**17. Equip animation, anim notify, equip flag, blend per bone (pose cache)**

1. Что будет из себя представлять анимация смены оружия и где будет лежать (в С++)? Какую создаем функцию и для чего?

2. Какой функционал у функции (реализация)? Где будем вызывать эту функцию?

3. Какая у нас есть анимация для смены оружия? Что из нее создали? Необходимо указать это в BP. Анимация проигрывается, но у нас есть ряд проблем – какие?

4. Мы хотим исправить проблемы из предыдущего пункта - что для этого есть у анимации = какой механизм?

5. Как создать нотифай? Какое сейчас поведение наблюдается (пока что все настроили через BP, мы выводим на экран сообщение, но сколько их)?

6. Создаем свой нотифай – от кого наследуемся, где его создаем (в плюсах)? Как мы можем узнавать, что анимация дошла до нашего нотифая? Что мы добавляем и как оповещаем наш код об этом событии?

7. Теперь наша задача - получить доступ к нашему Notify и подписаться на делегата, как это сделали?

8. Теперь нам надо добавить наш вид Notify на AnimMontage – как это сделали? Какую ошибку сразу наблюдаем и почему? Как пофиксили?

9. Подытожим, какая у нас логика отслеживания момента окончания анимации?

10. Завершим логику смены оружия – запретим стрелять и переключать оружие до конца анимации – как?

11. Какая у нас осталась последняя проблема с проигрыванием анимации (визуальная)? Почему это происходит? Как еще раз работает нода Slot?

12. Как фиксим наш визуальный баг – что добавляем на анимационный граф персонажа? Что нужно не забыть настроить в AnimMontage смены оружия?

13. Как теперь нам правильно соединять текущую анимацию и анимацию Equip – новая нода и ее настройка + как работает?

1. Анимация смены оружия так же будет представлять из себя AnimMontage, мы будем хранить его в WeaponComponent.

И дополнительно создадим еще одну функцию, которая будет проигрывать AnimMontage.

У нас позже будет еще анимация перезарядки, поэтому удобно сразу под проигрывание анимации выделить метод.

Изображение выглядит как текст

Автоматически созданное описание

2. В самой функции нам надо получить персонажа для вызова его метода PlayAnimMontage. Проверять указатель на аним монтаж не надо, так как он проверяется внутри самой функции.

Данную функцию будем вызывать внутри EquipWeapon.

Изображение выглядит как текст

Автоматически созданное описание

3. В наших ассетах имеется анимация Equip. На базе нее создаем AnimMontage. Называем AM\_Equip, переносим в нашу папку с анимациями.

В блюпринте персонажа в компоненте оружия устанавливает наш аним монтаж.

Теперь анимация смены оружия проигрывается, однако у нас есть ряд проблем.

Если мы быстро переключаемся между оружием, у нас постоянно вызывается PlayAnimation. Так же мы можем стрелять в процессе изменения анимации.

4. Мы можем при старте анимации блокировать стрельбу, смену оружия, но нам нужен механизм определения того, когда закончилась анимация, чтобы все разблокировать обратно.

У анимации есть такой механизм, он называется Notify.

Если мы откроем любой анимационный ассет, то в нижнем редакторе увидим таймлайн, где проигрываются кадры анимации, и слева подкатегорию, которая называется Notifies.

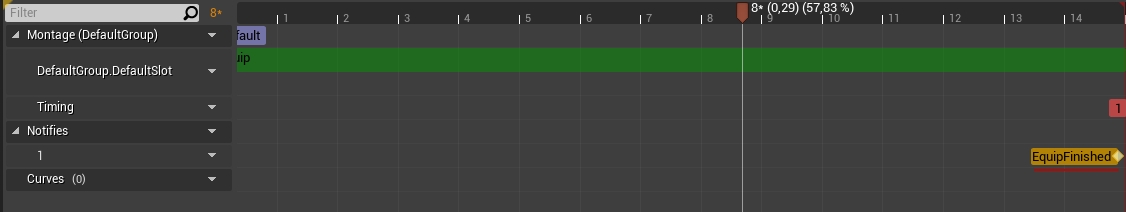
Мы можем размещать события, которые привязаны к определенным кадрам анимации.

