ЛЕКЦИЯ 4

Классы игрового процесса UE4

ЦЕЛИ И ИТОГИ ЛЕКЦИИ

Goals

Цели лекции:

- Опредение классов Gameplay Framework
- Показать класс Game Mode и его классы по умолчанию
- Показать классы Player Controller и Player State
- Представить класс Pawn и некоторые подклассы
- Знакомство с классами Game Instance и Game Session

Outcomes

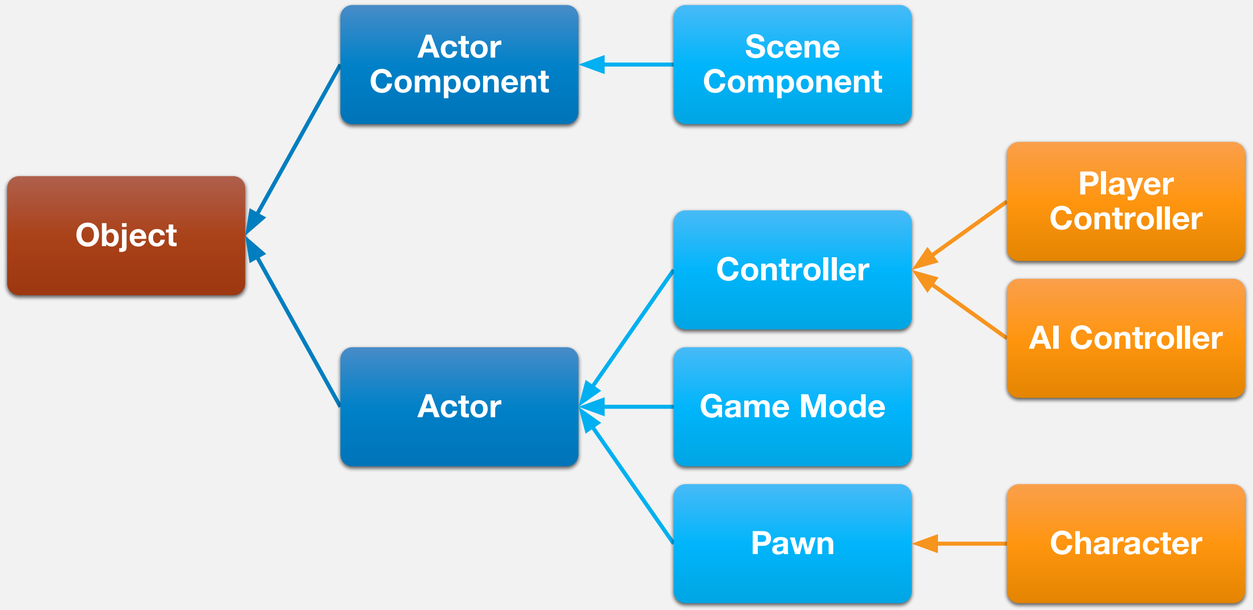
В конце этой лекции вы сможете:

- Определить общие классы Gameplay Framework
- Понять взаимосвязь между Game Mode и Game State
- Понять взаимосвязь между Player Controller и Player State
- Уметь владеть Pawn с помощью Controller
- Оформите простую информацию с помощью класса HUD

ОСНОВНЫЕ КЛАССЫ

При создании Blueprint вам будет представлен список классов, обычно используемых в качестве родительского класса Blueprint. Эти **основные классы** являются частью **Gameplay Framework** и используются для представления игроков, персонажей, контроллеров и правил игры.

Изображение справа показывает иерархию основных классов. Стрелка указывает, что класс справа наследуется от класса слева. В Unreal есть базовый класс, называемый классом **Object**.

