**9. Homework: run animation**

1. Мое решение для анимации бега, как реализовал?

2. Какой в моей реализации баг?

3. Реализация преподавателя: мэппинг, функции, забиндили их

4. Что нам необходимо понять о кнопке? Что для этого ввели и как определили поведение в функциях биндинга

5. Определяем, двигается ли персонаж вперед, первый способ (общий), наш способ (так как у нас только анимация бега вперед), что ввели дополнительно, как и где мы это определяем

6. Добавляем функцию, которая будет возвращать булевое значение – бежит наш персонаж или нет, что возвращает

7. Переходим в ивент граф анимационного блюпринта, что тут делаем

8. В анимационном графе что делаем? Как определяем новое состояние, как определяем переходы (и какие)?

9. Определяем анимацию для состояния бега – как?

10. Как реализуем изменение скорости? Создаем доп. компонент, что дальше?

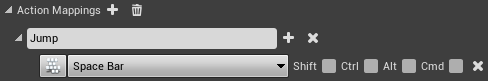
11. Что надо получить в функции и что вернуть из нее? Синтаксис

12. Создаем модификатор, как определить его диапазон значений

13. Последнее – необходимо поменять дефолтный класс компонента передвижения персонажа, как?

1. Мое решение:

Создал один Action mapping:



В коде – добавили компонент передвижения и две функции:

Изображение выглядит как текст

Автоматически созданное описание

Далее создал получил в переменную для компонента передвижения сам этот компонент:

Изображение выглядит как текст

Автоматически созданное описание

и забиндил функции на пробел при нажатии и отпускании:

Изображение выглядит как текст

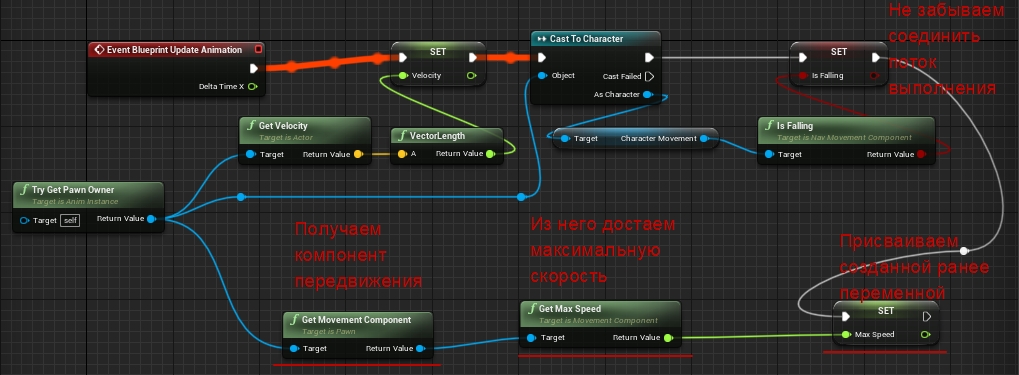
Автоматически созданное описание

Сами функции обращаются к полю компонента передвижения и записывают туда новое значение:

Изображение выглядит как текст

Автоматически созданное описание

В блюпринте анимации логика выглядит следующим образом:



На анимационном графе добавил анимацию бега и переходы из состояния Walk по изменению нашей переменной MaxSpeed, а переход из бега в прыжок такой же, как из ходьбы в прыжок:

Изображение выглядит как текст, электроника

Автоматически созданное описание

2. Однако в моей реализации нашлась одна проблема – я не отслеживаю то, двигается ли мой персонаж при нажатии клавиши Shift, поэтому у меня есть баг – персонаж бежит на месте когда мы нажимаем Shift без движения вперед.

3. Реализация преподавателя следующая:

Action Mapping задается аналогичным образом.

Далее мы создали две функции, которые мы биндили на клавишу:

Изображение выглядит как текст

Автоматически созданное описание

Забиндили их:

Изображение выглядит как текст

Автоматически созданное описание

4. Для начала нам необходимо понять, зажата ли наша клавиша, отвечающая за бег, для этого вводим булевую переменную bWantsToRun. Такое название потому, что зажатие Shift – это недостаточное условие для бега, так как наш персонаж должен двигаться при этом.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

5. Далее мы должны определить, что наш персонаж бежит вперед. Это можно сделать несколькими способами: в самом общем случае можно высчитать угол между вектором скорости и вектором того, куда смотрит наш персонаж, и по величине данного угла определять направление бега.

У нас всего одна анимация – анимация бега вперед, поэтому у нас решение проще.

Создаем еще один флаг, который будет отвечать за то, бежит наш персонаж или нет:

Изображение выглядит как текст

Автоматически созданное описание

Значение нашей переменной определяем в ранее определенной функции MoveForward следующим образом:

Изображение выглядит как текст

Автоматически созданное описание

6. Последнее – добавление функции, которую мы будем вызывать из нашего анимационного блюпринта. Она будет возвращать true, когда наш персонаж бежит и false в противном случае:

