ЛЕКЦИЯ 5

Простой пример игры

ЦЕЛИ И ИТОГИ ЛЕКЦИИ

Goals

Цели этой лекции:

- Реализовать простую игру
- Показать, как использовать класс игрового режима (Game mode) для определения правил игры
- Привести примеры использования макросов, настраиваемых событий и функций.
- Продемонстрировать использование таймеров в игре.

Outcomes

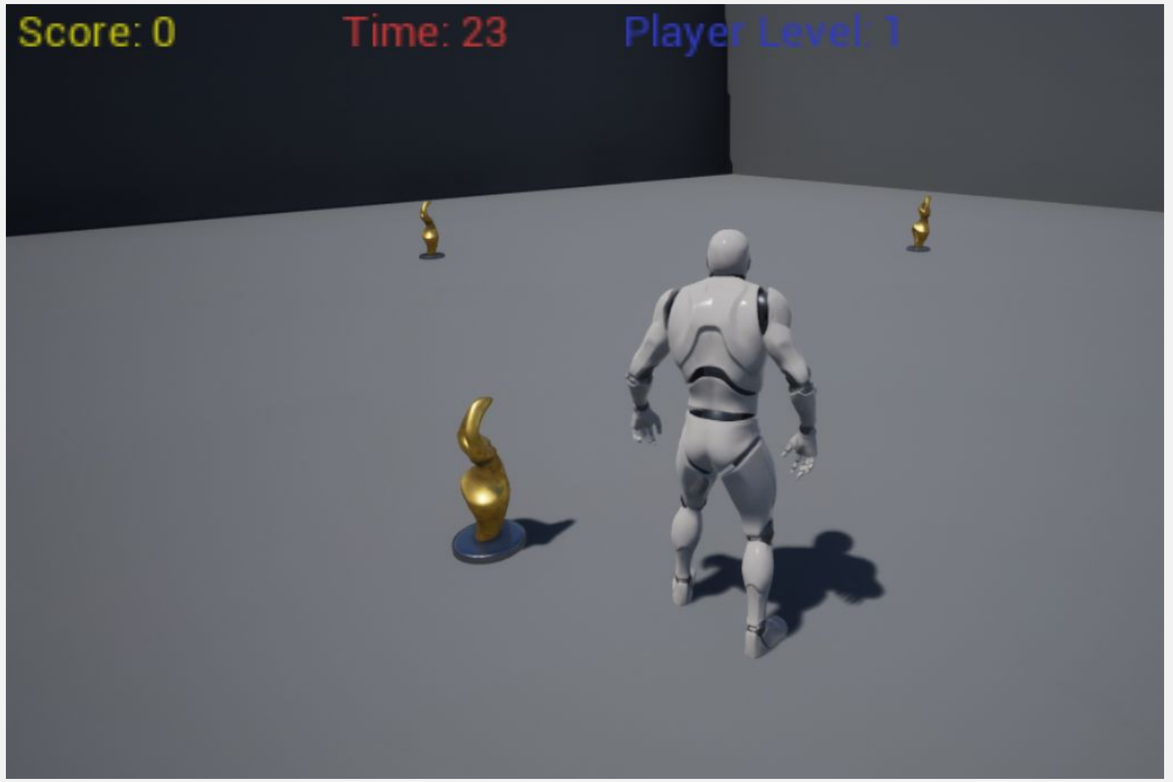
К концу этой лекции вы сможете

- Преобразуете правила игры в код Blueprint
- Организуете код в макросы, настраиваемые события и функции
- Оформите информацию с помощью класса HUD
- Определите таймеры с помощью функций или настраиваемых событий

ИГРОВЫЕ КЛАССЫ

Игра, созданная в этой лекции, основана на шаблоне от третьего лица **Third person**. В игре используются четыре основных класса Blueprint:

- **ThirdPersonCharacter**: Подкласс Pawn, который представляет игрока и является частью шаблона от третьего лица. Этот класс назначается параметру **Default Pawn Class** у Game Mode.
- **BP_Statue**: Класс Актора, представляющий статую. Он проверяет наличие столкновения с игроком и имеет логику, которая периодически меняет его положение.
- **BP_Guide_GameMode**: Контролирует состояние игры и сохраняет некоторые переменные, например, время, счет и уровень. Он также определяет базовые классы, такие как ThirdPersonCharacter.
- **BP_Guide_HUD**: Отвечает за отображение значений времени, очков и уровня игрока на экране.



ПРАВИЛА ИГРЫ

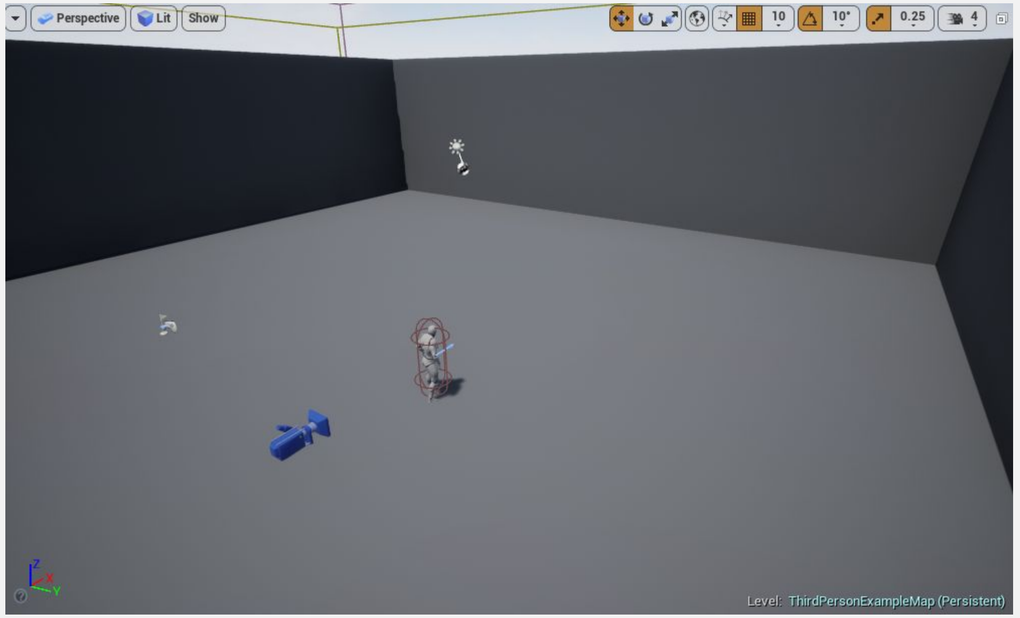
- Игрок должен собрать маленькие статуи, которые появляются на экране, прежде чем истечет время.
- Начальное время установлено на 30 секунд.
- Игра окончена, когда не остается времени.
- За каждые пять собранных статуй уровень игрока увеличивается, и ко времени добавляется 15 секунд.
- Игрок начинает с уровня игрока 1, а максимальный уровень, которого может достичь игрок, - это уровень игрока 5.
- В сценарии есть три статуи, которые периодически меняют положение.
- Когда статуя появляется в позиции, она остается там в течение периода времени, который зависит от текущего уровня игрока. Время в секундах, которое требуется статуе, чтобы изменить положение, является результатом выражения «6 - Уровень».
- Когда игрок получает статую, появляется другая.
- Количество очков за собранную статуию определяется выражением «10 x Уровень».
- Значения времени, очков и уровня игрока будут отображены на экране.

СОЗДАНИЕ ПРОЕКТА

Создайте новый проект используя шаблон **Third Person** с наличием **starter content**.

Удалите Static Mesh и Text Render Actors, которые находятся в середине сцены, оставив только пол и боковые стены, как показано на нижнем изображении справа.

В **Content Browser** создайте новую папку с названием “**BP_Guide**” которая будет хранить в себе все новые Blueprints.



