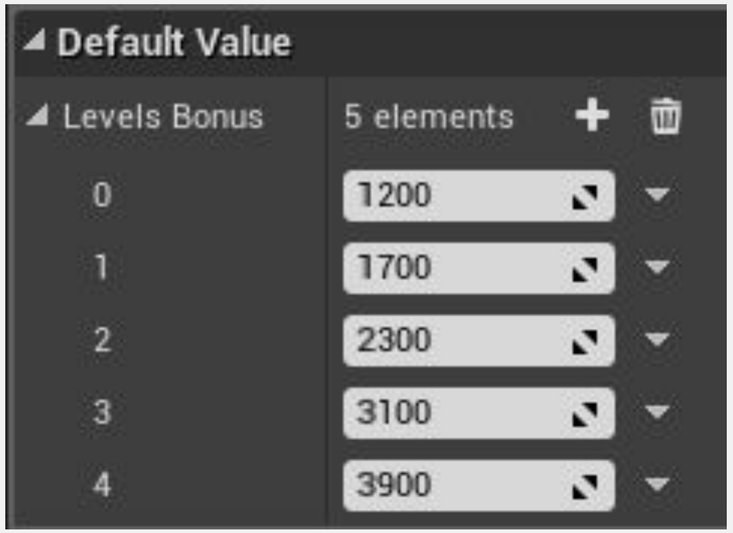
Использование массивов позволяет группировать переменные одного типа. Создать массив в Blueprints очень просто.

Сначала создайте новую переменную и выберите нужный тип.

Щелкните значок рядом с раскрывающимся списком **Variable Type** и выберите «**Array**» (см. Верхнее изображение справа).

После компиляции Blueprint можно заполнить элементы массива значениями по умолчанию, как показано на нижнем изображении.

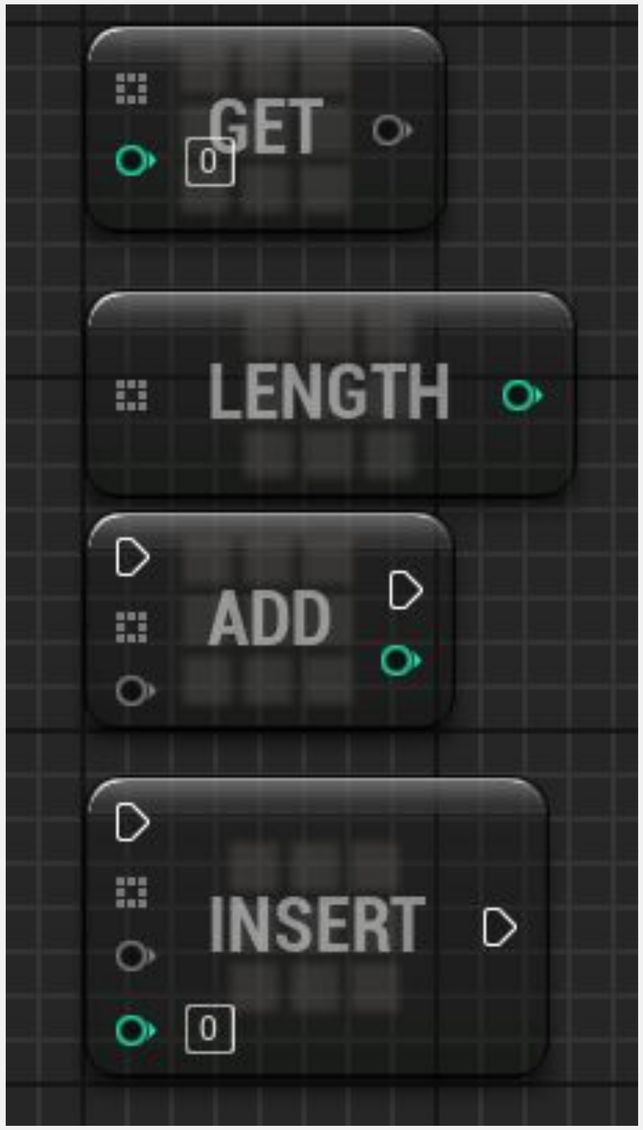


# МАССИВЫ: ОСНОВНЫЕ НОДЫ

Основные ноды, относящиеся к массивам, следующие:

- **Get**: возвращает элемент, который находится в позиции, указанной используемым индексом.
- **Length**: возвращает количество элементов массива.
- **Add**: добавляет новый элемент в конец массива.
- **Insert**: вставляет новый элемент в позицию, указанную параметром index.



# НАБОРЫ

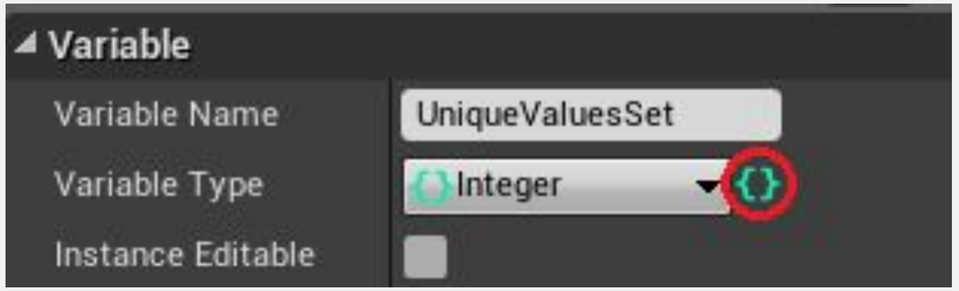
**Set** - это тип контейнера, подобный массиву, но в отличие от массива это неупорядоченный список элементов, поиск которых выполняется по их значению. Индекса нет. Его можно использовать для группировки переменных одного типа. Важное отличие состоит в том, что в наборах не допускается дублирование элементов.

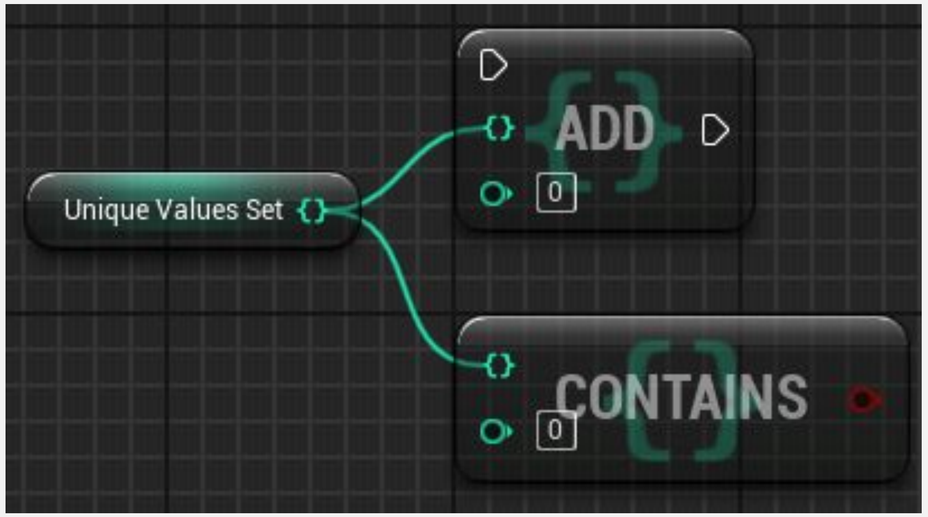
Ключевым значением, используемым для поиска предмета, является сам предмет.

