Импорт Live2D-моделей

В модах можно использовать кастомные Live2D-модели.

В этой статье будет рассмотрен процесс подготовки и импорта Live2D-модели в игру — о том, как непосредственно создавать Live2D-модели и работать с Live2D SDK, вы можете почитать в соответствующей официальной документации.

Требования

- Редактор **Unity 2022.3.6**;
- **Live2D Cubism SDK** для **Unity** (в проекте используется версия **4-r.7**);
- Исходник Live2D-модели — файлы **.moc3, model3.json,** текстуры и опционально **.motion3**-файлы анимации;
- Умение работать с редактором и аниматором **Unity**;

> Вы можете скачать **готовый проект с необходимыми плагинами**.

Введение

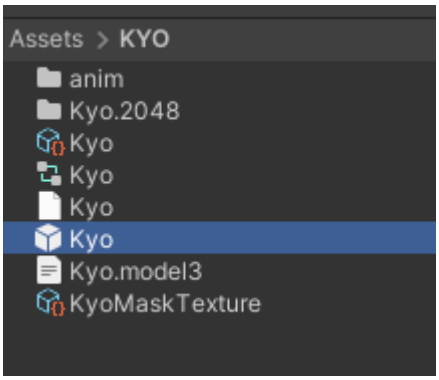
Для наглядности в данной статье будет описан процесс импорта модели в игру на примере спрайта второстепенного персонажа из старой демо-версии игры.

Вы можете скачать архив с исходниками по **этой ссылке**.
Ссылка на архив с финальным результатом расположена в конце статьи.

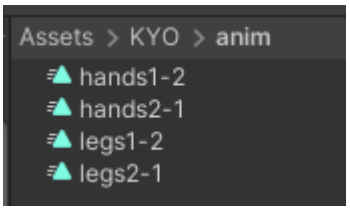
Импорт модели в Unity

Чтобы импортировать модель в Unity-проект, достаточно поместить её исходники внутрь папки Assets (или в любую её подпапку) — Live2D SDK импортирует модель и сгенерирует для неё префаб, а также контроллер анимации.

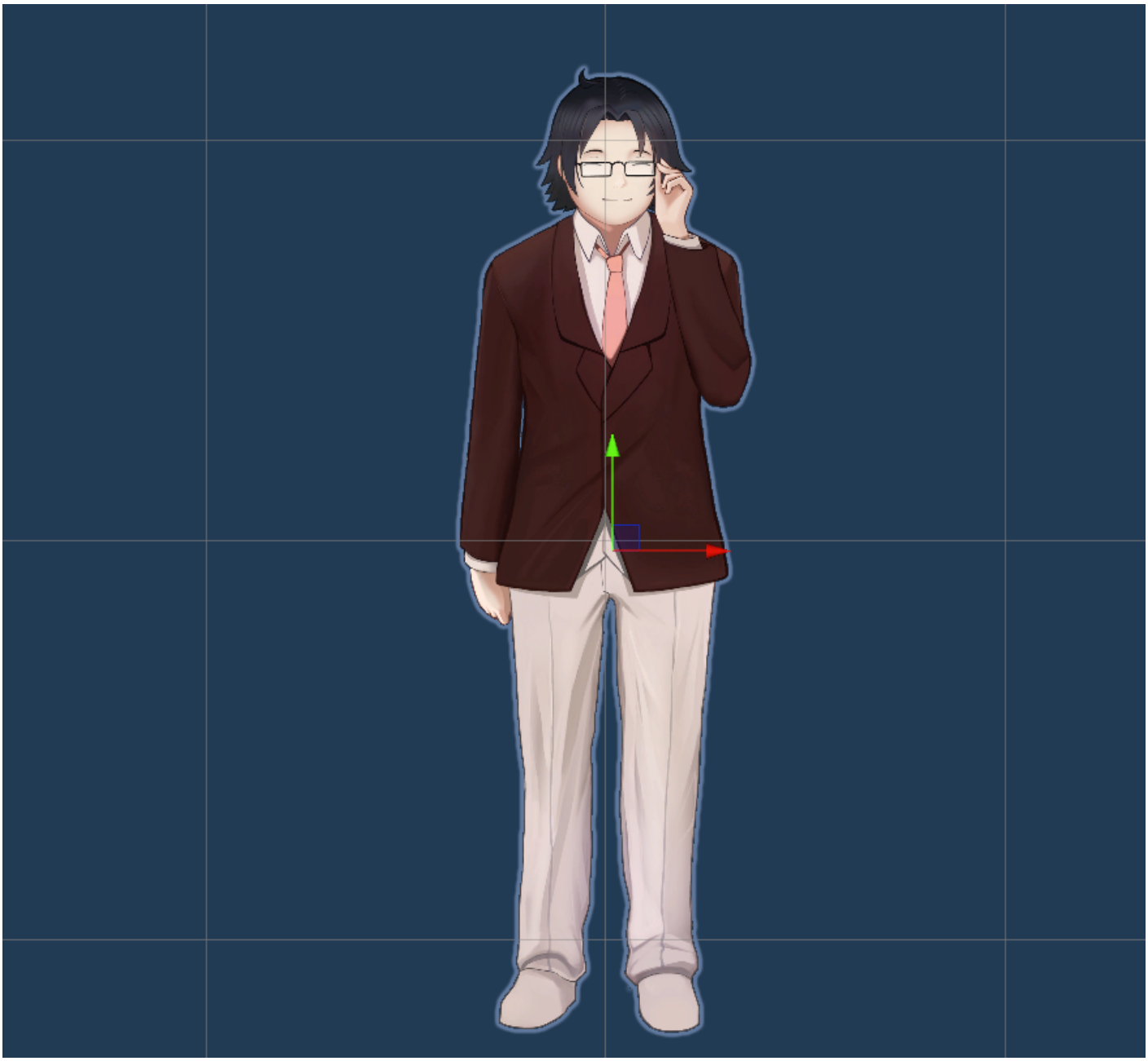
Распакуйте содержимое архива с исходниками спрайта в папку **Assets** вашего проекта. После того, как Live2D SDK закончит процесс импорта (он запустится автоматически при открытии окна редактора) — внутри папки с исходниками появится префаб, а также несколько вспомогательных ассетов:



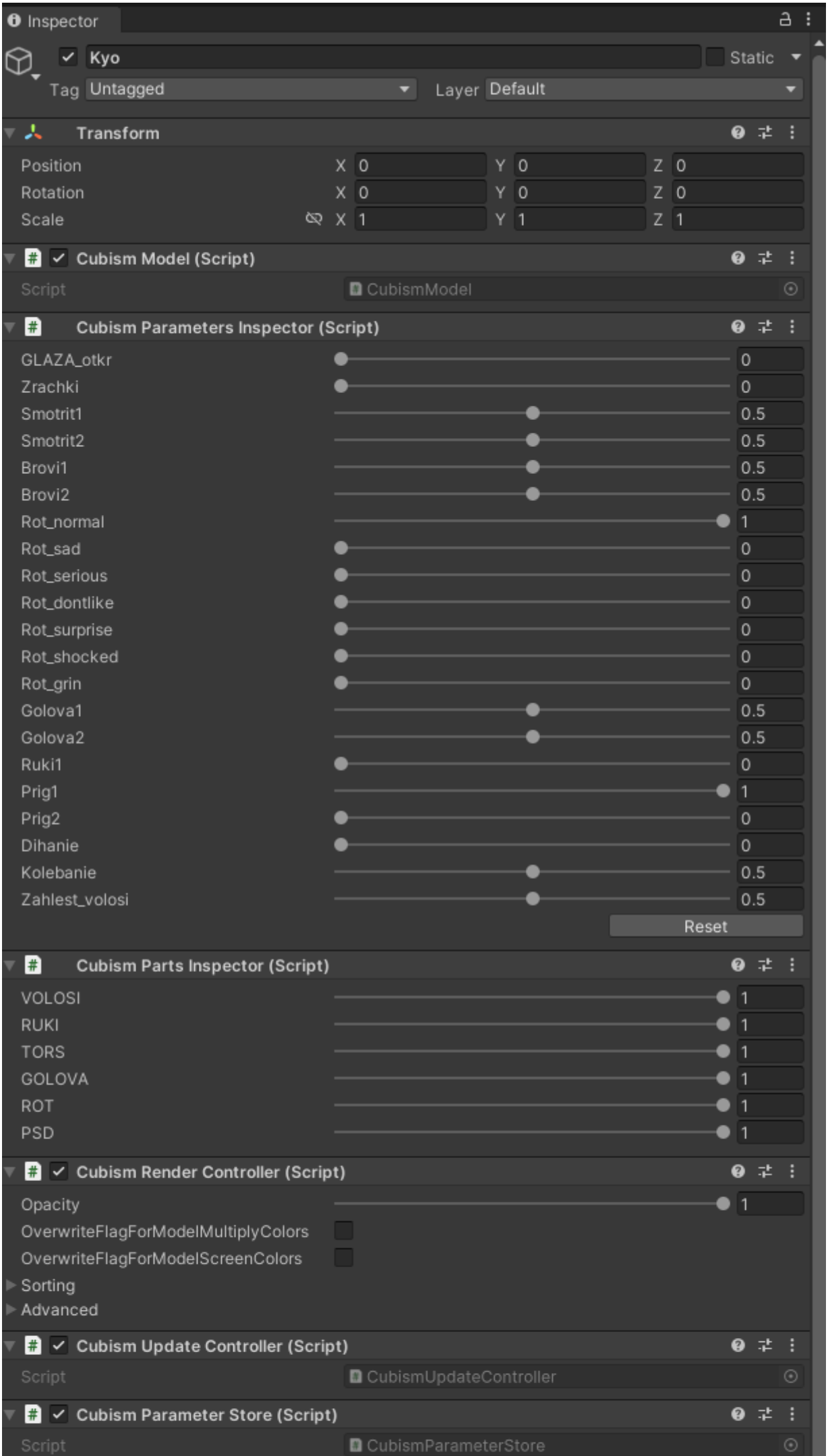
Для всех **.motion3**-файлов внутри папки **anim** также были сгенерированы анимационные файлы, поддерживаемые Unity:



Откройте файл **Kyo.prefab** в редакторе — вы увидите импортированную модель:



Справа в окне **Inspector** будут отображаться настройки различных компонентов Live2D SDK, включая анимируемые параметры нашей модели (компонент **Cubism Parameters Inspector**):


Inspector
Kyo
Static
Tag Untagged
Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Cubism Model (Script)
Script CubismModel
Cubism Parameters Inspector (Script)
GLAZA_otkr 0
Zrachki 0
Smotrit1 0.5
Smotrit2 0.5
Brovi1 0.5
Brovi2 0.5
Rot_normal 1
Rot_sad 0
Rot_serious 0
Rot_dontlike 0
Rot_surprise 0
Rot_shocked 0
Rot_grin 0
Golova1 0.5
Golova2 0.5
Ruki1 0
Prig1 1
Prig2 0
Dihanie 0
Kolebanie 0.5
Zahlest_volosi 0.5
Reset
Cubism Parts Inspector (Script)
VOLOSI 1
RUKI 1
TORS 1
GOLOVA 1
ROT 1
PSD 1
Cubism Render Controller (Script)
Opacity 1
OverwriteFlagForModelMultiplyColors
OverwriteFlagForModelScreenColors
Sorting
Advanced
Cubism Update Controller (Script)
Script CubismUpdateController
Cubism Parameter Store (Script)
Script CubismParameterStore

> Если модель не отображается после открытия префаба, попробуйте нажать кнопку **Reset** внутри **Cubism Parameters Inspector** (или просто измените значение какого-либо параметра). Также возможно, что модель просто не в фокусе камеры - нажмите **F,** чтобы сфокусировать камеру.

> Если же модель отображается некорректно — попробуйте в компоненте **Cubism Render Controller** во вкладке **Sorting** выставить параметр **Mode** в **Back To Front Order.**

Первое, что нужно сделать — прицепить сгенерированный контроллер анимации к компоненту **Animator** — просто перетащите его в соответствующие поле:

Сохраните изменения (если у вас отключено автосохранение) и закройте префаб.

