**РАЗРАБОТКА ИМИТАЦИОННОЙ МОДЕЛИ**

**ДЛЯ СИМУЛЯТОРА УСТАНОВКИ**

**СОЗДАНИЕ БЛОК-СХЕМЫ РАБОТЫ УСТАНОВКИ**

Для программирования действий любой установки, прежде всего, необходимо продумать и составить **СХЕМУ ВЗАИМОДЕЙСТВИЯ** **ОБЪЕКТОВ**, отражающую взаимодействие объектов внутри этой установки, при составлении которой необходимо пройти следующие этапы:

* ***Выделить события***, которые могут происходить (включение/выключение установки, поворот или перемещение какого-нибудь объекта, нажатие кнопки и т. д.).
* ***Выделить объекты***, которые должны инициировать эти события (например, нажатие по переключателю инициирует событие по включению/выключению установки).
* ***Выбрать элементы***, которые должны реагировать на то или иное событие (например, на включение/выключение установки может реагировать большое количество элементов, начиная от отображения показаний дисплеев до запуска механизмов).
* ***Указать*,** при каких должны происходить события и какие методы при этом будут вызываться.

Например, в работе симулятора лабораторной установки «НАБЛЮДЕНИЕ КОЛЕЦ НЬЮТОНА» используются следующие события:

* включение установки,
* выбор цвета света вращением барабана,
* переход к окуляру,
* заполнение таблицы результатами наблюдения,
* при необходимости обработка в таблице результататов наблюдения,
* возврат к выбору нового цвета и повторение наблюдения,
* просмотр таблицы,
* очистка таблицы.

**СИСТЕМА ЗАДАНИЙ**

Первое задание при начале работы с симулятором это, как правило, задание «Включить установку». После щелчка по кнопке «РАБОТА УСТАНОВКИ» выполняется событие вывода в текстовое окно сообщения: «Включите установку». Первое задание выполняется при совершении события включения установки, когда пользователь щелкнет мышью на объект «Переключатель», что и приведет к выполнению первого задания.

После выполнения первого задания программа должна выдать в информационное окно следующее задание (см. рисунок), которое должно выполняться в соответствии с появляющимся текстовым сообщением и т. д. вплоть до окончания работы с симулятором.



Результатом работы с симулятором установки являются полученные по ходу выполнения заданий измеренные значения, показания приборов и т. п., которые последовательно заносятся в таблицу с выполнением в ней предусмотренных в каждом конкретном эксперименте вычислений.

**МЕТОДЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ**

**МЕТОДЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ**

*Моделирование* является одним из основных инструментов исследования динамических объектов, процессов и явлений в живой и неживой природе.

В общем случае *методы моделирования* ориентированы на решение практических задач с целью получения оптимального решения.

Реализация методов *математического моделирования* должна включать следующие этапы:

1. Формализация исходной проблемы.
2. Построение математической модели.
3. Решение модели.
4. Проверка адекватности модели.
5. Реализация решения.

Альтернативой математическому моделированию сложных систем может служить *имитационное моделирование*как частный случай моделирования. Этот вид моделирования зачастую является наилучшим и единственным способом исследования реальных систем.

Под термином *имитационное моделирование* понимают метод исследования, при котором изучаемая система заменяется моделью, с достаточной точностью описывающей реальную систему, с которой проводятся эксперименты с целью получения информации об этой системе.

Различие между математической и имитационной моделями заключается в том, что в имитационной модели отношение между «входом» и «выходом» модели может быть явно не задано.

Кроме того, всегда существует класс объектов, для которых не разработаны аналитические модели, либо не разработаны методы решения полученной модели. В этом случае аналитическая модель заменяется имитатором или имитационной моделью.

К имитационному моделированию прибегают, когда:

* дорого или невозможно экспериментировать на реальном объекте;
* невозможно построить аналитическую модель поведения системы, но она развивается во времени, есть причинные связи, их последствия, нелинейные и стохастические (случайные) параметры;

необходимо просто имитировать поведение системы во времени для понимания ее сущности.