Чтобы определить переменную как набор, щелкните значок рядом с раскрывающимся списком **Variable Type** и выберите «**Set**» (см. Верхнее изображение справа).

Ниже приведены некоторые общие ноды, связанные с наборами:

- **Add**: добавляет элемент в набор.
- **Contains**: проверяет, содержит ли набор данный элемент.
- **Remove**: удаляет элемент из набора.
- **Length**: возвращает количество элементов в наборе.



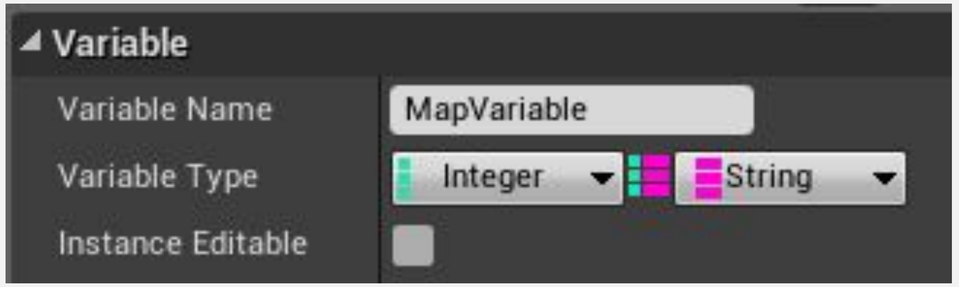
# КАРТЫ

Есть еще один тип контейнера, называемый **картой**. Чтобы определить переменную как карту, щелкните значок рядом с раскрывающимся списком **Variable Type** и выберите «**Map**».

Каждый элемент карты имеет ключ, связанный со значением. Карта не упорядочена, и поиск выполняется по ключевому значению. На верхнем изображении справа показана карта, тип ключа которой - «**Integer**», а тип значения - «**String**».

Ключевые значения карты должны быть уникальными.

На нижнем изображении показан пример карты, на которой число связано с названием игрового предмета.



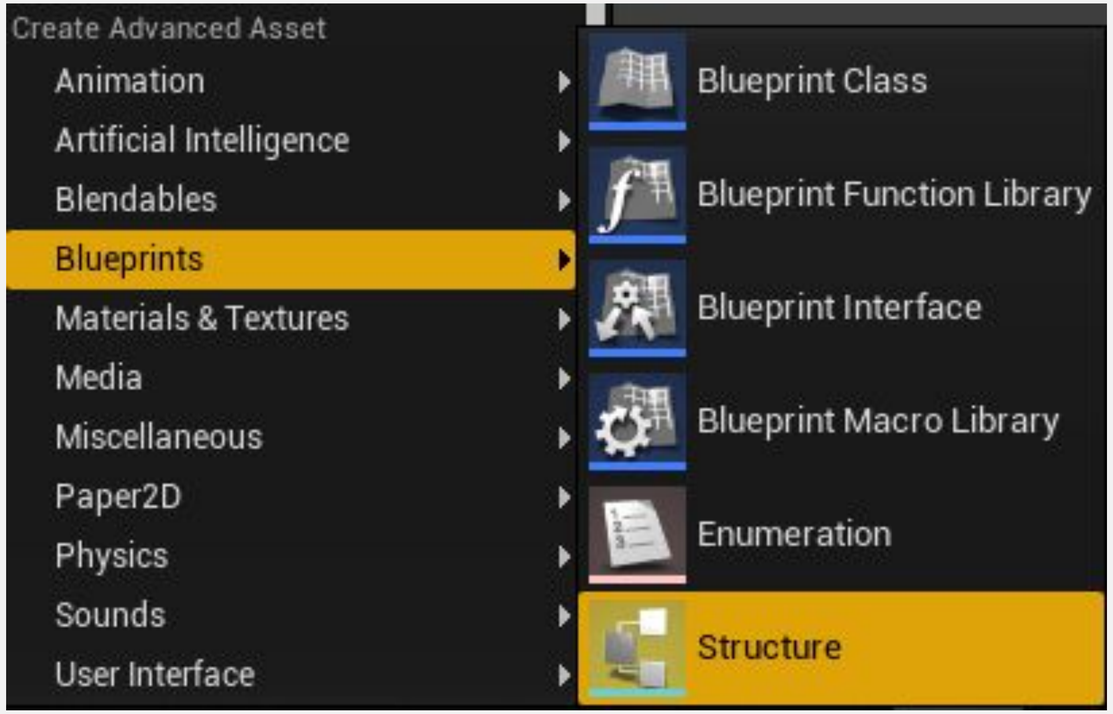

# СТРУКТУРЫ

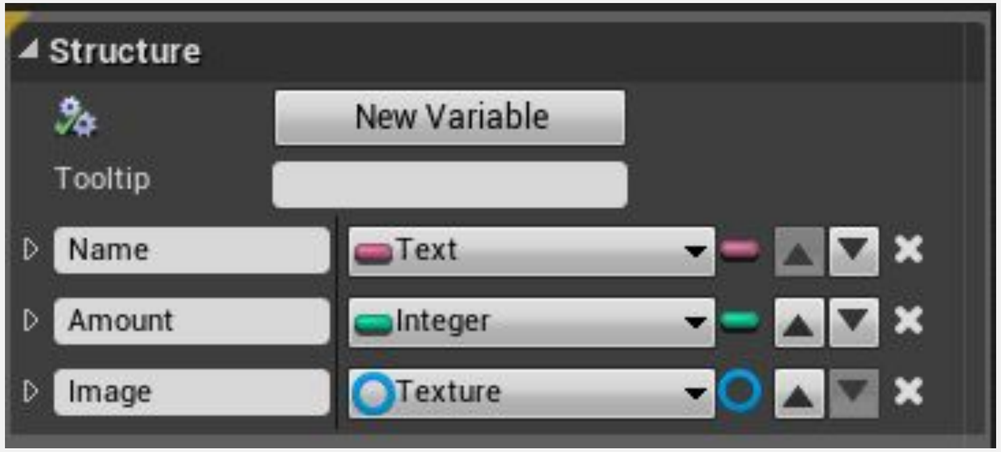
**Структуру (struct)** можно использовать для сбора в одном месте нескольких связанных переменных. Переменные, принадлежащие структуре, могут быть разных типов. У вас также могут быть структуры, содержащие другие структуры и массивы.

Чтобы создать новую структуру, нажмите зеленую кнопку **Add New** в **Content Browser** и в подменю **Blueprints** выберите “**Structure**”. Назовите ее “**Item Struct**”.

Дважды кликните по **Item Struct** для редактирования.

На панели **My Blueprint** нажмите кнопку “**+**” в разделе **Variables** чтобы добавить переменные в структуру.



