12. Обзор основных классов Unreal Engine

1. Все объекты имеют этот тип, это базовый класс UE, который можно добавить на сцену, что может быть представлено этим классом, все персонажи являются каким классом

2. Что представляют собой акторы, из чего состоят, каким бывает то, из чего они состоят

3. Компонентная модель, плюсы

4. Систематизирем: у нас есть \_, в котором есть \_, которые состоят из \_, какой тип на самом деле базовый в UE, можно ли его создать

5. Класс, которым можно управлять – название, от кого наследуется, какую доп. Функциональность имеет (пример)

6. Класс, который позволяет осуществлять контроль над предыдущими объектами (из пункта 5), имеет ли визуальное представление, чем является по сути, на каком промежутке времени существует, можем ли мы управлять различными паунами, переключаться между ними или не управлять никаким, имеет ли инпут, что там принято обрабатывать

7. Получившаяся схема между павнами и контроллером и актором

8. Класс павна по умолчанию, что тогда произойдет, что надо перед этим создать, что создается при старте

9. Класс персонажа, от кого наследуется, какую имеет доп. Функциональность по сравнению с павном

10. Установка дефолтного класса в чарактера, что происходит

11. Обновление схемы из 7-го пункта с учетом чарактера

12. Еще один класс контроллера, за что отвечает + обновление схемы из пункта 11 (наследование контроллеров)

13. Основной класс уровня, что в нем задается, определяется

14. Класс, отвечающий за пользовательский интерфейс, расшифровка, что используют вместо него, но как можно использовать данный класс все равно

15. Специальный класс, которым владеет наш павн после смерти

16. Класс для хранения различной статистики игрока, на каком отрезке времени существует

17. Класс, использующийся в многопользовательской логике, что происходит через него

18. Для чего мы определяем все данные классы, может ли их быть много, что происходит при переходе на другой уровень

19. Класс, который существует на протяжении всей игры, что в нем хранят

20. Итог:

1. Все объекты, которые находятся в нашем игровом мире имеют тип AActor. Это базовый класс Unreal, который можно добавить на сцену.

Это может произвольная геометрия, спецэффект, звук. Все персонажи являются акторами. Мы можем размещать актор вручную, либо динамические их спаунить в мире. Также actor может не иметь представления, внешнего вида, быть чисто логическим как, например, наш GeometryHubActor.

2. Акторы представляют собой композицию различных компонентов, которые в свою очередь и определяют его итоговый вид.

Компоненты также бывают визуальные как, например, скелетный меш, компонент системы частиц, компонент отображения текста и так далее.

И бывают чисто логические компоненты как, например, компонент движения. У данного компонента нет никакого представления, но имеются параметры и настройки, которые определяют, как актор будет перемещаться в пространстве, по каким законам.

3. Компонентная модель достаточно удобная. Компоненты легко переиспользовать. Мы можем создавать разные акторы, с разным набором компонентов и указывать разные настройки в данных компонентах.