Удаление исходников

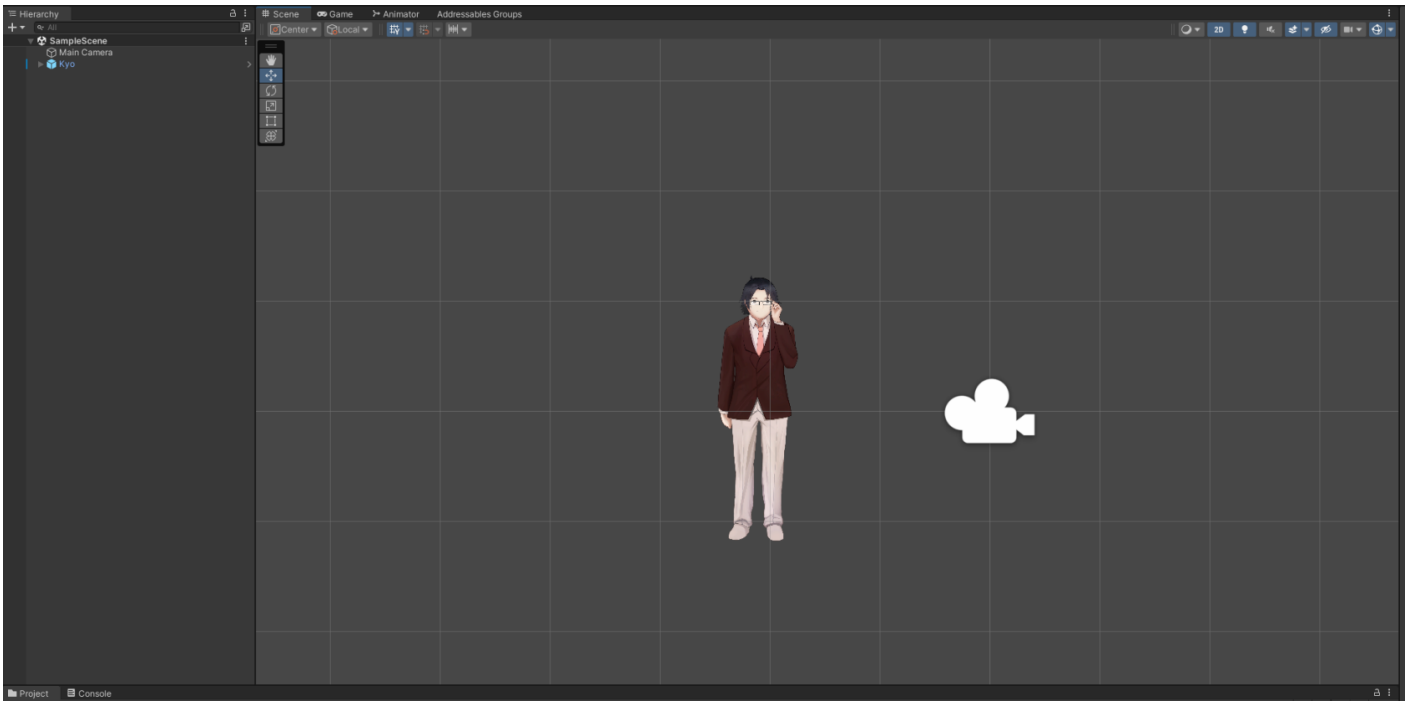
Очень важно после успешного импорта удалить исходники из проекта. На это есть две причины:

1. Вы можете случайно положить исходники модели в бандл;
2. По умолчанию каталог с бандлами не соберётся, если внутри будут неподдерживаемые Unity файлы;

Удалите все **.moc3, .model3.json** и **.motion3** файлы из проекта после завершения процесса импорта модели. **НЕ** удаляйте текстуры — они используются в префабе.

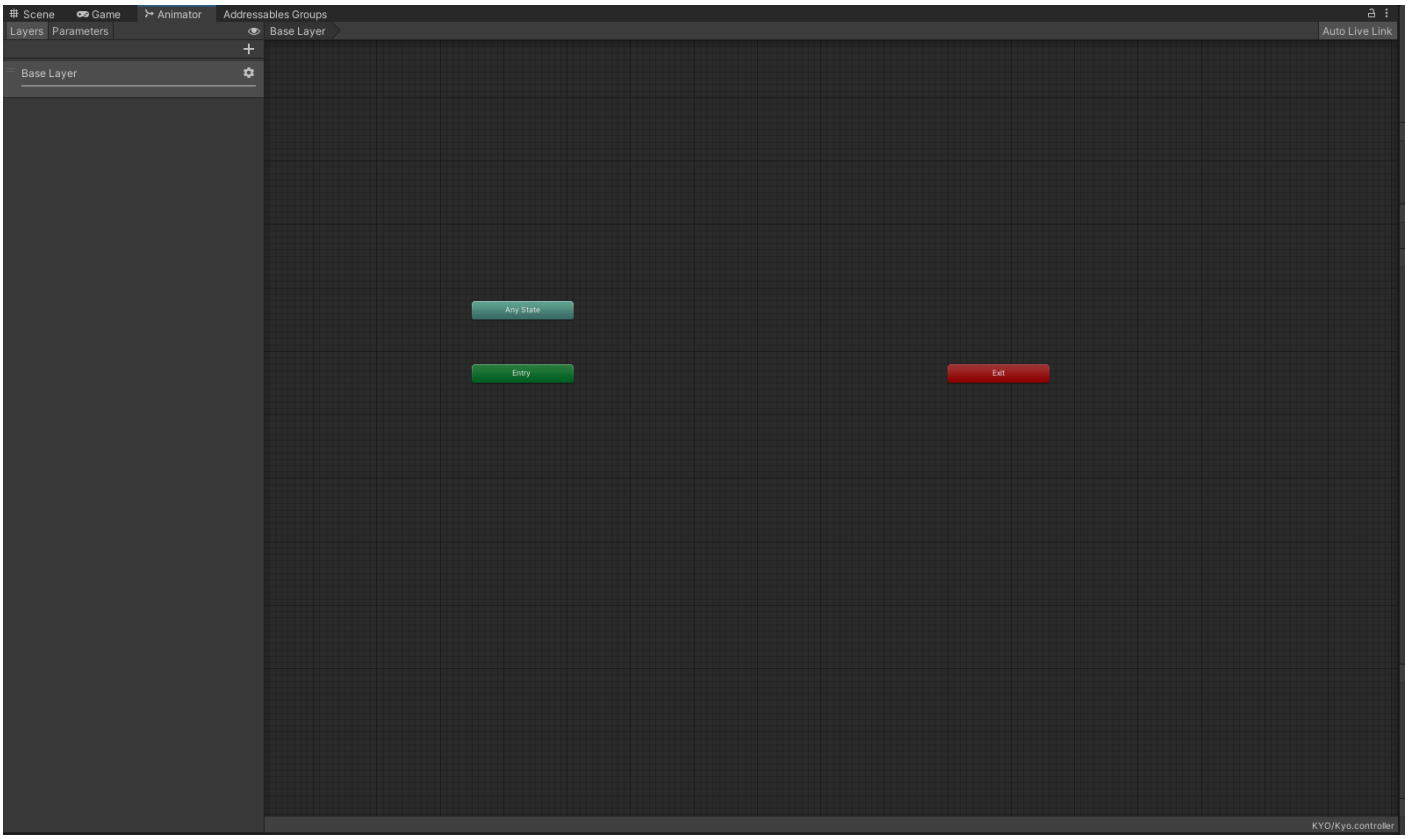
Анимации

Откройте любую сцену в редакторе и перетащите на неё префаб импортированной модели:



Выберите модель в иерархии сцены, затем откройте окно **Animator** (**Window -> Animation -> Animator**).

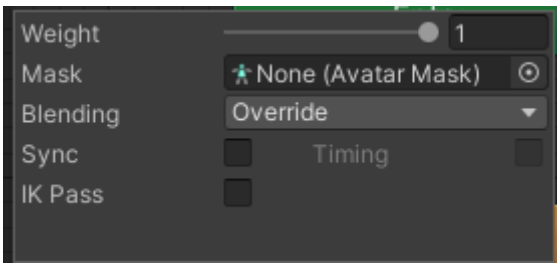
Вы увидите стандартный шаблон контроллера с базовым слоем:



У нашей модели есть заранее заготовленные анимации переходов между положениями рук и ног, однако нет самих анимаций конечных положений, которые будут проигрываться, когда модель находится в idle-состоянии (т.е. не меняет положение рук или ног).

Мы можем сделать анимации вручную — создадим их для рук.

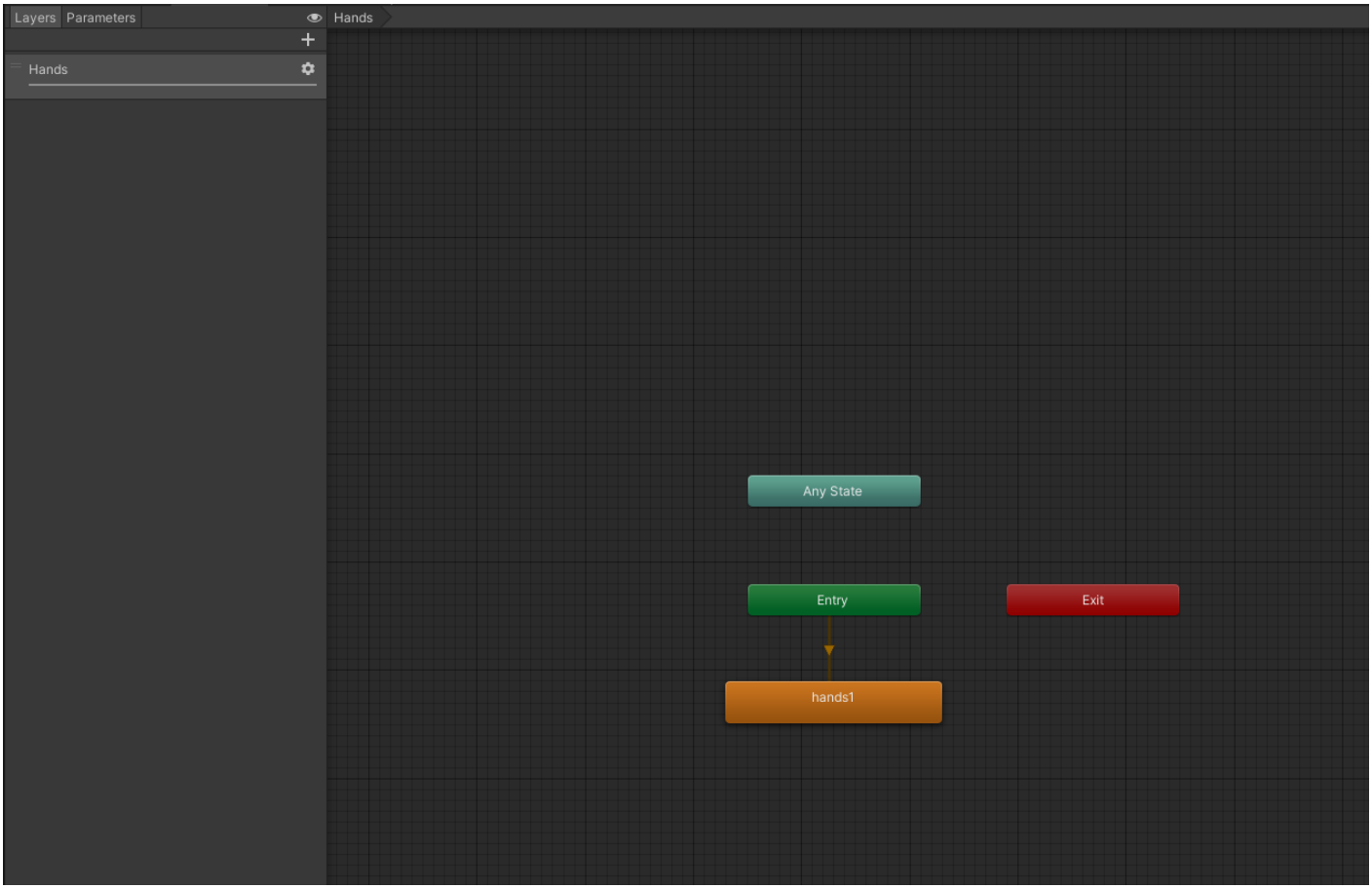
Переименуйте слой **Base Layer** в **Hands** (нужно кликнуть на название слоя), нажмите на шестерёнку справа и убедитесь, что параметр **Weight** у слоя равен единице:



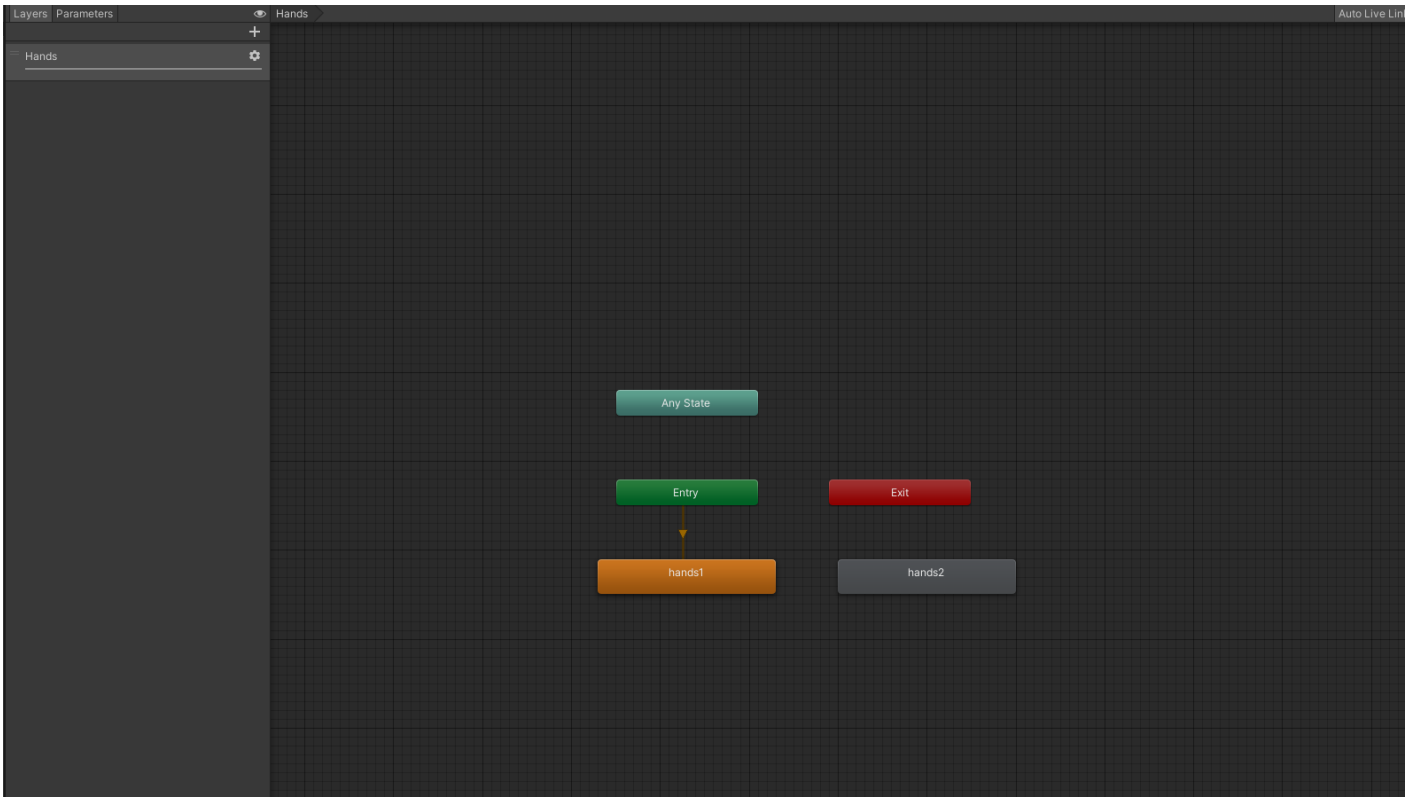
*При создании новых слоёв убедитесь, что значение **Weight** не равно нулю, иначе анимации на слое никак не будут воздействовать на модель в игре.*

Полоса под названием слоя также отображает значение этого параметра — если она серая — значит, что значение равно нулю.

Затем создайте на нём новый стейт и назовите его **hands1**:



Т.к. это единственный стейт на слое, он будет проигрываться по умолчанию при появлении модели на сцене. Создайте второй стейт под второе положение рук — **hands2**:

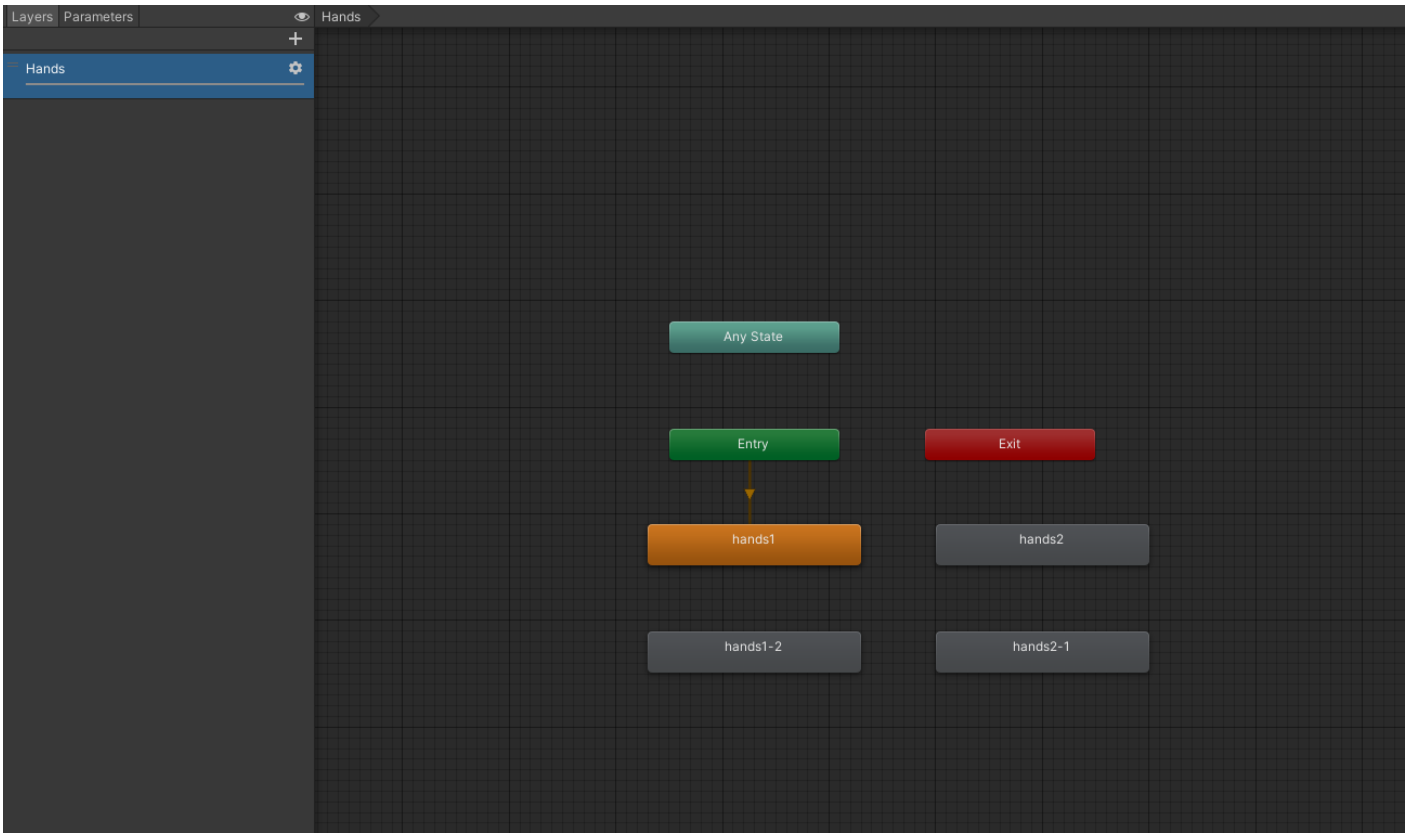


У нас есть два стейта под два положения рук — замечательно! Пока что, правда, они ничего не делают, т.к. у них нет анимаций — мы сделаем их чуть позже.

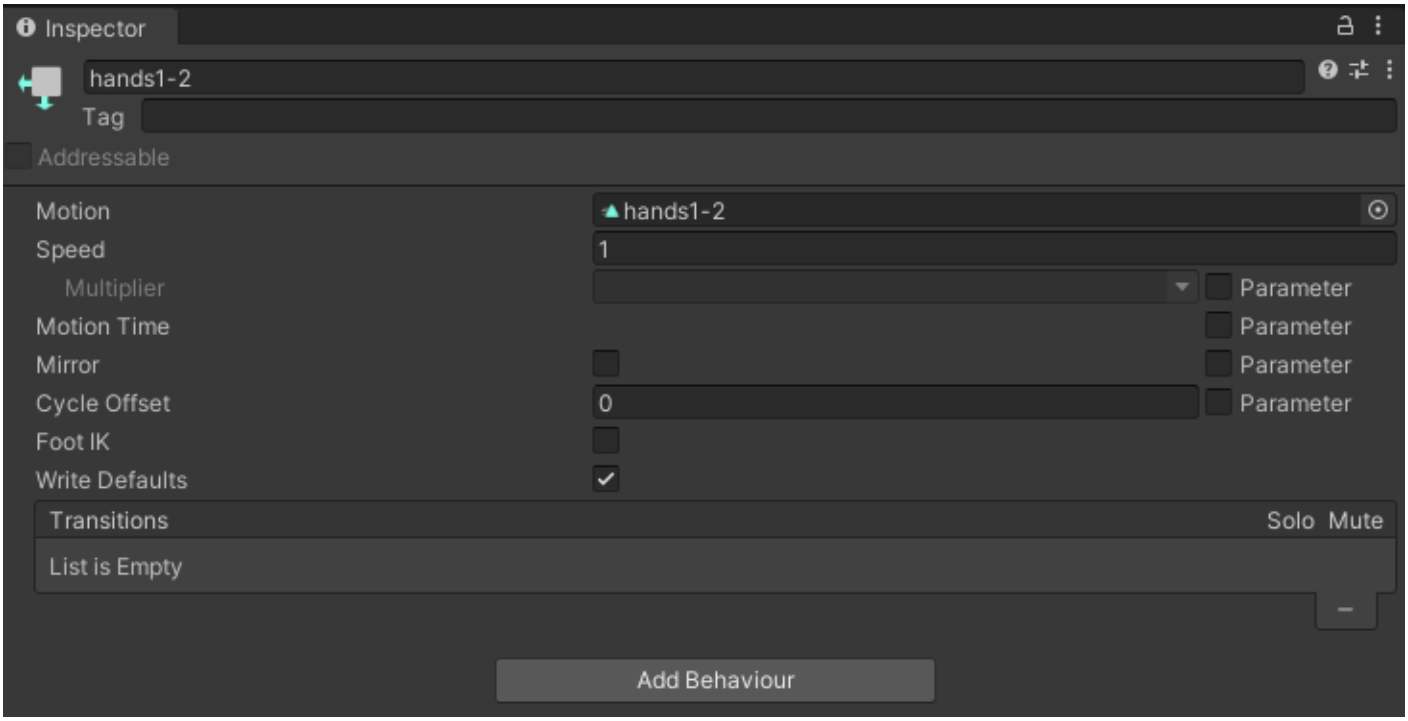
Откройте папку **anim** в папке с моделью и найдите там две сгенерированные анимации:

- hands1-2
- hands2-1

Это переходные анимации для различных положений рук. Первая — переход из 1 положения во 2, вторая — из 2 в 1. Перетащите оба этих файла на слой **Hands,** к созданным ранее стейтам:



В отличии от созданных вручную стейтов **hands1/hands2,** этим стейтам автоматически присвоились анимации. Вы можете увидеть их в инспекторе, в параметре **Motion,** кликнув по стейту:



Теперь мы можем сделать переходы между 2-мя положениями рук, которые будут проигрываться, когда модель меняет **hands1** на **hands2** и наоборот.