BP_GUIDE_GAMEMODE

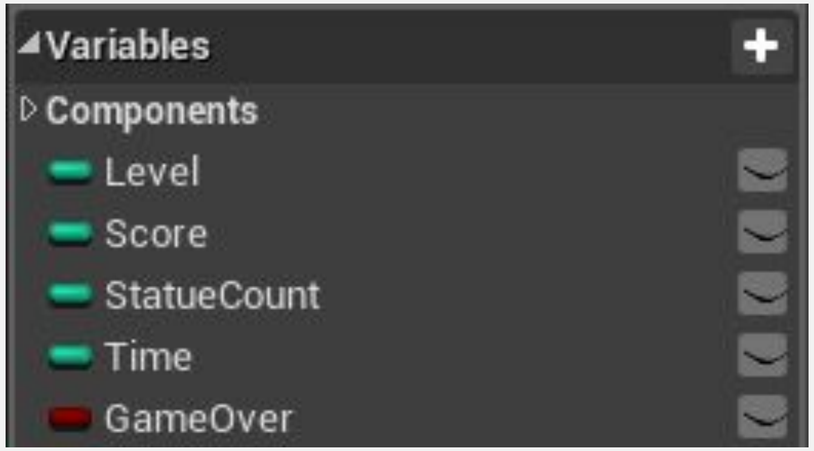
Создайте новый Блюпринт класс “**Game Mode Base**” как родительский класс. Назовите его “**BP_Guide_GameMode**”.

Создайте переменные **Integer** перечисленные ниже:

- **Level**: Сохраняет текущий уровень игрока в игре.
- **Score**: Сохраняет счет игрока.
- **StatueCount**: Отслеживает количество собранных статуй.
- **Time**: Сохраняет время, оставшееся до конца игры.

Создайте следующую **Boolean** переменную:

- **GameOver**: Указывает, закончилась ли игра.

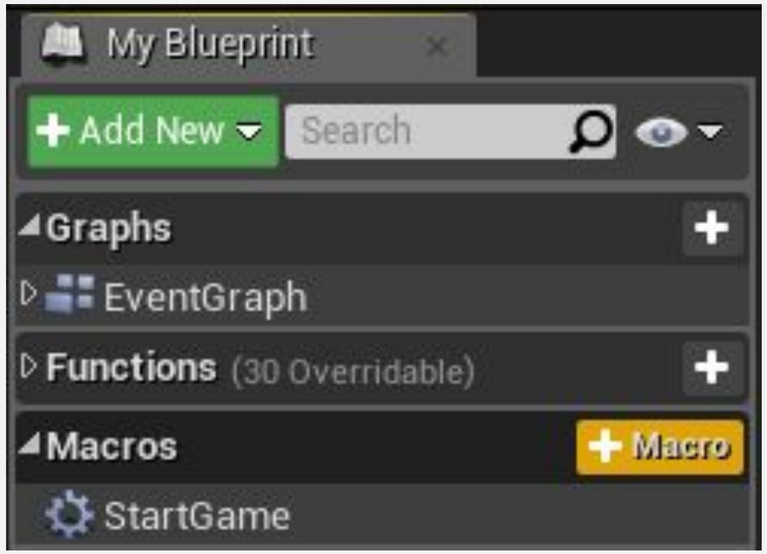


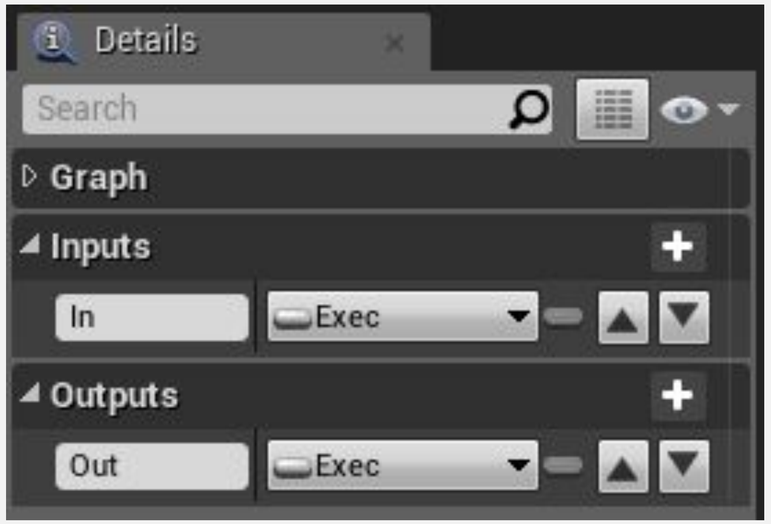
BP_GUIDE_GAMEMODE: МАКРОС СТАРТА ИГРЫ

На панели **My Blueprint** создайте макрос с именем “**StartGame**”.

Этот макрос отвечает за инициализацию переменных, управляющих состоянием игры.

На панели **Details** для макроса **StartGame** создайте входной параметр “**In**” и поставьте тип на “**Exec**”. Создайте выходной параметр “**Out**” и поставьте тип “**Exec**”.

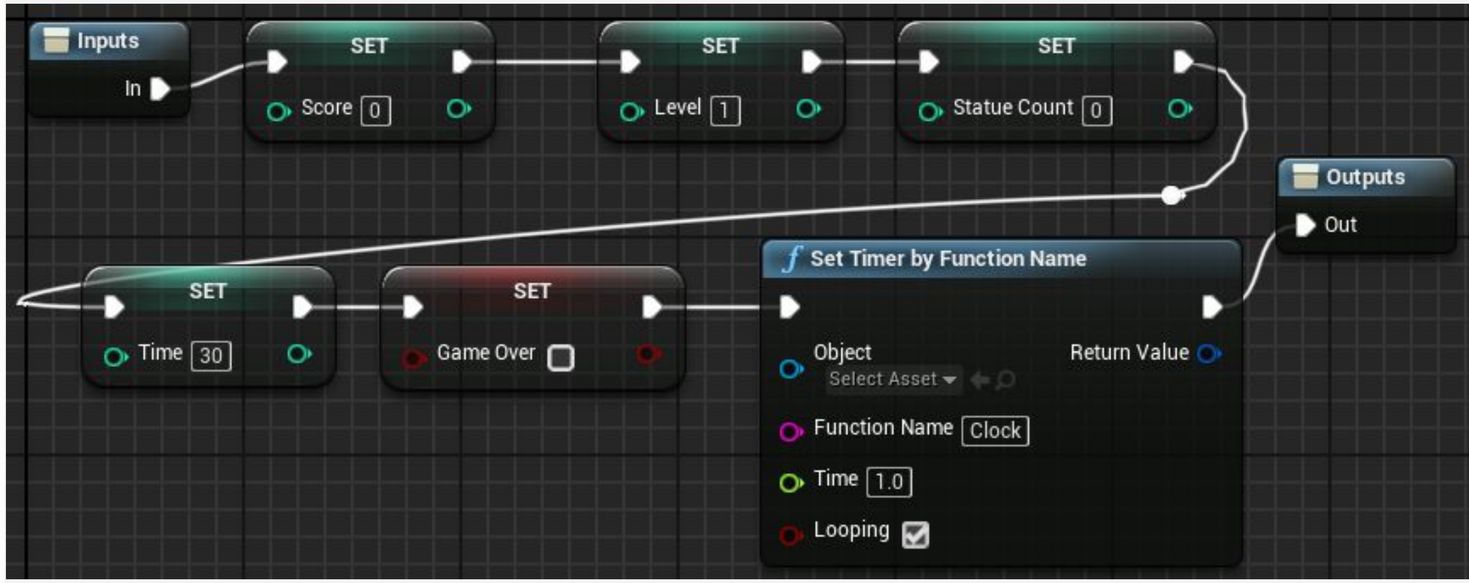




BP_GUIDE_GAMEMODE: МАКРОС СТАРТА ИГРЫ

Макрос **StartGame** установит начальные значения переменных.

Функция **Set Timer** создает объект **Timer**, который будет вызывать пользовательское событие с именем «**Clock**» каждые 1,0 секунды.

