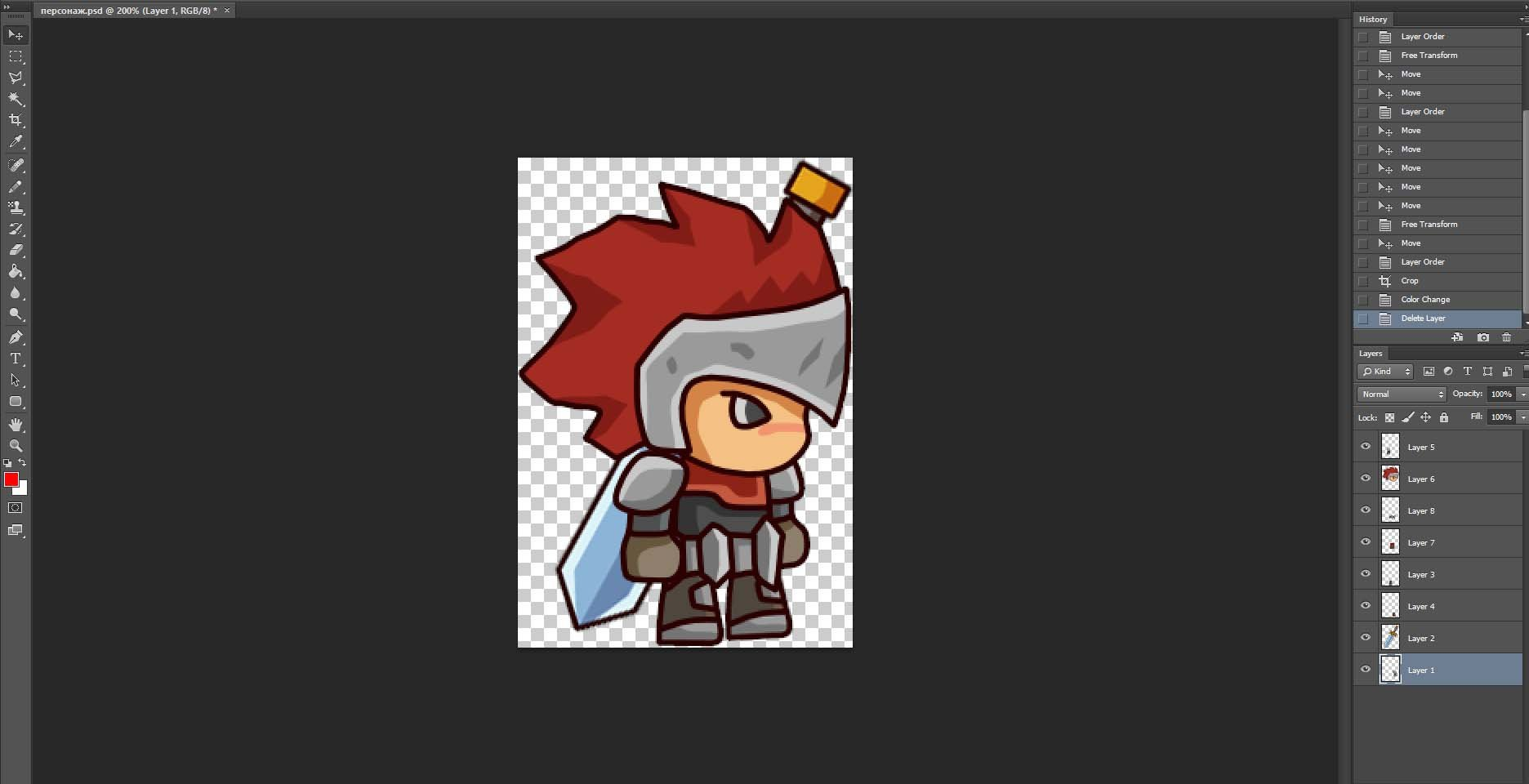
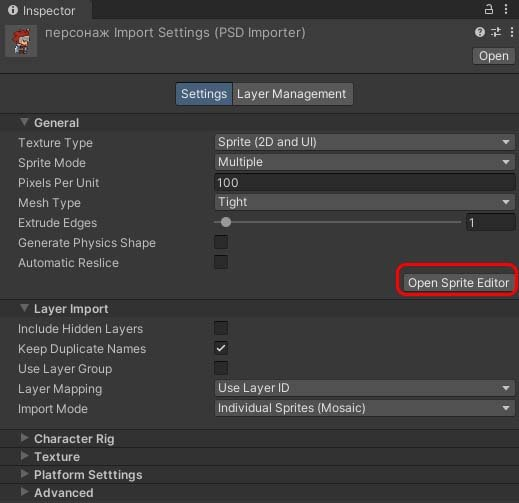
Перед началом урока включите видео:  
[Видео](https://drive.google.com/file/d/1K6Uf2CW_v34fZhZgudWOdTQSJ424eS-T/view?usp=sharing)  
  
  
На уроке мы рассмотрим одну из двух возможных видов анимаций. И речь пойдёт о костной анимации. Костная анимация в отличии от спрайтовой отличается тем, что в костной не нужно рисовать множество кадров, чтобы из них сделать анимацию. Здесь рисуется персонаж только один раз, с отдельными частями тела. А дальше с помощью программы его части тела анимируются.

Вся работа с костями происходит благодаря ассету **2D PSD Importer**. Если у Вас старая версия Unity, до 2020 года, то Вам нужно будет скачать этот пакет с помощью **Package Manager**, которая находится во вкладке **Window**. Если же Ваша версия Unity новая, то этот пакет скачивать не нужно, он встроен автоматически. Но на всякий случай всё же проверьте в **Package Manager**, установлен ли данный пакет или нет.

## **Подготовка персонажа**

Первым делом необходимо скачать персонажа, или нарисовать его самому, с отдельными конечностями, которые мы будем в дальнейшем вращать. Каждая конечность персонажа должна находится на отдельном слое. А фон должен быть прозрачным. Пример моего персонажа

  
Далее ОБЯЗАТЕЛЬНО сохраните этого персонажа целиком в **PSB** формате! И полученный **PSB** файл импортируем в окно **Project**. Далее кникнем по загруженному персонажу в окне **Project**, и в икне **Inspector** кликнем по **Open Sprite Editor**.  


Откроется окно **Sprite Editor** с вашими частями тела. Но это не совсем то, что нам нужно, поэтому жмём На вкладку **Sprite Editor**, и выбираем **Skinning Editor**.  
  
Далее в окне **Sprite Editor** поменяется вкладка на **Skinning Editor**, где мы и будем работать с костями. Всего у меня 8 конечностей. Но у вас их может быть как больше так и меньше.