Это можно сделать двумя способами:

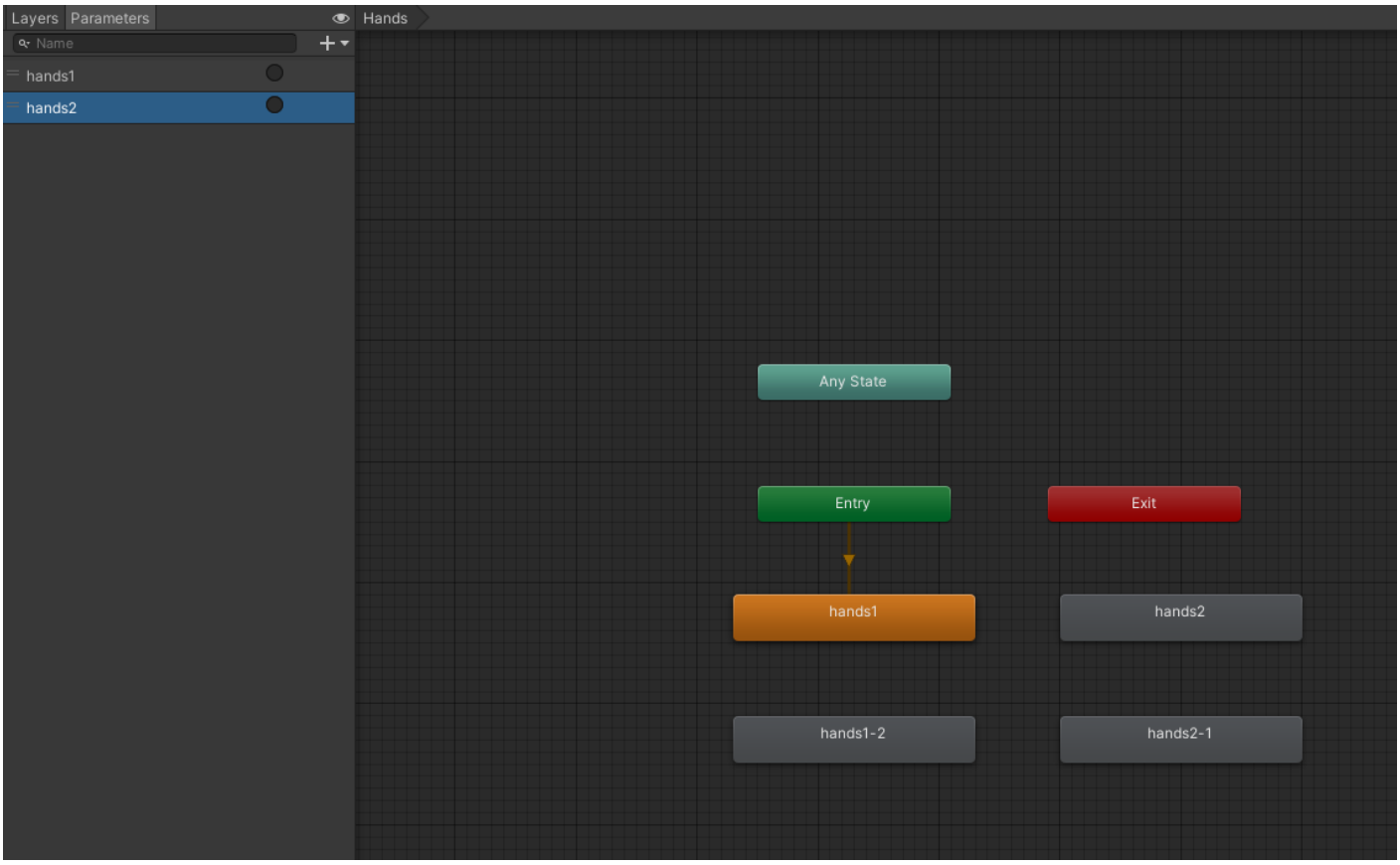
- Используя традиционную систему триггеров и переходов Unity;
- Используя упрощённую систему переходов, реализованную в LMR;

В этой статье будут рассмотрены оба способа, вы можете использовать тот, что кажется вам наиболее удобным.

Переходы

Реализация переходов при помощи триггеров

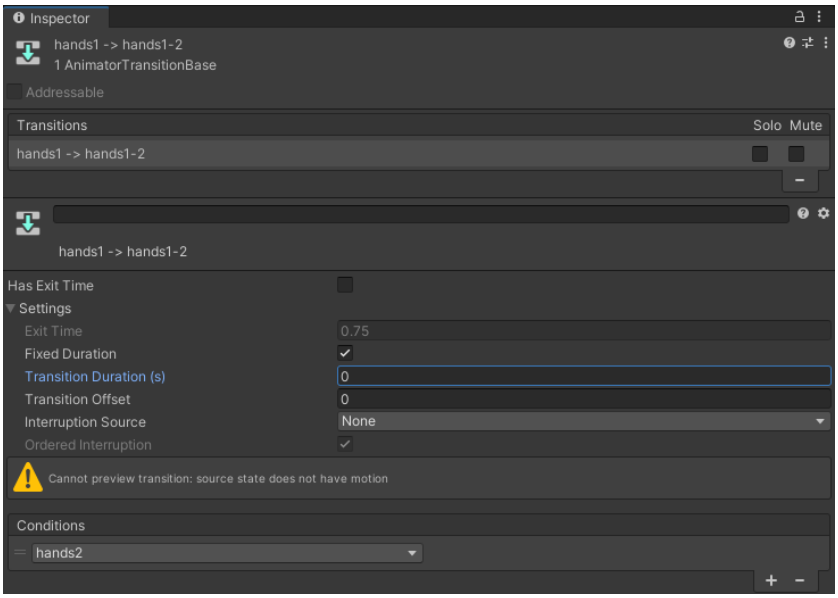
Создайте в параметрах аниматора триггеры, названия которых совпадают с названием стейтов основных положений рук (в данном случае — **hands1** и **hands2**):

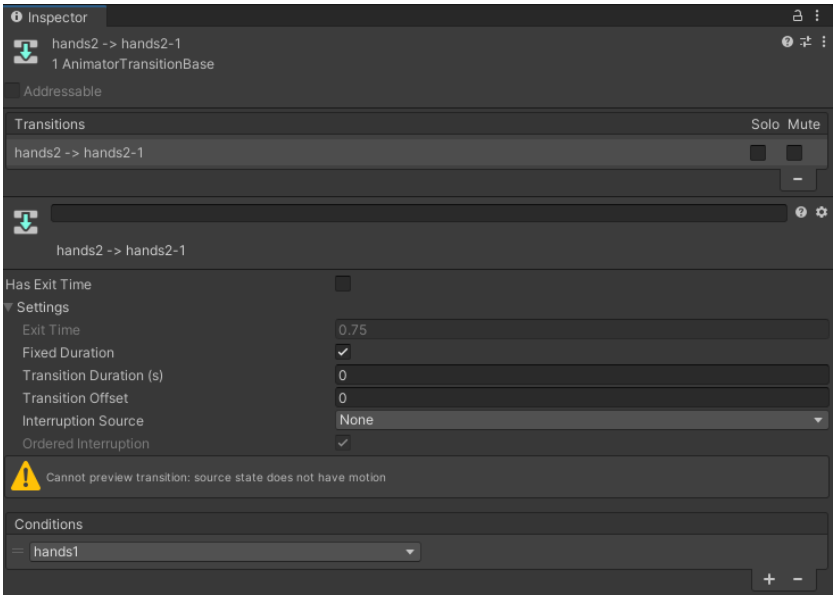


Далее создайте переходы между положениями рук, которые будут активироваться при помощи триггеров:

Переход из hands1 в hands2 активирует триггер **hands2,** из hands2 в hands1 — триггер **hands1.** Оба перехода происходят через стейты с анимациями смены положения рук.

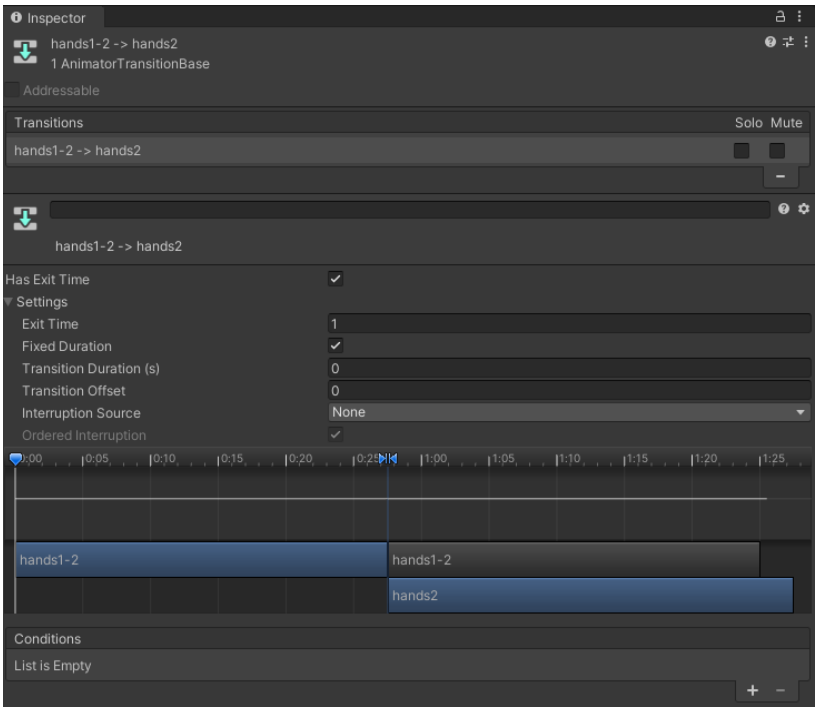
Также снимите чекбокс **Has Exit Time** у переходов **из** основных стейтов и поставьте **Transition Duration** 0:

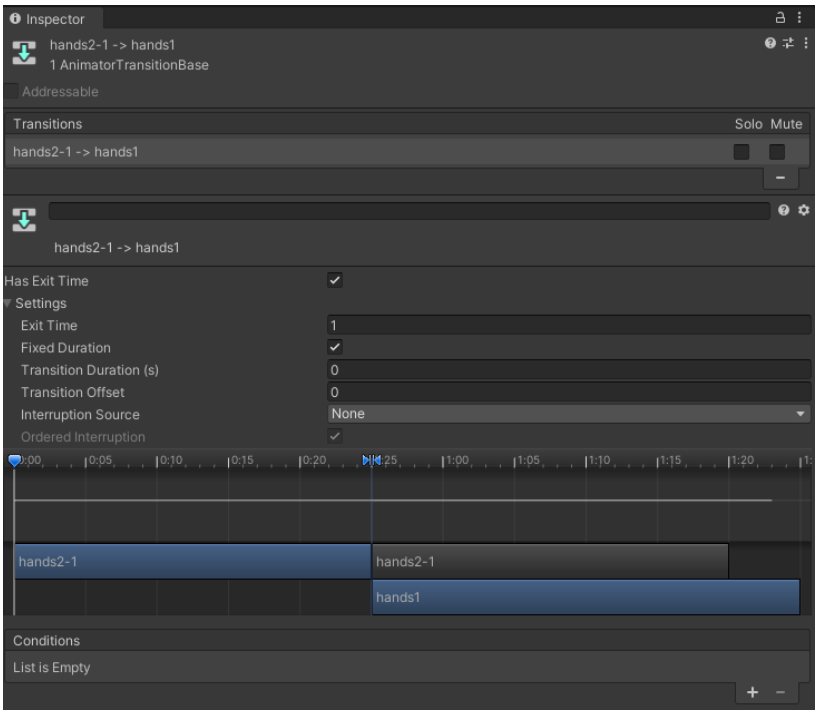




Предупреждение выводится, т.к. у основных стейтов пока что нет анимаций — мы скоро сделаем их.