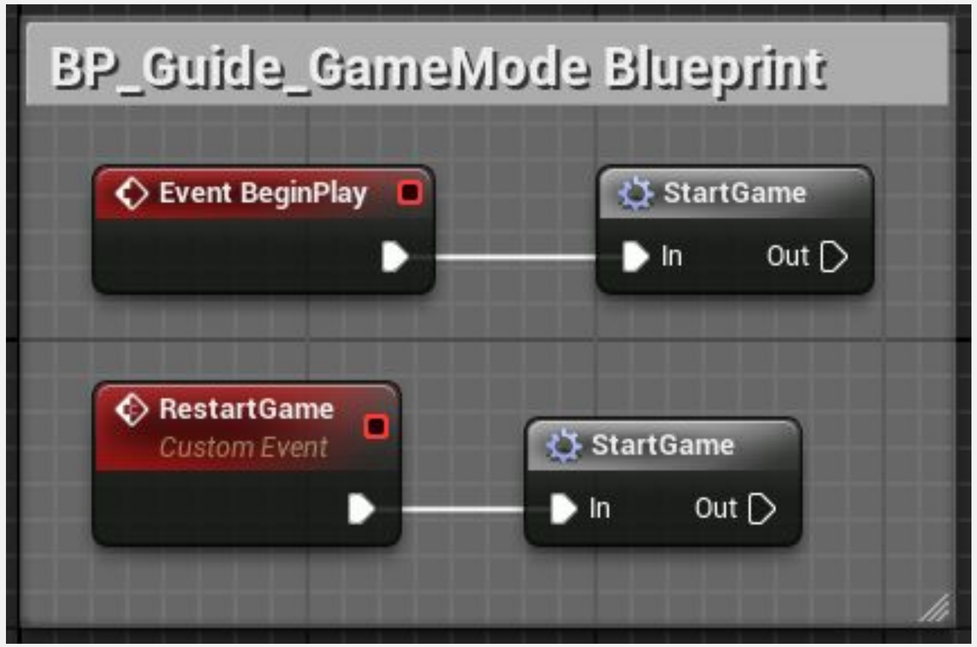


BP_GUIDE_GAMEMODE: ИСПОЛЬЗОВАНИЕ МАКРОСА

Макрос **StartGame** вызывается из события **BeginPlay** в **BP_Guide_GameMode** Blueprint.

Было создано настраиваемое событие с именем «**RestartGame**», которое также вызывает макрос **StartGame.**

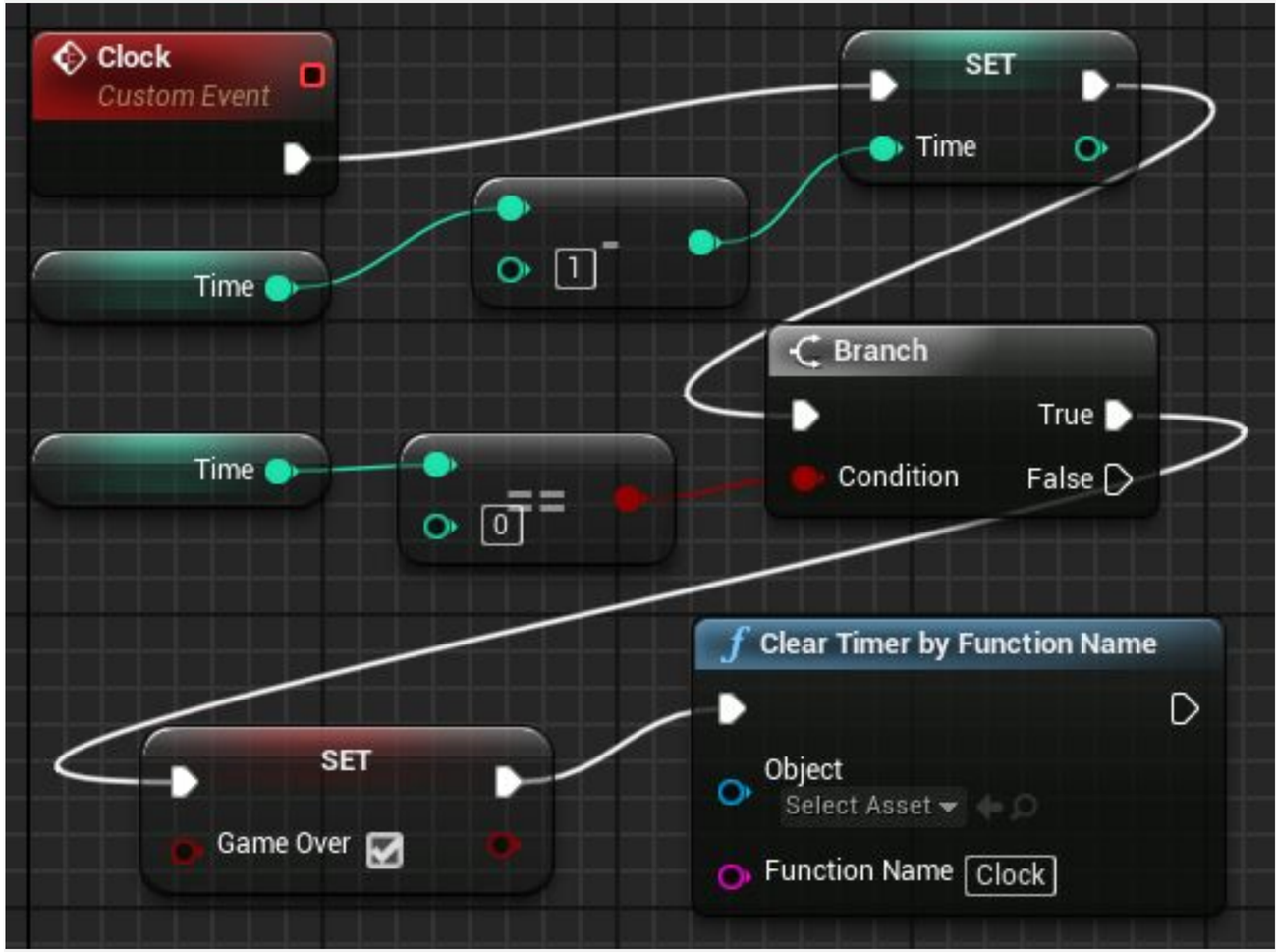
Пользовательское событие **RestartGame** вызывается из **ThirdPersonCharacter** Blueprint, когда игрок нажимает клавишу **Enter**, как показано на нижнем изображении справа.



BP_GUIDE_GAMEMODE: СОБЫТИЕ CLOCK

Clock - это настраиваемое событие, которое **Timer** вызывает каждую секунду. Он имеет следующие обязанности:

- Уменьшить на «**1**» значение переменной **Time**
- Чтобы проверить, равно ли значение переменной **Time** «**0**»; если «**true**», выполнить следующие действия:
- Установите для **Boolean** переменной **GameOver** значение «**true**».
- Очистите таймер, чтобы он перестал вызывать событие **Clock**