При имитационном моделировании вместо явного математического описания взаимоотношения между входными и выходными переменными математической модели реальная система, как правило, разбивается на ряд достаточно малых в функциональном отношении элементов или модулей. Затем поведение исходной системы имитируется как поведение совокупности этих элементов, определенным образом связанных в единое целое путем установления соответствующих взаимосвязей между элементами системы. При этом реализация такой модели начинается с входного элемента, далее проходит по всем элементам, пока не будет достигнут выходной элемент модели.

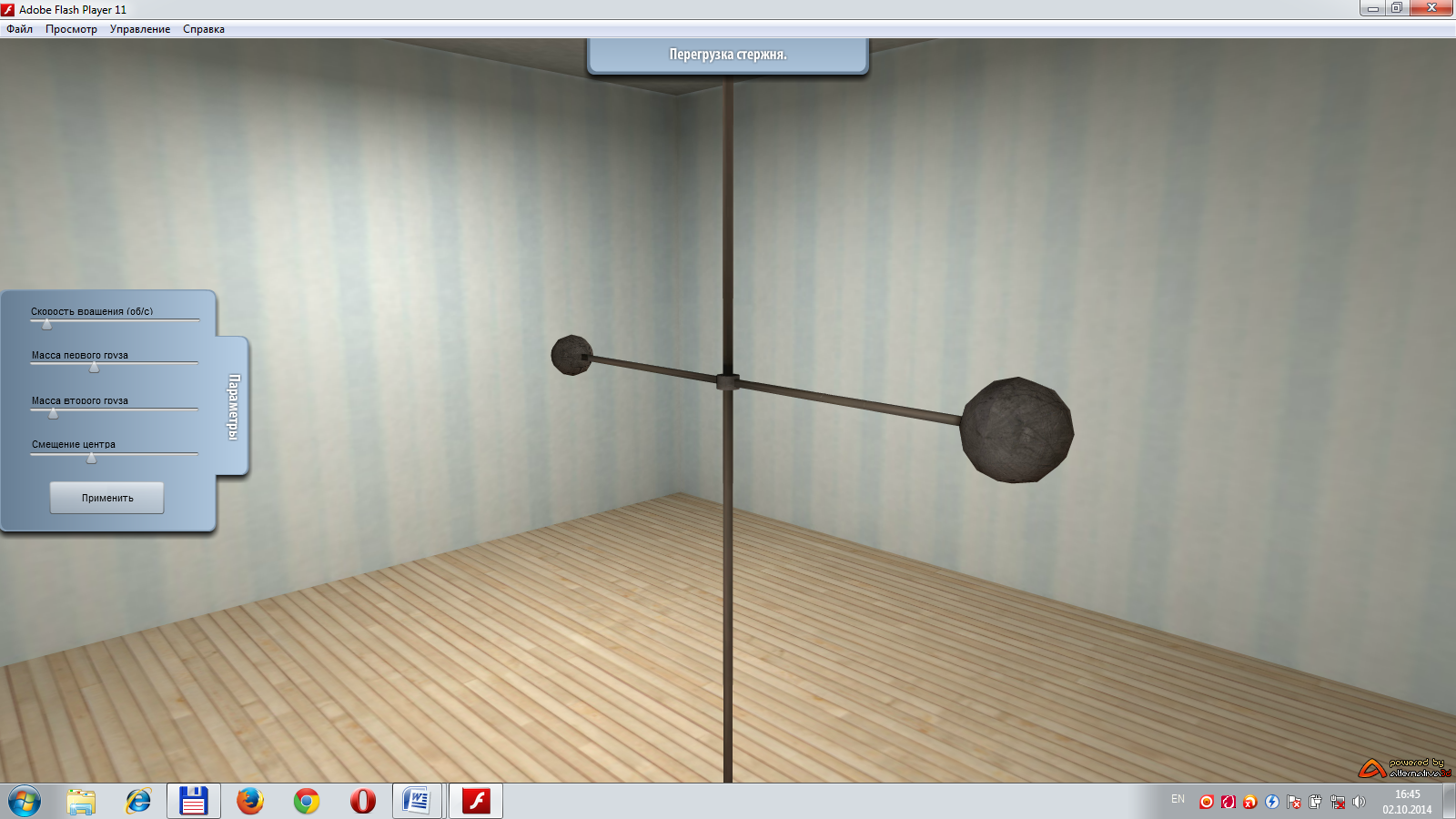
Цель имитационного моделирования состоит в воспроизведении поведения исследуемой системы на основе результатов анализа наиболее существенных взаимосвязей между ее элементами или, другими словами, разработке *симулятора* исследуемой предметной области для проведения различных экспериментов.