У переходов **на** основные стейты поставьте значение **Exit Time** на 1 и **Transition Duration** на 0: _

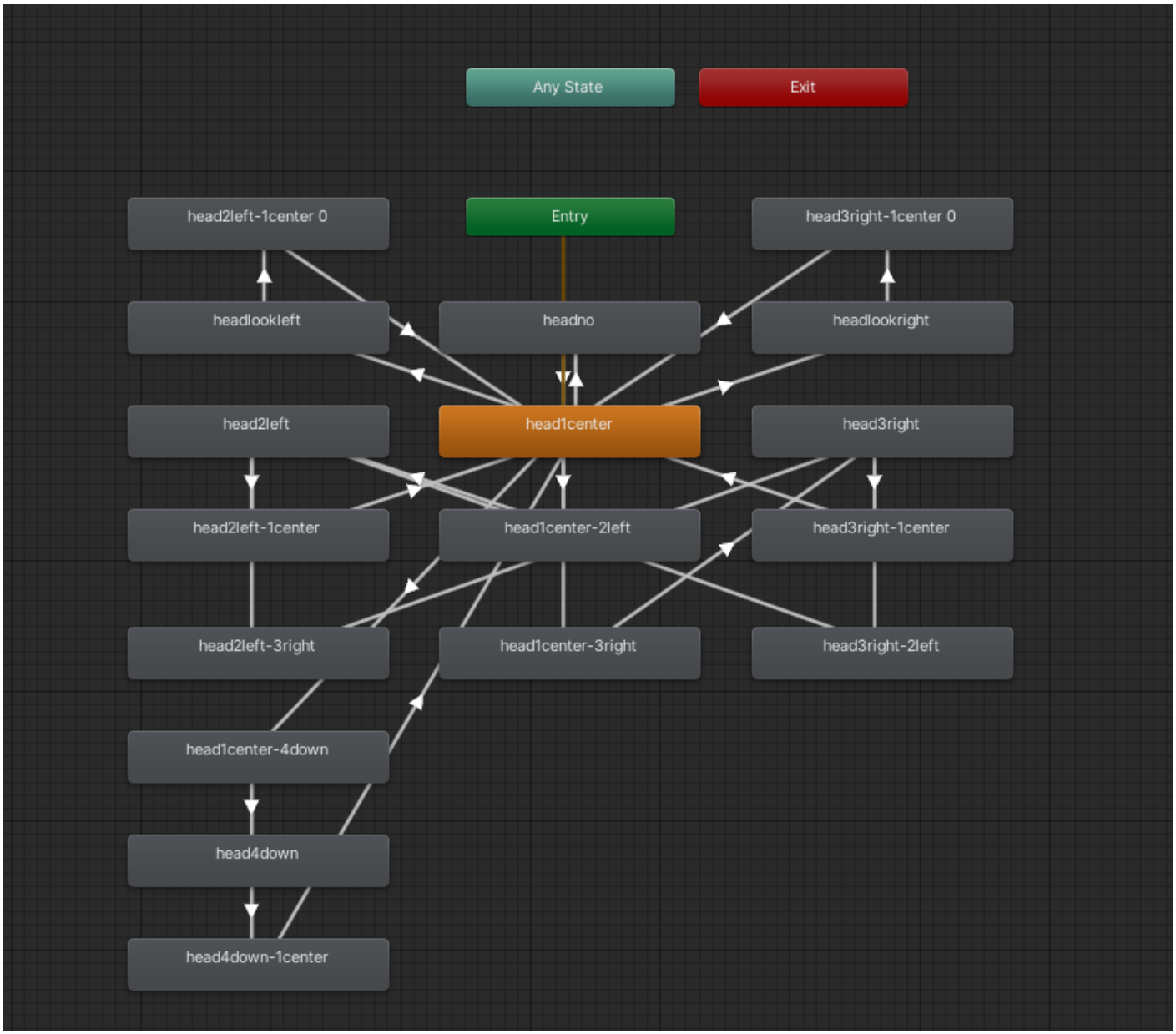




После проигрывания анимаций переходов, аниматор сразу перейдёт на основной стейт без задержек.

Реализация переходов при помощи упрощённой системы

Этот способ был придуман, когда мы столкнулись с неудобствами первого. При большом количестве анимаций и переходов между ними, схема может превратиться во что-то подобное:



Так выглядит слой Head у спрайта himitsu_front

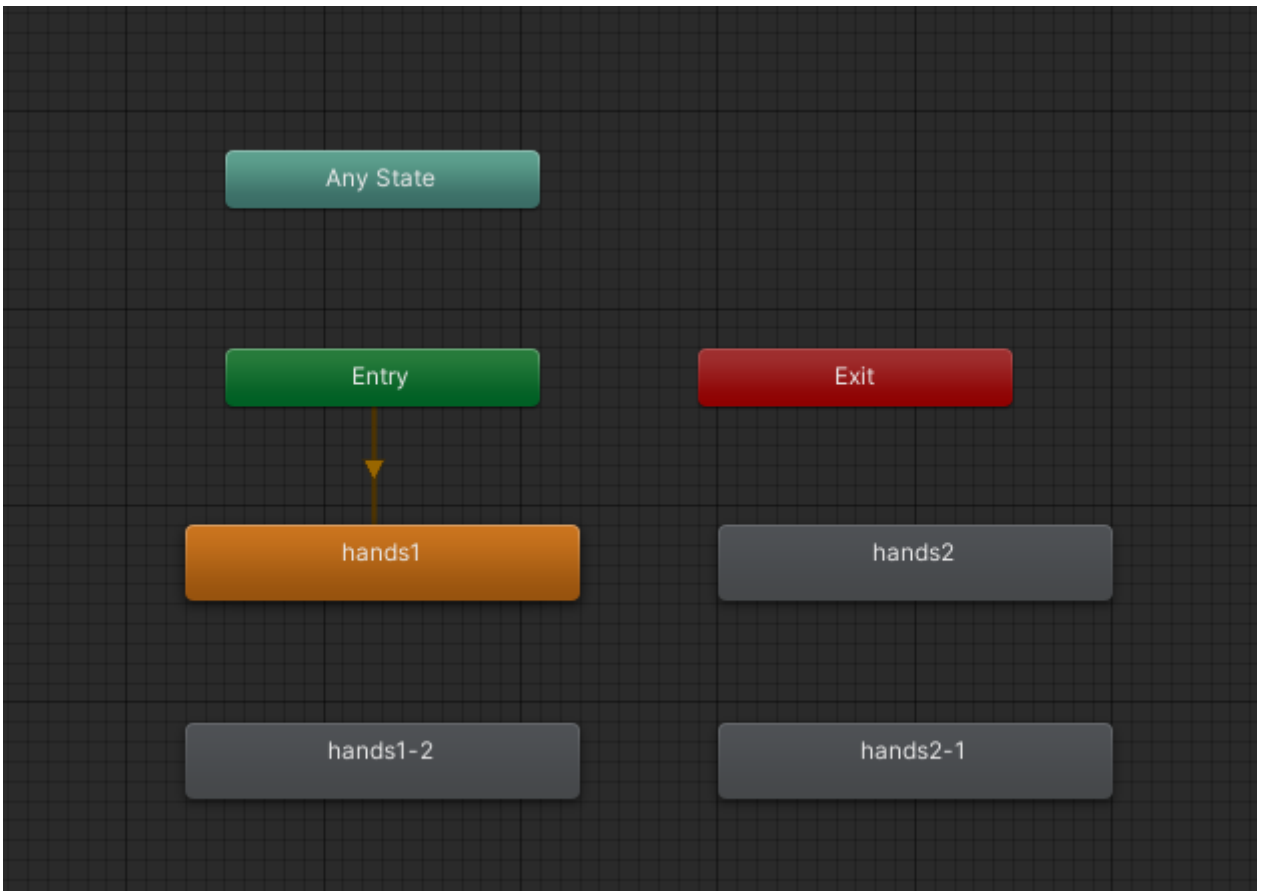
С этим не очень удобно работать и очень легко запутаться откуда и куда идёт тот или иной переход, поэтому был разработан альтернативный вариант.

Суть заключается в том, что стейты, которые должны проигрываться при переходах между основными стейтами (положениями рук **hands1** и **hands2** в нашем случае), называются определённым образом, используя достаточно простой синтаксис:

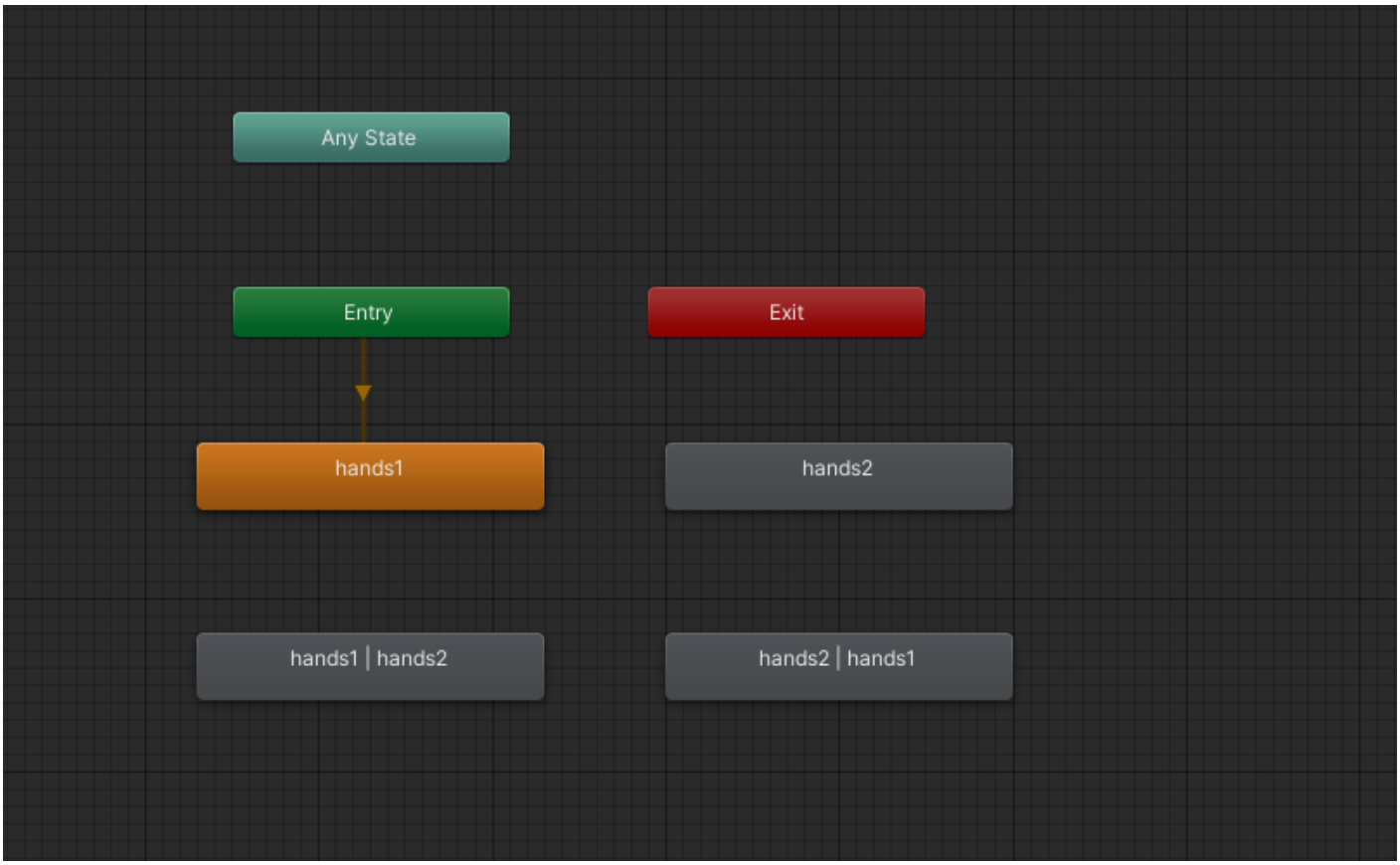
```
стейт_из_которого_переход | стейт_в_который_переход
```

Т.е. два стейта — "Стейт А" — из которого нужно перейти — и "Стейт Б" — в который нужно перейти, разделяются символом `|`.

Создадим такие переходы для наших положений рук. Сейчас у нас есть следующее:



Переименуйте **hands1-2** в **hands1 | hands2,** а **hands2-1** в **hands2 | hands1**:



И всё! Никаких триггеров, стрелочек переходов и прочих настроек. Игра автоматически будет проигрывать наши анимации при смене положений рук. Данный способ очень удобен, если у вас есть анимации переходов между различными состояниями модели и у вас нет необходимости делать сложные системы переходов при помощи аниматора Unity.

Автоматически переходы

Если мы не создадим вообще никаких стейтов для переходов, игра автоматически будет пытаться перейти из одной анимации в другую за фиксированное время. Подобная система не подходит для чего-то сложного (например, такие переходы для конечностей будут выглядеть не очень), но для чего-то простого, вроде эмоций, вполне подходит.

Создание анимаций

В идеале, все анимации для вашей модели вы должны делать в редакторе Live2D, а затем импортировать в Unity. Однако, если по какой-то причине вы решили этого не делать, существует также возможность создавать анимации прямо в редакторе Unity.

Средствами Live2D SDK для Unity создавать анимации крайне проблематично, т.к. окно записи анимаций не будет реагировать на изменение параметров модели (и сама модель не будет реагировать на проигрывание анимаций вне контроллера, который работает только в Play-моде), поэтому для этих целей нами был написан специальный плагин, позволяющий анимировать Live2D-модели внутри Unity.

Скачайте и импортируйте его в проект.