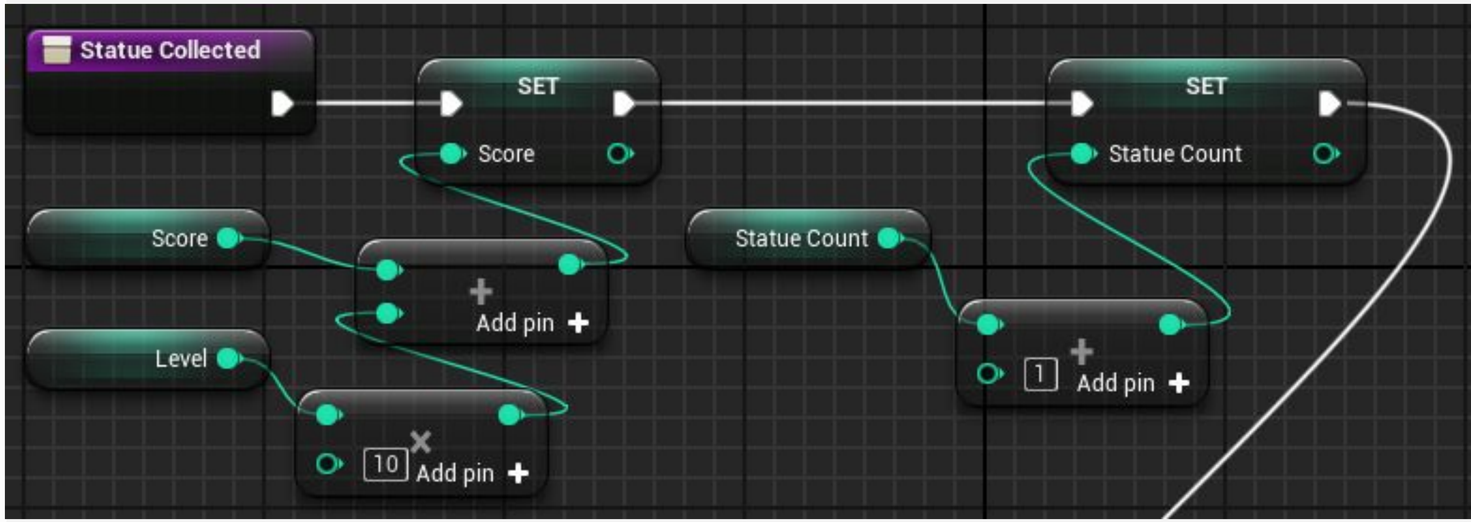


BP_GUIDE_GAMEMODE: СОБРАННЫЕ СТАТУИ 1/2

“**Statue Collected**” - это функция в **BP_Guide_GameMode**, которая вызывается классом **BP_Statue**, когда игрок получает статую.

Первая часть функции выполняет следующие действия:

- Добавляет к значению переменной **Score** очки, полученные, когда игрок собирает статую, которая рассчитывается с помощью выражения «**10 x Уровень**».
- Добавляет значение «**1**» в переменную **StatueCount**, в которой хранится количество собранных статуй.



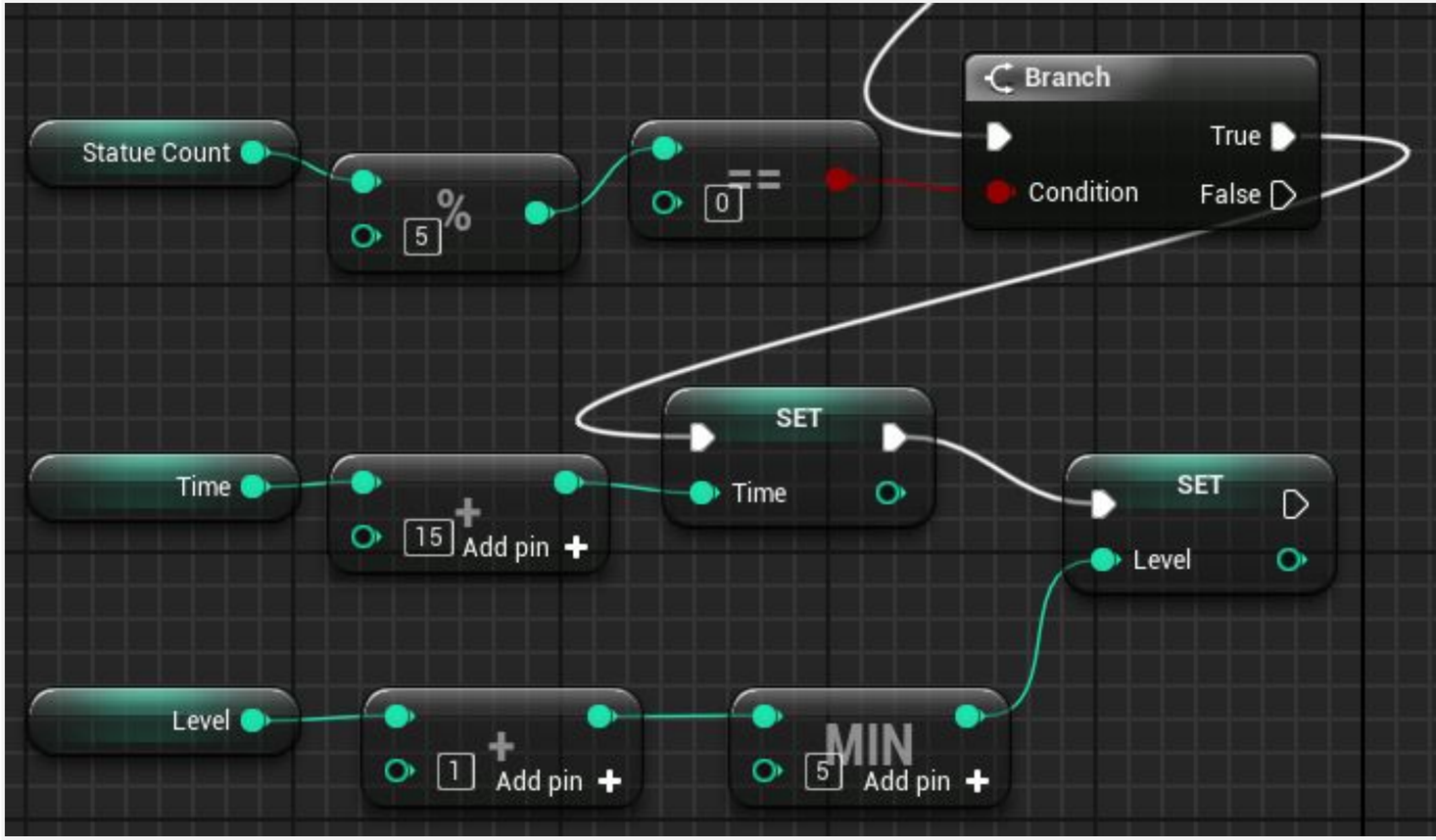
BP_GUIDE_GAMEMODE: СОБРАННЫЕ СТАТУИ 2/2

Вторая часть функции Statue Collected выполняет следующие функции:

- Проверяет, кратно ли значение переменной **StatueCount** 5. Если «**true**», выполняется следующие действия:
  - Добавляет значение «**15**» к переменной **Time**.
  - Добавляет значение «**1**» к переменной уровня, при этом максимальное значение переменной уровня ограничено «**5**».

Это означает, что за каждые пять собранных статуй игрок продвигается на уровень и получает 15 дополнительных секунд времени.

Оператор по **модулю** (%) возвращает остаток от деления.



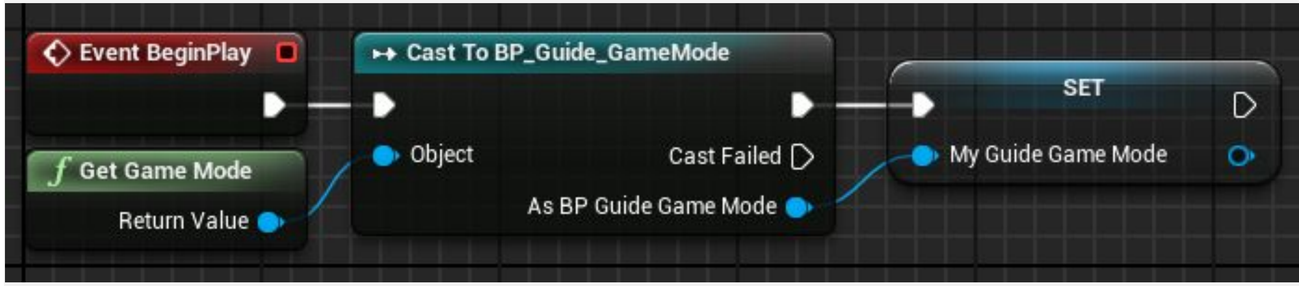
BP_GUIDE_HUD

Создайте новый класс Blueprint и выберите «**HUD**» в качестве родительского класса. Назовите его «**BP_Guide_HUD**».

Создайте переменную с именем «**My_Guide_GameMode**» типа «**BP_Guide_GameMode Object Reference**».

Изображение справа показывает, что событие **BeginPlay** получает ссылку на игровой режим, приводит ссылку на **BP_Guide_GameMode** и сохраняет эту ссылку в переменной.

Эта ссылка будет использоваться для доступа к переменным, которые будут отображаться на экране.