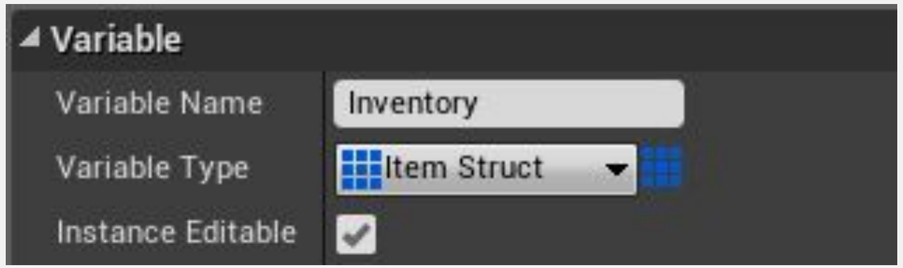



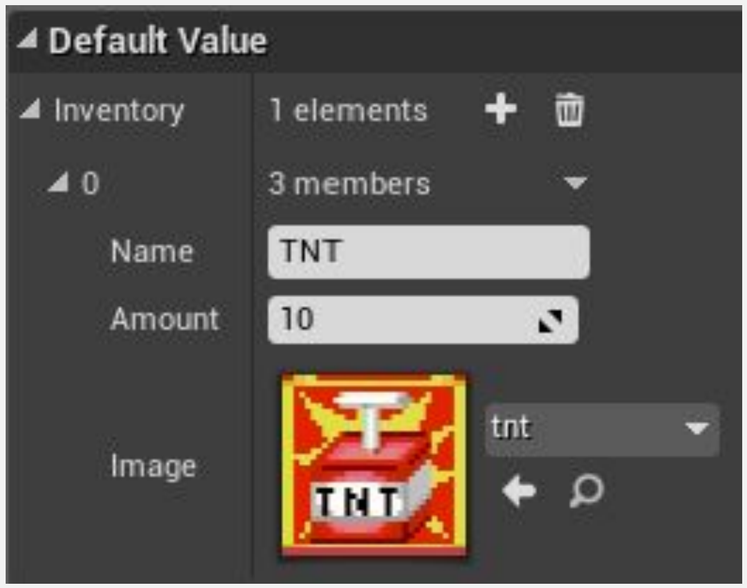
# СТРУКТУРЫ: ПРИМЕР

В качестве примера создайте новую переменную в Blueprint, которая представляет инвентарь игрока. Назовите ее “**Inventory**” и используйте тип переменной “**Item Struct**” которая была создана на предыдущем слайде. Щелкните значок рядом с раскрывающимся списком **Variable Type** и выберите “**Array**”.

Скомпилируйте Блюпринт.

Новые элементы могут быть добавлены в массив в разделе **Default Value** на панели **Details** у переменной **Inventory**. Каждый элемент будет содержать переменные, определенные в структуре **Item Struct**.



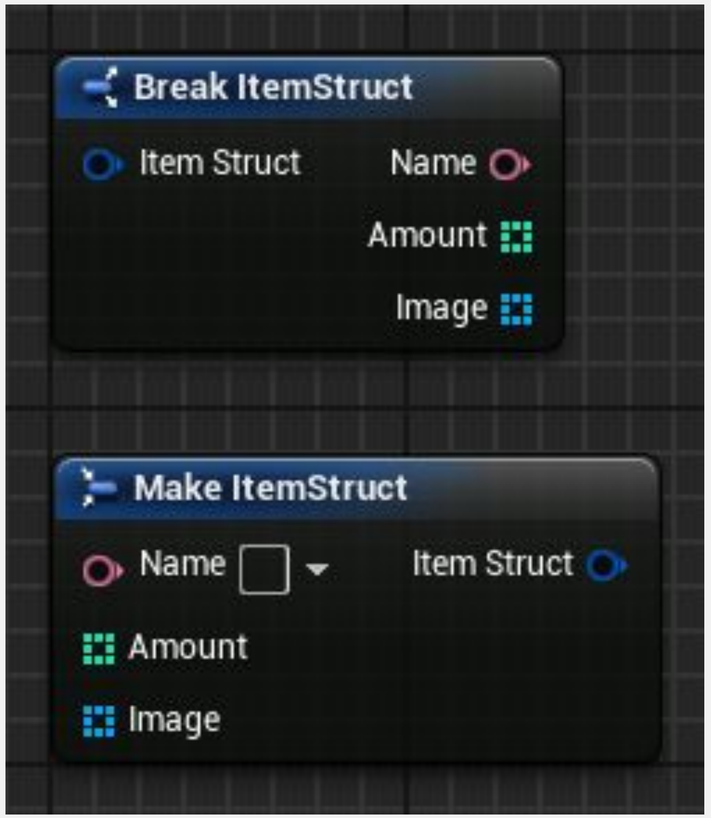
# СТРУКТУРЫ:
# НОДЫ BREAK И MAKE

При ссылке на структуру ноды **Break** и **Make** становятся доступными для использования в Blueprint.

Нода **Break** получает в качестве входных данных структуру и разделяет ее элементы.

Нода **Make** получает отдельные элементы в качестве входных данных и создает новую структуру.

На изображении справа показаны ноды **Break** и **Make** структуры **Item Struct**.



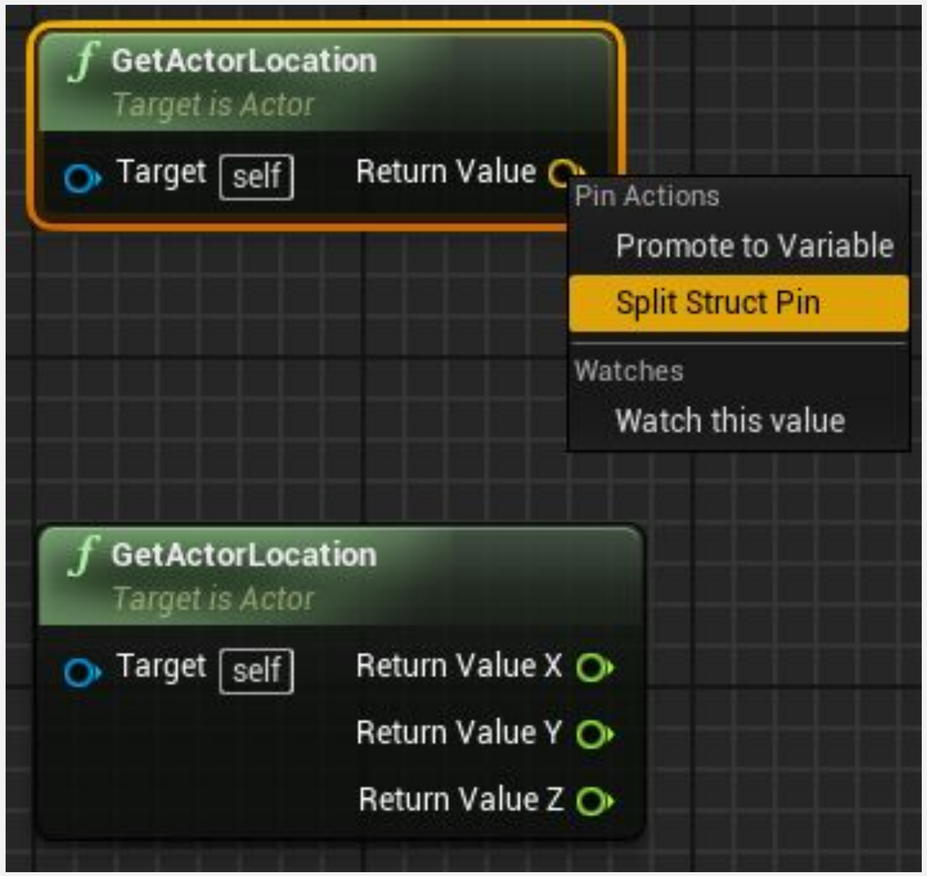
# СТРУКТУРЫ:
# SPLIT STRUCT PIN

Когда структура является входным или выходным параметром функции, ее вывод можно разделить, чтобы создать вывод для каждого элемента структуры.