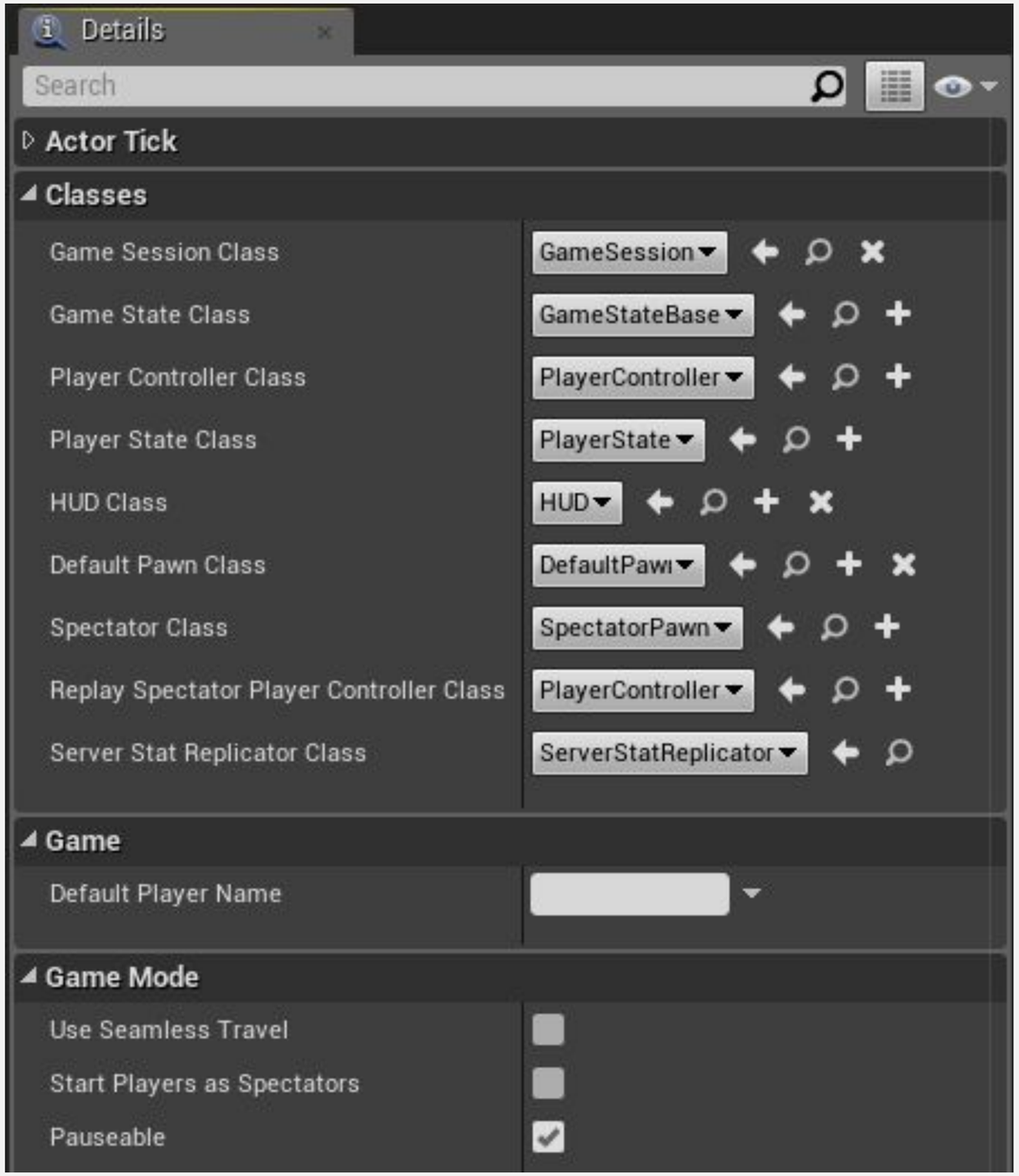


КЛАСС GAME MODE

Класс **Game Mode** используется для определения правил игры. Game mode также определяет классы по умолчанию, которые будут использоваться для создания **Pawn**, **Player Controller**, **Game State**, **HUD** и других классов, как показано на изображении справа.

У каждого уровня может быть свой Game mode. Если для уровня не указан Game mode, он будет использовать игровой режим, установленный для проекта.

В многопользовательской игре игровой режим существует только на сервере и не копируется на клиентов.