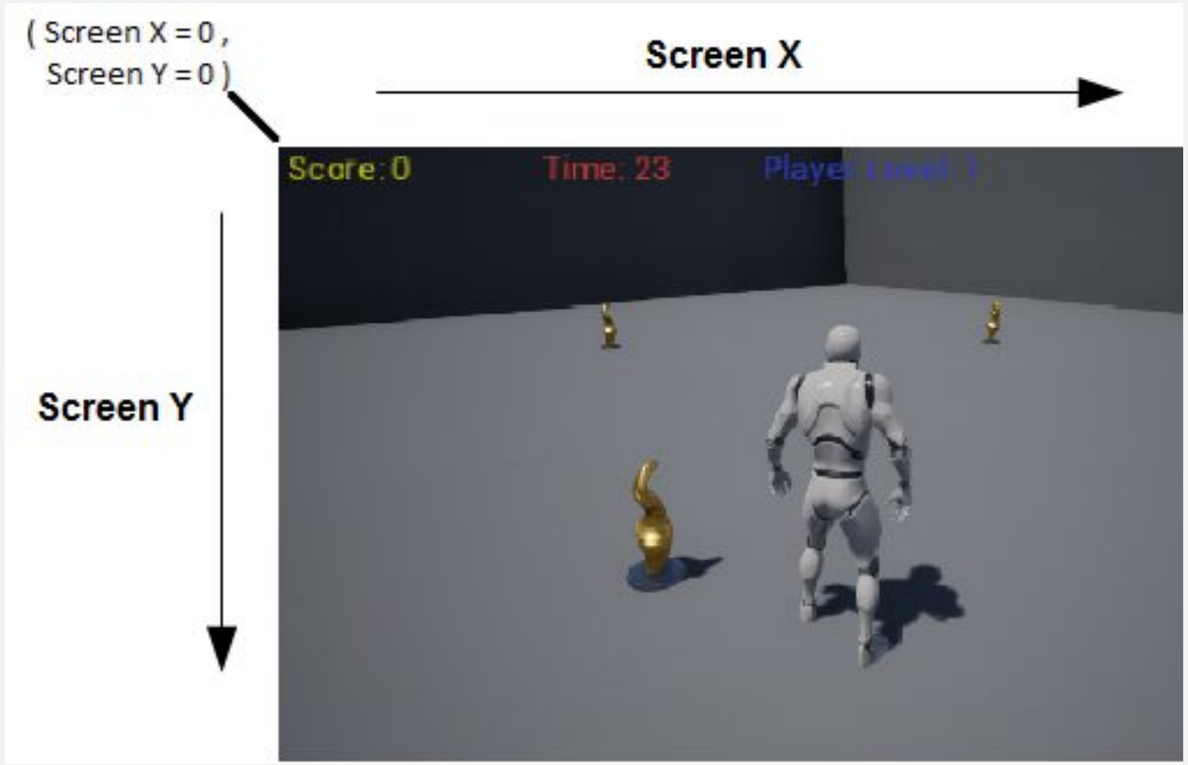
BP_GUIDE_HUD: ЭКРАННЫЕ КООРДИНАТЫ

Единственное действие, которое будет использоваться для рисования в этой игре, - это функция «**Draw text**». Эта функция принимает некоторые входные значения, в том числе значения для двух параметров, известных как «**Screen X**» и «**Screen Y**», которые зависят от разрешения экрана. Например, обычным разрешением является 1280 x 720 пикселей.

Эти значения представляют собой координаты экрана, на котором будет нарисован текст. Верхнее левое положение экрана - это начало координат, где значения **Screen X** и **Screen Y** равны «**0**». Изображение справа показывает, как значения этих двух параметров определяют, где будет отображаться текст.

Значение **Screen Y** , используемое переменными **Score**, **Time** и **Level**, равно «**10**», поэтому соответствующий текст отображается в верхней части экрана.

Используемые значения **Screen X**: «**10**» для переменной **Score**, «**300**» для переменной **Time** и «**550**» для переменной **Level**.

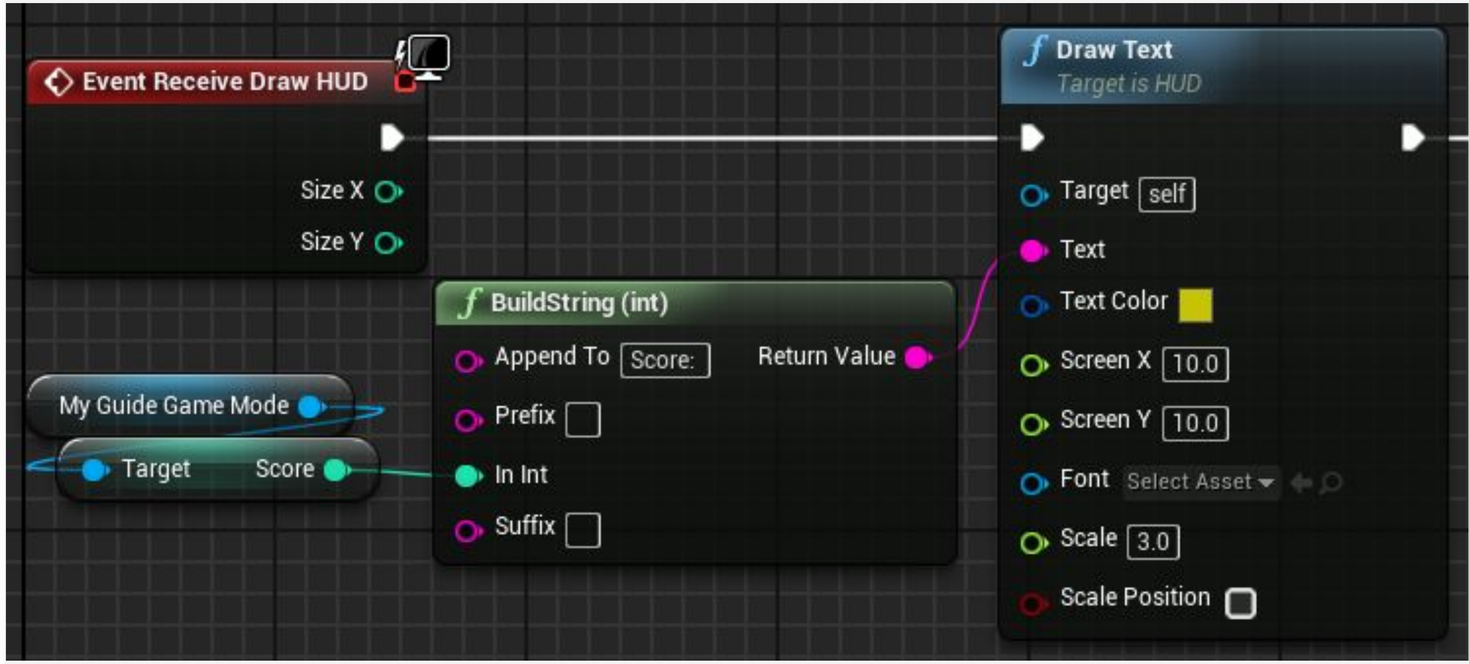


BP_GUIDE_HUD: ПОЛУЧЕНИЕ ОТРИСОВКИ HUD 1/3

Чтобы рисовать на экране, событие **Receive Draw HUD** должно быть добавлено в Event Graph. Это событие доступно только в Blueprints на основе класса HUD.

Функция **Draw Text** класса HUD используется для печати счета на экране, как показано на изображении справа.