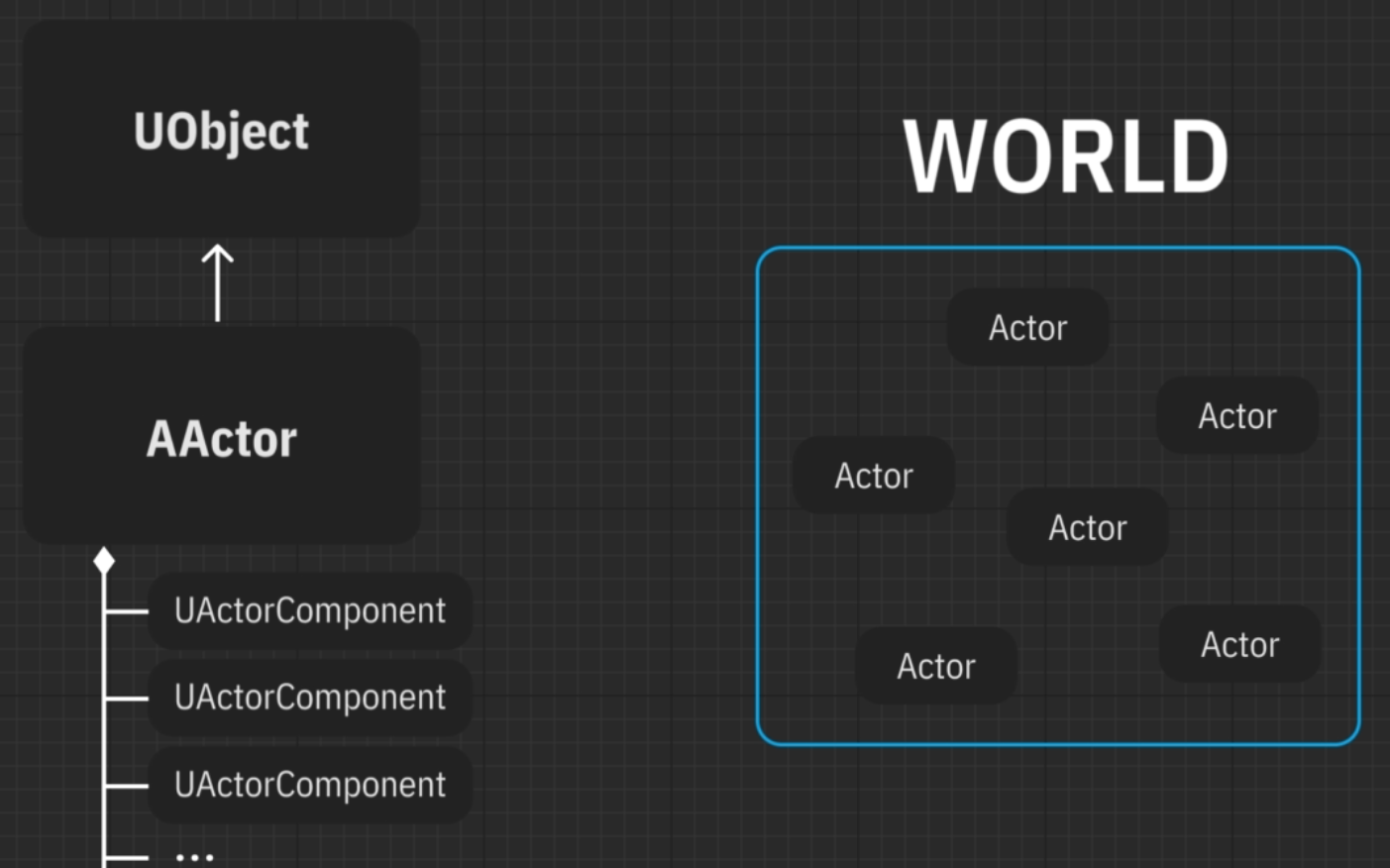
4. Давайте систематизируем немного информацию.

Итак, у нас имеется игровой мир, который состоит из акторов.

Акторы в свою очередь состоят из компонентов.

Компонентов может быть бесконечно много, и с помощью данных сущностей мы будем проектировать игру.

На самом деле актор является не самым базовым типом в unreal engine. Архетипом является тип UObject, но объект типа UObject создать на сцене у нас не получится, но можно создать динамически коде и инкапсулировать в нем какую-либо логику. С UObjectами мы с вами в данном курсе работать не будем.



5. Первый класс, который мы изучим уже в данном разделе называется APawn. Pawn - это Actor, которым можно управлять. То есть данный класс наследуется, расширяет класс Actor. И имеет дополнительную функциональность. Например, он может принимать input с различных устройств ввода и каким-то образом на данный input реагировать.

6. Класс, который позволяет нам осуществлять контроль над паунами, называется APlayerController.