* Голова
* Туловище
* Мечь
* Правая рука
* Левая рука
* Правая нога
* Левая нога
* Пояс

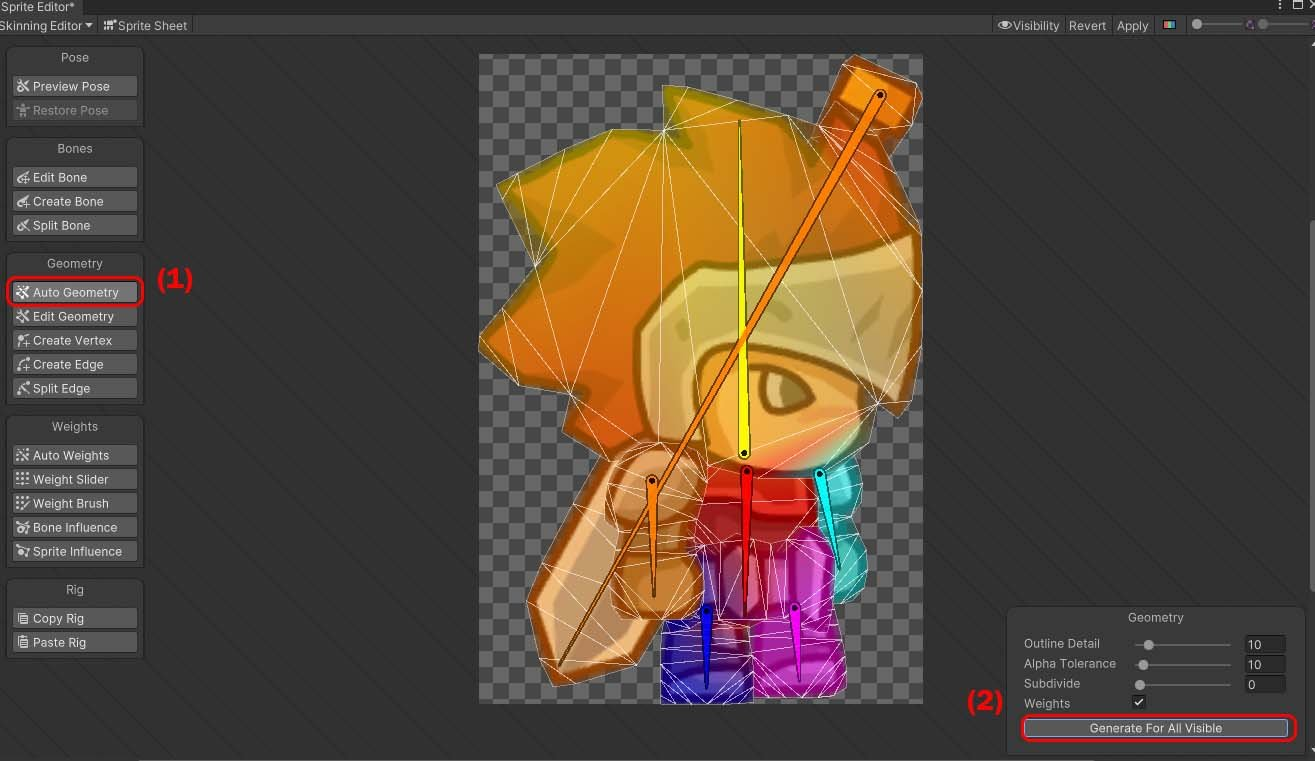
Исходя из количества конечностей, нам необходимо сделать 8 костей(хотя их может быть любое количество). Для создании кости кликнем на кнопку **Create Bone**, которая находится с левой стороны меню.

  
А потом на самом персонаже рисуем кость. Для отмены рисования кости, нужно нажать на правую кнопку мыши. После рисования 8-ми костей под каждую часть тела, у Вас должна получится примерно следующее:

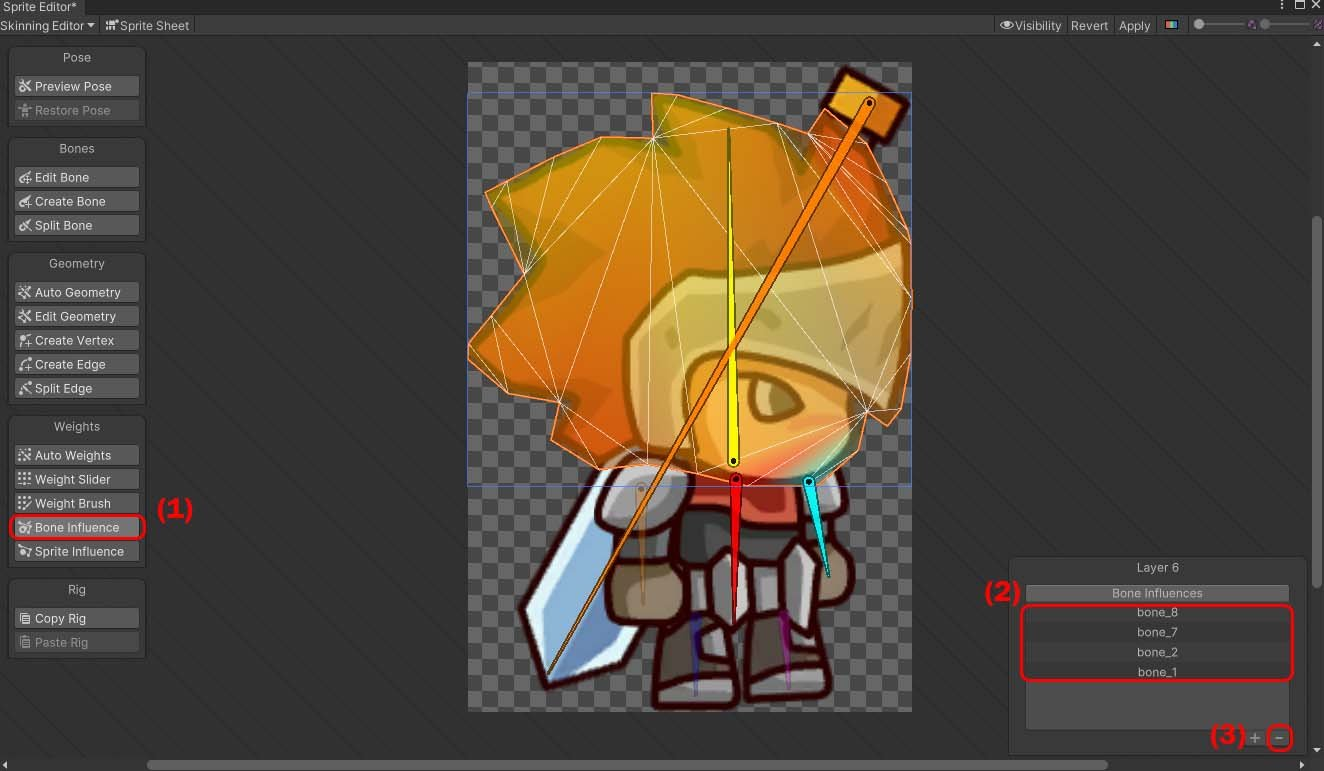


Обратите внимание, что красная кость соединена с костями рук и ног полупрозрачной еле заметной линией. Это нужно для того, чтобы при вражении основной красной кости, вращались и все остальные конечности. А вот голова и мечь к телу никак не прикреплён, эти кости сами по себе. По желанию их так же можно соединить, но в данном примере мы это делать не будем.

После того как кости созданы, у нас они не прикреплены к картинкам. То-есть картинки сами по себе, а кости сами по себе. Поэтому нам необходимо склеить наши кости с нашими картинками. Для этого доверимся автоматическому склеиванию, и в левом меню жмём кнопку **Auto Geometry**. После этого снизу жмём **Generate For All Visible**. И каждая частичка персонажа окрашивается в цвет кости.

