**Подготовка рабочего пространства**

Итак, начнем. Создадим новый проект в Unity. Выберем папку для расположения проекта, импортировать дополнительные пакеты не будем. Обязательно укажем, что мы создаем проект, настроенный на 2D игру (*Setup defaults for: 2D*). Проект создан. В окне *Project* у нас должна быть одна папка — *Assets*. Давайте создадим в ней подпапку *Sprites*, где будем хранить спрайты — графические файлы, необходимые для отображения персонажей, фона, пола, бонусов и прочих игровых объектов. Нам нужен спрайт, для отображения платформы, по которой будет бегать наш персонаж. Для этого подойдет любое прямоугольное изображение. Это немного спрайтов из игры Capitan Claw. Файл спрайта платформы называется *Platform.png*. Скопируем его в папку *Sprites*. Теперь нам надо перетащить спрайт*Platform* на окно *Scene*. В нашем проекте есть камера с именем *Main Camera*. Она-то и будет отображать то, что мы видим в игре. Перетащим спрайт платформы так, чтобы она оказалась в нижнем углу поля зрения камеры (если кликнуть по камере, то внизу сцены появится окошко *Camera Preview*, по которому можно контролировать, что в данный момент видит камера). Unity автоматически создаст игровой объект с двумя компонентами — *Transform* и *Sprite Render*. Первый отвечает за положение нашей платформы, второй — за ее отрисовку. Но нам еще нужно, чтобы персонаж не падал сквозь эту платформу, поэтому добавим к объекту платформы компонент *Box Collider 2D*, из раздела *Physics 2D*.

**Подготовка персонажа / Объекта**  
Теперь займемся персонажем. Создадим пустой игровой объект (*Game Object* — *Create Empty*) и перетащим его так, чтобы он висел над левой частью платформы. Переименуем этот объект как *Character* и добавим к нему компонент *Rigidbody 2D*, для придания нашему персонажу физических свойств твердого тела. В компоненте *Rigidbody 2D* установим флажок *Fixed Angle*, чтобы предотвратить случайные вращения нашего персонажа, например, от столкновения с другими твердыми телами. Затем установим в поле *Interpolate* значение *Interpolate*. Документация Unity рекомендует устанавливать это значение для персонажей, управляемых игроком, особенно, если за ним следует камера. Это связано с синхронизацией расчета физики и отрисовкой графики. Подробности в документации.

**Нарезка Sprite Sheet персонажа**  
Следующим шагом нам нужно добавить компонент *Sprite Render*, для отрисовки персонажа. Почему мы не можем просто перенести нужный спрайт, и получить автоматически сгенерированный *Sprite Render*, как в случае с платформами? Потому что наш персонаж, в отличии от платформ, будет отрисовываться не одним, а несколькими спрайтами, чтобы получился анимированный персонаж. Для этого нам придется выполнить ряд действий и первое из них — достать подходящие спрайтшиты (*Sprite Sheet*). Спрайтшит — это изображение, на котором содержаться кадры анимации для нашего персонажа. Думаю, ни для кого не секрет, что анимация — последовательное и быстрое отображение неанимированных кадров, каждый из которых немного отличается от предыдущего. Погуглите по запросу *Sprite Sheet*, и вы сразу поймете, что это такое. Нам нужны спрайтшиты для состояний покоя, бега и прыжка. В архиве со спрайтами есть файлы *Idle.png*, *Run.png*, и *Jump.png*. Скопируем их в папку *Sprites*. На данном этапе должно быть следующее: Приступим к анимированию персонажа, а конкретно — к анимированию состояния покоя, когда персонаж просто стоит и ничего не делает. Точнее говоря, он ничего не делает с точки зрения игровой логики, но он может переминаться с ноги на ногу, моргать, делать жесты, показывающие, что ему скучно так просто стоять и так далее. Для анимации покоя нам понадобиться файл *Idle* из нашей папки*Sprites*. Выделим этот файл. В окне *Inspector* отображаются свойства этого файла. Свойство *Texture Type* задано как *Sprite*, и это то, что нам нужно, а вот значение свойства *Sprite Mode* надо изменить с *Single* на *Multiple*. Таким образом, мы указали, что файл играет роль не одиночного спрайта, а представляет собой коллекцию спрайтов. Однако, эту коллекцию еще надо инициализировать. Для этого чуть кликнем по кнопке *Sprite Editor*, которая находится все в том же окне *Inspector* чуть ниже свойства *Pixels To Units*. Откроется новое окно. В нем мы видим содержимое нашего спрайтшита для состояния покоя: несколько похожих друг на друга кадров. Нам нужно их нарезать на отдельные изображения. Для этого нажмем на кнопку *Slice* в левом верхнем углу окна. Во-первых, нам надо задать способ (*Type*) нарезки изображения: *Grid* или *Automatic*. Первый способ нарежет наше изображение сеточкой с настраиваемыми размерами ячеек (*Pixel Size* — X… Y.). То есть, в этом режиме надо подобрать такие значения, чтобы все кадры нормально уместились в ячейках, чтобы ничего лишнего не было отрезано и т.п. Во втором режиме нарезка на кадры будет произведена автоматически. Сама нарезка произойдет после нажатия кнопки *Slice*. Попробуйте применить разные способы нарезки и посмотрите, что из этого получается. В случае с моим спрайтшитом нормально подходит способ *Automatic*. Даже если какой-то из кадров вышел немного неудачно — его можно отредактировать, кликнув по нему и изменив значения высоты/ширины/расположения и других параметров в соответствующем окне или при помощи мышки. Подтвердим нарезку нажатием на кнопку *Apply* в правом верхнем углу и закроем это окно.