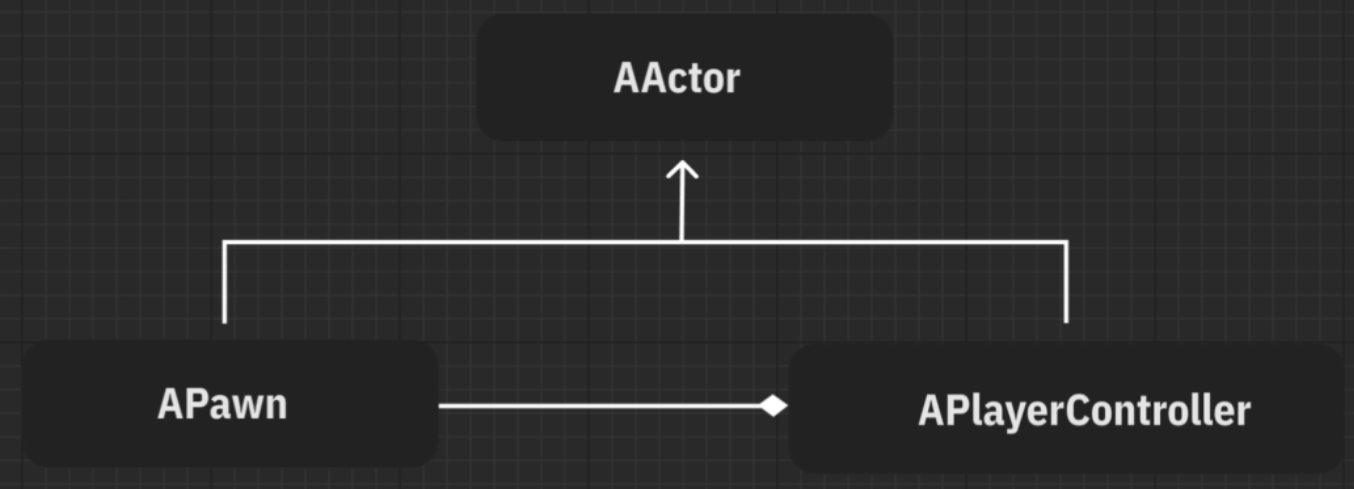
Данный класс не имеет визуального представления - это логический класс. По сути это интерфейс между нами пользователями и пауном, которым мы управляем.

APlayerController существует на протяжении всего игрового уровня, в то время как пауны могут уничтожаться, респауниться.

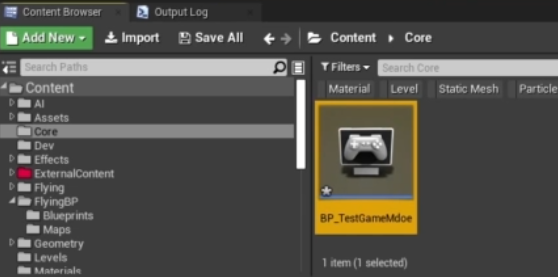
Мы можем управлять различными паунами, переключаться между ними или не управлять никаким пауном вообще.

APlayerController также имеет input. В нем принято обрабатывать события, которые не относятся к конкретному пауну. Например, выход в меню, выключение звука или вовсе закрытие игры.

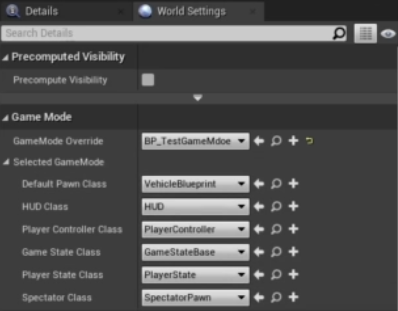
7. У нас получается следующая схема: pawn и playercontroller наследуются от актора. Оба они существуют в игровом мире, но контроллер не имеет никакого внешнего представления. Также playercontroller владеет пауном, хотя возможна ситуация, когда он не владеет ни одним из паунов.