Для этого щелкните ПКМ на выводе структуры и выберите “**Split Struct Pin**”.

Например, вектор в Blueprint - это структура, содержащая три переменные типа **float** с именами «**X**», «**Y**» и «**Z**».

На изображении справа показана функция **GetActorLocation** с пином структуры и пинами для каждого элемента структуры.



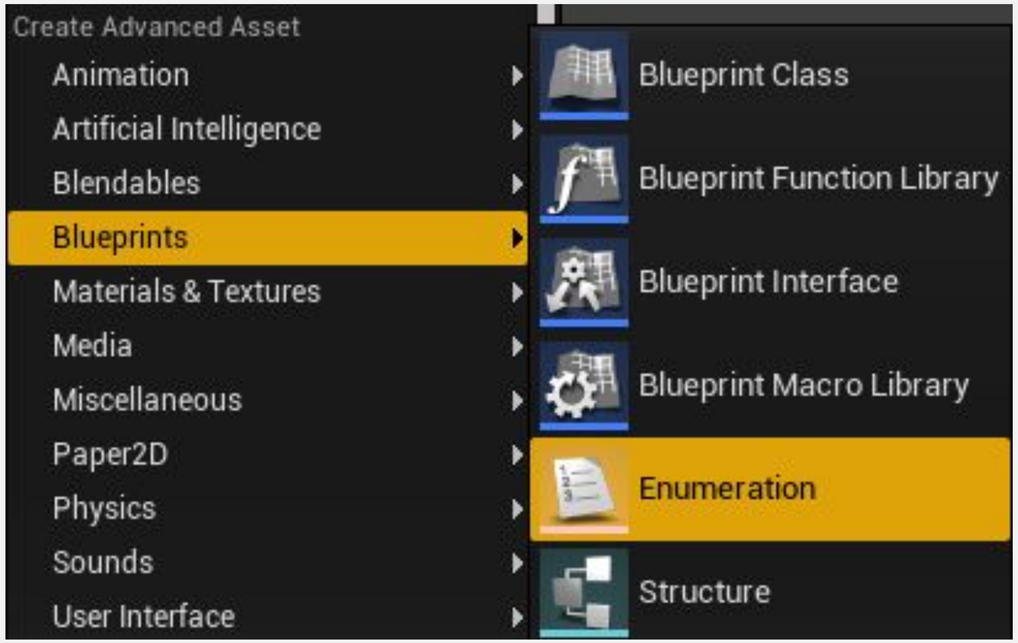
# ПЕРЕЧИСЛЕНИЯ

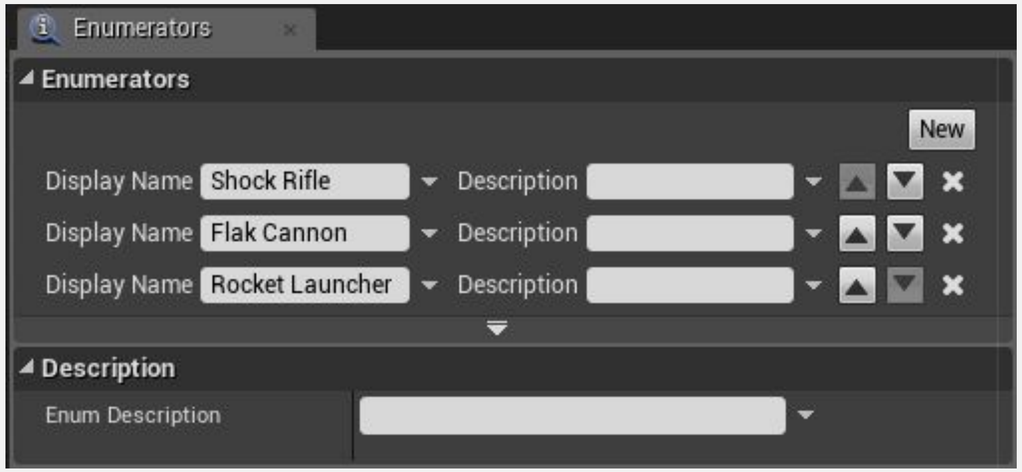
Перечисление (**enum**) - это набор констант с осмысленными именами, который определяет все возможные значения, которые может иметь переменная.

Чтобы создать новое перечисление, нажмите зеленую кнопку «**Add New**» в **Content Browser** и в подменю Blueprints выберите «**Enumeration**».

Дважды щелкните созданное перечисление, чтобы отредактировать его.

Нажмите кнопку **New** чтобы добавить имена, которые являются частью перечисления.



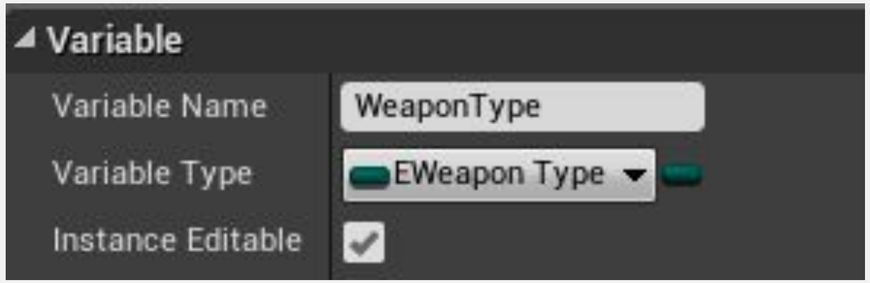



# ПЕРЕЧИСЛЕНИЯ: СОЗДАНИЕ ПЕРЕМЕННЫХ

Чтобы создать новую переменную с помощью **перечисления**, просто выберите имя перечисления в раскрывающемся списке **Variable Type**. На верхнем изображении справа показано перечисление с именем «**EWeaponType**», используемое в качестве типа переменной.

Установите для свойства **Instance Editable** значение «**true**», чтобы значение перечисления можно было выбрать в редакторе уровней Level Editor.

На нижнем изображении показаны значения, которые можно выбрать для переменной enum.