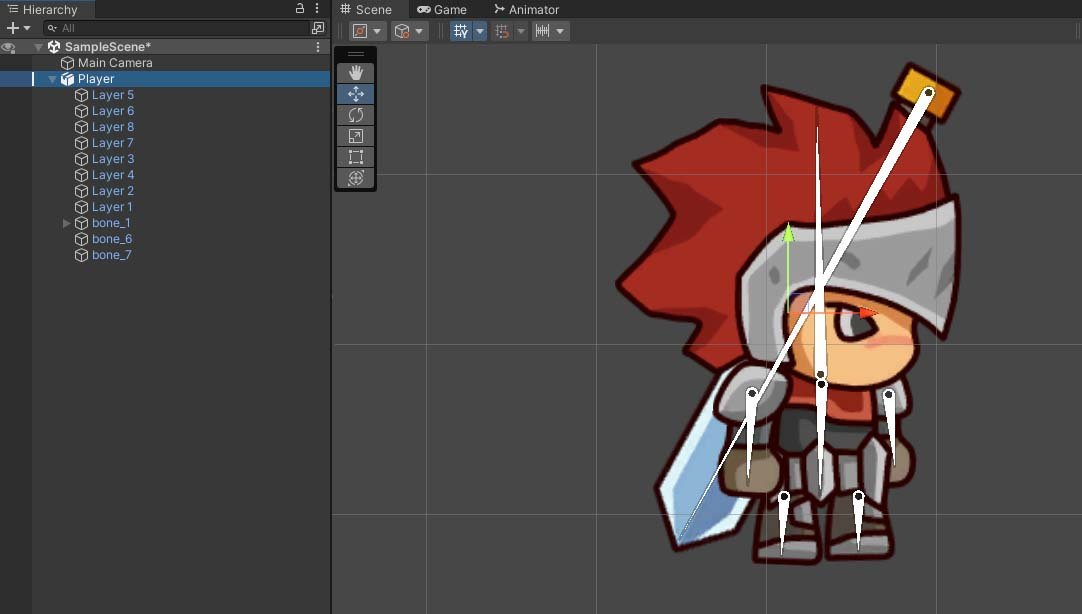
  
Поскольку некоторые кости задевают сразу несколько слоёв персонажа, то мы можем увидеть, что и цвет этих слоёв так же смешан разными цветами. Например, через голову проходит две кисти, жёлтая(для головы), и оранжевая(для меча). Из-за этого произошло смешивание - чего нам не нужно. Поэтому сейчас в ручном режиме это смешивание отделим.

Для этого кликнем 2 раза по одной из части тела, например голове, и нажмём в левом меню, кнопку **Bone Influence**. И снизу мы видим 4 кости, которые задели и влияют на поворот нашей головы. Необходимо оставить только кость для нашей головы, а остальные 3 кости удалить. После чего ваша голова окрасится в чисто жёлтый цвет, под цвет жёлтой кости.

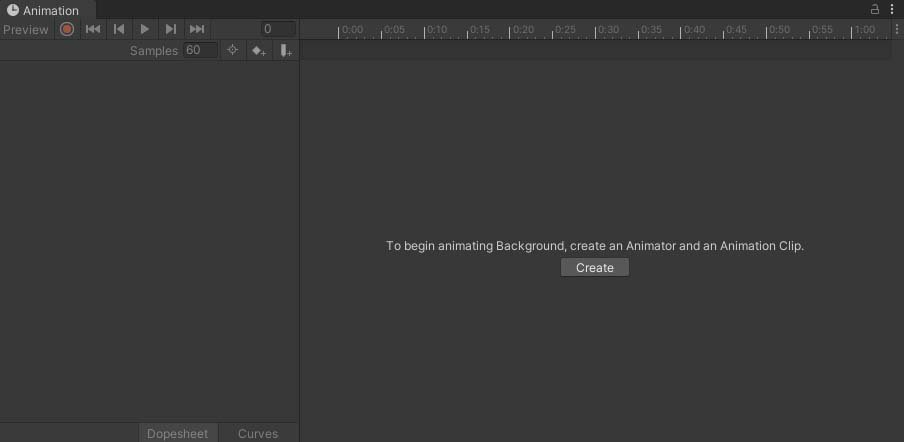
  
Такую же процедуру проводим и со всеми остальными частями тела, удаляя лишние кости. То-есть 2 раза кликаем на мечь, кликаем кнопку **Bone Influence**, потом удаляем лишные кости. Так же проводим и со всеми остальными 6 слоями персонажа, чтобы каждый слой имел чётковыраженный цвет, под свою кость, без примесей.

После всех этих манипуляций, жмём кнопочку **Apply** в верхнем-правом углу окна **Sprite Editor**, и закрываем данное окно.

В окне **Inspector** Вы увидите персонажа с добавленными дочерними элементами(костями). А в игровой сцене персонаж изображён вместе с костями, которые вы можете вращать.

  
  
**Создание Анимации (Animation)**

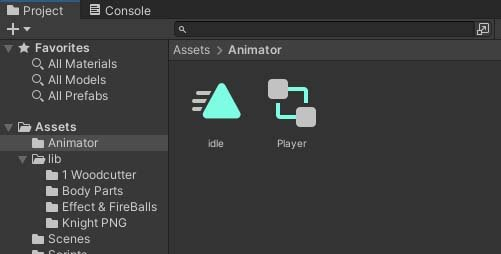
Теперь на основе данного персонажа, мы будем создавать различные интересные анимации. И начнём мы с анимации состояния покоя. Для этого откроем окно **Animation**, с помощью вкладки **Window - Animation - Animation**. Откроется данное окно.

  
В этом окне и будет создаваться вся анимация.

Для начала выбирите игровой объект, который необходимо анимировать. В нашем случае это персонаж с именем **Player**. После чего нажимаем кнопку **Create**, которая изображена на рисунке выше. Далее откроется окно, в котором необходимо указать название для нашей первой анимации. Первой анимацией будет анимация покоя, и назовём её **idle**.