Изображение выглядит как текст, оранжевый, набор

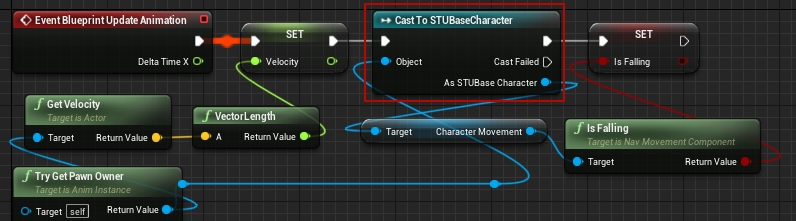
Автоматически созданное описание

Изображение выглядит как текст, оранжевый, закрыть

Автоматически созданное описание

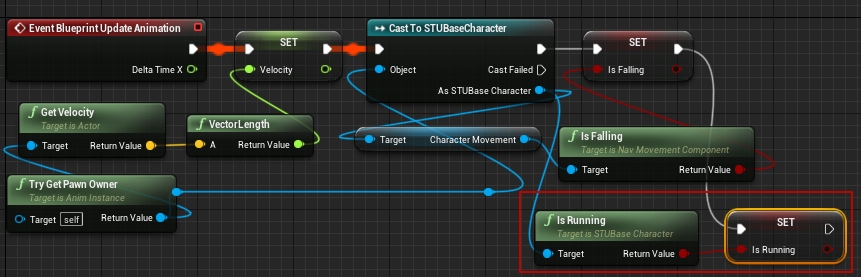
Последняя проверка на то, что вектор скорости персонажа не равен нулю нужен для того, чтобы, даже если у нас зажата клавиша Shift и движения вперед, но наш персонаж в ловушке или где-то застрял, у нас не срабатывала анимация бега.

7. В ивент графе анимационного блюпринта создаем переменную IsRunning. Начнем с того, что **поменяем наш каст к Character на каст к нашему конкретному STUBaseCharacter**:



С зажатым Ctrl можно перемещать связи от одной ноды к другой.

Теперь мы можем вызвать нашу функцию IsRunning и на ее выход подаем новую переменную:



8. В анимационном графе добавляем новое состояние (не переносим анимацию напрямую). Назовем его Run. И добавляем переход в него и из него при условии, что переменная IsRunning устанавливается в true и false соответственно.

Условие перехода в JumpStart из Run такое же, как для Walk. Мы можем продублировать наше условие или же сделать следующее: у перехода из Walk в JumpState в параметре Transition Rule Sharing в поле Promote to shared вводим IsFalling. По сути, мы из условия перехода сделали функцию, которую мы можем переиспользовать. При этом переход окрашивается в другой цвет – это индикатор «расшаренности» нашего перехода. Для перехода из Run в JumpStart в поле Transition Rule Sharing выбираем IsFalling.

Изображение выглядит как текст, электроника

Автоматически созданное описание

9. Нам осталось определить анимацию для состояния бега. Для этого в Content Browser переименуем наш старый Blend Space и создадим на его основе новый, просто заменив финальное состояния с Run\_Fwd на нашу анимацию бега. Конечная скорость у нас потом будет больше чем 600 (правая граница на оси), но значение скорости будет clamp-ится и анимация будет проигрываться самая правая. Далее в нашем состоянии добавляем наш свежий Blend Space и не забываем подать ему на вход нашу Velocity:

Изображение выглядит как текст, электроника

Автоматически созданное описание

10. Осталось реализовать изменение скорости. В этой реализации это делается с помощью кастомного CharacterMovement-компонента. Создаем этот класс в папке Public и назовем STUCharacterMovementComponent и создаем его в поддиректории Components.

Из-за последнего в Build-C# файл добавляем строчку:



Далее переходим в ЗФ нашего нового компонента и находим внутри него функцию GetMaxSpeed(). Переопределим ее поведение:

Изображение выглядит как текст, снимок экрана, экран

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

11. Теперь нам надо получить указатель на нашего персонажа и проверить, бежит он или нет – в зависимости от этого изменить MaxSpeed.

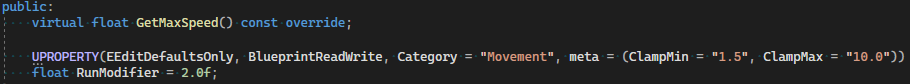
Для начала подключаем ЗФ нашего персонажа. Создаем указатель на него. Нам понадобится каст, так как компонент хранить указатель только на класс Pawn’a. Компонент при этом имеет специальную функцию для возвращения указателя на владельца данного компонента – GetPawnOwner().

Изображение выглядит как текст

Автоматически созданное описание

Если мы указатель не нулевой и персонаж бежит, то мы умножаем нашу скорость на специальный модификатор, в противном случае – возвращаем MaxSpeed.

12. Этот модификатор надо создать. Мы даем возможность менять его в блюпринтах, но если ему поставят значение 0 или отрицательное, логика будет нарушена. Для таких случаев у UPROPERTY есть механизм, позволяющий избавиться от таких ситуаций. В аргументе meta мы можем указать два спецификатора, ограничивающих минимальное и максимальное значение параметра:



13. Последнее – необходимо поменять дефолтный класс CharacterMovementComponent в классе персонажа на наш кастомный STUCharacterMovementComponent.

Для этого в cpp-файле персонажа подключаем ЗФ нашего компонента и для того, чтобы явно указать класс для CMC мы воспользуемся специальным видом конструктора с параметром, в который передается объект инициализации.

Тип объекта FObjectInitializer:

Изображение выглядит как текст

Автоматически созданное описание

Нам надо не забыть вызвать конструктор родительского класса, в который передаем тот же аргумент и сразу вызываем от него функцию SetDefaultSubobjectClass<>(), которая принимает в качестве параметра шаблона класс подобъекта, на который мы хотим заменить компонент, который передаем в качестве параметра самой функции – нам надо указать имя Movement-компонента, имя находится в статической переменной базового класса персонажа с именем CharacterMovementComponentName (аналогичные имена есть и для компонент меша, капсулы). При этом данная функция SetDefaultSubobjectClass<>() возвращает объект FObjectInitializer, поэтому все чики бамбони:

Изображение выглядит как текст

Автоматически созданное описание

В блюпринте видим изменение компонента движения на наш, а RunModifier не может выйти за границы установленного нами диапазона.