Имитационное моделирование позволяет имитировать поведение системы во времени, причем временем в модели можно управлять: замедлять в случае с быстропротекающими процессами и ускорять для моделирования систем с медленной изменчивостью.

Имитационная модель рассматривается как специальная форма математической модели, в которой:

* декомпозиция системы на компоненты производится с учетом структуры проектируемого или изучаемого объекта;
* в качестве законов поведения могут использоваться экспериментальные данные, полученные в результате натурных экспериментов;
* поведение системы во времени иллюстрируется *динамическими графическими образами*.

Одним из главных достоинств систем *визуального моделирования* является то, что они позволяют создавать на компьютере удобную среду, в которой можно создавать виртуальные функционирующие системы и проводить эксперименты с ними. При этом графическая среда становится похожей на физический испытательный стенд, только вместо тяжелых металлических ящиков, кабелей и реальных измерительных приборов, осциллографов и самописцев пользователь имеет дело с их образами на [экране дисплея](file:///C:\Users\ngour\Desktop\КМС%202022%20весна\Лекция%201%20КМСвИД\laba63\laba63.swf).

[](file:///C:\Users\ngour\Desktop\КМС%202022%20весна\Лекция%201%20КМСвИД\laba63\laba63.swf)

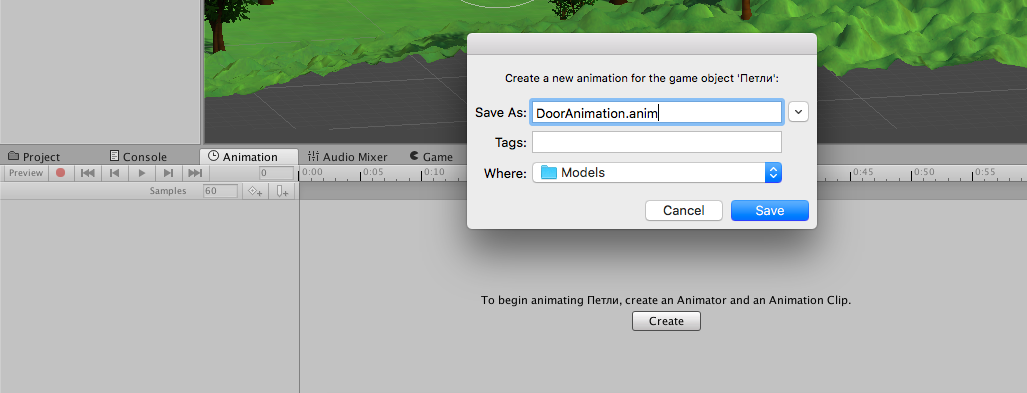
3D-симулятор для изучения явления перегрузки стержня

*Имитационные модели* значительно гибче в представлении реальных систем, чем их математические «конкуренты». При имитационном моделировании исходная система рассматривается на элементном уровне, в то время как математические модели стремятся описать системы на глобальном уровне.

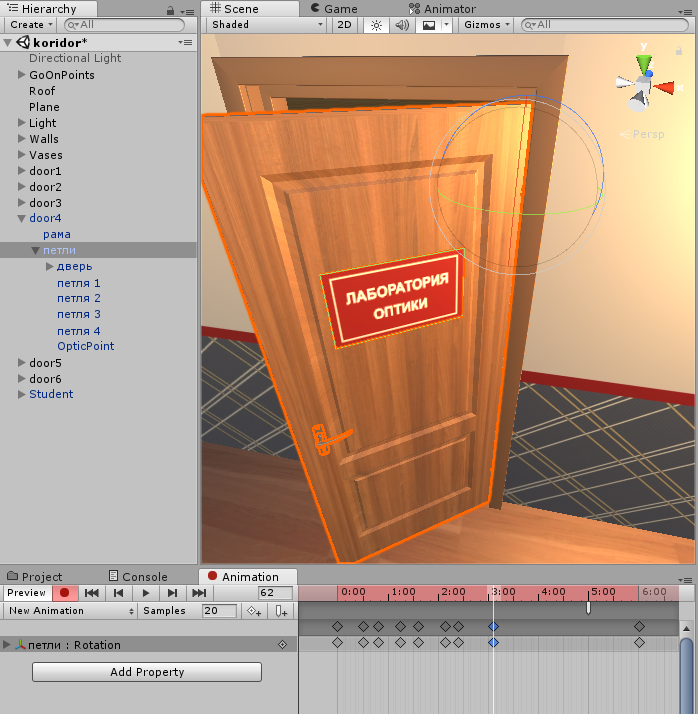
В действительности сложные многоэлементные модели с многообразными связями очень редко могут быть описаны уравнениями. Зачастую зависимости выражаются табличными и логическими соотношениями, правилами «если – то». При этом построенная на их основе *имитационная модель* исследуемого явления или поведения установки точно так же, как и любая вычислимая функция, может быть использована для многовариантных расчетов в процессе разработки любых изделий.