**Отрисовка персонажа**  
теперь нам надо найти наш импортированный файл *Idle* в окне *Project*. В правой части файла есть треугольник (или в левой, при самом мелком мастабе значков). Кликнув по треугольнику, мы развернем коллекцию изображений, полученных в результате нарезки. Они будут иметь имена *Idle\_0*, *Idle\_1* и т. д. Теперь в окне *Hierarchy* выберем наш *Character*, и перетащим изображение *Idle\_0* в поле*Sprite* компонента *Sprite Rende*r. Наш персонаж отобразиться на сцене. Если он получился маленьким — можно увеличить его размеры до необходимых. Вот так: Давайте сразу добавим к нашему персонажу компонент *Box Collider 2D*, что не проваливаться сквозь платформу. При этом откорректируем размеры и местоположение коллайдера так, чтобы он не был слишком большим и располагался на уровне ног персонажа. Этого достаточно, чтобы персонаж не падал сквозь платформу. Можете запустить игру и проверить.

**Анимация персонажа в состоянии покоя**  
вернемся к анимации покоя. Для этого добавим к нашему *Character* еще один компонент — *Animator* (раздел *Miscellaneous*). Изменим некоторые его свойства — снимем флаг с *Apply Root Motion* и установим флаг *Anymate Physics*. *Apply Root Motion* позволяет изменять положение объекта из самой анимации (что нам сейчас не нужно), а включенный флаг *Anymate Physics* задает выполнение анимации в цикле расчета физики, что как раз рекомендовано для движущихся твердых тел. Теперь создадим в папке *Assets* файл *Animator Controller*. Назовем его *CharacterController*. В окне *Hierarchy* выделим нашего персонажа *Character* и перетащим *CharacterController* в поле *Controller* компонента *Animator*: Кликнем дважды по *CharacterController* — откроется новое окно *Animator*. В нем мы будем создавать различные состояния анимации (покой, бег, прыжок) и задавать условия перехода между ними. Для создания непосредственно анимаций нам нужно окно *Animation*. Если оно у вас еще не отображается его можно включить из главного меню Unity (*Window* — *Animation*). Теперь выберем нашего персонажа *Character* в окне *Hierarchy*, а в окне *Animation* нажмем кнопку для создания новой анимации и выберем *Create New Clip*. На скриншоте ниже я отметил эту кнопку красной окружностью: в стандартном диалоге сохранения файла сперва создадим папку *Animations*, а в нее сохраним наш файл анимации, назвав его I*dle*.   
  
После сохранения, в окне *Animator* появится наша анимация *Idle* в виде прямоугольника оранжевого цвета. Оранжевый цвет означает, что это будет анимация по умолчанию — как раз то, что нам сейчас нужно.  
  
Осталось всего пара шагов. Переходим в папку *Sprites*, разворачиваем спрайт *Idle*, выделяем первое изображение *Idle\_0*, зажимаем шифт и выделяем последнее изображение *Idle\_7*. Все выделенные изображения переносим мышью на окно *Animation*. Зададим значение *Sample* равное 10 — этот параметр означает количество кадров анимации в секунду. Как известно, для хорошей анимации необходимо, чтобы она отображалась со скоростью не менее 24 кадров в секунду, однако, в нашем случае анимация состоит из довольно маленького числа кадров и при значении 24 она будет отображаться слишком быстро.  
  
Запустим игру! Если все сделано правильно, наш Персонаж должен стоять на платформе, и анимироваться.  
  
  
На этом пока все. В следующий раз поговорим о реализации бегущего вправо-влево персонажа и соответствующей анимации.