8. Мы можем указать класс пауна по умолчанию. Тогда он будет создан во время старта игры. Перед этим нам необходимо создать собственный класс GameModeBase. О нем мы поговорим чуть позднее.



Укажем созданный класс в проперти Game Mode Override. После этого нам становятся доступны все остальные классы. И в проперти Default Pawn Class мы, например, можем указать машинку.



При запуске создался дефолтный pawn, также создался контроллер - класс для него мы пока что не меняли. Мы контролируем паун по умолчанию, но также можем переключаться между другими паунами.

9. Следующий класс, который мы рассмотрим, называется класс ACharacter. Данный класс наследуется класса APawn. То есть им также можно управлять. В данный класс добавляется дополнительная функциональность.

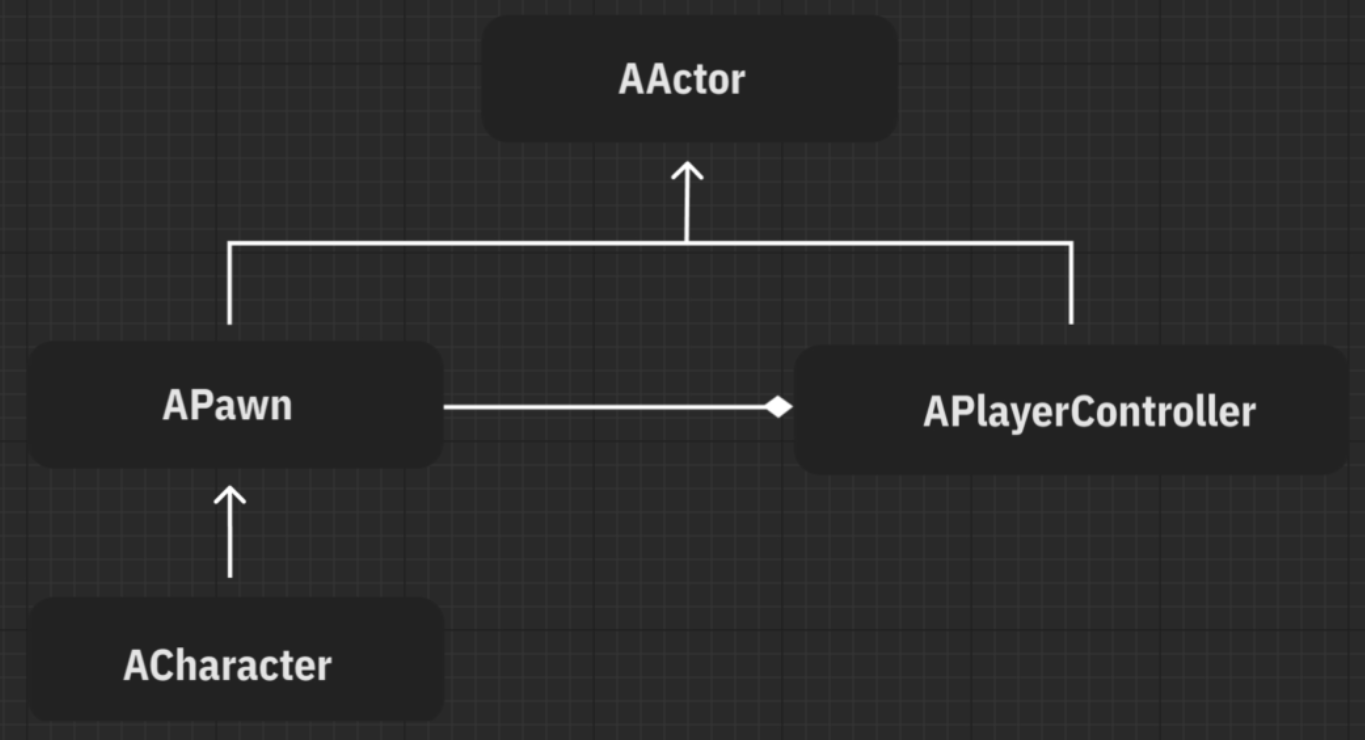
У него имеется компонент капсула, который отвечает за физические взаимодействия с внешним миром.