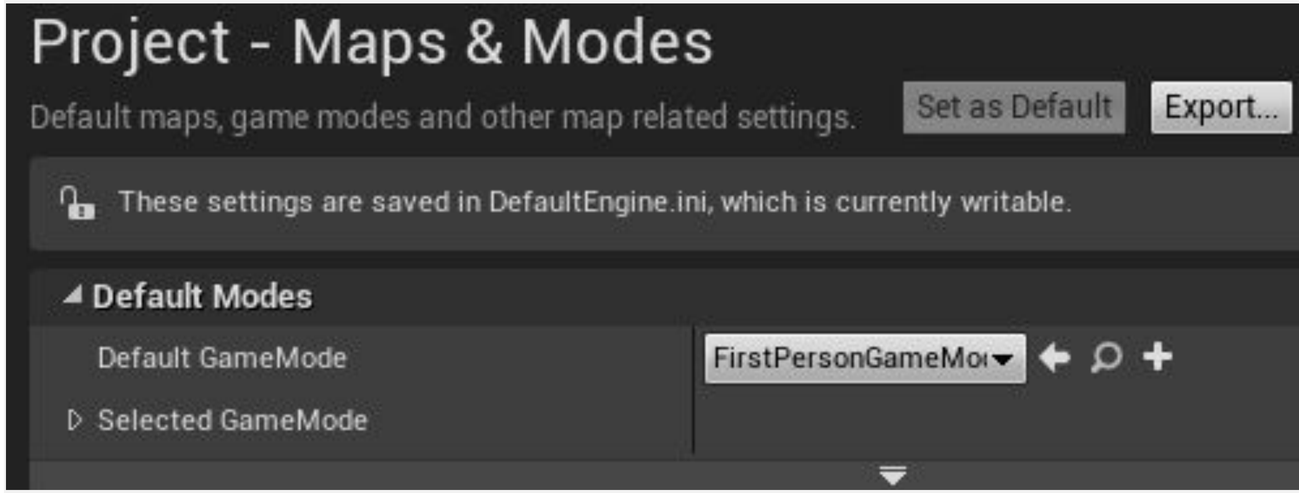


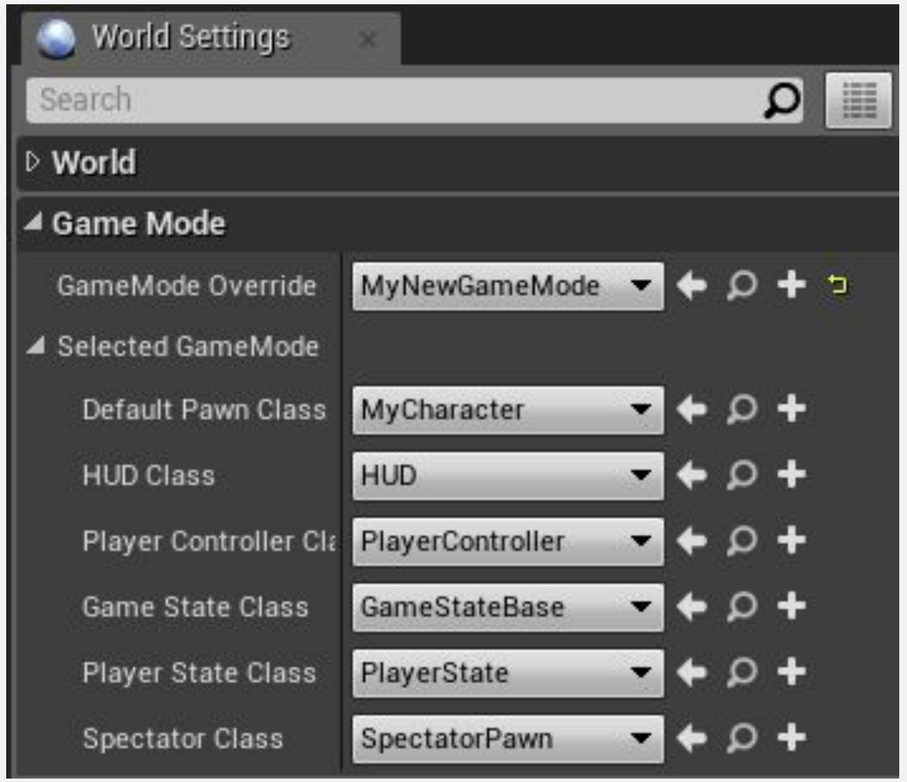
НАЗНАЧЕНИЕ ИГРОВОГО РЕЖИМА

Чтобы указать игровой режим по умолчанию для проекта, выберите **Edit > Project Settings...** В **Level Editor** и в разделе **Project** выберите параметр **Maps & Modes**. Выберите игровой режим в раскрывающемся списке свойств **GameMode** по умолчанию, как показано на верхнем изображении справа.

Чтобы указать игровой режим уровня, нажмите кнопку **Settings** в **Level Editor** и выберите параметр **World Settings**. Выберите игровой режим в раскрывающемся списке свойств **GameMode Override**, как показано на нижнем изображении.

Игровой режим уровня переопределяет игровой режим проекта по умолчанию.





КЛАСС GAME STATE

Класс **Game State** используется для отслеживания переменных, которые представляют текущее состояние игры, и используется всеми клиентами в многопользовательской игре.

Основная идея состоит в том, что игровой режим определяет правила на сервере, а состояние игры управляет информацией, которая изменяется в игре и должна быть отправлена клиентам.

Чтобы использовать новый Blueprint, основанный на классе **Game State**, вы должны назначить свой собственный класс **Game State** параметру **Game State Class** в Game Mode.