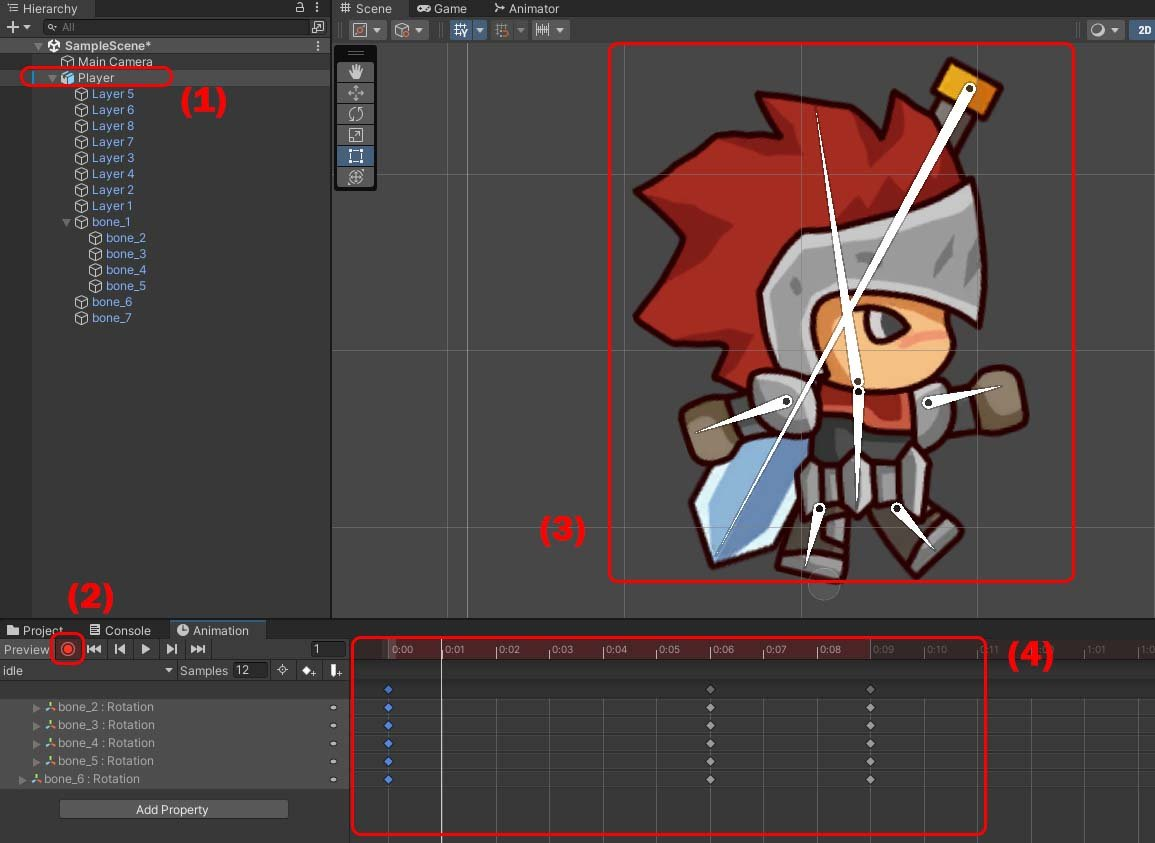
Отлично, анимация создана. Но пока она ещё пустая, и мы это сейчас исправим..

**Важно №1**. После создания ПЕРВОЙ анимации, юнити автоматически создаёт и контроллер для наших анимаций. Контроллер анимаций принято называть **Animator**. Имя этого контроллера имеет такое же, как и имя объекта, для которого Вы создаёте анимацию. В моём случае название контроллера будет **Player**. Контроллер анимаций хранит ВСЕ состояния анимаций данного объекта(например: покой, бег, прыжок и тд.), и благодаря ему происходит смена этих состояний анимаций с одной на другую. На рисунке у нас создана пока ещё пустая анимация покоя **idle**, и контроллер анимаций **Player**.

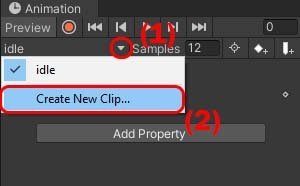


**Важно №2**. После создания первой анимации, юнити так же создаёт и компонент **Animator**, в **Inspector**-е данного объекта, который несёт в себе ссылку на наш контроллер. Благодаря этому компоненту чуть позже мы будем менять состояния анимаций с одной в другую.  
  
  
Вернёмся к нашему окну **Animation**. Для анимирования нашего костного персонажа делаем следующие действия (см. рис.):

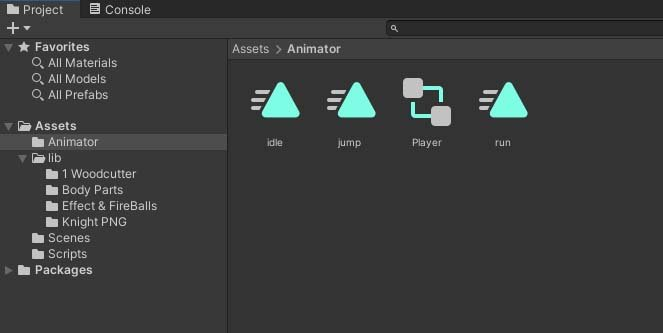
* Выбираем наш объект для анимации либо на игровой сцене, либо в окне **Hierarchy**.
* Нажимаем в окне **Animation** красную кнопку записи, чтобы все наши значения персонажа записывались в область #4
* Как только Вы начнете двигать кости персонажа на игровой сцене, в область #4 будут записываться ключи.
* В области #4 перетаскивайте шкалу времени в те места, где хотите изменять ключи для костей вашего персонажа, после чего снова поворачивайте кости персонажа как Вам удобно.
* Как только Вы полностью создадите состояние покоя idle, нажмите опять красную кнопку записи, чтобы выключить запись.

  
После нажатия на кнопку "**пробел**", ваш персонаж начнёт воспроизводить Вашу первую анимацию **idle**.

**Совет**: Чтобы персонаж плавно возвращался в исходное состоние, необходимо чтобы ключи последнего кадра были такими же как и первого кадра. Для этого скопируйте все ключи в первом кадре, и вставьте в самый конец анимации.

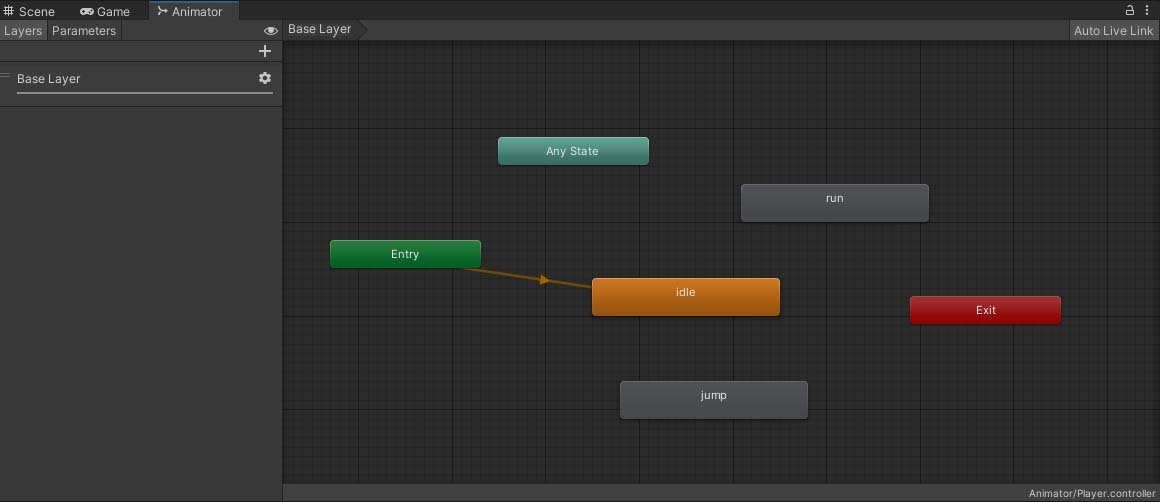
Поздравляем, анимация состояния покоя **idle** создана. Таким же образом создаём ещё 2 анимации: бег(**run**) и прыжок(**jump**). Для этого в окне **Animation** жмём треугольничек, напротив названия Вашей анимации, и в выпадающем списке жмём **Create New Clip**. После чего прописываем название Вашей анимации бега, например **run**, и сохраняем.  
  
Теперь у нас создалась вторая анимация, анимация бега, **run**. Но она пока ещё пустая. Потому, делаем всё тоже самое, что и в предыдущем примере, а именно, вращаем костями нашего персонажа, создавая ключи во временной шкале. Аналагичную работу проделываем и с новой анимацией прыжка, **jump**.