Имеется компонент скелетал меш, в котором мы настраиваем модель и анимации персонажа.

Также имеется логический компонет, который называется character movement. Данный компонент отвечает за то, как наш персонаж двигается в окружающей среде. За его скорость, высоту прыжка и другие настройки.

10. Чарактер является подклассом пауна - поэтому проперти Default Pawn Class мы можем поменять на класс чарактера. Я укажу класс персонажа, который мы создадим в данном курсе и также поменяю контроллер, хотя сейчас это значение не имеет. Запускаем игру и создался указанный в настройках персонаж.

11. Можно обновить диаграмму классов. Теперь она выглядит следующим образом. В курсе я иногда буду называть нашего чарактера пауном, иногда буду называть актором. Как мы видим это соответствует действительности. Чарактер является пауном, который в свою очередь является актором.

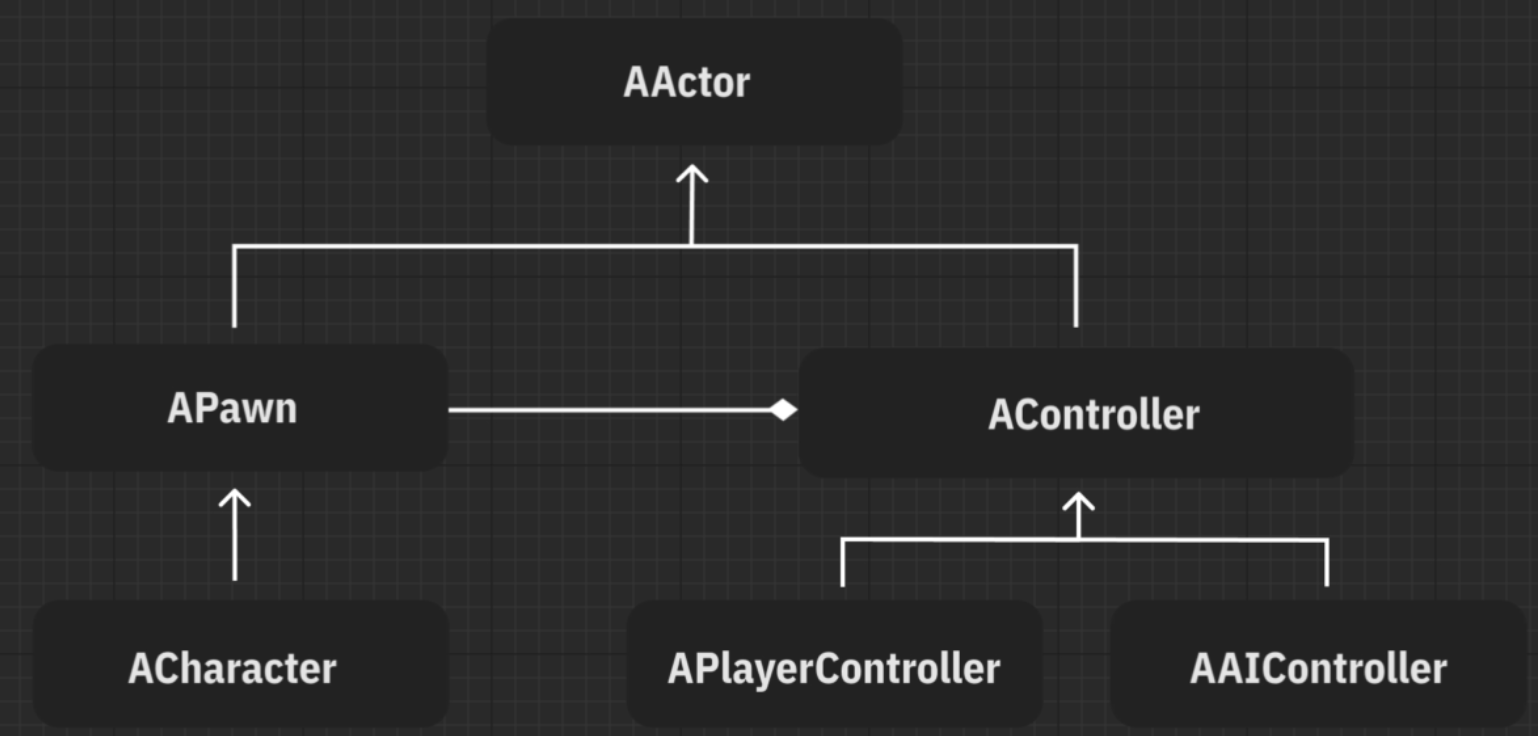


12. Двигаемся дальше. Пауном может владеть не только playercontroller, но и другой тип контроллера, который называется AIController.

Данный контроллер отвечает за искусственный интеллект. Одна из его основных задач - это запускать дерево поведения, которые мы видим на экране. Оно является мозгами бота. По сути паун может управляться как игроком, так и искусственным интеллектом.

В настройках пауна можно указать интересующий нас AIController класс в соответствующем проперти. И при запуске игры одним персонажем управляем мы, а другим искусственный интеллект.

PlayerController является подклассом базового класса AController и от него также наследуется AIController. Поэтому наша схема обновляется следующим образом.



13. Основным классом уровня является класс GameMode. В нем задаются различные настройки игрового уровня и определяются общие правила игры. Например, время игры, сколько игроков может участвовать, сколько ботов, как происходит распределение по командам, может ли игрок зареспаунится и так далее.

14. Давайте рассмотрим следующий класс. Он называется HUD. Аббревиатура расшифровывается как Head Up Display. Данный класс отвечает за пользовательский интерфейс. Это класс немного устарел. Его практически напрямую не используют. Все интерфейсы в unreal engine делаются с помощью виджетов. Мы с вами их также будем рассматривать, но как минимум HUD можно использовать как класс менеджер всех виджетов.