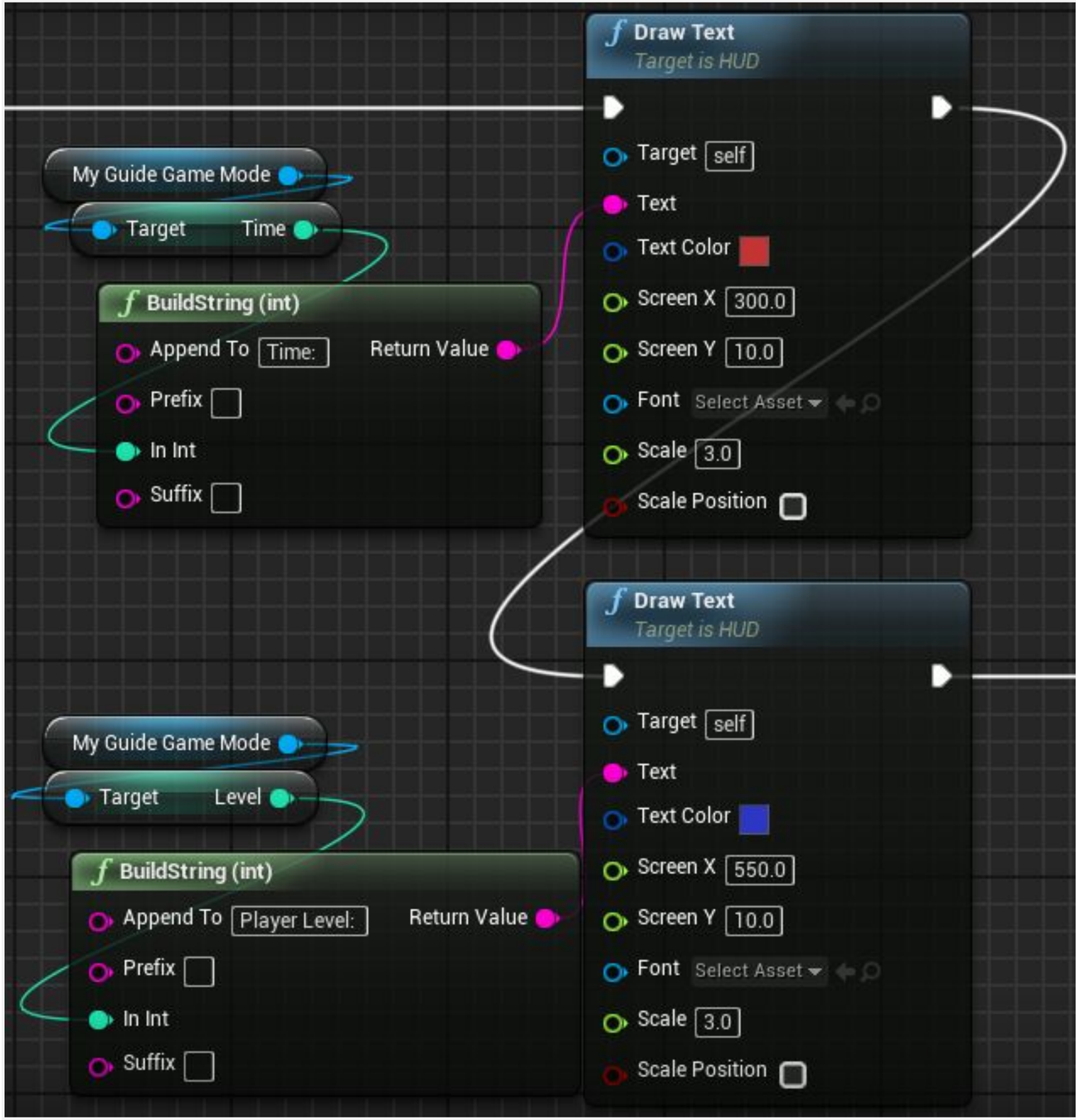
Функция **BuildString** (int) используется для создания строки, содержащей текст «**Score:**» плюс текущее значение переменной **Score**. Строка с результатом передается в параметр **Text** функции **Draw Text**.



BP_GUIDE_HUD: ПОЛУЧЕНИЕ ОТРИСОВКИ HUD 2/3

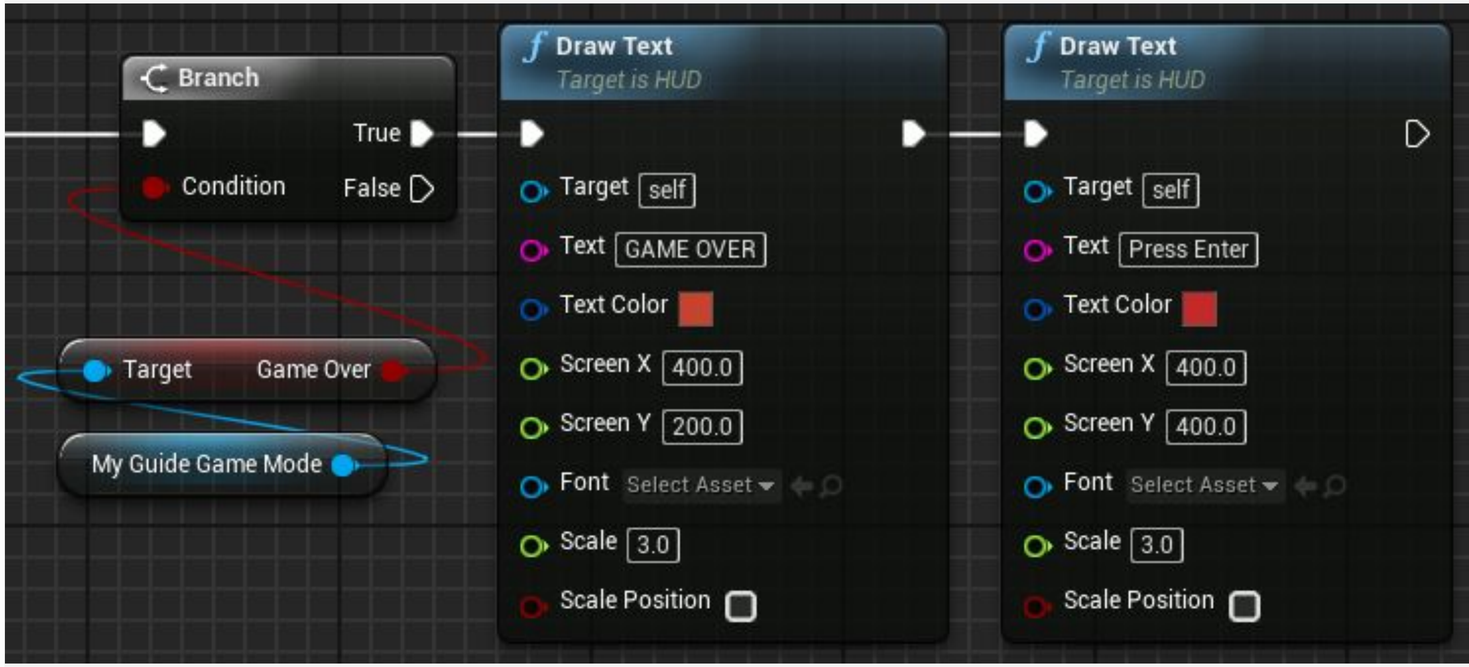
Настройка рисования текста для переменных **Time** и **Level** очень похожа на настройку рисования текста для переменной **Score**.

Единственные изменения в функции **Draw Text** касаются параметров **Text**, **Text Color** и **Screen X**.



BP_GUIDE_HUD: ПОЛУЧЕНИЕ ОТРИСОВКИ HUD 3/3

Последняя часть события **Receive Draw HUD** проверяет, имеет ли значение переменной **GameOver** значение «**true**». Если это так, то он рисует на экране строки «**GAME OVER**» и «**Press Enter**».



BP_STATUE

Создайте новый класс Blueprint и выберите «**Actor**» в качестве родительского класса. Переименуйте его в «**BP_Statue**».