После всех проделанных манипуляций, у нас должно получится 3 анимации(**idle, run, jump**) и один контроллер(Animator) **Player**.

  
**Контроллер Анимаций (Animator)**

Большая часть работы уже позади. Теперь осталось настроить условия переходов между созданными анимациями. Для этого открываем наш контроллер, кликнув 2 раза по нашему аниматору **Player** в окне **Project**. Или же можете запустить **Animator** через вкладку **Window - Animation - Animator**.

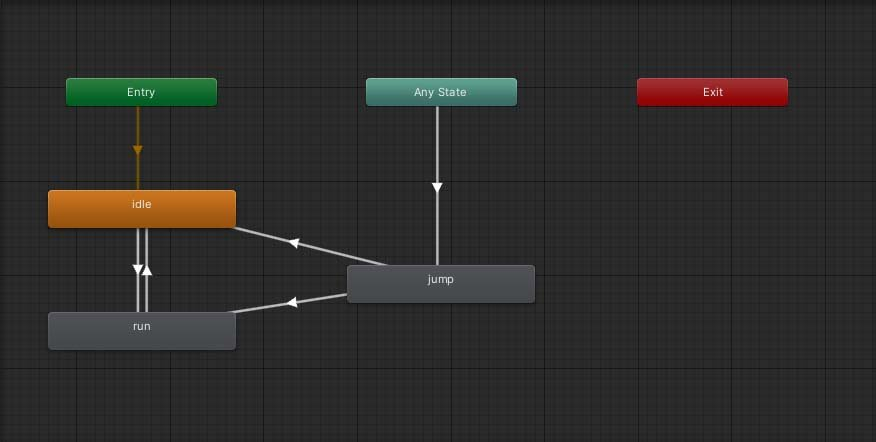
Окно контроллера(**Animator**) выглядит примерно так:



Обратим своё внимание на зелёный блок **Entry**. Он указывает на ту анимацию, которая будет запускаться по умолчанию. Обычно это анимация состояния покоя, **idle**. Вы можете эту стрелочку перенаправить на другую Вашу анимацию, которая будет стартовой.

Сейчас нам нужно стрелочками указать, между какими анимациями возможны переходы. Для этого правой кнопкой мыши жмём на анимацию **idle**. Далее в выпадающем списке жмём **Make Transition**, и указываем к какому блоку будет вести стрелочка. Укажем на блок **run**. Стрелочка перехода создана.

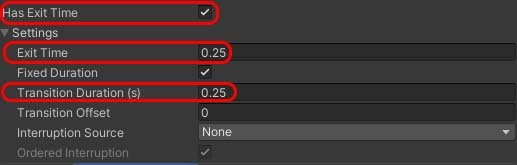
Создаём такие же стрелочки и между другими анимациями. У Вас должно получится так:



Обратите внимание на блок **Any Stay**, который соответствует *Любому Состоянию*. Тоесть она указывает, что к анимации **jump** можно перейти от ЛЮБОЙ анимации.

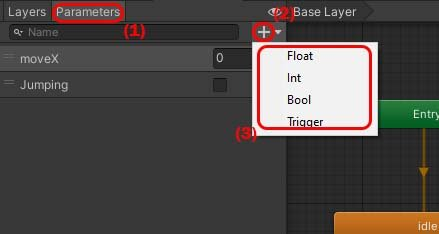
Теперь необходимо наши переходы настроить. Для этого жмём на любую БЕЛУЮ стрелочку, и в окне **Inspector** обратим внимание на следующие параметры:

* Если хотите, чтобы смена анимаций происходила мгновенно, то в поле **Has Exit Time** необходимо поставить галочку.
* В поле **Exit Time** указываете время перехода от первой анимации ко второй.
* В поле **Transition Duration** указываем плавность перехода из одной анимации в другую.

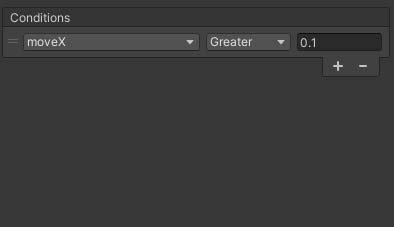
  
  
Для начала оставьте те значения, которые указаны на скриншоте. В дальнейшем можете с ними поэксперементировать, впрочем, как и с другими параметрами. Так же не забудьте указать эти настройки для остальных белых стрелочек. Эти настройки для каждой стрелочки могут отличатся друг от друга.

Готово, настройки переходов созданы, теперь создадим условия переходов, при которых будет осуществляться переход из одной анимации к другой. Для этого создадим переменную **moveX** с типом **Float**, в которой будет хранится информация о скорости движения персонажа по оси *Х*. А так же переменную **Jumping** с типом **Bool**, которая будет хранить информацию о том, находится ли персонаж в прыжке в данный момент.

Для этого в этом же окне выбираем **Parametrs**, жмём на плюсик, и выбираем тип переменной. Далее указываем название для созданных переменных.

  
Переменные созданы, теперь необходимо создать для всех наших белых стрелочек условия перехода. Например, чтобы анимация с **idle** перешла на **run**, необходимо, чтобы значение переменной **moveX** было отличное от нуля. Давайте это условие и создадим.

Для этого жмём на белую стрелочку идущая от **idle** к **run**, и в окне **Inspector** находим отдел **Conditions**, в котором указываем, что переход осуществляем, когда переменная **moveX** имеет значение больше чем **0.1** - то-есть находится в движении по оси X. Значение можете поставить любое, но не 0.

  
По такой же примеру создаём ещё 4 условия для остальных белых стрелочек перехода.

* **От run к idle**: переменная moveX имеет значение меньше чем 0.1
* **От Any State к jump**: переменная jumping имеет значение true.
* **От jump к idle**: переменная jumping имеет значение false.
* **От jump к run**: переменная jumping имеет значение false.

## **Запуск анимаций через скрипт**

Проделана огромная и самая сложная работа. Теперь дело осталось за малым. Сейчас нам необходимо через скрипт C# при определённых событиях заносить значения в переменные **moveX** и **Jumping**.

Для этого откройте свой *C# скрипт*, в котором прописана ваша функция передвижения персонажа, и для начала объявите переменную **anim**, в котором будет хранится ссылка на наш контроллер(Animator).

public Animator anim;

В методе **Start()** или **Awake()** присвойте ссылку к нашей переменной:

anim = GetComponent<Animator>();

Теперь самое главное, запись данных в переменную **moveX**. Для этого, используя метод **SetFloat()** установим нашей переменной **moveX** значения полученное от нашего передвижения **Mathf.Abs(Input.GetAxisRaw("Horizontal"))** по модулю:

anim.SetFloat("moveX", Mathf.Abs(Input.GetAxisRaw("Horizontal")));

Готово! Как только персонаж начнёт своё движение по оси Х, данные занесутся в переменную, и будет воспроизведена анимация бега **run**. А если персонаж остановится, то анимация бега перейдёт в состояние покоя **idle**.

Анимацию передвижения мы настроили, теперь перейдём к анимации прыжка **jump**.

В вашем скрипте должна быть переменная, которая отвечает за то, находится ли персонаж на земле или нет. У меня эта переменная называется **isGround**. Необходимо делать проверку на то, находится ли персонаж на земле или нет.