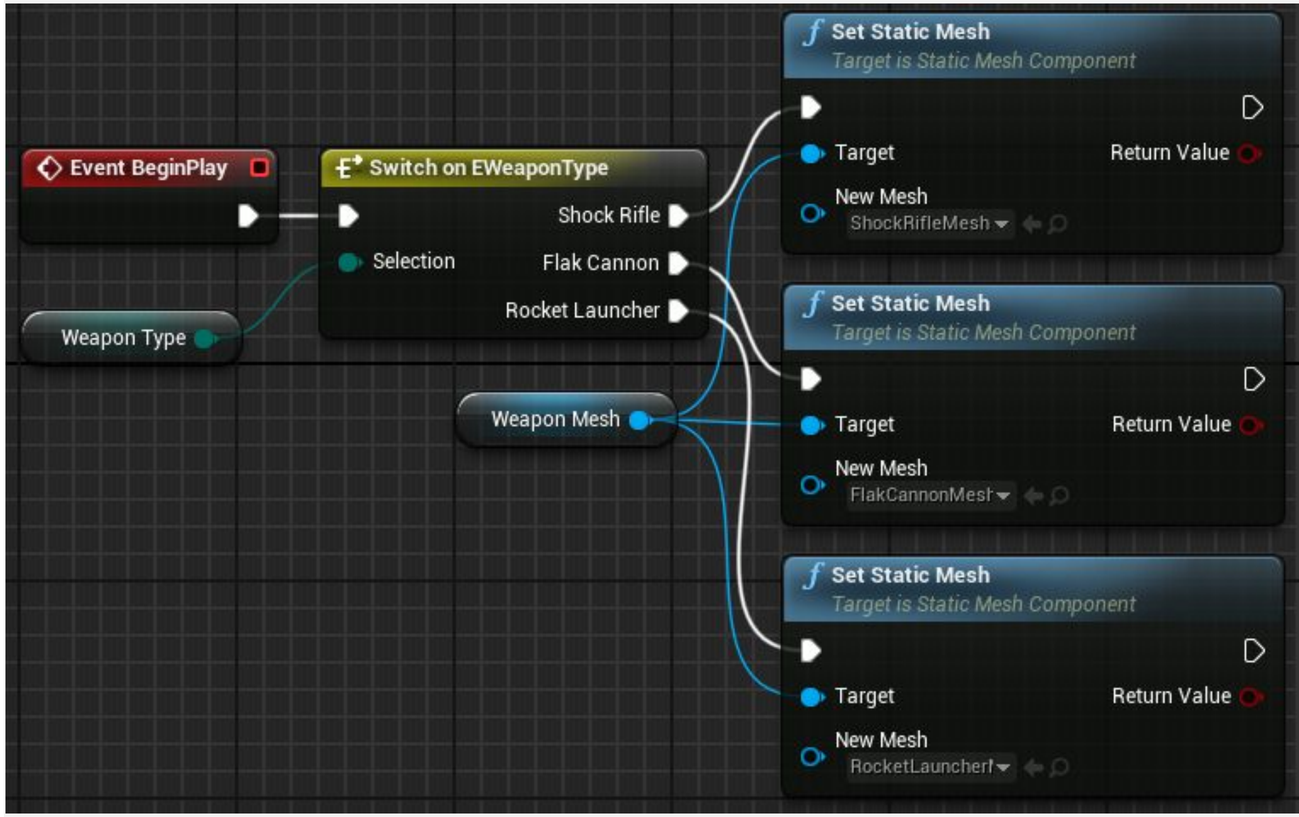



# ПЕРЕЧИСЛЕНИЯ: SWITCH ON

Существует тип **переключателя** ноды, который определяет поток выполнения в соответствии со значением перечисления.

На изображении справа “**Weapon Type**” - это переменная перечисления, а “**Weapon Mesh**” компонент статик меша.

Статик меш будет установлена в соответствии с типом оружия.



# ТАБЛИЦЫ ДАННЫХ

**Таблицу данных** можно использовать для представления документа электронной таблицы. Это полезно для игрового процесса, управляемого данными.

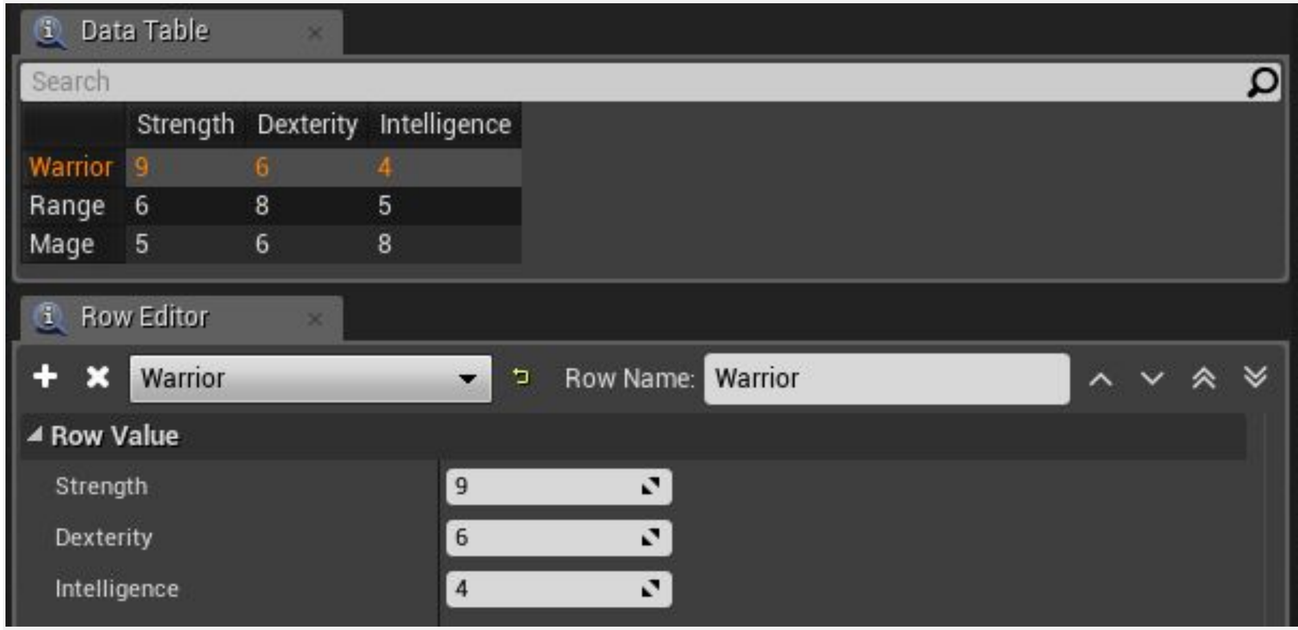
Чтобы создать новую таблицу данных, нажмите зеленую кнопку **Add New** в **Content Browser** и в подменю **Miscellaneous** выберите “**Data Table**”.

После создания таблицы данных необходимо выбрать структуру, которая будет представлять содержимое таблицы (см. Верхнее изображение справа).

После создания новой таблицы данных дважды щелкните ее, чтобы отредактировать.

В примере справа используется структура с переменными **Strength (сила)**, **Dexterity (ловкость)** и **Intelligence (интеллект)**.

Эта таблица данных состоит из трех строк, каждая из которых представляет класс RPG.