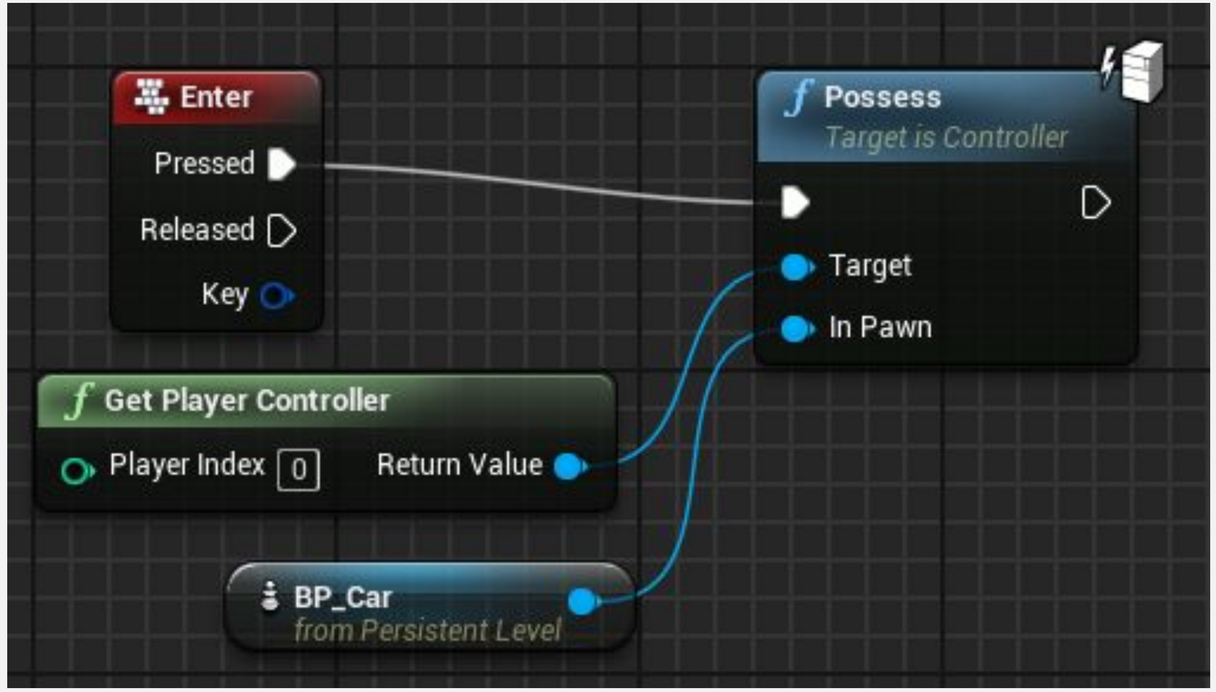
Класс Game State расширяет класс Game State Base и добавляет некоторые многопользовательские функции.

КЛАСС
PLAYER CONTROLLER

Класс **Controller** имеет два основных подкласса. Класс **Player Controller** используется игроками-людьми, а класс **AI Controller** использует искусственный интеллект для управления **Pawn**.

Классы **Pawn** и **Character** получают входные события только в том случае, если они принадлежат **Player Controller**.

Класс **Pawn**, которым владеет игрок, может быть изменен в игре. Изображение справа взято из **Level Blueprint** и показывает использование функции **Possess**. В этом примере Контроллер Игрока будет обладать Актором Pawn **BP_Car** на Уровне, когда нажата клавиша **Enter**.



КЛАСС PLAYER STATE

Класс **Player State** используется для отслеживания информации о конкретном игроке, которая должна быть передана другим клиентам в многопользовательской игре.

Player Controller существует только на клиентах, в то время как состояние игрока реплицируется на всех клиентов с сервера.

Чтобы использовать новый Blueprint, основанный на классе **Player State**, вы должны установить его в параметре **Player State Class** в игровом режиме.

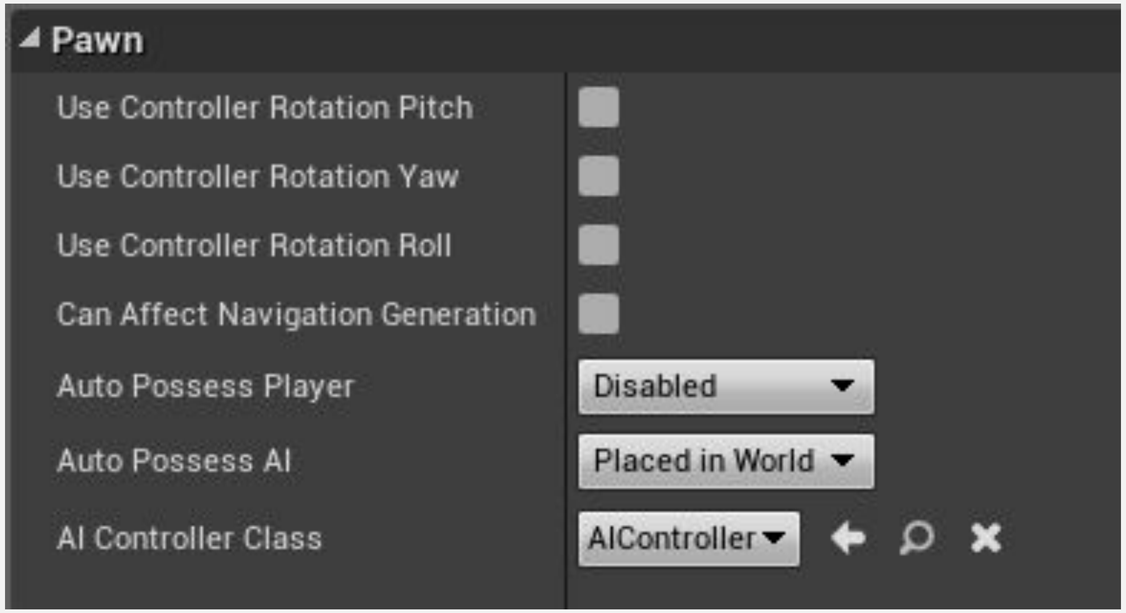
PAWN

Pawns - это Акторы, которыми могут управлять (владеть) контроллеры (Игрок или ИИ).