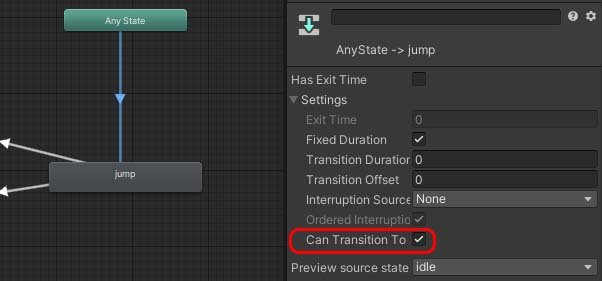
* Если персонаж находится на земле, то в переменную **Jumping** заносим *false*, и анимация прыжка не воспроизводится
* Если персонаж находится не на земле, то нужно в переменную **Jumping** записать *true* - которая и воспроизведёт анимацию прыжка:

if(isGround) { anim.SetBool("Jumping", false); } else { anim.SetBool("Jumping", true); }

Данный код нужно впихнуть в метод **Update()** или **FixedUpdate()**.

Поздравляем, работа с анимацией завершена полностью!

**Важно**: В некоторых случаях, в зависимости от анимации прыжка, нам необходимо отменить циклическое воспроизведение анимации прыжка. Поэтому нажмите на стрелочку, исходящую от **Any State** ведущая к **jump**, и в окне **Inspector** в поле **Can Transition To** поставьте галочку.



## **Домашнее задание**

мы разберём анимацию на примере трёх самых популярных состояний в 2д платформере:

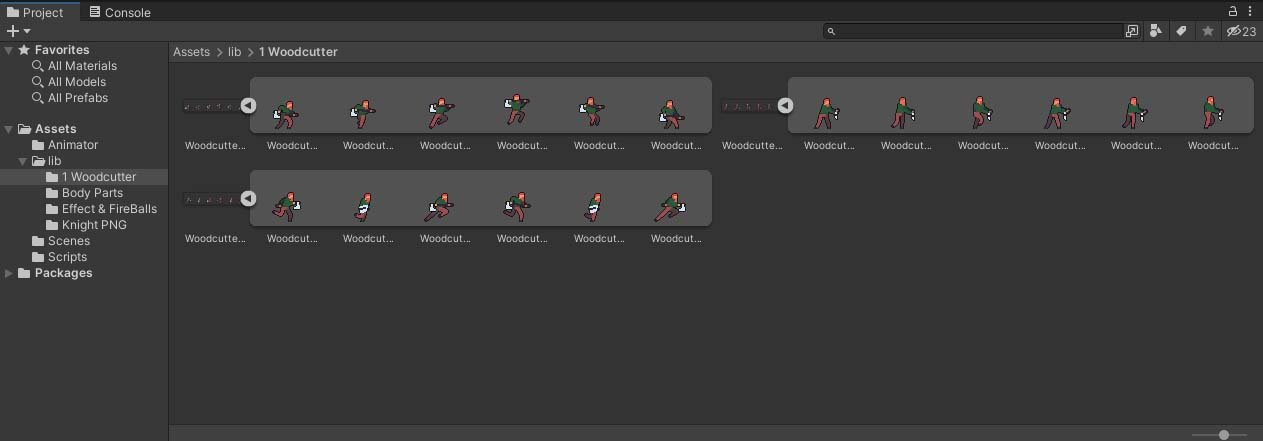
* Состояние покоя
* Бег, или ходьба
* Прыжок

## **Подготовка кадров**

Первым делом нам необходимо загрузить в окно **Project** спрайты для наших анимаций.

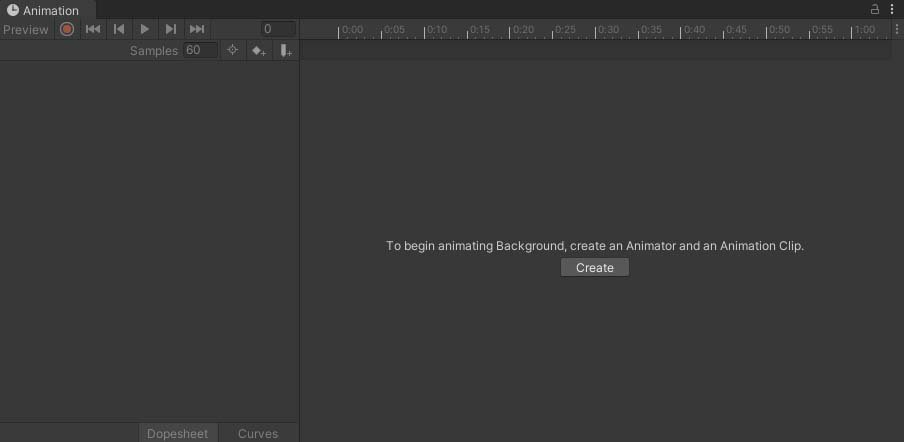


На изображении ниже, все необходимые кадры анимаций в три спрайта для каждого состояния(покоя, прыжок, бег). Вы можете все кадры использовать отдельными спрайтами, а можете сгруппировать как у меня. Это в принципе не важно.



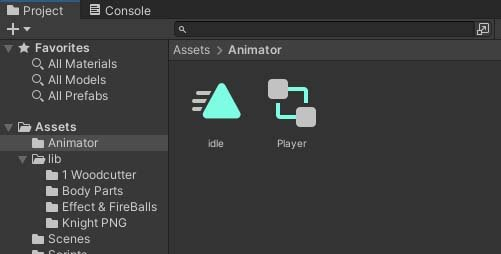
## **Создание Анимации (Animation)**

Теперь нам необходимо каждое из состояний анимировать. Для этого открываем окно **Animation**, с помощью вкладки **Window - Animation - Animation**. Откроется данное окно.



Благодаря этому окну, мы и будем превращать наши кадры в анимацию. Для этого выбираем игровой объект, который анимируем, в нашем случае это персонаж с именем **Player**. А далее нажимаем кнопку **Create**, которая изображена на рисунке выше. Далее откроется окно, в котором необходимо указать название для нашей первой анимации покоя. Назовём её **idle**. Готово, наша анимация создана, но пока она ещё пустая, и не имеет никаких кадров.