Так же здесь можно добавлять треки, которые позволяют, насколько я понял, для удобства группировки событий (то есть можно сгруппировать событий в одном треке таким образом, во втором – другим, и выбирать между треками в плюсах, например).

Изображение выглядит как текст, электроника, снимок экрана, компьютер

Автоматически созданное описание

5. Через ПКМ на любом треке (кликнули если что в саму область), выбираем AddNotify->NewNotify и назовем наш EquipFinished и поместим его в самый конец анимации.



Перейдем в анимационный блюпринт. Чтобы подписаться на событие Notify достаточно выбрать его в категории Add Anim Notify Event.

Напечатаем сообщение при возникновении ивента EquipFinished:

Изображение выглядит как текст

Автоматически созданное описание

У нас выводится два сообщения (в начале игры), потому что на сцене два персонажа. Далее при нажатии на Tab под конец анимации у нас печатается одно сообщение.

6. Теперь создадим свой AnimNotify через C++ и запрограммируем логику в WeaponComponent. Создаем свой класс STUEquipFinishedAnimNotify, наследующийся от AnimNotify. Положим в папку Animations.

В сгенерированном коде сразу переходим в базовый класс UAnimNotify.

В классе UAnimNotify нету никакого C++ ивента, на который мы могли бы подписаться и узнавать, когда произошел данный Notify. Однако имеется виртуальная функция, которая вызывается, когда срабатывает Notify:

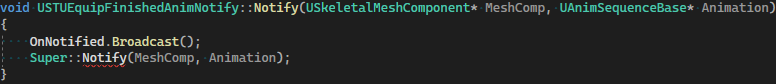
Изображение выглядит как текст

Автоматически созданное описание

Мы можем переопределить ее в нашем Notify и добавить в нее вызов нашего делегата.

Создаем делегат для нашего класса.

По сути мы тут будем использовать паттерн Декоратор – добавляем дополнительную логику в существующую функцию.



7. Переходим к нашему компоненту оружия. Наша задача – получить доступ к нашему Notify и подписаться на делегат.

Создадим две функции:

Изображение выглядит как текст

Автоматически созданное описание

В InitAnimations() мы найдем наш AnimNotify и подпишемся на него.

Вторая – callback, который мы должны забиндить на делегат OnNotify.

Не забудем подключить ЗФ нашего Notify в cpp-файле компонента оружия.

В функции OnEquipFinished пока просто залогируем, что мы закончили смену оружия.



**InitAnimations надо добавить в функцию BeginPlay, она будет вызываться один раз при спауне персонажа.**

Изображение выглядит как текст

Автоматически созданное описание

В теле функции следующая логика

У нас есть указатель на AnimMontage, доступ к AnimNotify можно получить с помощью специального поля Notifies (данное проперти есть почти у всех анимационных ассетов).

Если мы найдем его в файлах движка, мы попадем в UAnimSequenceBase (один из базовых классов анимации). Самое первое проперти и есть массив Notifies стуктур FAnimNotifyEvent.

Изображение выглядит как текст

Автоматически созданное описание

Если перейдем уже внутрь них, то увидим, что это стуктура, в которой хранится дополнительная информация об анимационном ивенте и указатель на объект AnimNotify, который нам непосредственно нужен.

Изображение выглядит как текст

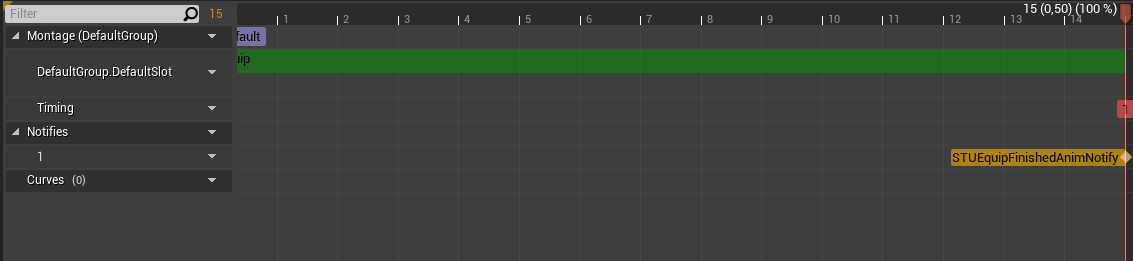
Автоматически созданное описание

Получив данный массив в нашу функцию, мы пройдемся по нему и найдем нужный нам. Проверяли мы это с помощью каста. Если преобразование произошло успешно, именно этот нам и нужен. Тогда мы на него биндим нашу функцию.