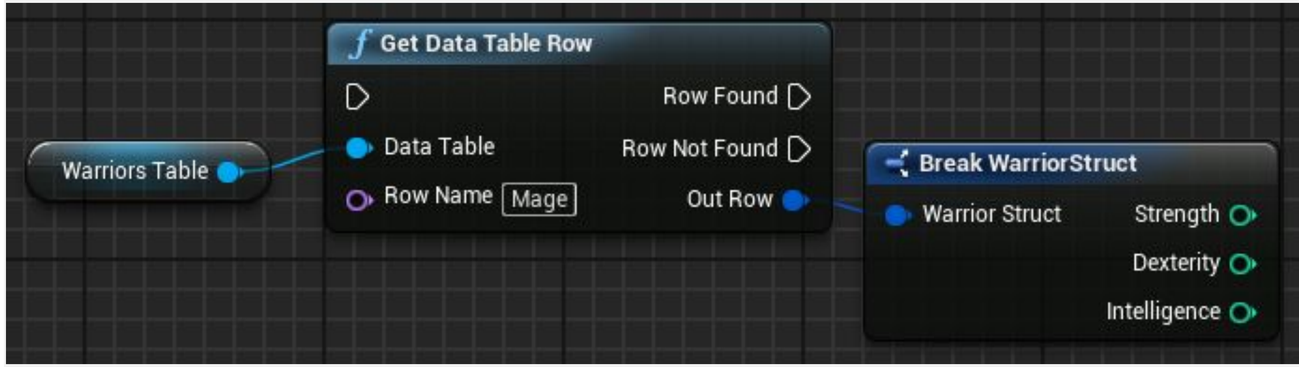



# ТАБЛИЦЫ ДАННЫХ: ДЕЙСТВИЯ

Можно создать переменную для представления таблицы данных в Blueprint.

Функция **Get Data Table Row** использует параметр **Row Name** для возврата строки таблицы в виде структуры.

Нода **Break Struct** может использоваться для доступа к атрибутам структуры.



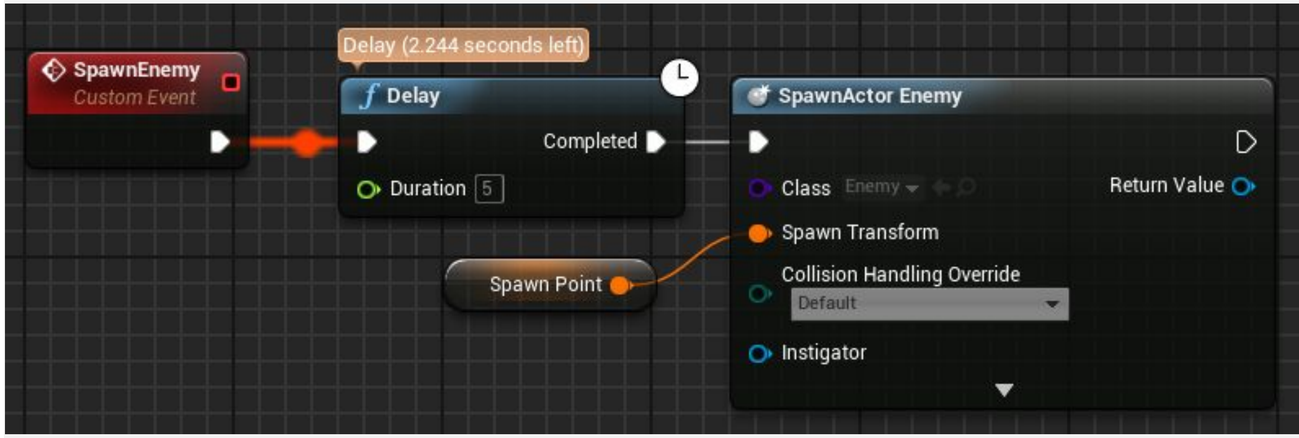
# СКРЫТЫЕ ФУНКЦИИ: ЗАДЕРЖКА

Функция **Delay** - это скрытая функция, которая выполняет действия, связанные с пином **Completed**, только по истечении времени, указанного в параметре **Duration**.

Скрытые функции не соответствуют нормальному процессу выполнения Blueprints. Они выполняются параллельно, и до их завершения может потребоваться несколько тиков.

Пример справа показывает настраиваемое событие под названием «**SpawnEnemy**», которое имеет функцию **Delay**, чтобы гарантировать, что прошло не менее пяти секунд, прежде чем оно создаст нового вражеского актора. Даже если событие **SpawnEnemy** вызывается снова менее чем через пять секунд, функция **Delay** не позволяет создать нового врага.

Поиграйте в редакторе, чтобы увидеть оставшееся время в функции задержки (см. Пример).



# СКРЫТЫЕ ФУНКЦИИ: ПЕРЕЗАПУСКАЕМАЯ ЗАДЕРЖКА

Функция **Retriggerable Delay** - это скрытая функция, которая выполняет действия, связанные с пином **Completed**, только по истечении времени, указанного в параметре **Duration**.

Разница между функциями **Retriggerable Delay** и **Delay** заключается в том, что значение обратного отсчета для параметра **Duration** будет сброшено, если функция **Retriggerable Delay** будет вызвана снова.

# СКРЫТЫЕ ФУНКЦИИ: ТАЙМЕР

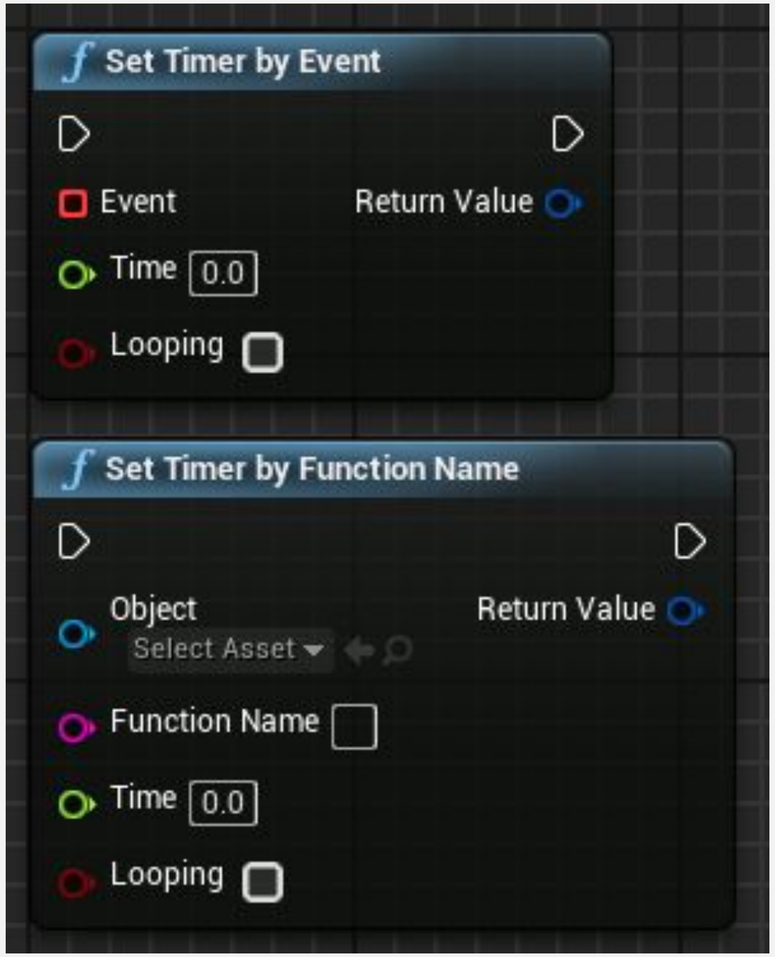
**Таймер** запрограммирован на выполнение заданной функции (или пользовательского события) по истечении заданного времени.