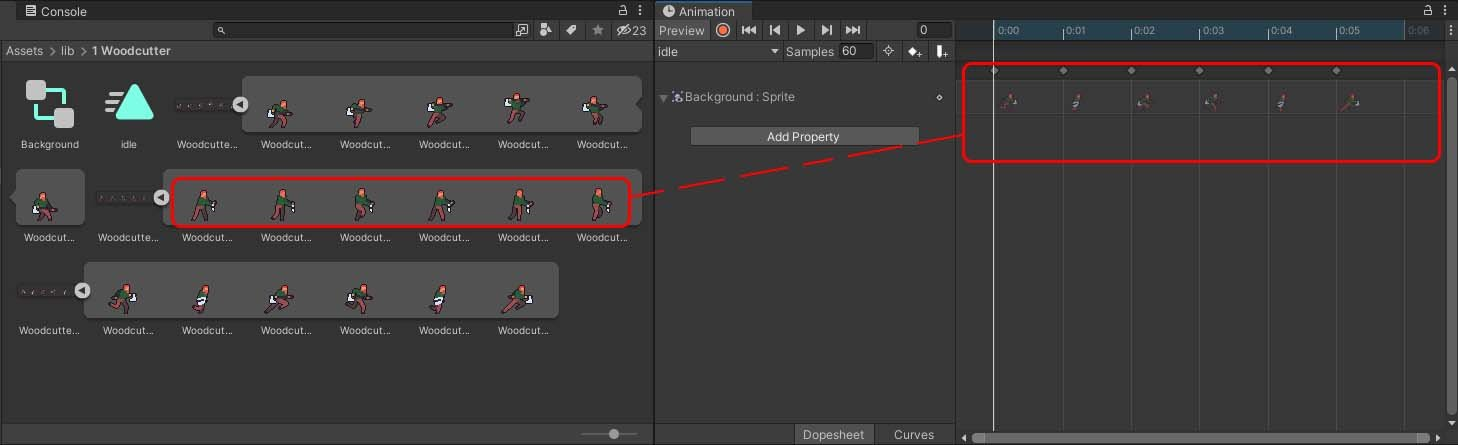
**Важно №1**. После создания ПЕРВОЙ анимации, автоматически создаётся и контроллер анимаций(Animator), с таким же названием, как и у объекта, которому мы и создаём анимацию. В моём случае название аниматора будет **Player**. Контроллер анимаций хранит ВСЕ состояния анимаций данного объекта(покой, бег, прыжок), и благодаря ему происходит смена этих состояний анимаций с одной на другую. На рисунке у нас создана пока ещё пустая анимация покоя **idle**, и контроллер анимаций **Player**.



**Важно №2**. Так же, после создания первой анимации, в **Inspector**-е данного объекта создаётся и компонент **Animator**, который несёт в себе ссылку на наш контроллер анимаций. Благодаря этому компоненту мы и будем в дальнейшем через *скрипт C#* менять состояния анимаций с одной в другую.

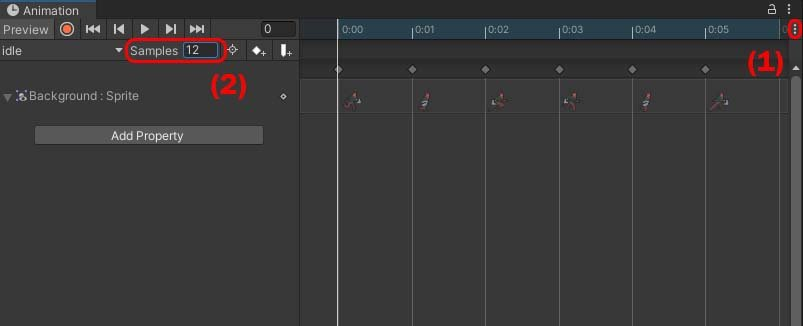


Перейдём к окну **Animation**. Перенесём наши спрайты из окна **Project**, в пустое пространство окна **Animation**. После нажатия клавиши "**пробел**", Вы можете обнаружить, что ваш персонаж на игровой сцене начинает анимироваться.

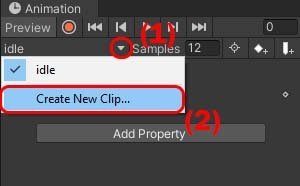


Скорее всего смена кадров у Вас происходит очень быстро, и чтобы исправить эту проблему, необходимо ЛИБО вручную увеличить расстояние между Вашими кадрами, ЛИБО снизить частоту кадров в секунду. Обычно используют второй вариант.

Для этого открываем меню в правом верхнем углу(три точки), и ставим галочку на **Show Sample Rate**. А дальше снижаем значение кадров в поле **Samples**, например, до 12.

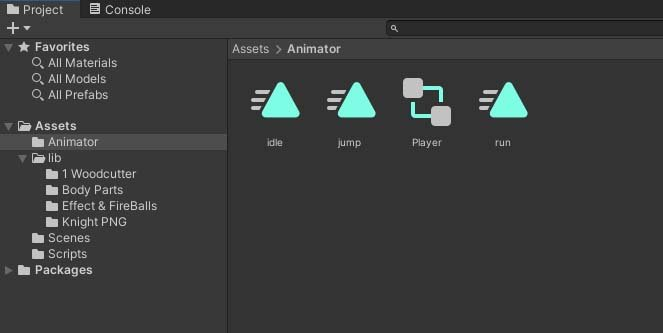


Отлично. Анимация состояния покоя **idle** готова! Теперь создадим вторую анимацию, *бег*. Для этого в окне **Animation** жмём треугольничек, напротив названия Вашей анимации, и в выпадающем списке жмём **Create New Clip**. После чего прописываем название Вашей анимации бега, например **run**, и сохраняем.



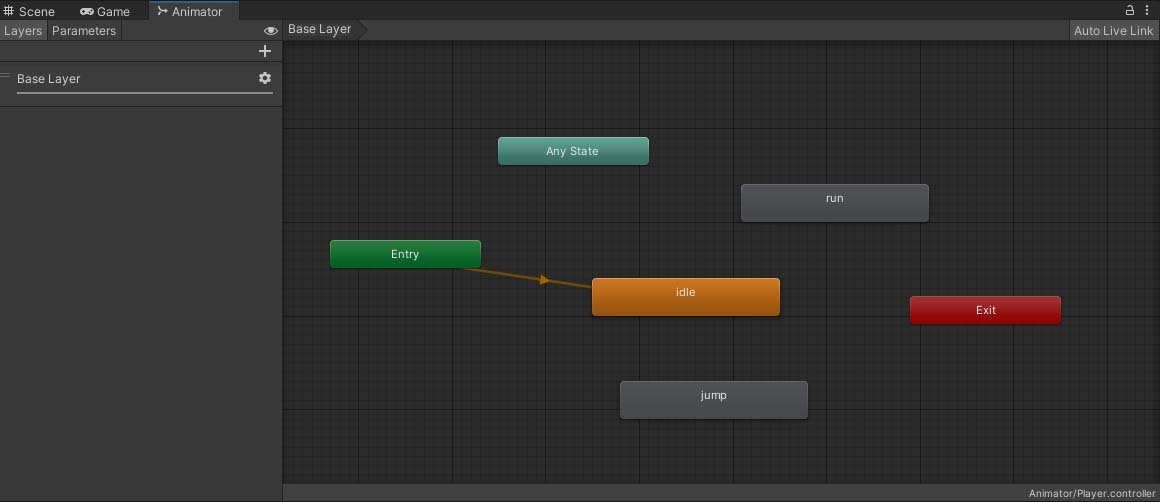
После чего была создана новая и пустая анимация бега. Теперь всё так же как и в предыдущем примере, перетаскиваем кадры бега в окно **Animation**, уменьшаем количество кадров, и всё. Таким же образом создаём анимацию прыжка, с названием, например, **jump**.

После всех проделанных манипуляций, у нас должно получится 3 анимации(**idle, run, jump**) и один контроллер(Animator) **Player**.