Класс Pawn представляет физическое тело, а класс Controller представляет мозг.

На изображении справа показаны некоторые параметры, унаследованные от класса Pawn. Класс Pawn может использовать значения вращения Контроллера, который им владеет.

Другие атрибуты показывают, каким образом Pawn находится во владении Контролера.

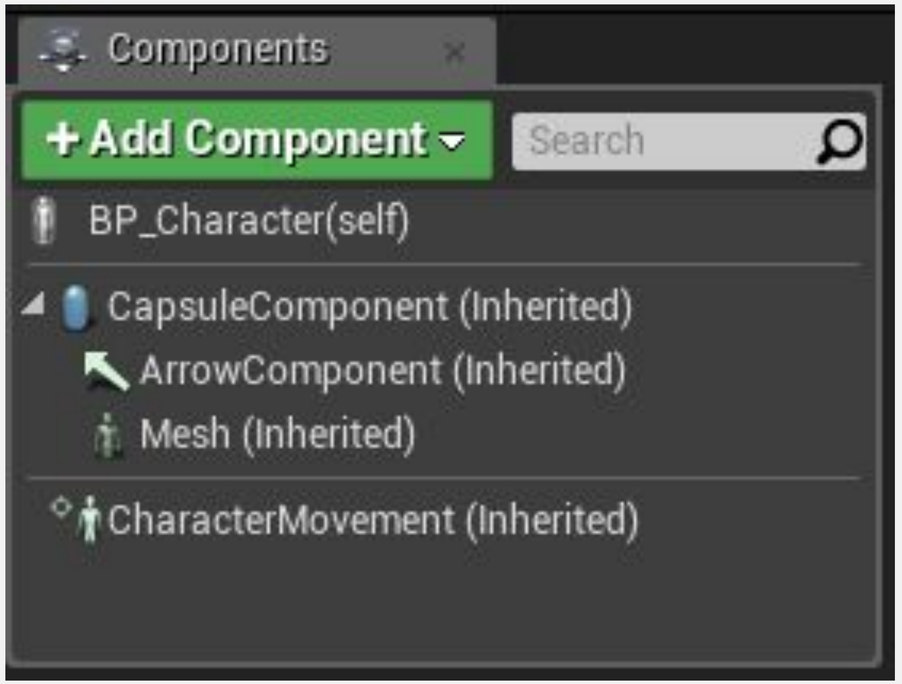


PAWN: КЛАСС CHARACTER

Класс **Character** является подклассом класса **Pawn**, который предназначен для представления двуногих персонажей, которые могут ходить, бегать, прыгать, плавать и летать. У этого класса уже есть набор компонентов, помогающих в достижении этой цели.

На изображении справа показаны компоненты, присутствующие в классе Character.