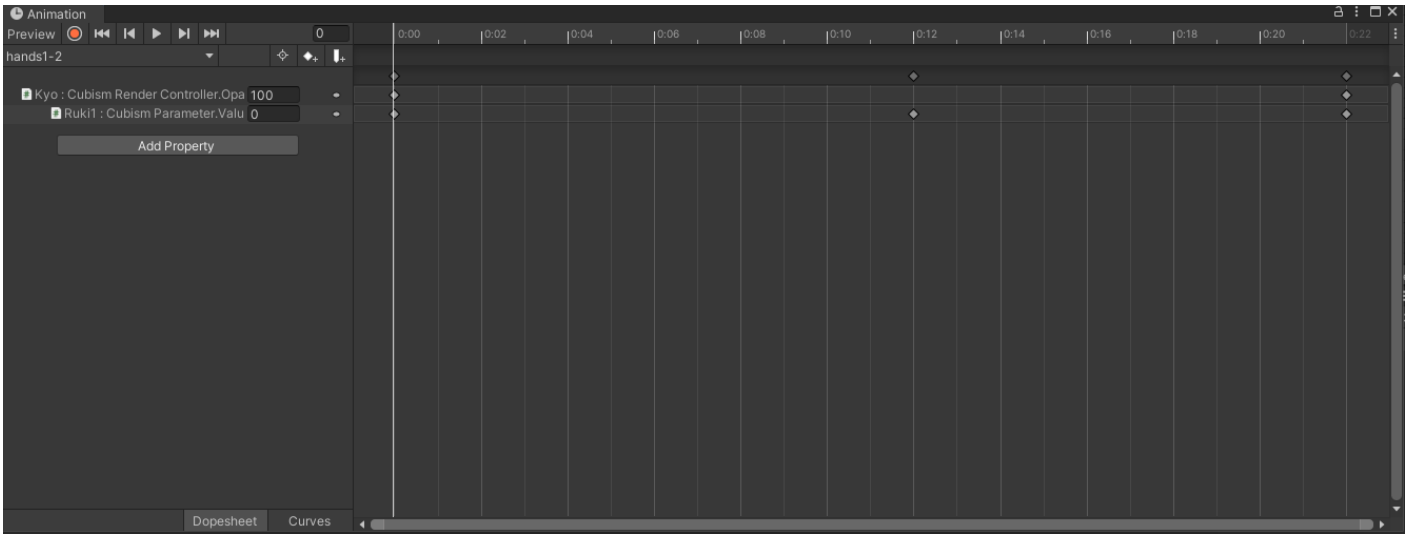
> Если вы скачали готовый проект по ссылке в начале статьи, плагин можно не качать — он идёт в комплекте.

Выберите модель на иерархии сцены и добавьте к ней компонент **CubismPoser**. После этого компонент **Cubism Parameters Inspector** должен исчезнуть, а вместо него, в самом низу, появится новый компонент с теми же параметрами:

Анимации для положений рук

Сделаем анимации для основных положений рук - **hands1** и **hands2.**

Выберите модель на иерархии сцены, затем откройте окно **Animation** (**Window -> Animation -> Animation,** или **Ctrl + 6**). Вы увидите окно, в котором будут ключи анимаций-переходов:



Мы можем посмотреть значения параметров в начале и в конце анимации.

У первого положения рук (hands1) параметр **Ruki1** должен быть равен **0**, у второго (hands2) — **1**.

Кликните на выпадающий список в левом верхнем углу окна и создайте новый анимационный файл, нажав на **Create New Clip...**

Назовите его **hands1** и сохраните в папку с остальными анимациями.

Убедитесь, что на таймлайне установлен 0 кадр:

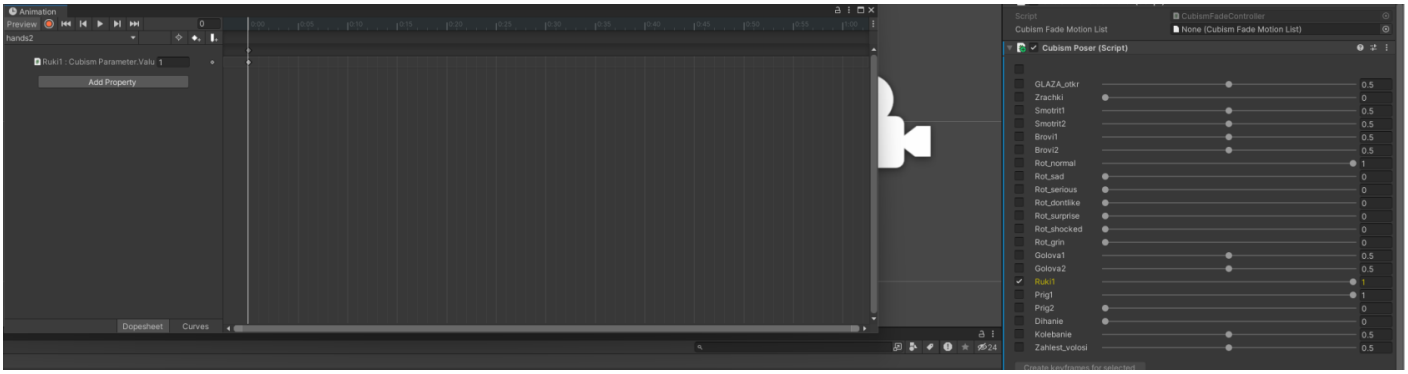
Далее включите режим записи, выберите в компоненте **CubismPoser** параметр **Ruki1** и установите его значение на 0:

Затем нажмите кнопку **Create Keyframes for Selected** (активна только в режиме записи).

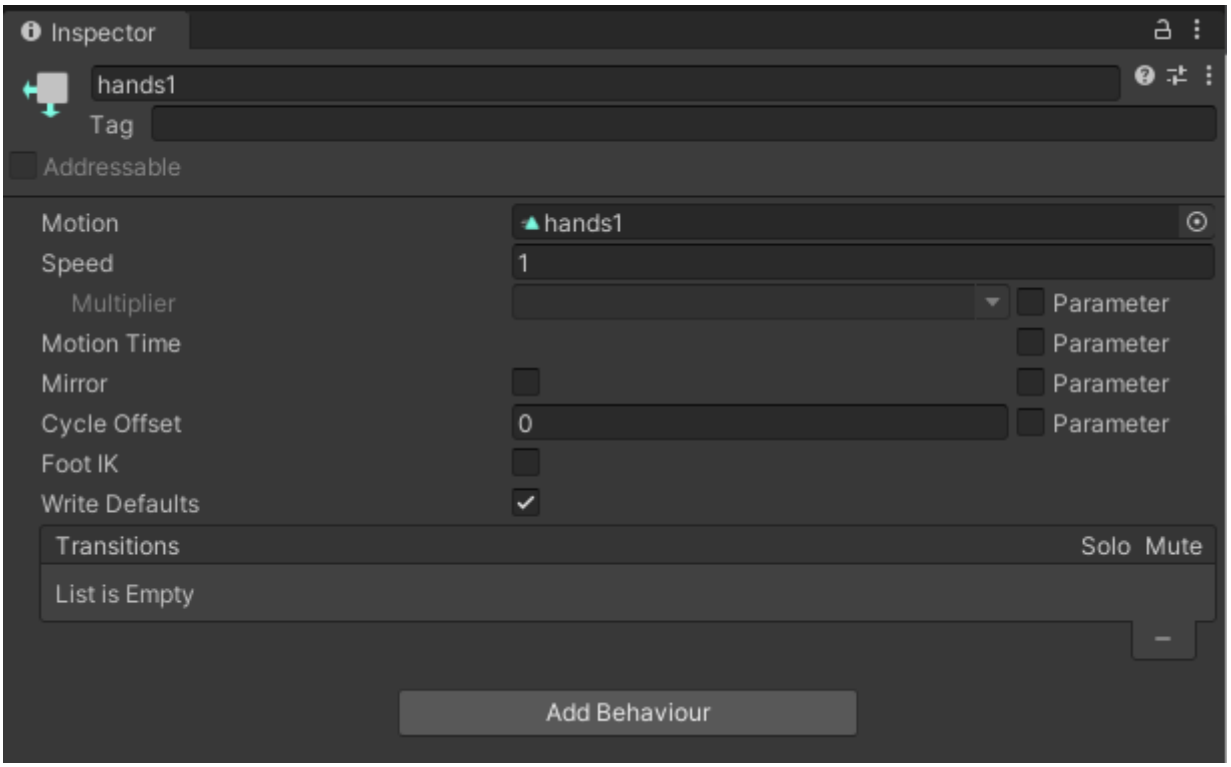
На таймлайне должен появится ключ со значением нашего параметра:

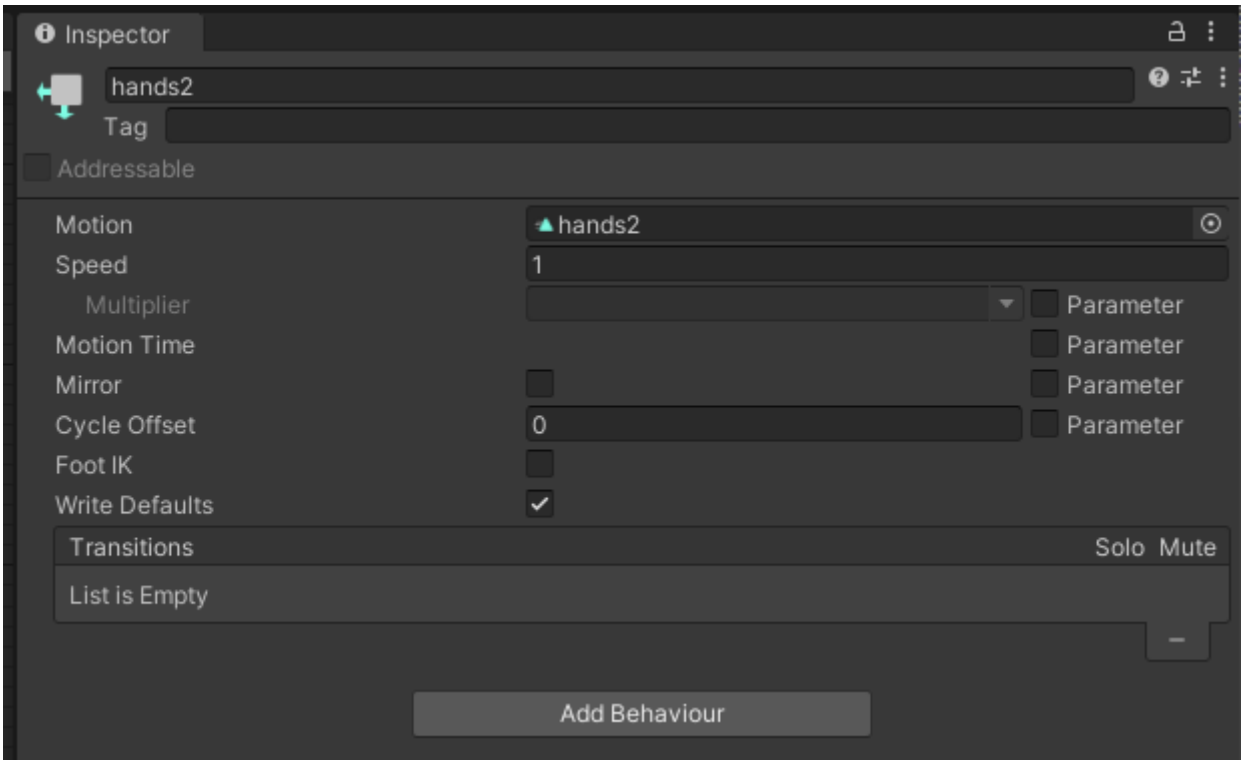
Поздравляю - вы только что создали анимацию для Live2D-модели в Unity (пусть она и состоит только из одного кадра).

Сделайте то же самое для **hands2,** установив значение **Ruki1** на 1:



Анимации готовы. Теперь осталось только добавить их к соответствующим стейтам в аниматоре (по идее, они должны добавиться сами, если названия стейтов и анимаций совпадают):





Готово. Вы сделали полностью функционирующие положения рук для модели, а также переходы между ними.