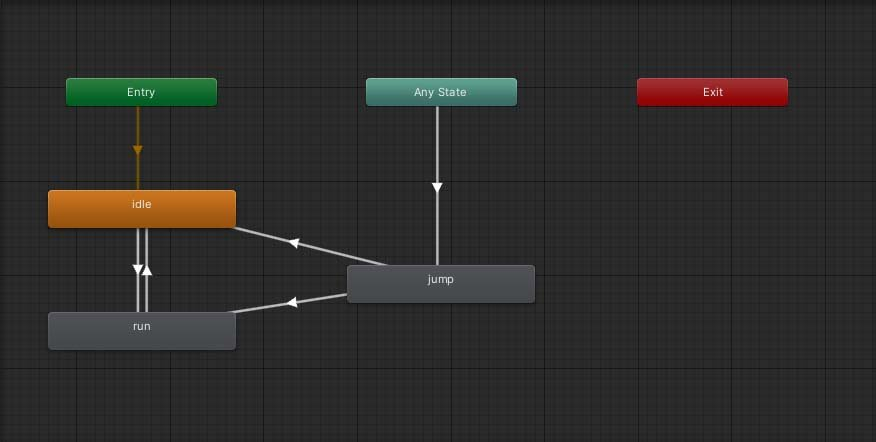


## **Контроллер Анимаций (Animator)**

Теперь поработаем с контроллером анимаций. Открываем наш контроллер либо через вкладку **Window - Animation - Animator**, либо кликнув 2 раза по нашему аниматору **Player** в окне **Project**. Откроется окно **Animator**, которое будет выглядеть примерно так.



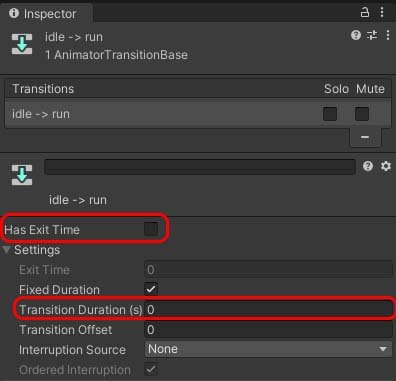
Для начала обратим своё внимание на зелёный блок **Entry**, это начальная точка аниматора. А оранжевая стрелочка, которая исходит от блока **Entry** указывает на стартовую анимацию, которая будет воспроизведена сразу после запуска сцены. С помощью таких стрелочек, нам нужно указать все возможные переходы от одних анимаций к другим.

Например, от состояния покоя **idle** анимация может быть переключена на анимацию бега **run**. И это нам нужно указать через стрелочки. Для этого правой кнопкой мыши нажмём на анимацию **idle**. Далее в выпадающем списке жмём **Make Transition**, и далее указываем к какому блоку будет вести стрелочка. Укажем на блок **run**. Ура, стрелочка прехода создана. Создаём такие же стрелочки и между другими анимациями. У Вас должно получится так:

Так же обратите внимание на блок **Any Stay**, что означает *Любое Состояние*. Например, чтобы не создавать много стрелочек от множества анимаций к анимации **jump**, можно сделать всего одну стрелочку от **Any State**. Это будет тоже самое.

Теперь настроим наши переходы. Для этого жмём на любую БЕЛУЮ стрелочку, и в окне **Inspector** проводим следующие настройки:

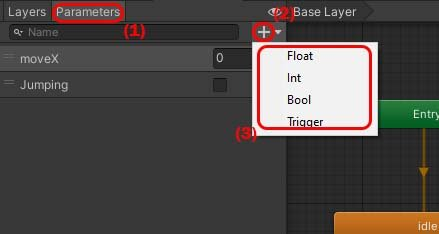
* В поле **Has Exit Time** убираем галочку, чтобы анимация прерывалась сразу же, как произойдёт смена анимаций на другую.
* В поле **Transition Duration** указываем значение 0, чтобы переход между анимациями был не плавный, а мгновенный.



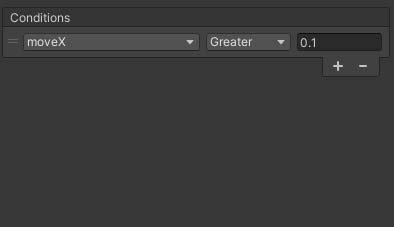
Такие же настройки нужно проделать со ВСЕМИ БЕЛЫМИ стрелочками.

Готово! Теперь необходимо создать условия, при которых будет осуществляться переход из одной анимации к другой. Для этого создадим две переменные. Одна переменная будет хранить информацию о том, бежит ли наш персонаж - если да, то будем воспроизводить анимацию бега. А вторая переменная будет хранить информацию о том, находится ли наш персонаж в прыжке - если да, то воспроизводим анимацию прыжка.

Создадим эти две переменные с названиями **moveX** и **Jumping** с типами **Float** и **Bool** соответственно. Для этого в этом же окне выбираем **Parametrs**, жмём на плюсик, и выбираем тип. Далее указываем название этих переменных.



Переменные созданы, теперь необходимо создать 5 условий перехода, для каждой белой стрелочки. Приведу пример, как создать условие перехода с **idle** на **run**. Для этого жмём на белую стрелочку перехода между этими анимациями, и в окне **Inspector** находим отдел **Conditions**, в котором указываем, что переход осуществляется тогда, когда переменная **moveX** имеет значение больше чем **0.1** - то-есть находится в движении по оси X:



По такой же примеру необходимо создать ещё 4 условия для остальных белых стрелочек перехода.

* **От run к idle**: переменная moveX имеет значение меньше чем 0.1
* **От Any State к jump**: переменная jumping имеет значение true.
* **От jump к idle**: переменная jumping имеет значение false.
* **От jump к run**: переменная jumping имеет значение false.

## **Запуск анимаций через скрипт**

Мы почти завершили. Отсалось совсем немного. Теперь нам необходимо в эти переменные занести данные из нашего скрипта C#, во время передвижения персонажа и прыжка.

Для этого откройте свой C# скрипт, в котором прописана ваша функция передвижения персонажа, и для начала объявите переменную **anim**, в котором будет хранится ссылка на наш контроллер.

public Animator anim;

Далее в методе **Start()** или **Awake()** присвоим ссылку к нашей переменной:

anim = GetComponent<Animator>();

Отлично, теперь с помощью метода **SetFloat()** установим нашей переменной **moveX** значения полученное от нашего передвижения **Mathf.Abs(Input.GetAxisRaw("Horizontal"))** по модулю:

anim.SetFloat("moveX", Mathf.Abs(Input.GetAxisRaw("Horizontal")));

Теперь как только персонаж начнёт своё движение, будет воспроизведена анимация бега. А если персонаж остановится, то анимация бега перейдёт в состояние покоя.

Теперь работа с прыжком:

В вашем скрипте должна быть переменная, которая отвечает за то, находится ли персонаж на земле или нет. У меня эта переменная называется **isGround**. Необходимо проверять, если персонаж на земле, то в переменную **Jumping** заносим *false*, и анимация прыжка не воспроизводится. А если персонаж находится не на земле, то нужно в переменную **Jumping** записать *true* - которая и воспроизведёт анимацию прыжка:

if(isGround) { anim.SetBool("Jumping", false); } else { anim.SetBool("Jumping", true); }

На это всё! Поздравляем, анимация готова.

**Важно**: В некоторых случаях, в зависимости от анимации прыжка, нам необходимо отменить циклическое воспроизведение анимации прыжка. Поэтому нажмите на стрелочку, исходящую от **Any State** ведущая к **jump**, и в окне **Inspector** в поле **Can Transition To** поставьте галочку.