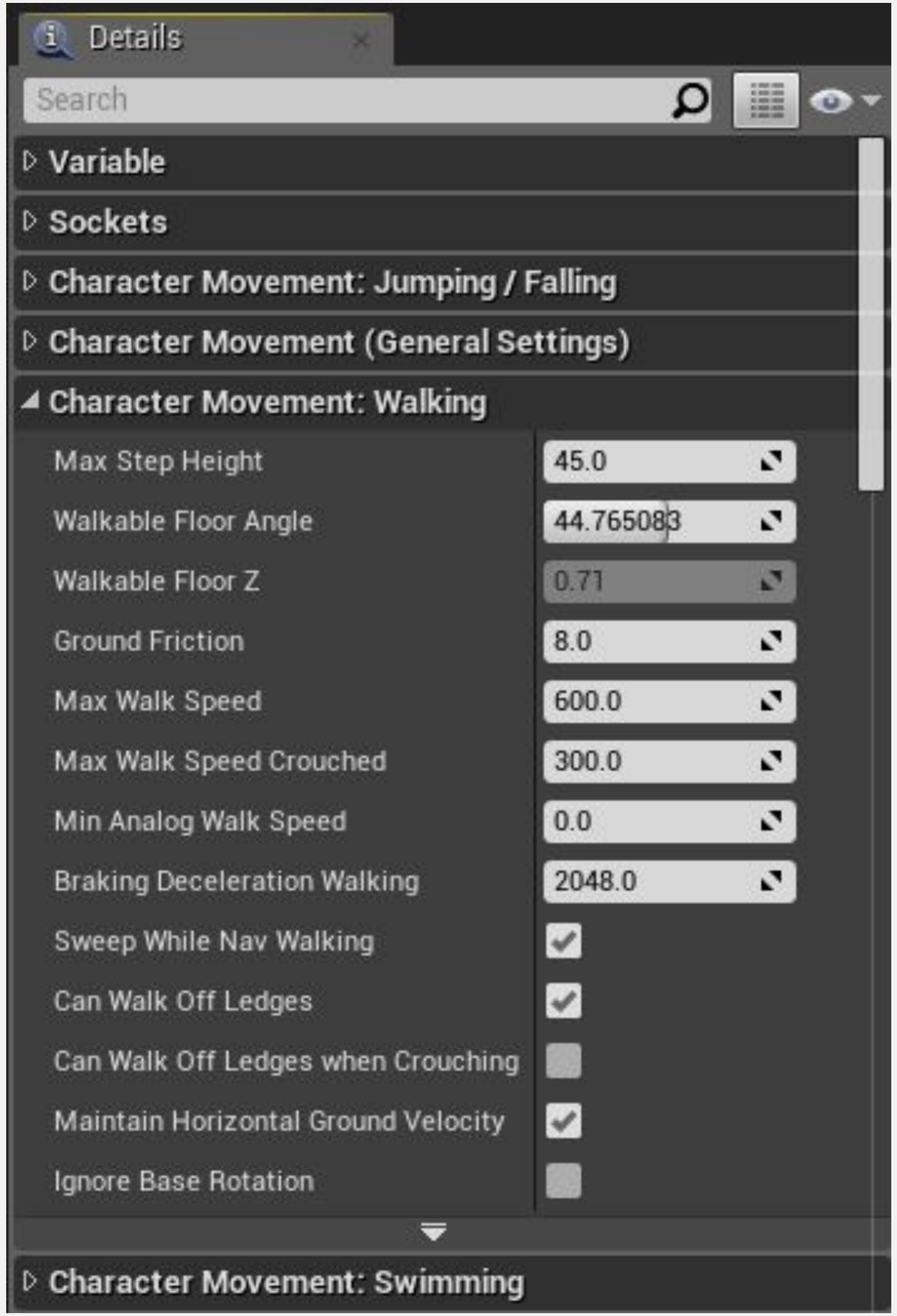
- **CapsuleComponent** используется для тестирования столкновений.
- **ArrowComponent** указывает текущее направление персонажа.
- Компонент **Mesh** - это скелетный меш, который визуально представляет персонажа. Анимация компонента Mesh управляется **animation Blueprint**.
- Компонент **CharacterMovement** используется для определения различных типов движения персонажа, таких как ходьба, бег, прыжки, плавание и полет.



КОМПОНЕНТ CHARACTER MOVEMENT

Компонент **CharacterMovement** написан на C ++ и обрабатывает движение, а также репликацию и предсказание в многопользовательских играх.

Существует множество свойств для различных типов движений, которые можно настроить на компоненте.



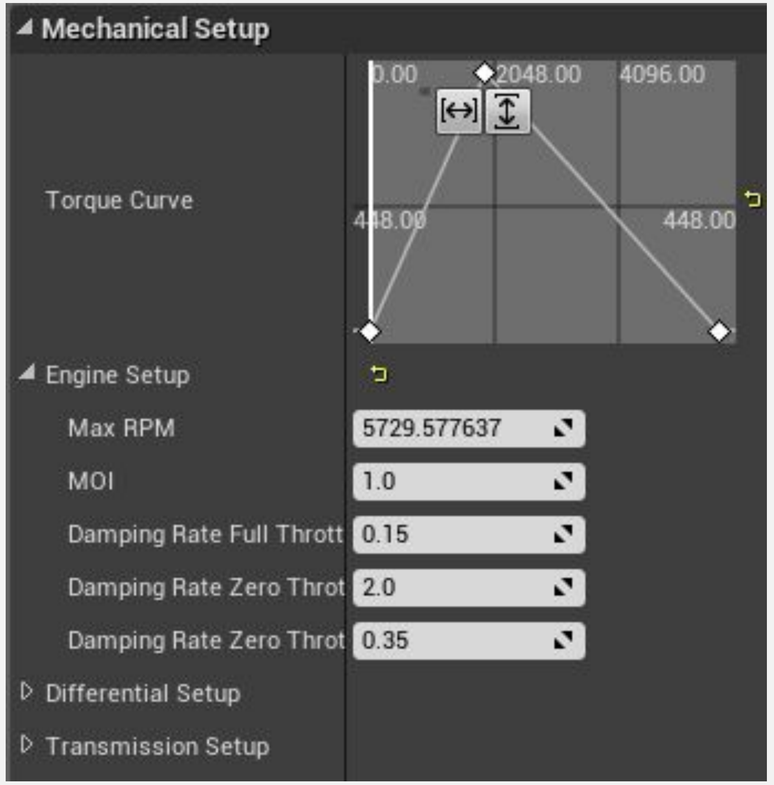
PAWN: ТРАНСПОРТ

Класс **Wheeled Vehicle** - еще один пример подкласса класса **Pawn**.