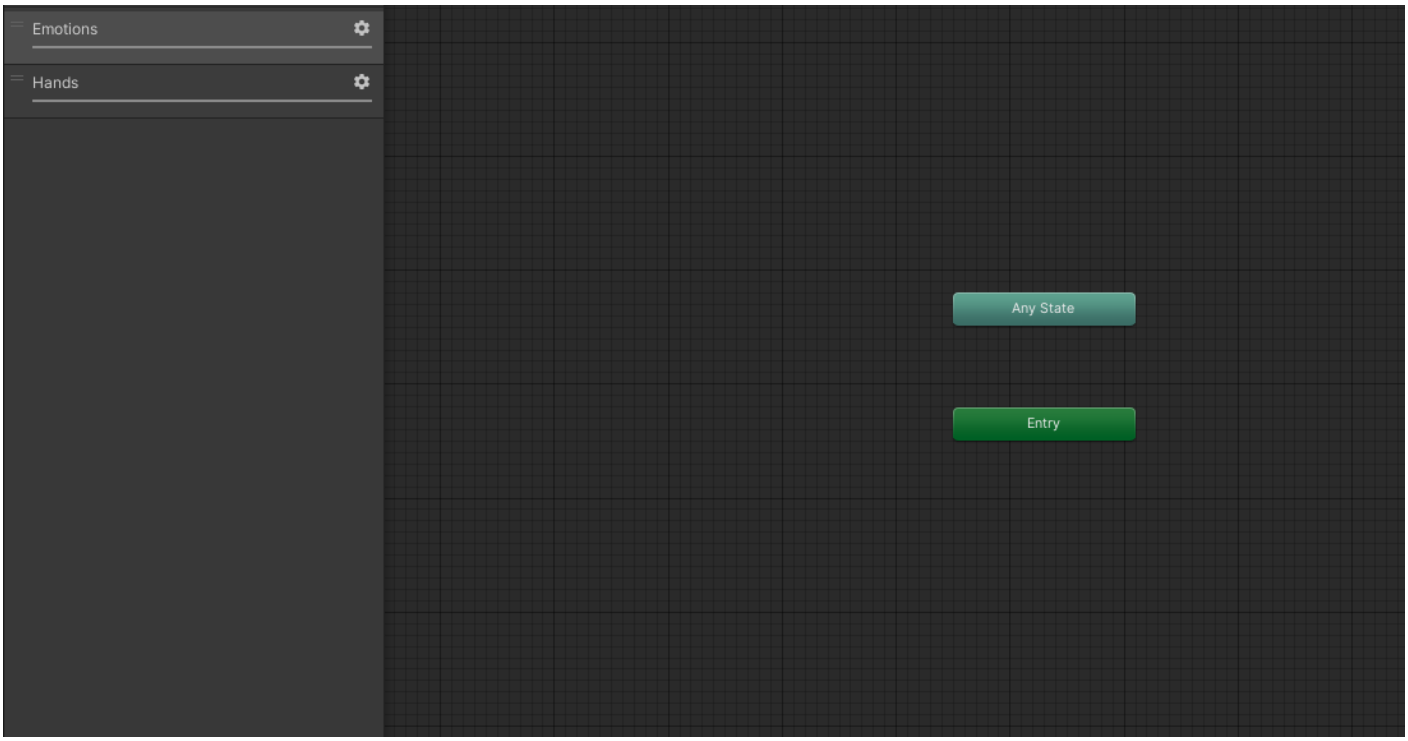
В исходниках модели также лежат анимации переходов ног — вы можете попрактиковаться на них, процесс абсолютно идентичен.

Эмоции

В архиве с исходниками нет никакой анимации эмоций, но вы можете попробовать создать свои. Функциональность модели ограничен (всё же, это спрайт из старой демо-версии игры), но у неё есть несколько параметров, отвечающих за эмоции.

При создании анимации эмоций очень важно создавать ключи для всех параметров, отвечающих за эмоции — иначе при переходах какие-то значения, которые не используются в конкретной анимации, могут не изменяться после предыдущей анимации.

Сделаем несколько эмоций для нашего спрайта: создайте в аниматоре новый слой **Emotions** и переместите его в самый верх:

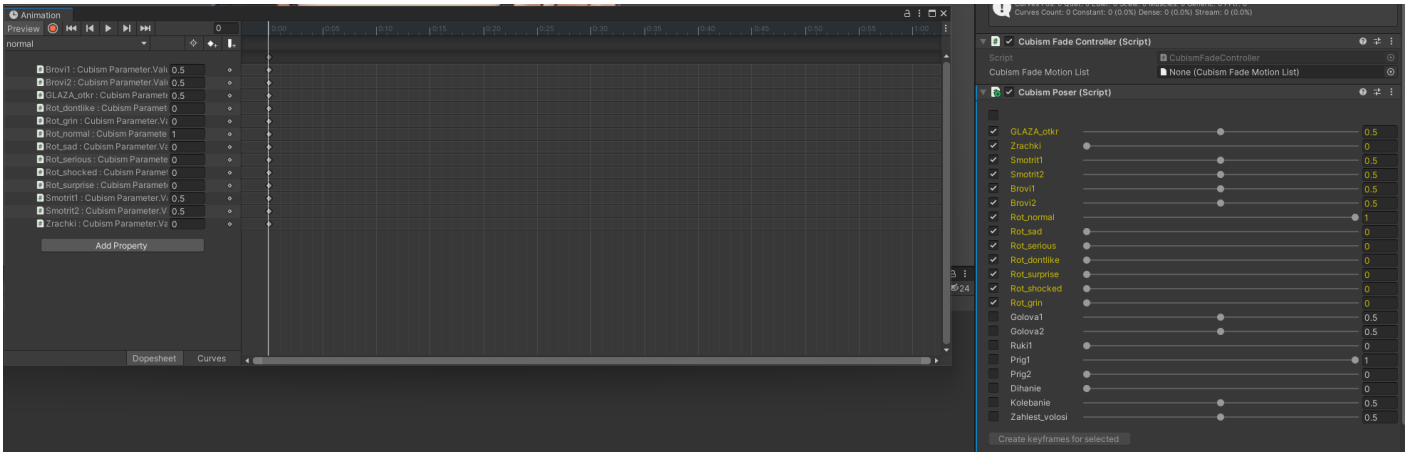


Далее создадим несколько анимационных файлов:

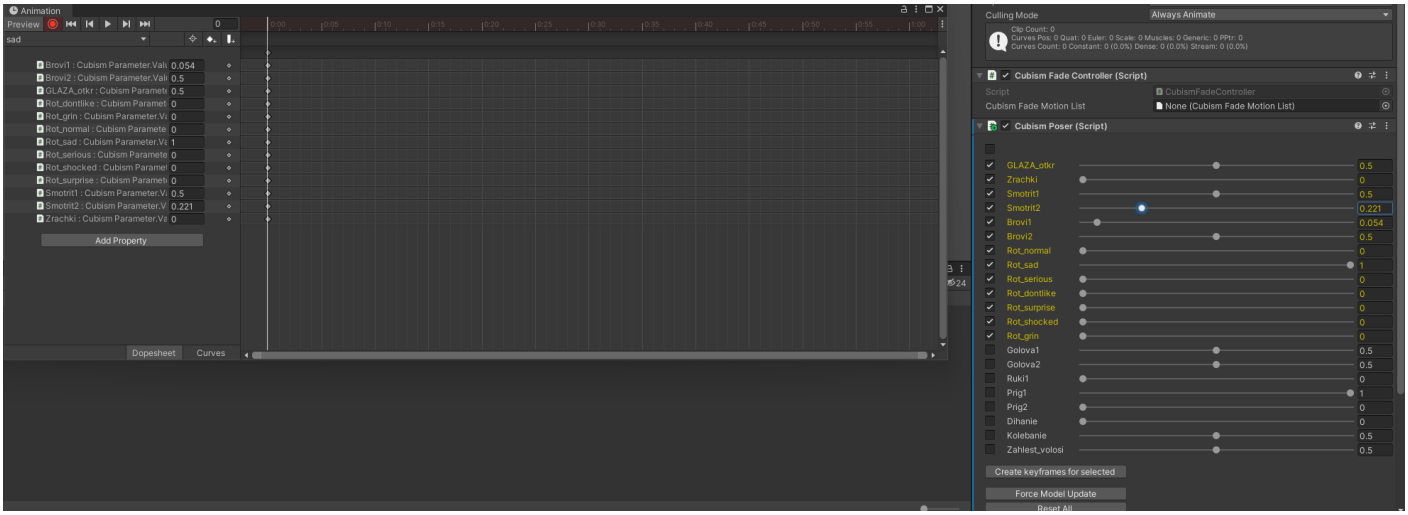
normal

Обычное (стандартное) выражение лица персонажа, которое будет активно по умолчанию. Выделяем все необходимые параметры, не

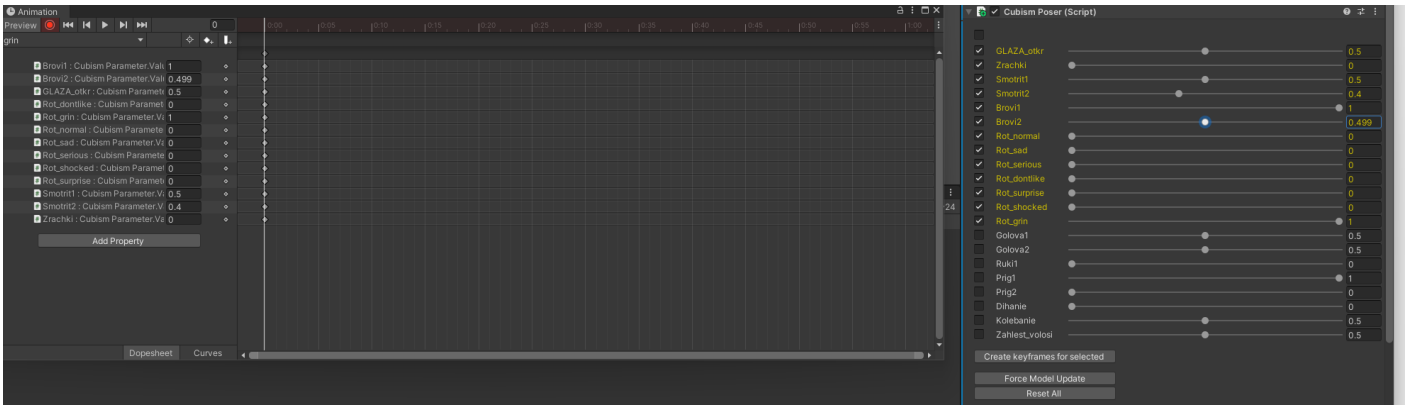
меняя их значений, и создаём новый файл анимации с ключами:



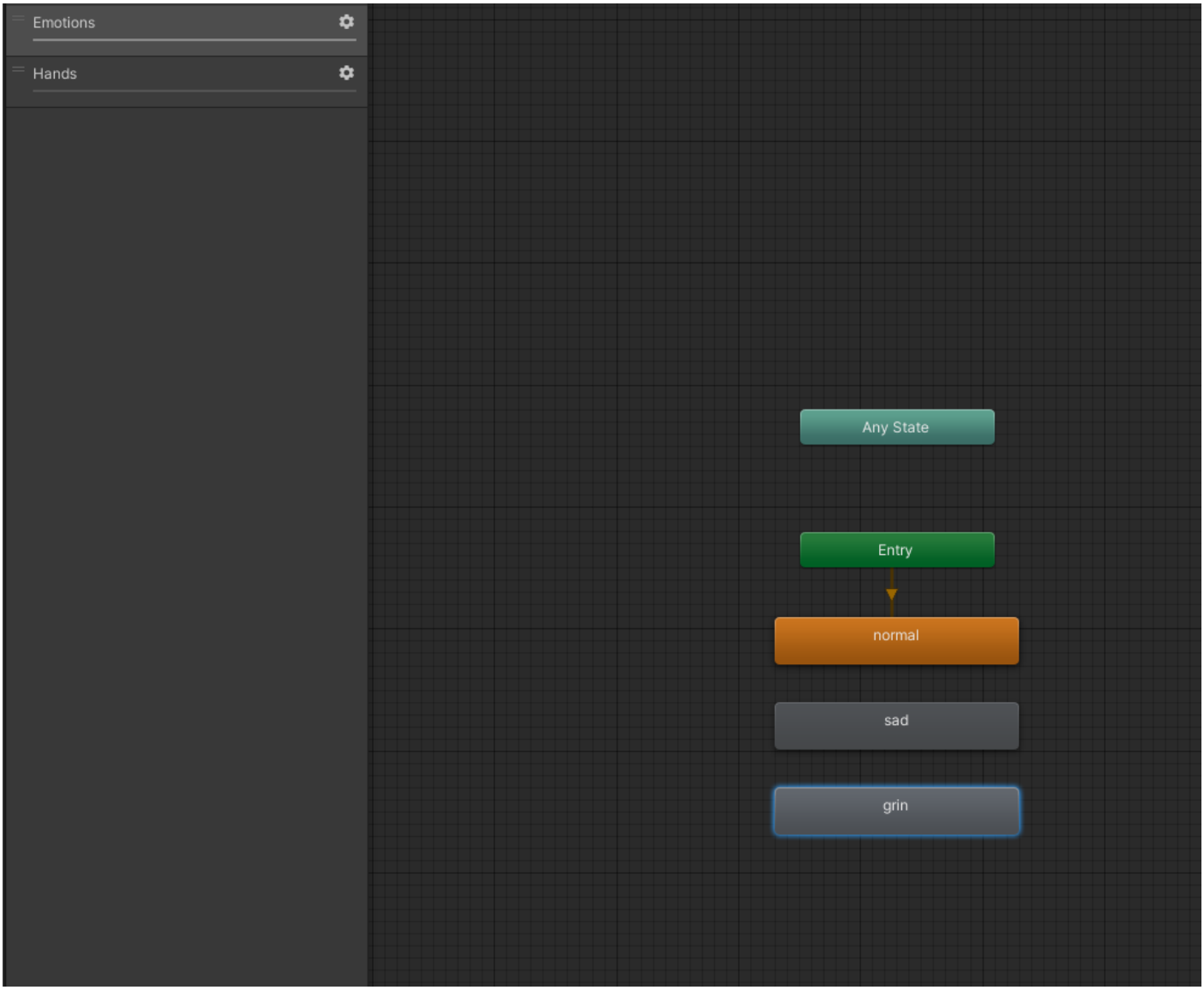
sad



grin



Для всех созданных анимаций в аниматоре были автоматически проставлены одноимённые стейты — по умолчанию они создаются на самом верхнем (первом) слое:



Нам не нужно ничего менять — для смены эмоций в игре достаточно автоматических переходов.