Изображение выглядит как текст

Автоматически созданное описание

8. Теперь нам надо добавить наш вид Notify на AnimMontage.



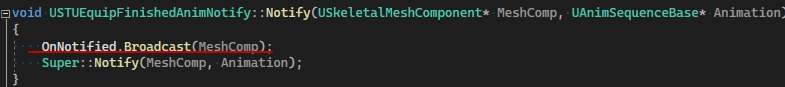
Однако, если мы сейчас запустим игру, у нас в лог выведется сразу 4 строчки. А если нажмем на Tab, то приходят две строчки.

Все дело в том, что наш EquipAnimMontage является указателем на анимационный ассет и оба нашего персонажа на сцене подписываются на ивент от него. То есть меняет оружие наш персонаж, начинает проигрываться анимация и когда она доходит до нашего Notify, оповещаются все персонажи, которые подписаны на данный AnimMontage.

Функция Notify, которую мы переопределили, имеет два параметра. Один из них – SkeletalMesh-компонент. Именно этим мы и воспользуемся. Мы можем немного изменить наш делегат, добавив ему указатель на этот SkeletalMesh-компонент.

По данному указателю мы можем фильтровать Notify. Если анимация проигрывается не для нашего скелета, то мы данные Notify обрабатывать не будем.





Не забываем изменить в WeaponComponent в ЗФ сигнатуру функции.

В самой функции проверяем, совпадает ли тот меш, который пришел в колбеке, с нашим мешем персонажа:

Изображение выглядит как текст

Автоматически созданное описание

9. Подытожим, какая тут логика – нам нужно как-то отследить момент окончания анимации. Для этого можно добавить Notify на анимационный таймлайн. Когда до него доходит анимация, она вызывает виртуальный метод Notify. Мы хотим, чтобы при вызове Notify происходила какая-то логика, которая находится в WeaponComponent. Именно поэтому мы переопределяем поведение метода Notify, добавляя ему делегат, к которому закрепляем функцию из WeaponComponent, а Broadcast мы вызываем в функции Notify.

10. Завершим логику смены оружия. Создадим дополнительную булевую переменную, которая будет отражать то, проигрывается сейчас анимация смены оружия или нет.

А так же создадим две дополнительные функции, которые будут говорить нам, можем мы стрелять или менять оружие соответственно.

Изображение выглядит как текст

Автоматически созданное описание

Наш новый флаг мы будем устанавливать в true перед тем, как начать проигрывать анимацию в EquipWeapon (то есть когда начинаем менять оружие), а в false – когда срабатывает наш делегат из Notify (то есть когда анимация заканчивается – вызов функции OnEquipFinished).

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Функция CanEquip будет возвращать отрицание данной переменной – то есть мы можем экипироваться только тогда, когда предыдущая анимация уже закончилась.

Изображение выглядит как текст

Автоматически созданное описание

Как мы помним, смена оружия у нас происходит по клавише Tab (или колесику), поэтому в функции, которая привязана к этому биндингу мы проверяем, можно ли сейчас сменить оружие:

Изображение выглядит как текст

Автоматически созданное описание

В функции CanFire так же будем проверять флаг bEquipAnimInProgress, но еще перенесем сюда проверку самого CurrentWeapon из функции StartFire.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

11. Сейчас у нас имеется проблема с анимацией смены оружия – когда мы меняем оружие, у нас при беге анимация бега немного ломается. Это происходит, потому что мы вставляем данную анимацию в последний слот, который перетирает всю информацию о текущей анимационной позе. По сути, так у нас происходит с анимацией смерти.