Он содержит компонент **VehicleMovement**, написанный на C ++ и содержащий код для имитации поведения транспортного средства. Он имеет множество параметров, используемых для определения движения и физики транспортного средства. Он также может обрабатывать репликацию и прогнозирование в многопользовательских играх.

Верхнее изображение справа взято из шаблона Vehicle template.

На изображении внизу показаны некоторые свойства компонента VehicleMovement.



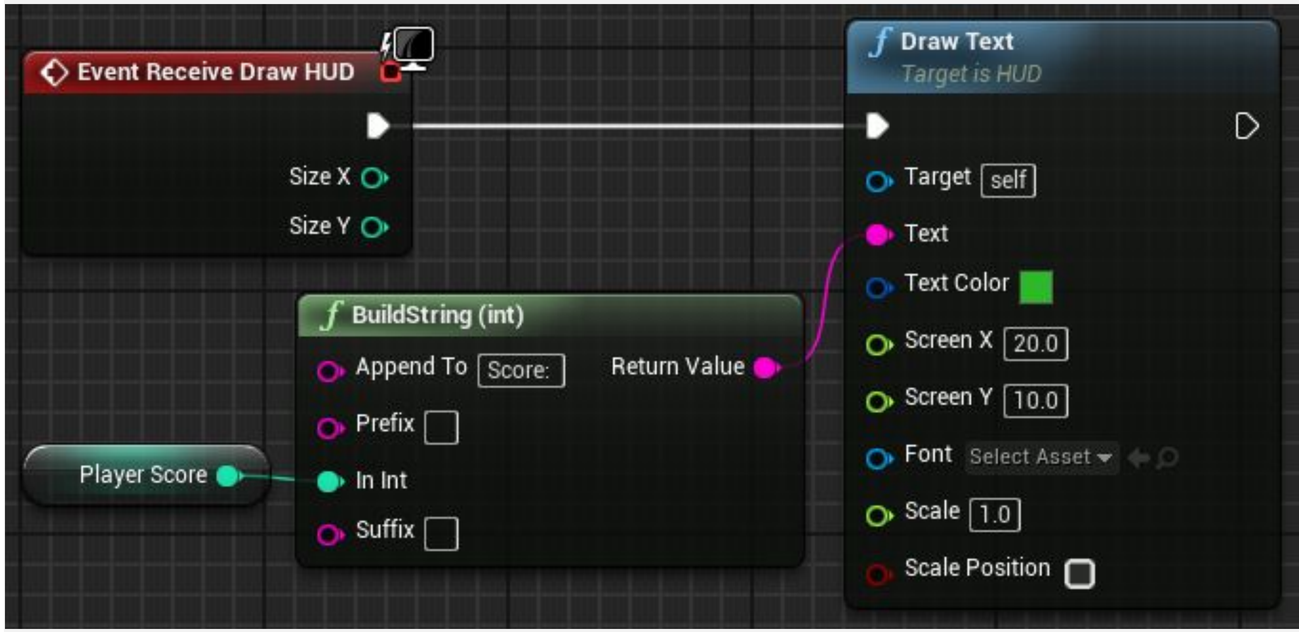
HUD (HEAD-UP DISPLAY)

Класс **HUD** используется для создания хедз-ап-дисплея (HUD), который является своего рода экранным дисплеем, обеспечивающим быстрый доступ к важной информации. HUD используется в играх для отображения различной информации для игроков, такой как их счет, время, энергия и так далее.

В Unreal Engine класс HUD - это базовый класс, содержащий **Canvas**, который представляет собой объект, на котором можно рисовать примитивы, такие как текст и текстуры.

Класс HUD содержит событие под названием «**Receive Draw HUD**», которое используется для рисования примитивов в каждом кадре.

Класс HUD существует только на каждом клиенте и не реплицируется.