**СОЗДАНИЕ АНИМАЦИИ ОБЪЕКТА В СРЕДЕ UNUTY**

Для создания анимации открытия двери в редакторе **Unity** нужно выбрать объект, который будет вращаться, открыть для него вкладку **Window/Animation** и создать, прежде всего, отдельный файл анимации в открывающемся окне, нажав кнопку **Create**. После сохранения файла состояния анимации двери, например **DoorAnimation.anim,** в соответствующей папке проекта создается непосредственно анимация вращения двери.



При этом необходимо учесть, что, если дверь создана в среде **3dsMAX,** то точка центра вращения такого объекта **Pivot** по умолчанию будет расположена по центру, т. е. дверь будет вращаться через собственный центр, а не на “петлях”, как это бывает обычно. Чтобы дверь вращалась правильно нужно в **Unity** создать пустой объект типа **Empty Object** (невидимый при проигрывании) с именем, например «Петли», который будет выполнять функцию дверных петель, и в окне **Hierarchy** сделать дверь дочерним элементом этого пустого объекта, при этом разместив край двери с «петлями» в центре пустого объекта. Таким образом при вращении такого объекта дверь будет вращаться “правильно”, но в этом случае контроллер нужно добавлять именно на этот пустой объект «Петли» и вращать уже его.