То есть у нас получается, что нода Slot не привязана к конкретному AnimMontage – если персонаж умирает или меняет оружие, нода Slot работает для обоих AnimMontage (если совпадает пункт в Slot и в AnimMontage).

12. Теперь будем проигрывать анимацию смены оружия в другом Slot. Заменим имя слота на DefaultGroup.UpperBody:

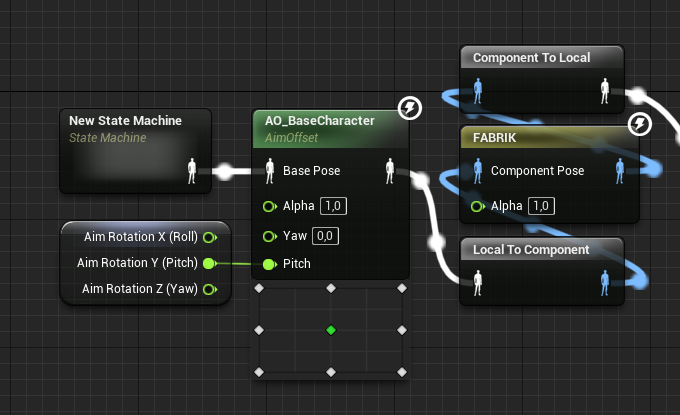
Изображение выглядит как текст

Автоматически созданное описание

Далее в AM\_Equip так же меняем имя слота. При такой настройке AM\_Equip будет проигрываться в созданном нами Slot.

13. В анимационном графе имеется возможность в любой момент сохранять текущую позу, так называемый **анимационный кеш**. К данной ноде можно относиться как к переменной, в которой хранится текущая поза.

И теперь данную ноду мы можем использовть в любом месте нашего анимационного графа. То есть разницы между этим:



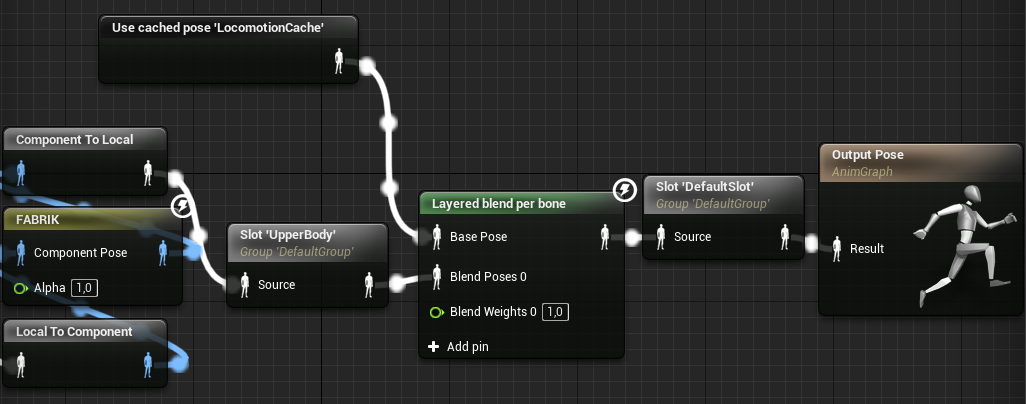
И этим **нет**:

Изображение выглядит как текст, внутренний, черный

Автоматически созданное описание

То есть данная нода не кеширует конкретную позу, она как бы ретранслирует ее в различные места как будто.

Когда проигрывается анимация смены оружия, мы можем сблендить эту позу с нашим кешем анимации с помощью ноды **Blend Per Bone**. Данная нода смешивает анимационный позы по заданной позе из скелета. То есть мы теперь настроили анимационный граф таким образом, что верхняя часть тела никак теперь не разрушит нашу анимацию бега, потому что теперь она всегда будет добавляться в финальную позу.



Идем в панель Details данной ноды и добавляем элемент массива в Branch Filters, где нам не необходимо указать **имя кости**, **по которой мы будем блендить** данные анимации. Выберем для этого b\_Spine – фактически, она разделяет верхнюю и нижнюю часть персонажа.

