Open Dynamics Engine

Руководство пользователя

От составителя

Открытый Динамический Движок (Open Dynamics Engine, или ODE) - это бесплатная библиотека промышленного качества, предназначенная для симуляции динамики составных твердых тел. Она хорошо подходит для транспортных средств, существ с ногами и движущихся объектов в виртуальном пространстве. Она быстра, гибка и проста, имеет встроенное определение столкновений. ODE разрабатывается Расселом Смитом - http://www.q12.org

Данное руководство составлено мной на основе документа "Открытый Динамический Движок. Руководство пользователя v0.5" ("Open Dynamics Engine v0.5 User Guide" by Russell Smith. Copyright © 2001-2004 Russell Smith) в переводе Дениса Татаринцева aka asmzx и оптимизировано под нужды пользователей графического движка Xtreme3D, в состав версии 2 которого входит ODE v0.8.14.0. Все названия функций даются в том виде, в каком они были донесены до конечного пользователя автором Xtreme3D. Список функций ODE из Xtreme3D v2.0.2.0 отличается от списка из Xtreme3D v2.0.0.0. Некоторые структуры элементов и концепции ODE в Xtreme3D отличаются от таковых в официальных дистрибутивах и документации ODE. Настоятельно рекомендуется принять это к сведению во избежание отдельных несовместимостей и "непоняток" в целом.

Удачи!

Тимур Гафаров aka Gecko

Лицензия ODE

Словарь терминов

Этот краткий технический словарь содержит описания некоторых основных специфических терминов концепции ODE. Если вы только начали изучение ODE, рекомендуется обращаться к словарю, только если вы встретили термин, доселе вам незнакомый, или описание функций ODE, из которого состоит основная часть настоящего руководства, на ваш взгляд, содержит недостаточно подробные разъяснения особенностей ODE и его элементов. Суть многих терминов и понятий намного легче уяснить, уже освоившись с частью приведенного материала.
Более подробную информацию о ODE, а также встречающихся понятиях из физики, механики и математики можно найти в сети.

Симуляция (simulation) - виртуальное моделирование механических свойств и процессов, характерных для динамики физических объектов (тел).

Интеграция (integration) - процесс симуляции системы твердых тел во времени. Каждый шаг интеграции увеличивает текущее время на заданный шаг и устанавливает новые значения состояний для всех жестких тел. Существует две главные проблемы при интеграции:
1. Насколько оно точно, а именно, насколько точно поведение имитируемой системы совпадает с тем, что происходит в реальной жизни?
2. Насколько оно стабильно, а именно, как ошибки в вычислениях сказываются на физически некорректном поведении (т.е. вынуждая системы “взрываться” без причины)?
Текущая реализация интеграции в ODE очень стабильна, но не очень точна, даже несмотря на маленький шаг времени. Для большинства случаев это не является проблемой – с точки зрения физики, поведение ODE почти во всех случаях смотрится прекрасно. Тем не менее не стоит использовать ODE для квантовой физики, до тех пор, пока в следующих версиях проблемы с точностью не будут решены.

Шаг времени интеграции (Step) - заданный временной отрезок до следующего обновления состояния элементов системы симуляции (значение delta). Чем меньше шаг, тем выше точность, но ниже стабильность.

Сочленение (joint) - структура, используемая для соединения двух тел. Это такое взаимоотношение между двумя телами, при котором тела могут занимать определенную позицию и ориентацию друг относительно друга. Такое взаимоотношение называется соединением (constraint) – слова сочленение и соединение часто взаимозаменяемы.
Типы сочленений в ODE: шарик-в-разъеме (ball-and-socket), сгибание (hinge), скольжение (slider), сгибание-2 (hinge-2), универсальное (universal), фиксированное (fixed).

Твердое тело (rigid body) - физический объект, имеющий относительно постоянные объем и форму. Твердые тела в ODE разделяются на два типа: динамичные (dynamic) и статичные (static). Динамичные тела имеют массу, могут перемещаться и изменять свою ориентацию в пространстве, к ним могут быть приложены силы, и они могут сталкиваться друг с другом. Статичные тела всегда находятся в одном положении, но могут участвовать в столкновениях с динамичными телами.
Каждое тело имеет локальную систему XYZ-координат, которая перемещается и вращается вместе с телом. Начало этой системы координат совпадает с точкой расположения тела. Некоторые значения в ODE (вектора, матрицы и т.д.) задаются в локальной системе координат тела, а другие в глобальной. Учтите, что *форма* жесткого тела не является динамическим свойством. Она имеет значение только для определения столкновений.

Геометрия (geometry) - фундаментальный объект в системе столкновений. Любая геометрия может столкнутся с другой геометрией и образовать ноль или более точек контакта. Каждая геометрия относится к какому либо классу, такому, как сфера, плоскость или прямоугольный параллелепипед.

Поверхность (surface) - подструктура, определяющая свойства взаимодействия геометрий.

Параметр уменьшения ошибки (Error Reduction Parameter - ERP) - Параметр, контролирующий силу возвращения тел на правильные позиции во время каждого шага для компенации ошибок при сочленении двух тел. Это необходимо, чтобы тела занимали определенное положение и ориентацию друг относительно друга. ERP принимает значение от 0 до 1. ERP определяет пропорцию, в которой ошибка в сочленении будет исправляться в следующем шаге симуляции. Если ERP=0, то корректирующая сила прилагаться не будет, и тела будут перемещаться в соответствии с ходом симуляции. Если ERP=1, то будет предприниматься попытка исправить все ошибки в сочленениях на следующем шаге симуляции. Тем не менее, устанавливать ERP=1 не рекомендуется, поскольку ошибки в сочленении нельзя полностью устранить из-за различных внутренних округлений. Рекомендуется устанавливать значения от 0.1 до 0.8 (0.2 - значение по умолчанию). Глобальное значение ERP воздействует на большинство сочленений симуляции. Тем не менее, некоторые сочленения имеют локальные значения ERP, которые контролируют определенные аспекты их поведения.