Две функции определяющие Таймер:

- **Set Timer by Event**: имеет в качестве входного параметра ссылку на настраиваемое событие.
- **Set Timer by Function Name**: имеет в качестве входных параметров имя функции и объект, содержащий функцию.

Обе функции имеют следующие параметры:

- **Time**: представляет продолжительность таймера в секундах.
- **Looping**: указывает, будет ли таймер продолжать выполнение или будет выполняться только один раз.



# ЗАГРУЗКА УРОВНЕЙ

Функцию **Open Level** можно использовать для загрузки уровней в игре.

Значением параметра **Level Name** может быть имя папки плюс имя уровня. Если функция **Open Level** не имеет полного пути, она попытается открыть первый найденный уровень (.umap) с указанным именем. Если существует несколько файлов .umap с одинаковым именем, он выберет первый найденный.

Эту функцию можно использовать в Blueprint, который при пересечении игроком переносит их на другой уровень.

Его также можно использовать в меню Start игры, где могут быть загружены разные уровни в зависимости от выбранной опции.

# ВЫХОД ИЗ ИГРЫ

Функцию **Quit Game** можно использовать для выхода из игры или перемещения приложения в фоновый режим.

Эта функция содержит следующие входные параметры:

- **Specific Player**: представляет игрока, который выйдет из игры. “**Player 0**” - значение по умолчанию.
- **Quit Preference**: показывает, как игрок хочет выйти из игры. Значение может быть “**Quit**” или “**Background**”.

# СОХРАНЕНИЕ И ЗАГРУЗКА ДАННЫХ

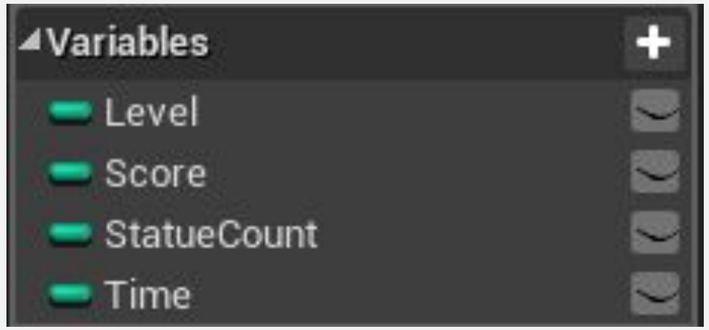
# SAVE GAME BLUEPRINT

Чтобы сохранить игру, информация, которая будет сохранена, должна быть собрана в переменных Blueprint типа «**SaveGame**».

Первым шагом к сохранению и загрузке игр с помощью Blueprints является создание нового Blueprint с использованием «**SaveGame**» в качестве родительского класса.

Следующим шагом является создание переменных в Save Game Blueprint для хранения информации, которая будет сохранена.

На нижнем изображении справа показан пример игры, созданной в Лекции 5.

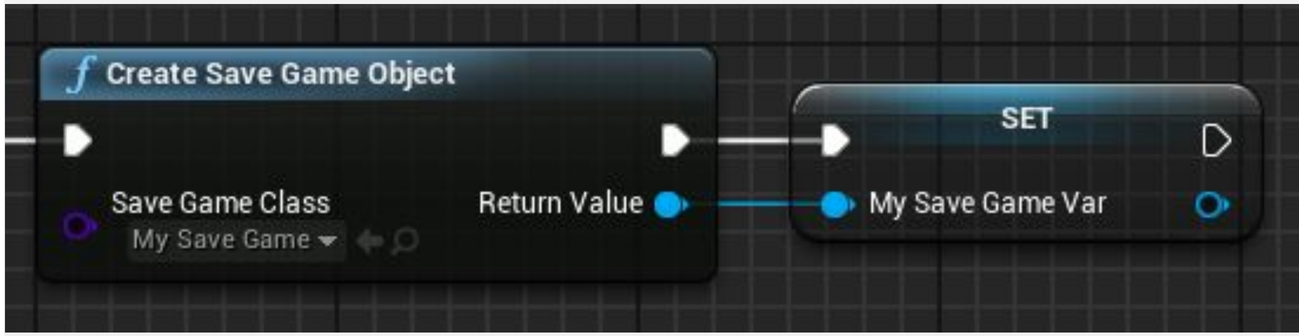



# ФУНКЦИЯ CREATE SAVE GAME OBJECT

Функция **Create Save Game Object** создает объект Save Game на основе класса Save Game.

Информация, которая будет сохранена, будет установлена в переменных этого Объекта.