Дыхание и моргание

Дыхание и моргание реализуются при помощи стандартных компонентов Live2D SDK:

- Cubism Harmonic Motion Controller
- Cubism Harmonic Motion Paramater
- Cubism Eye Blink Controller
- Cubism Eye Blink Input
- Cubism Auto Eye Blink Input

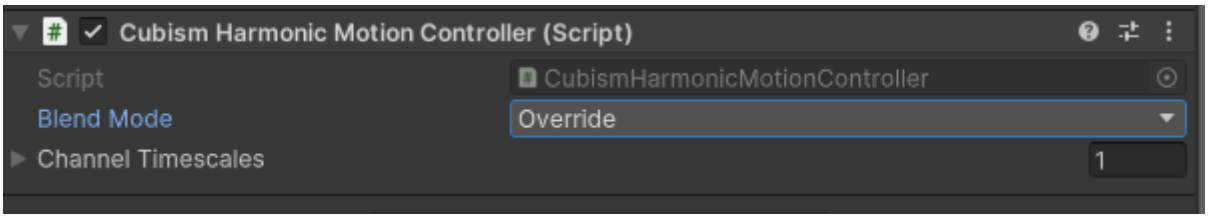
Подробнее о настройках вы можете прочитать в официальной документации. Далее будет рассмотрен процесс настройки этих компонентов с общими параметрами, которые используют модели в игре.

Все настройки должны проходить в префабе модели — чтобы изменения сохранились не только на сцене.

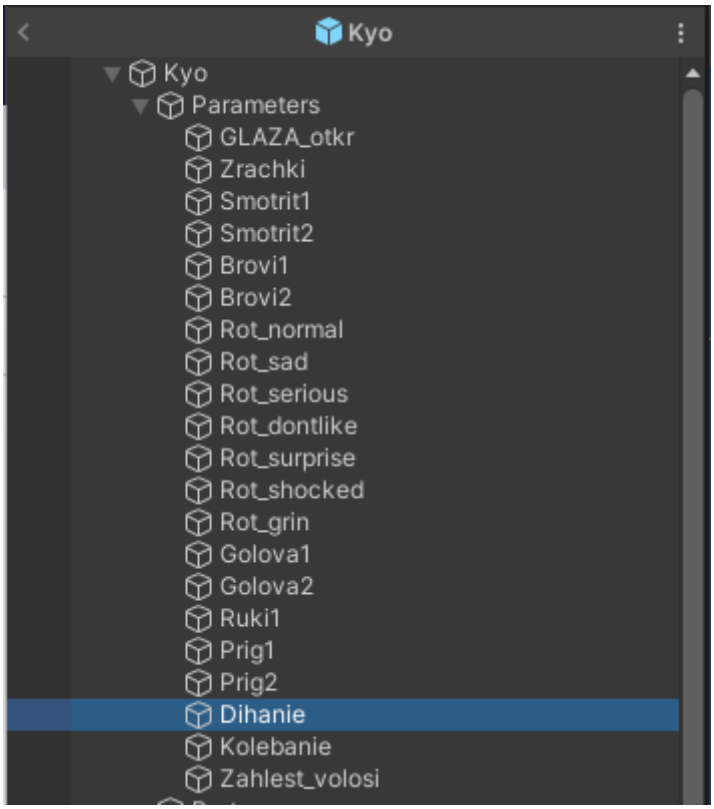
Дыхание

Добавим нашей модели анимацию дыхания.

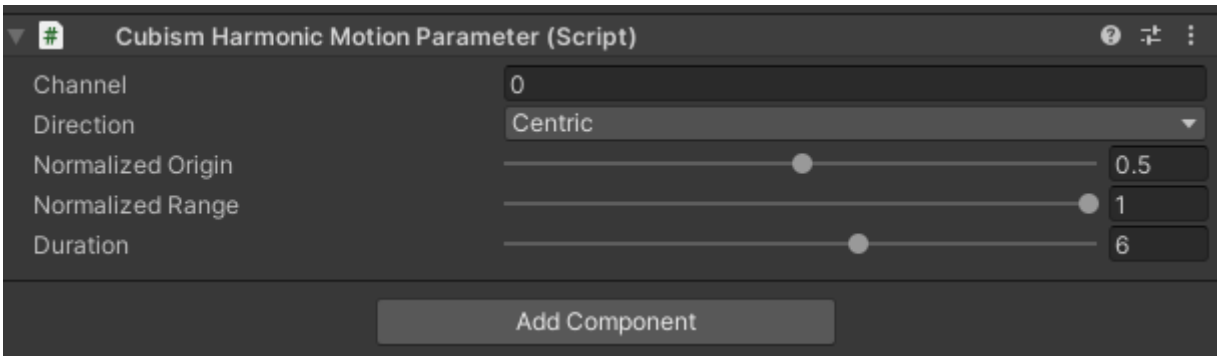
Откройте префаб и добавьте к нему компонент **Cubism Harmonic Motion Controller,** значение **Blend Mode** установите в **Override:**



Далее в иерархии модели внутри **Parameters** найдите объект **Dihanie** и добавьте к нему компонент **Cubism Harmonic Motion Parameter:**

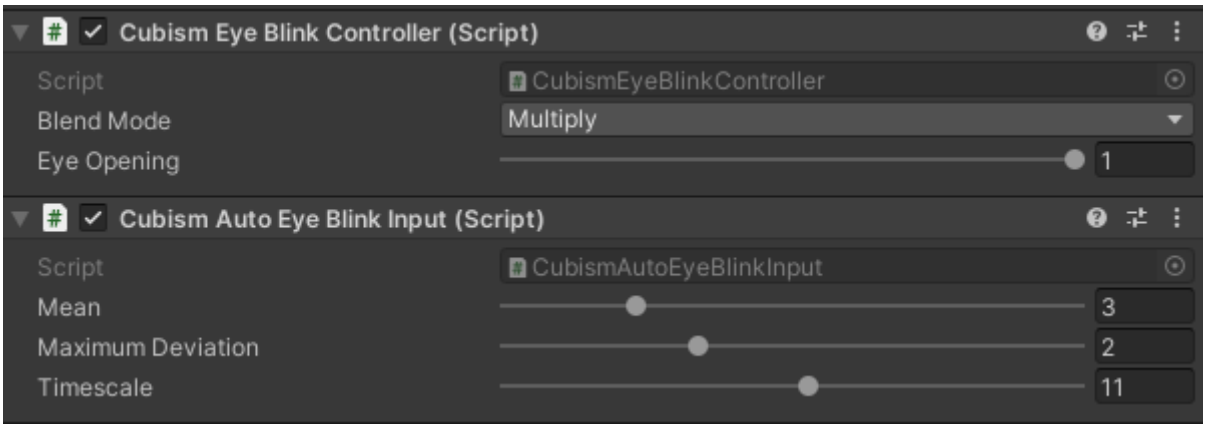


Установите параметры следующим образом:

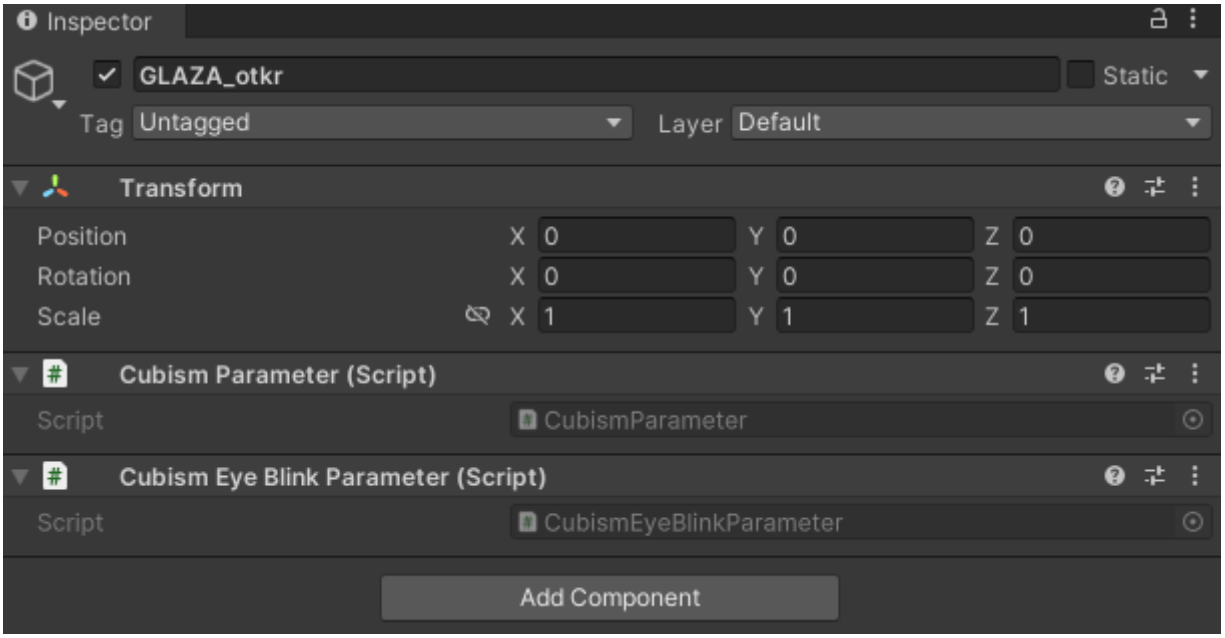


Моргание

Добавьте к префабу модели два компонента: **Cubism Eye Blink Controller** и **Cubism Auto Eye Blink Input** со следующими параметрами:



Далее в иерархии модели внутри **Parameters** найдите объект **GLAZA_otkr** и добавьте к нему компонент **Cubism Eye Blink Parameter:**

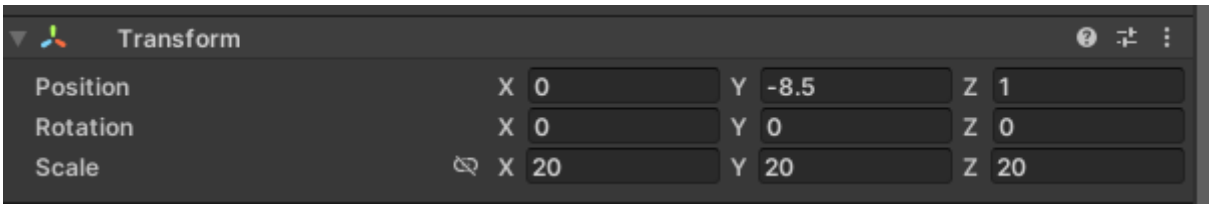


Сохраните изменения и закройте префаб.

Проверка

Вернитесь на сцену, расположите модель перед камерой, задав ей подходящий размер.

Если вы скачали готовый проект, то вы можете воспользоваться следующими параметрами:



Включите Play-мод. Если всё сделано верно, модель должна дышать и моргать.

Финальный результат

Архив с моделью, анимациями и двумя контроллерами (kyo.controller — с триггерами, kyo_simple.controller — без) можно **скачать здесь**.

Экспорт в игру

1. Соберите каталог с моделью и всеми необходимыми ассетами.
2. Добавьте каталог в ресурсы мода.
3. Добавьте в ресурсы мода спрайт персонажа, вписав модель, необходимые слои и стейты.

Если вы всё сделали верно, модель и анимации будут корректно отображаться в игре.

Revision #84
Created 10 August 2023 01:50:40 by Admin
Updated 21 January 2025 16:25:18 by Admin