15. Spectator class - это класс наблюдателя. Это специальный паун, которым наш контроллер владеет, после того как наш персонаж погибает. Обычно это просто летающая камера. У нас дополнительно с вами будет черно-белый эффект у данной камеры.

16. Следующий класс PlayerState. У него также не имеется представления. Класс обычно используются для хранения различной статистики игрока: сколько противников уничтожил, сколько раз сам погиб и так далее. Данный класс также существует на протяжении всего уровня

17. Следующий класс - GameState. Он используется в многопользовательской логике, через него клиентам передается различная информация по игровому процессу, которая находится в GameMode. В многопользовательской игре класс GameMode находится только на сервере.

18. Все данные классы мы определяем для одного уровня. Уровней в нашей игре может быть много, поэтому в другом уровне данные классы могут отличаться. У другой карты данный набор классов мы можем настроить по-другому, например, при старте одного уровня мы управляем летающей тарелкой, при старте другого машиной, при старте третьего персонажем. В данном курсе мы также научимся открывать уровни программно. При открытии нового уровня все классы пересоздаются: GameMode, PlayerState, PlayerController и так далее.

19. Но в unreal также имеется класс, который существует на протяжении всей игры. Он называется game instance, в нем можно хранить различную глобальную информацию, которая не зависит от конкретного уровня: графические настройки, настройки звука и также непосредственно информацию по загружаемым уровням. Данный класс указывается в project settings. Но об этом мы с вами поговорим попозже.

20. Давайте подведем итог. Игра может состоять из нескольких уровней. Базовый класс каждого уровня это GameMode. Он отвечает за создание всех базовых классов: пауна, контроллера, спектейтора, плеер стейта, гейм стейта и hud. При открытии нового уровня создается указанный в данном уровне гейм мод и также пересоздаются все основные классы. А на протяжении всей игры, вне зависимости от текущего уровня, существует объект класса UGameInstance.