**MySaveGame** - это Blueprint класса Save Game.

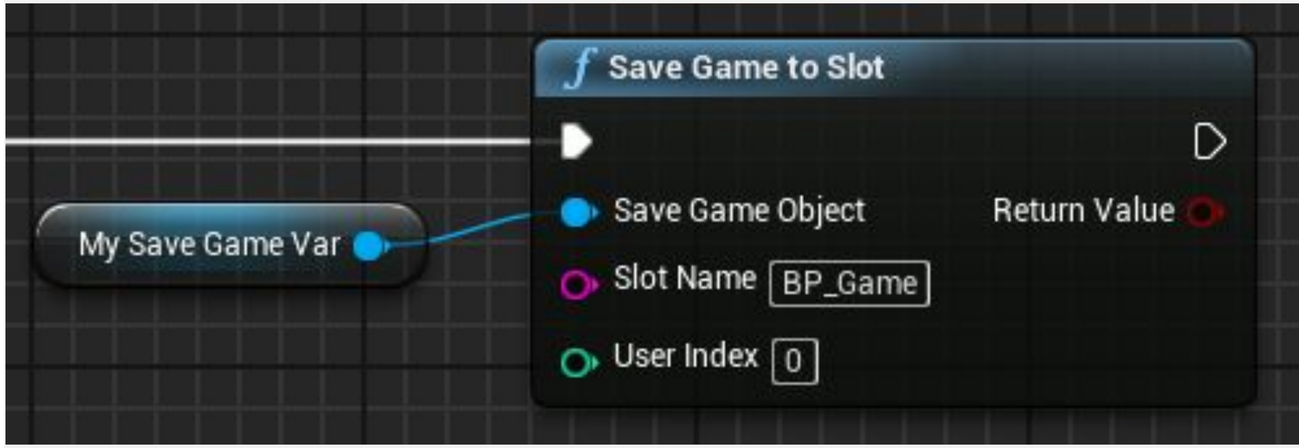


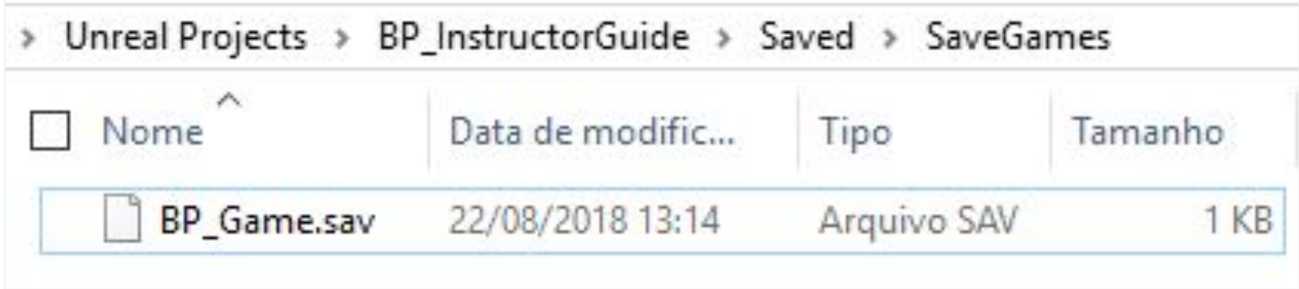
# ФУНКЦИЯ
# SAVE GAME TO SLOT

Функция **Save Game to Slot** сохраняет содержимое Save Game Object в файл на диске.

Картинка слева показывает параметр **Slot Name** со значением “**BP_Game**”. Это значит что файл с именем “**BP_Game.sav**” будет создан с информацией которую нужно сохранить.

Файл “**BP_Game.sav**” хранится в папке “**ProjectName > Saved > SaveGames**”.

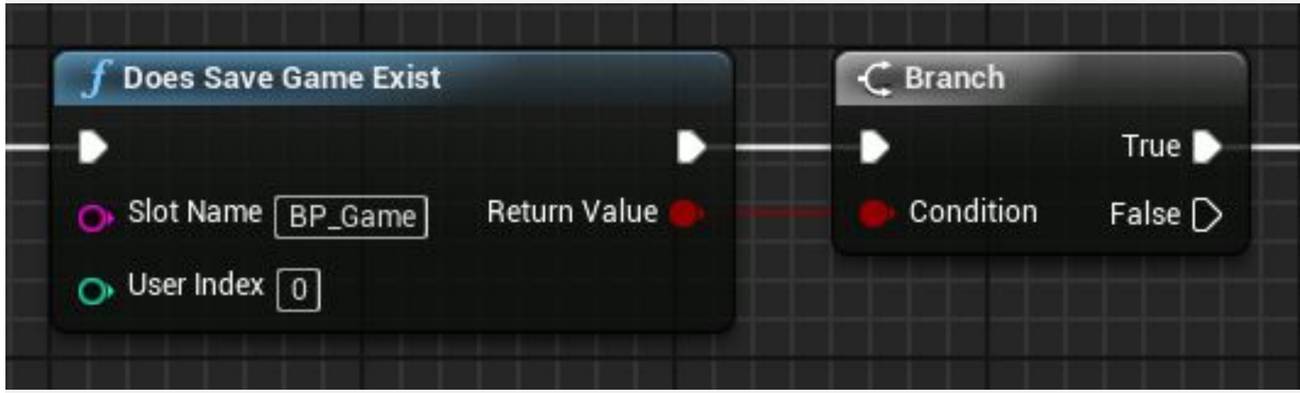


# ФУНКЦИЯ
# DOES SAVE GAME EXIST

Функция **Does Save Game Exist** проверяет, существует ли Save Game в заданном параметре **Slot Name**.

Это полезный тест, который нужно сделать перед попыткой загрузить Save Game.

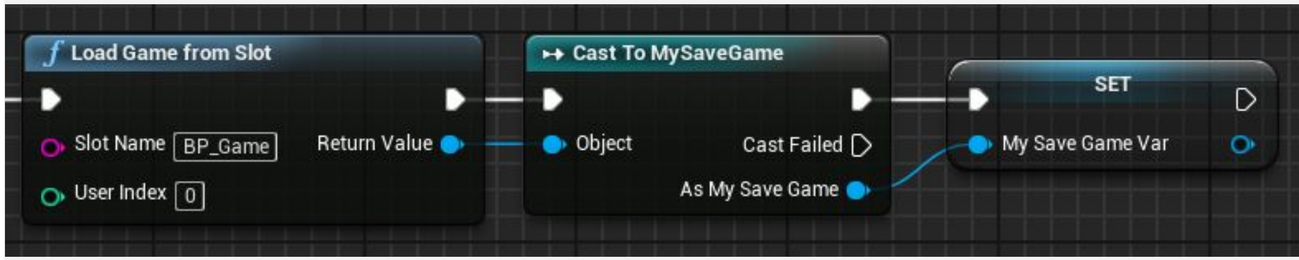


# ФУНКЦИЯ
# LOAD GAME FROM SLOT

Функция **Load Game from Slot** загружает содержимое слота и создает с ним объект Save Game Object.

Выходной параметр **Return Value** - это ссылка на общий объект Save Game, поэтому необходимо привести ссылку на правильный Save Game Blueprint, чтобы получить доступ к переменным.

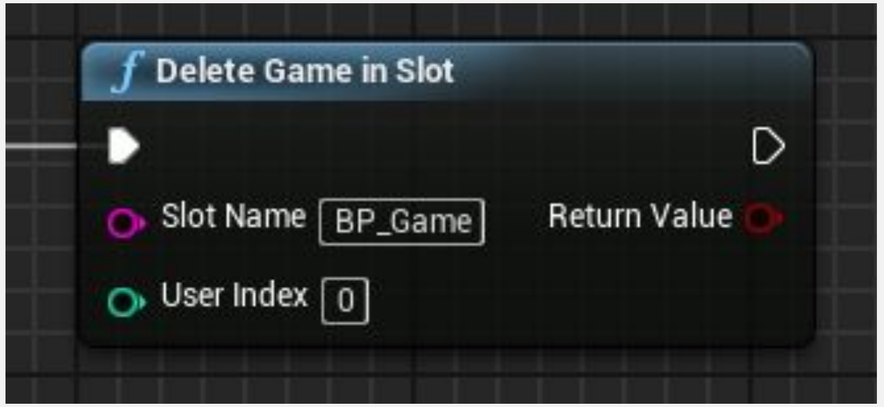
В примере справа функция **Load Game from Slot** приводит к **MySaveGame**. После этого ссылка на объект MySaveGame устанавливается в переменной.



# ФУНКЦИЯ
# DELETE GAME IN SLOT

Функция **Delete Game in Slot** используется для удаления данных сохранения игры в слоте.

Параметр **Return Value** является “**true**”, если файл был найден и удален.



# ИТОГ

В этой лекции были представлены многие передовые концепции Blueprint и представлены структуры, массивы, наборы, карты, перечисления, скрытые функции и таблицы данных.

Были объяснены функции, необходимые для реализации базовой системы сохранения / загрузки.