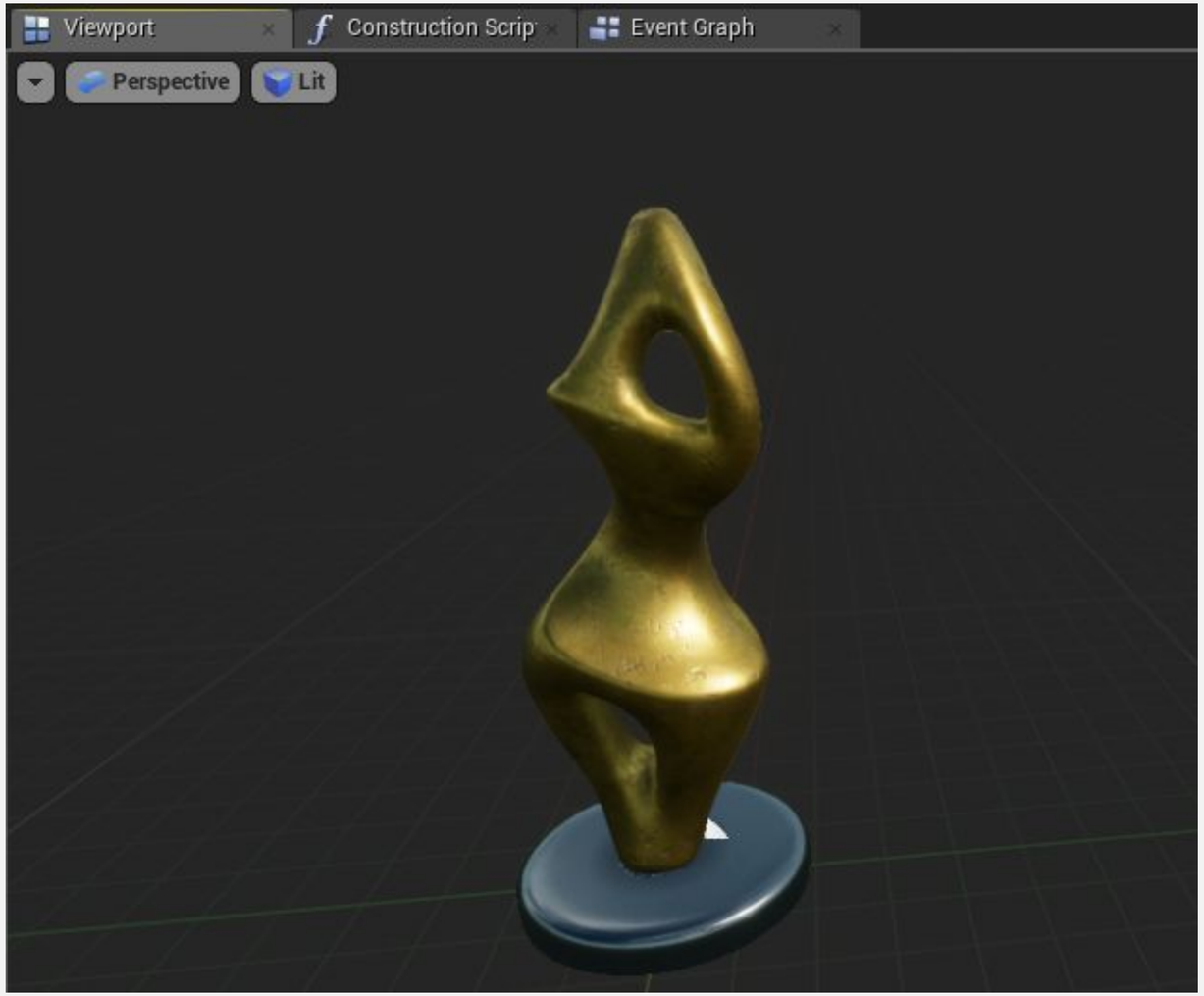
Добавьте компонент **Static Mesh**. На панели Details для компонента **Static Mesh** установите для свойства **Static Mesh** значение «**SM_Statue**».

Установите для свойства **Element 0** в разделе **Materials** значение“**M_Metal_Gold**”.

Установите свойство **Collision Presets** на “**OverlapAllDynamic**”.

Скомпилируйте и сохраните Блюпринт класс.

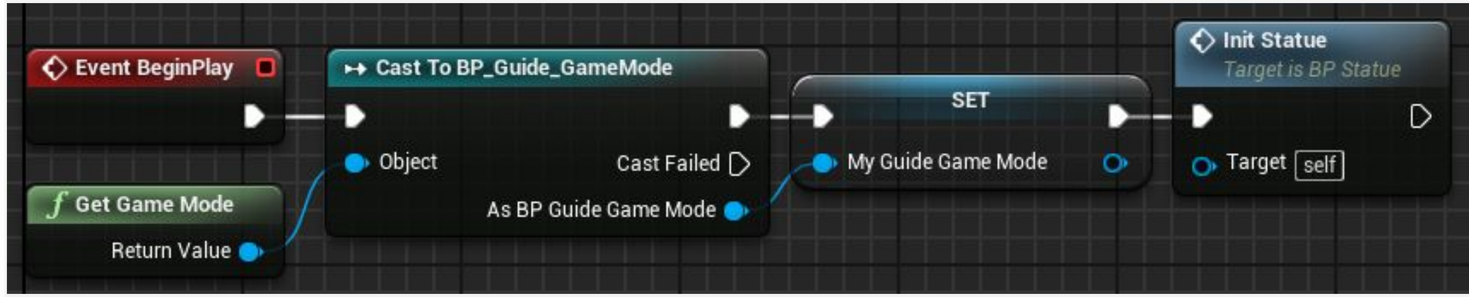
Добавьте три экземпляра **BP_Statue** Blueprint на уровень в любом месте.



BP_STATUE: СОБЫТИЕ BEGIN PLAY

Изображение справа показывает, что событие **BeginPlay** получает ссылку на Game Mode, приводит ссылку на **BP_Guide_GameMode** и сохраняет эту ссылку в переменной.

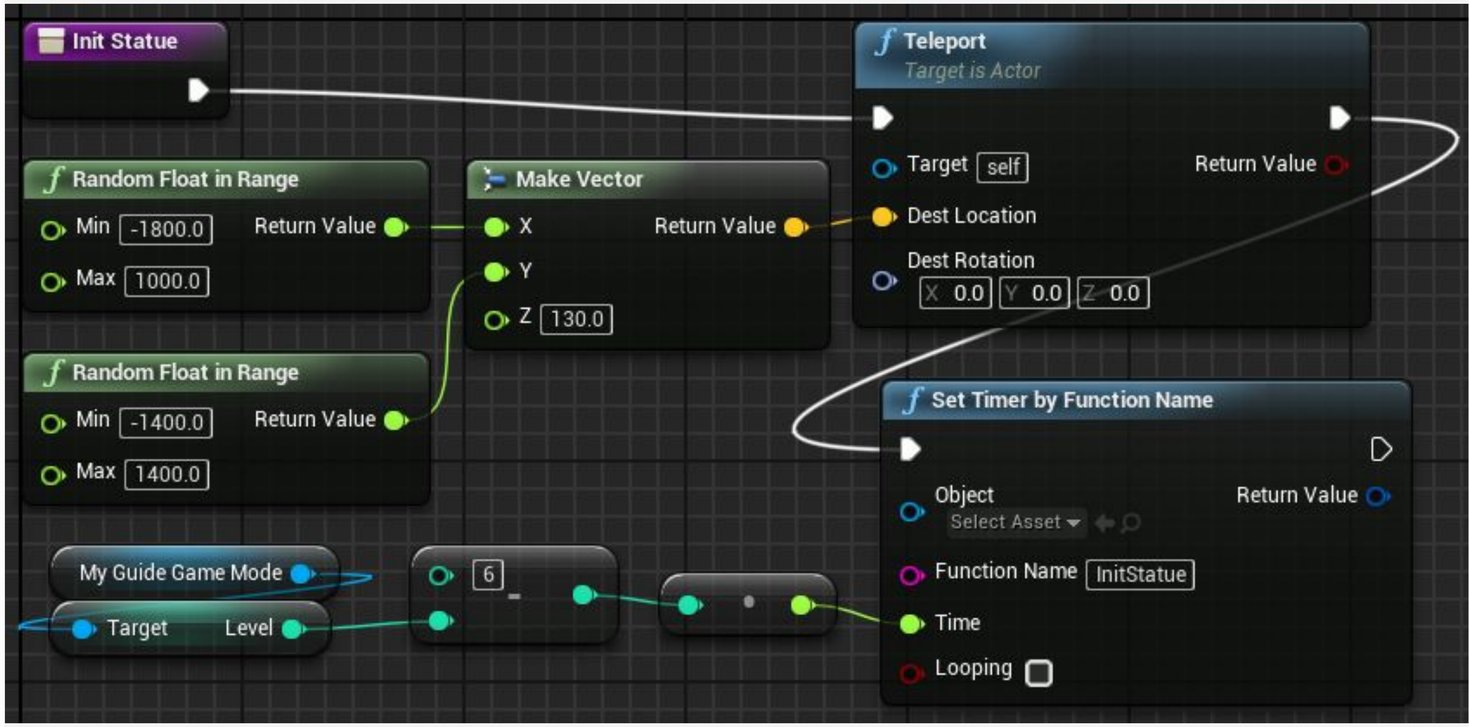
После этого вызывается функция **InitStatue**.