КЛАСС GAME INSTANCE

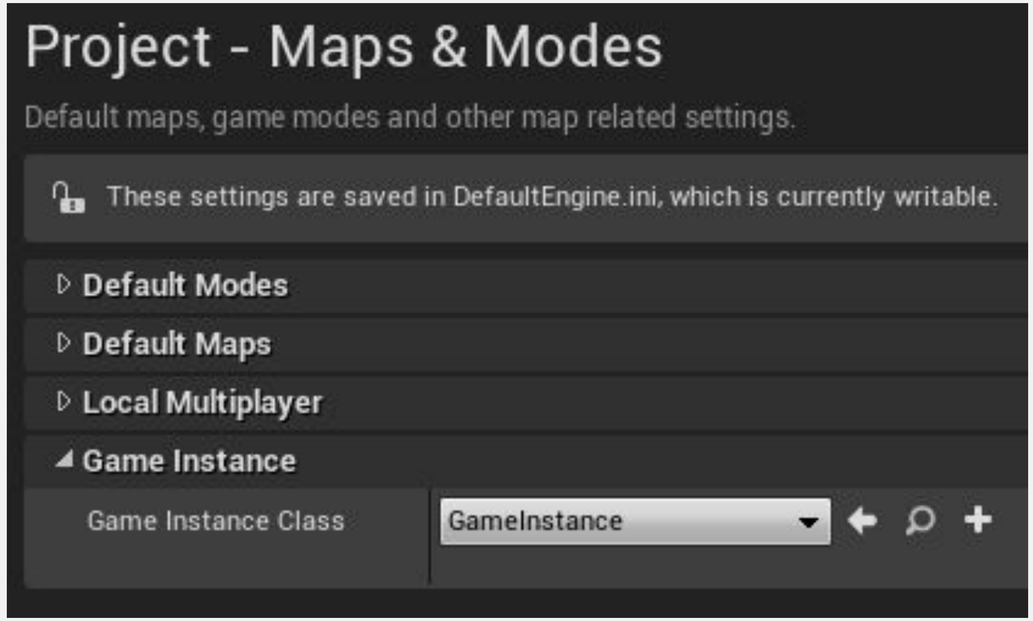
Экземпляр класса **Game Instance** создается в начале игры и удаляется только при закрытии игры.

Все акторы и другие объекты на уровне полностью уничтожаются и респавнятся каждый раз при загрузке уровня.

Класс Game Instance и данные, которые он содержит, сохраняются между уровнями.

Экземпляр игры существует только на каждом клиенте и не реплицируется.

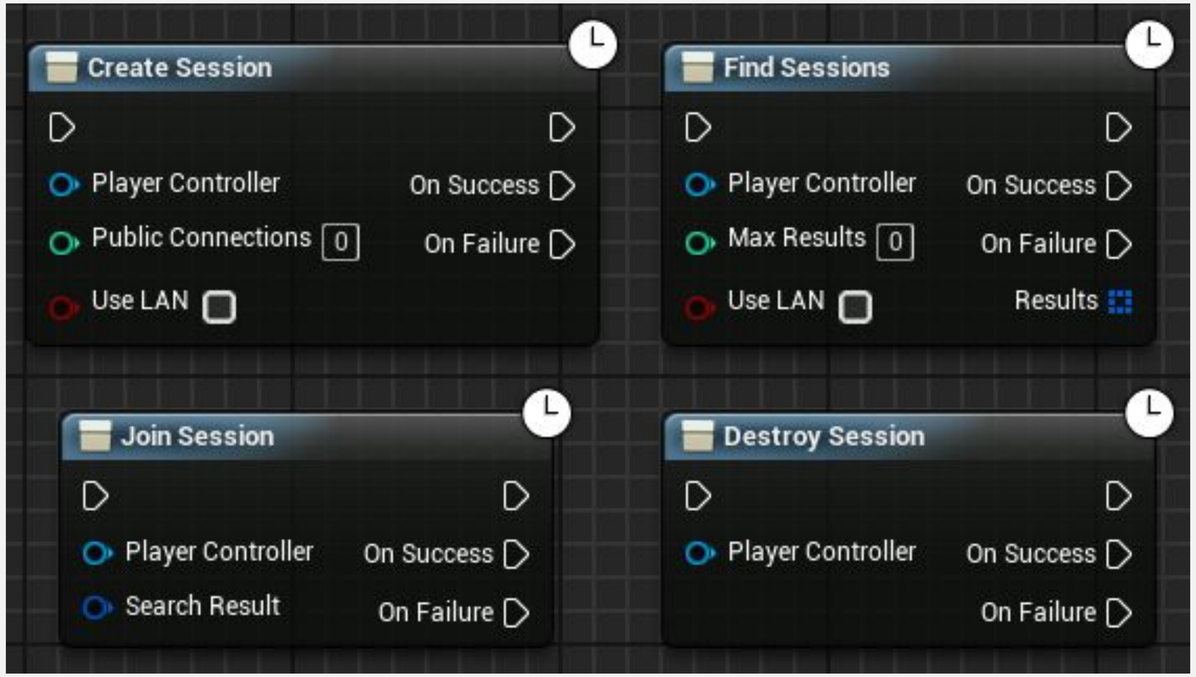
Чтобы назначить класс **Game Instance** для использования в игре, измените настройки проекта, выбрав **Edit > Project Settings > Maps & Modes**.



GAME SESSION

Проще говоря, **игровая сессия** - это интерфейс для игры, запущенной на сервере, который клиенты могут использовать для обнаружения и присоединения к игре.

На изображении справа показаны некоторые функции Blueprint, которые можно использовать для управления сессией.



ИТОГ

В этой лекции были представлены классы Gameplay Framework и отношения между некоторыми общими классами.

Он показал, как владеть Pawn с помощью контроллера и как оформить простую информацию с помощью класса HUD.