**8. Jump animation, state machine**

1. Биндим клавишу на прыжок (какой вид биндинга)

2. Биндим функцию к клавише, какую? Как реализована внутри?

3. Начинаем настраивать анимацию прыжка – для начала надо получить объект персонажа + как изменять нашу кривую в блюпринтах (добавлять доп. узлы)

4. Какую переменную состояния (встроенную) нам надо получить?

5. Далее нам надо создать из нее переменную – как?

6. Переходим на анимационный граф – какую специальную ноду создали и куда подключили

7. Перешли в нее – что содержит в началае, какое добавили состояние

8. Переходим в это состояние – что содержит внутри, какое поведение тут определяем?

9. Какие у нас есть анимации для прыжка? что происходит с ними, когда мы их переносим в машину состояний?

10. Создаем переходы – как, что у них появляется? Как определили переход из Walk в JumpStart

11. Далее надо перейти к JumpLoop – как это сделали?

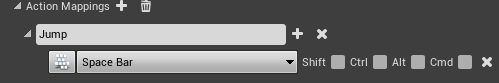
12. Переход из JumpLoop в JumpEnd – как?

13. Переход JumpLoop в Walk – как?

14. Все работает норм, только есть подергивания – почему и как исправили?

15. Какой есть параметр и где, отвечающий за высоту прыжка у персонажа?

1. Биндим клавишу на прыжок – выбираем Action Mapping на Space Bar, называем Jump.



2. Биндим его нашему персонажу в коде, при этом будем использовать встроенную функцию Jump класса Pawn, которая отвечает за физику прыжка:

Изображение выглядит как текст

Автоматически созданное описание

Так данная функция реализована внутри:

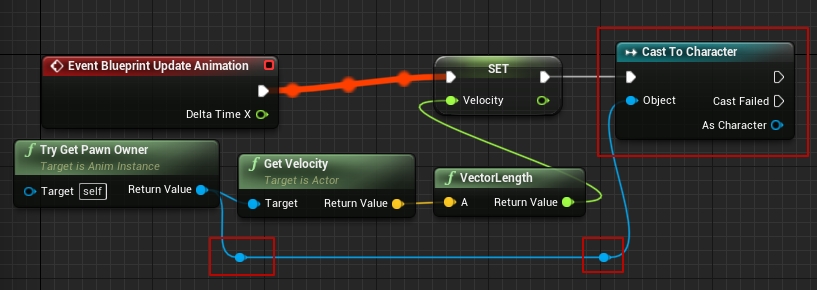
Изображение выглядит как текст

Автоматически созданное описание

В ней устанавливается флаг и в соответствии с ним на следующий тик будет добавлен импульс нашему персонажу по оси Z.

3. Переходим в анимационный блюпринт персонажа и настроим анимацию прыжка. Первым делом создадим булевую переменную, которая будет определять, находится наш персонаж в воздухе или нет. Скастим ссылку на Pawn’a, которого нам выдает функция Try Get Pawn Owner к Character’у. Мы скастили Pawn к Character, при чем нам не надо каститься дальше до BaseCharacter.

Нажимая два раза на ниточку в блюпринте можно выделить дополнительный булет для изменения ее изгиба.



4. Достаем из персонажа переменную Is Falling, которая находится в CharacterMovement-компоненте и равняется true, когда персонаж находится в воздухе и false в противном случае.

Изображение выглядит как текст, внутренний, снимок экрана

Автоматически созданное описание

5. Далее выбираем Promote to variable из выхода Is Falling – то есть создаем переменную, назовем ее также IsFalling и будем использовать ее на анимационном графе, то есть это очередная расшаренная переменная между двумя графами:

Изображение выглядит как текст, электроника, проектор

Автоматически созданное описание

6. Переходим на анимационный граф. Посмотрим, как можно добавить анимацию прыжка (пока у нас тут занято анимацией бега). Нам потребуется специальная нода StateMachine – она позволяет управлять несколькими анимациями и настраивать переходы между ними. Назовем ее Locomotion и подключаем ее к OutputPose.

Изображение выглядит как текст, внутренний

Автоматически созданное описание

7. Кликнув два раза по нашей StateMachine, в открывшемся графе видим лишь одну ноду – EntryPoint – начальная точка нашей StateMachine. Вытягиваем стрелочку и выбираем AddState. Называем его Walk.

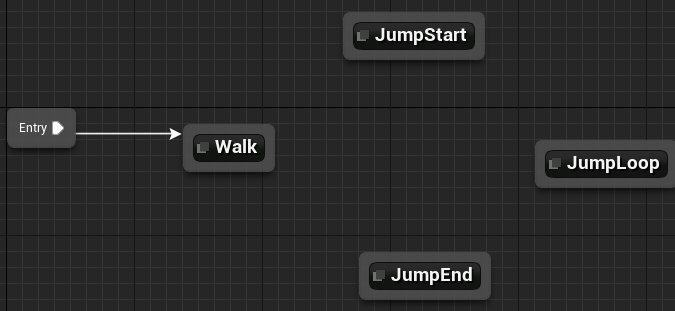


8. Кликаем два раза по ноде Walk и внутри ее графа замечаем, что выходное значение у нас также AnimationPose. Поэтому сюда вставляем логику, которое мы отключили от OutputPose.

Изображение выглядит как текст, внутренний

Автоматически созданное описание

9. У нас в AssetBrowser есть следующие анимации прыжка: JumpStart, JumpLoop, JumpEnd. Первая будет проигрываться один раз, вторая – пока в воздухе, третья – по приземлению. Скомбинируем данные анимации в нашей машине состояний. Добавим их в нашу State Machine (при выбрасывании они будут автоматически инкапсулированы в стейты):



10. Теперь создаем переходы: вытаскиваем стрелочку из Walk и указываем на JumpStart. При этом сверху стрелочку автоматически создалось правило перехода (Transition Rule):

Изображение выглядит как текст, часы, устройство, с плиткой

Автоматически созданное описание

Открываем его, кликнув два раза. В этом правиле вычисляется значение булевой переменной, при которой осуществляется переход (при значении true, разумеется).

Изображение выглядит как текст

Автоматически созданное описание

Сюда мы подключаем нашу переменную IsFalling:



11. Далее нам необходимо перейти к анимации JumpLoop. Для этого воспользуемся специальной функцией TimeRemaining (в скобочках у нас сразу подсказка, что данная функция будет вычисляться для JumpStart). Она возвращает количество секунд до конца анимации. Будем проверять, что данное время меньше 0.1:

Изображение выглядит как текст, электроника

Автоматически созданное описание

12. Переход из JumpLoop в JumpEnd – когда IsFalling равняется false:

Изображение выглядит как текст, внутренний, электроника, проектор

Автоматически созданное описание

13. Переход JumpLoop в Walk – аналогично JumpStart в JumpLoop:



14. Все работает, однако наблюдаются некоторые подергивания. Для исправления этого поведения переходим в StateMachine->JumpStart и убираем флажок Loop Animation в панели Details нашей анимации (чтобы анимация старта проигрывалась только один раз). Аналогично делаем и для JumpEnd.

15. Готово! Еще у нас в CharacterMovement-компоненте можно настроить высоту прыжка – параметр Jump Z Velocity.