BP_STATUE: ФУНКЦИЯ INIT STATUE

В функции **InitStatue** есть две основных ноды:

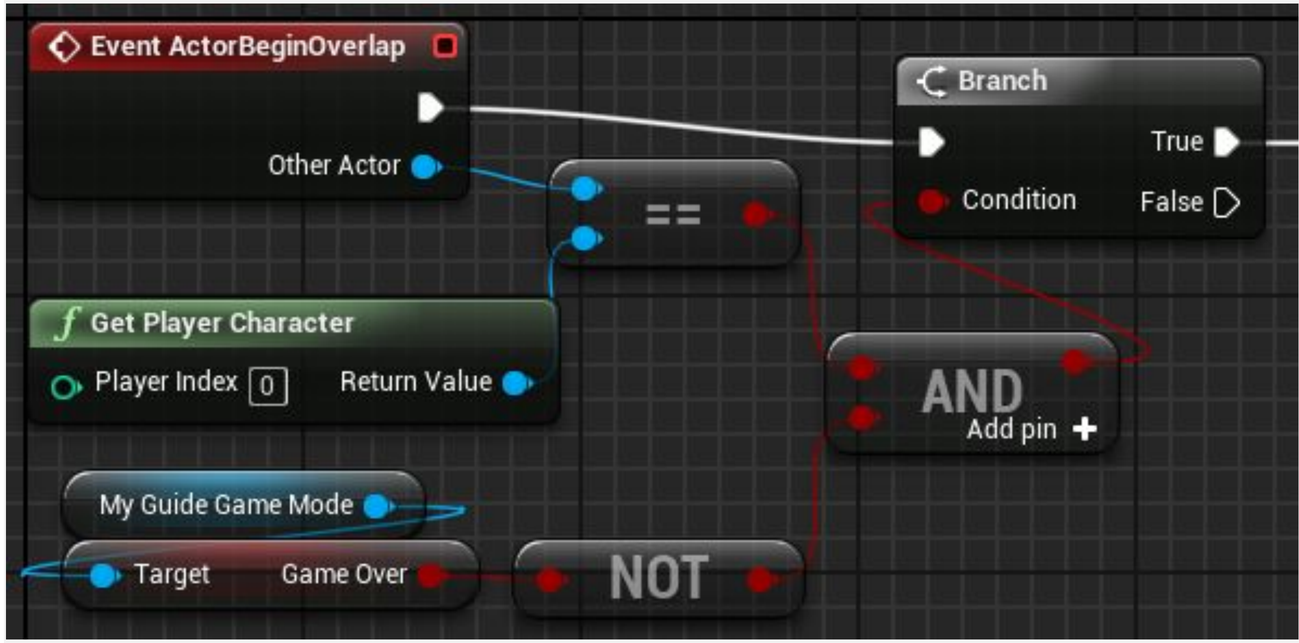
- **Teleport**: этот узел устанавливает новое место для статуи. Местоположение представлено вектором (**X, Y, Z**), где значения параметров **X** и **Y** случайны. Минимальные и максимальные значения параметров **X** и **Y** представляют собой область игры.
- **Set Timer by Function Name**: этот таймер представляет время в секундах, которое потребуется статуе, чтобы изменить положение. По истечении этого времени вызывается функция **InitStatue** для установки новой позиции. Значение параметра **Time** является результатом выражения «**6 - Level**». «**Level**» - это переменная из класса **My_Guide_GameMode**, представляющая уровень игрока.



BP_STATUE:
ACTOR BEGIN OVERLAP 1/2

Событие столкновения **ActorBeginOverlap** запускается, когда Актор в игре перекрывает статую.

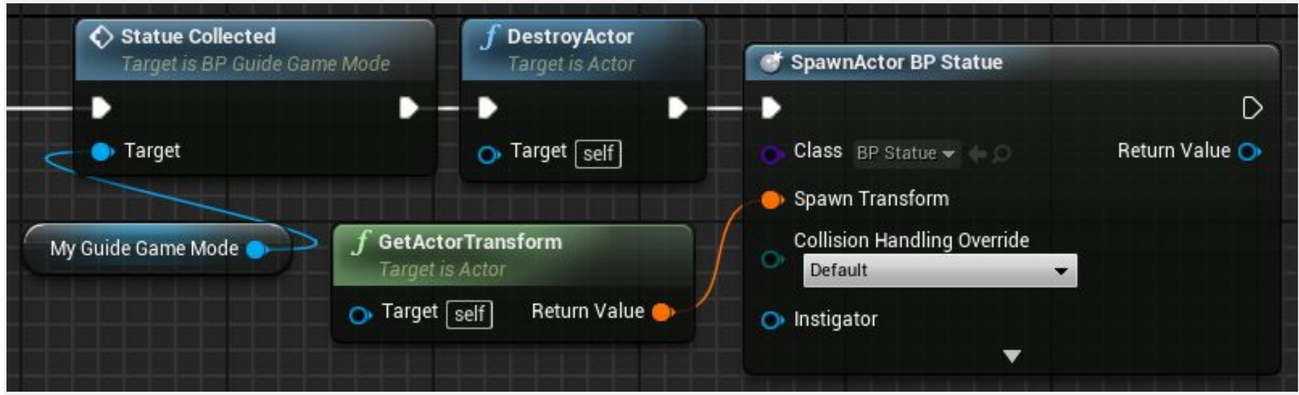
Значение, переданное входному параметру **Condition** ноды **Branch**, является результатом выражения, которое проверяет, является ли Актор, перекрывающий статую, **Player Character**, и не установлено ли для переменной **GameOver** в Игровом режиме значение «**true**». . Статуя будет собрана, только если оба эти условия выполнены.



BP_STATUE:
ACTOR BEGIN OVERLAP 2/2

Если статуя собрана, вызывается функция **Statue Collected** класса **My_Guide_GameMode** для управления счетом игрока и уровнем игры.

После этого текущая статуя разрушается и создается новая.



УСТАНОВКА РЕЖИМА ИГРЫ

Откройте **BP_Guide_GameMode** Blueprint и нажмите кнопку **Class Defaults**. Поставьте свойство **Default Pawn Class** на “**ThirdPersonCharacter**” и свойство **HUD Class** на “**BP_Guide_HUD**”**.**

В редакторе **Level Editor**, нажмите кнопку **Settings** и выберите “**World Settings**”. На панели **World Settings** установите свойству **GameMode Override** значение “**BP_Guide_GameMode**”.

Теперь игра готова к запуску.

ПРОЦЕСС ИГРЫ

Если все правильно, во время игры игрок будет управлять персонажем и должен получить статуи.

В сценарии есть три статуи, которые периодически меняют положение.

Когда игрок перекрывает статую, он получает счет, и статуя снова появляется в другом месте.

Идет обратный отсчет, и игра заканчивается, когда не остается времени.

За каждые пять собранных статуй уровень игрока увеличивается, и ко времени добавляется 15 секунд.

ИТОГ

В этой лекции была реализована простая игра, чтобы показать, как классы Blueprint взаимодействуют друг с другом в игре.

В этой игре используются макросы, пользовательские события, функции и таймеры. Класс HUD использовался для вывода некоторой информации на экран.

Кроме того, игровой режим использовался для определения правил игры и использует логику для обеспечения соблюдения этих правил.