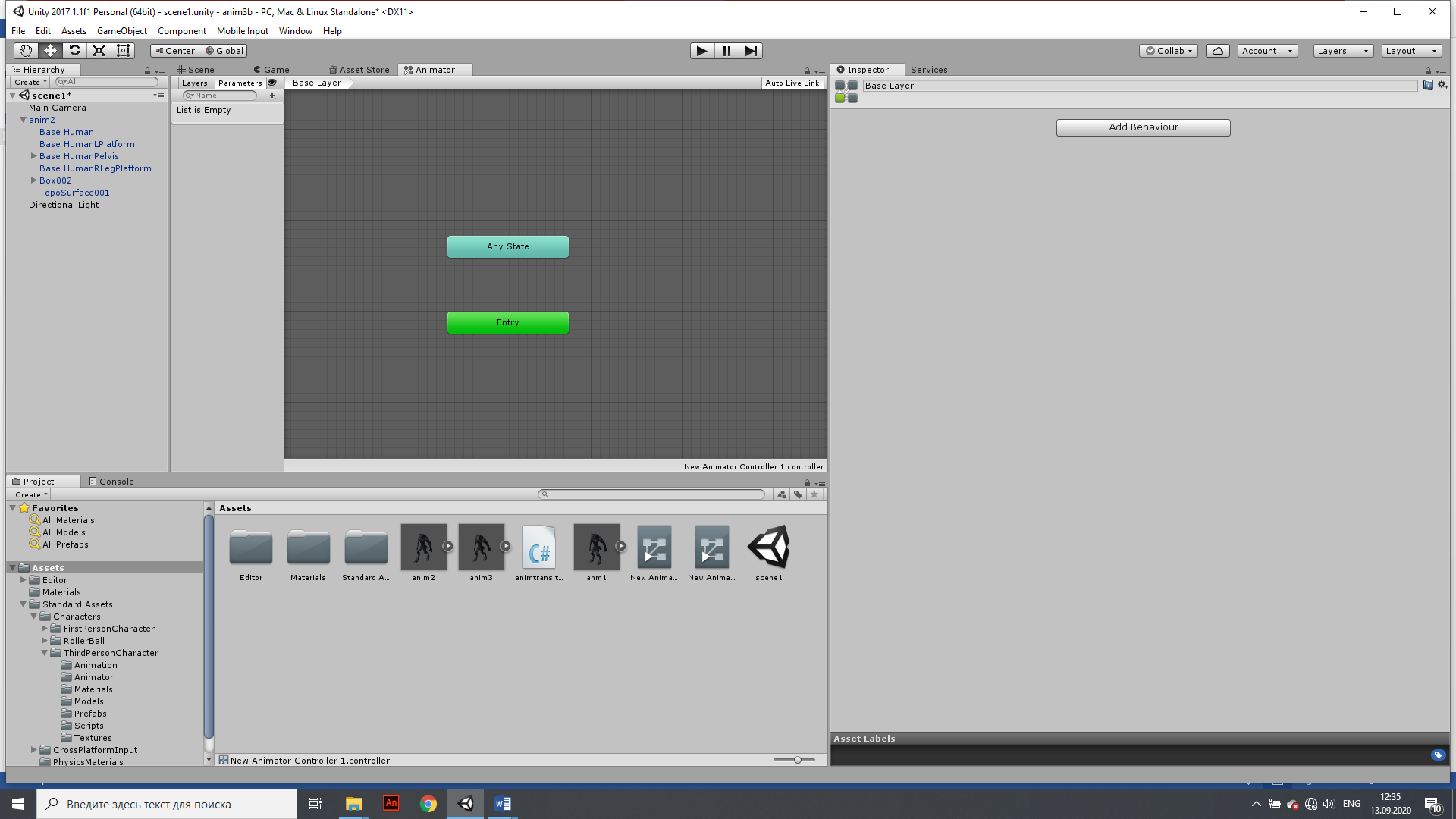


**СОЗДАНИЕ КОНТРОЛЛЕРА АНИМАЦИИ**

Затем добавить к пустому объекту, к которому привязана дверь, контроллер анимации **Animator** и объявить триггерную переменную для открытия двери, по аналогии так же как у персонажа.

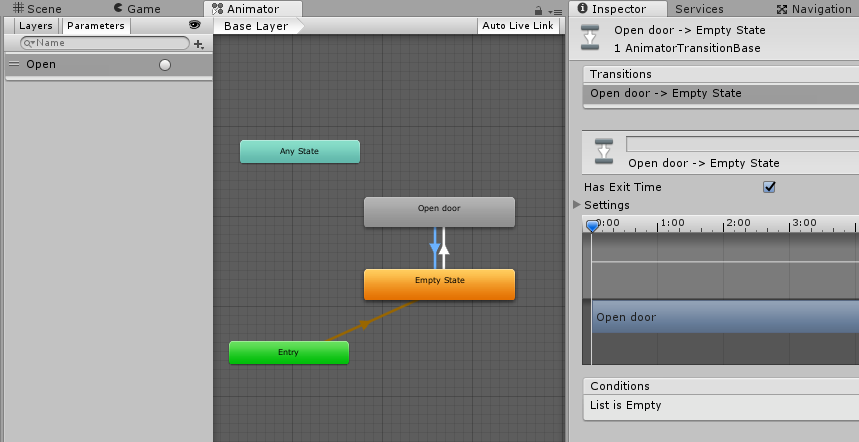
Для создания контроллера анимации нужно кликнуть правой кнопкой мыши по свободной области в папке **Assets**, выполнить команду **Create->Animator Controller** и дать имя контроллеру. Дважды кликнув на созданный контроллер, перейти в окно редактора.

В начале есть два блока, это **Entry**, с которого будут начинать проигрываться анимации и **Any State** – особое состояние, которое всегда существует и позволяет перейти от любого состояния к конкретному, а также блок выхода **Exit**.



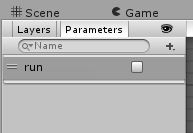
Создать **Empty State** как состояние по умолчанию, щелкнув правой кнопкой мыши в свободной части окна аниматора и выполнив команду **Create State/Empty,** в качестве состояния покоя у двери и добавить созданную анимацию отдельным состоянием.

Переходы к состоянию анимации открытия двери и обратно должен происходить по булевой переменной **Bolean**, причем обратно - по условию установки режима **Has Exit Time**. При этом после создания анимации двери необходимо *снять зацикливание* в контроллере анимации.

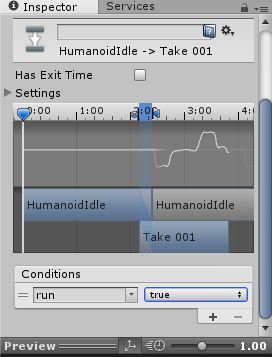


**УПРАВЛЕНИЕ АНИМАЦИЕЙ В UNITY НА ЯЗЫКЕ C#**

Для программного управления анимацией непосредственно в окне редактора необходимо создать программный код на языке C# и, прежде всего, в окне аниматора **создать переменную** открыввкладку **Parameters,** нажать плюс (добавить), выбрать тип создаваемой переменной (для перехода к другой анимации подойдет тип **Bool)**, затем дать имя переменной, например **run**.



После этого выбрать в окне редактора щелчком мыши переход от анимации ожидания, например с именем **HumanoidIdle,** к анимации движения, например с именем **HumanoidRun**. Затем в Инспекторе для перехода в списке **Conditions** и добавляем в него нашу переменную **run** со значением **true**, нажав кнопку **+**.



Нажать правой кнопкой мыши на анимацию ходьбы и выбрать **Make transition**, чтобы создать обратную связь на анимацию по умолчанию. Выбрать новый переход и опять снять галочку **Has exit time**. Затем в список **Conditions** добавляем нашу переменную **run**, но со значением **false**.

**ДОБАВИТЬ СКРИПТ ДЛЯ ПЕРЕКЛЮЧЕНИЯ МЕЖДУ АНИМАЦИЯМИ**

В окне инспектора **Inspector** добавляем **Add Component** для персонажа новый скрипт **New Script** и переходим в редактор кода с шаблоном заготовки.

**Добавляем в шаблон следующий код:**

**Animator anim;** //переменная типа **Animator** для ссылки на анимацию

**void Start(){**

**anim = GetComponent<Animator>();** //контроллера анимации

**}**

**void Update()**

**{**

**If(Input.GetKeyDown(KeyCode.Q))** //если нажата клавиша **q**

**{**

**anim.SetBool(“run”, true);** // переменная, отвечающая за переход имеет значение **true**

**}**

**If(Input.GetKeyDown(KeyCode.W))** //если нажата клавиша **w** отпускается

**{**

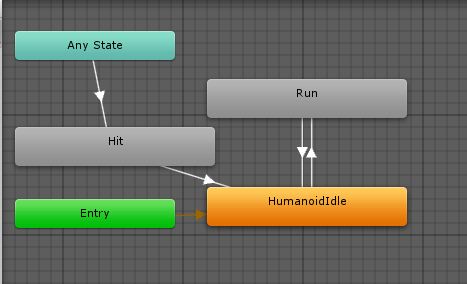
**anim.SetBool(“run”, false);** // переменная, отвечающая за переход имеет значение **false**

**}**

**ПРОИГРАТЬ АНИМАЦИЮ ИЗ ЛЮБОГО СОСТОЯНИЯ**

Чтобы проиграть какую-либо анимацию можно было из любого состояния, необходимо:

1. создать переход от **Any State** к этой анимации, например с именем **Hit**. Для возвращения в состояние ожидания необходимо создать переход от новой анимации **Hit** к анимации по умолчанию **HumanoidIdle** (на рисунке).
2. создать еще одну переменную типа **Trigger** (например **hitten**), которая будет отвечать за щелчок мышью для запуска анимации. Добавим к переходу от **Any State** эту переменную, а для обратного перехода оставим галочку **Has Exit Time**, чтобы переход совершался автоматически (без использования переменной) один раз после проигрывания анимации.

в

1. Добавить в метод **Update()** код проигрывания новой пользовательской анимации по клику мыши:

**if (Input.GetMouseButtonDown(0))  {  anim.SetTrigger("hitten"); }**