Смешивающая сила соединения (Constraint Force Mixing - CFM) - Параметр, контролирующий так называеоме "мягкое" соединение тел. Большинство соединений по своей природе “твердые” (hard). Это значит, что соединение находится в определенных условиях, которые никогда не могут быть нарушены. Например, шарик всегда должен быть в разъеме, а сгибание должно происходит вдоль одной линии. На практике соединения могут быть нарушены непреднамеренным возникновением ошибок в системе, но с помощью параметра уменьшения ошибки можно откорректировать эти ошибки.
Не все соединения жестки. Некоторые “мягкие” (soft) соединения разработаны для того, чтобы быть нарушенными. Например, контактное соединение, которое предотвращает сталкивающиеся объекты от взаимного проникновения, по умолчанию жестки, это выглядит так, как будто сталкивающиеся поверхности сделаны из стали. Но для симуляции материалов помягче можно сделать мягкое соединение, тем самым позволив при взаимодействии двух объектов иметь место естественному проникновению.
Если CFM установлен в ноль, соединение будет жестким. Если в CFM установлено положительное число, то появляется возможность нарушать соединение “смещением”. Другими словами, соединение становится мягким, и мягкость будет нарастать с увеличением CFM. Происходит здесь следующее: соединению позволяется быть нарушенным пропорционально CFM раз, восстанавливая силу, которая нужна для удержания соединения. Учтите, что установка в CFM отрицательного значения приведет к непредсказуемым отрицательным последствиям, например к нестабильности. Не делайте этого.

Аккумулятор силы (force accumulator) - совокупность сил, приложенных к телу. Между каждым шагом интегрирования пользователь может вызывать функции приложения сил к жесткому телу. Эти силы накапливаются в аккумуляторе силы твердого тела. Когда происходит следующий шаг интегрирования, сумма всех приложенных сил вызывает перемещение тела. Аккумулятор силы устанавливается в ноль после каждого шага интегрирования.

Функции ODE

Ниже приведен полный список доступных в Xtreme3D функций ODE с подробным описанием. Некоторые понятия могут потребовать отдельного разъяснения, поэтому иногда даются более развернутые описания принципа действия той или иной функции. Часть сведений собрана в словаре терминов (см. стр. 3).
Почти все функции принимают и возвращают действительные значения (real). Только некоторые функции принимают в качестве аргументов логические значения (boolean), то есть 0 или 1 (true и false). Упоминаемые константы (встроенные постоянные значения) в некоторых исходниках GM6 могут отсутствовать. Названия функций, присутствующих в обеих версиях Xtreme3D V2 (2.0.0.0 и 2.0.2.0) выделены красным цветом, присутствующих только в Xtreme3D v2.0.2.0 - фиолетовым, типы значений - синим, встроенные константы - бордовым.

К некоторым функциям на данный момент нет описаний. В частности, это следующие функции:
OdeManagerSetMaxContacts(max);
OdeManagerGetNumContactJoints();
OdeDynamicCalibrateCenterOfMass(obj);
OdeDynamicGetContact(obj,index);
OdeStaticGetContact(obj,index);
OdeJointInitialize(joint);

Менеджер ODE

real = OdeManagerCreate();
Создает новый менеджер.

real = OdeManagerDestroy();
Уничтожает менеджер и все что в нем находится.

real = OdeManagerSetSolver(solver as real);
Устанавливает метод интеграции. Доступны следующие значения:
osmStep = 0
osmStepFast = 1
osmQuickStep = 2
osmStep: используется метод "большой" матрицы (big matrix), который требует времени расчета порядка m^3 и памяти порядка m^2, где m - общее количество строк матрицы. Для больших систем требуется много памяти, скорость работы будет низка, но на текущий момент это самый точный метод.
osmStepFast: для больших систем osmStep может потребовать много памяти и времени для расчета. osmStepFast обеспечивает альтернативный путь решения этой проблемы, принося в жертву точность. osmStepFast не зависит от количества итераций в одном шаге. Используйте этот метод, когда у вас имеется немного параметров, влияющих на стабильность, и вы хотите использовать преимущества скорости или расхода памяти. Другой совет использования osmStepFast - это сразу ориентироваться на этот метод, если вы знаете что будете строить большие миры с большим количеством физических объектов. Последний совет по использованию osmStepFast - это использовать его только там где это необходимо. Поскольку osmStepFast использует такие же структуры тела и мира как и osmStep, то можно переключатся между ними.
osmQuickStep: используется метод итераций, который требует времени порядка m*N и памяти порядка m, где m - общее количество строк матрицы и N - количество итераций. Для больших систем это работает намного быстрее чем osmStep, но менее точно. osmQuickStep отлично подходит для большого количества объектов, особенно вместе с авто-выключением (см. ниже). Тем не менее, имеет не очень хорошую точность для неустойчивых систем (near-singular). Система может становиться неустойчивой, когда используются контакты с сильным трением, двигатели и определенные составные структуры. Например, робот с несколькими ногами, сидящий на земле. Увеличение итераций в osmQuickStep может немного помочь, но не сильно, если система неустойчива.

real = OdeManagerSetIterations(it as real);
Устанавливает и получает количество итераций, которые использует метод osmQuickStep на каждом шаге расчета. Значение по умолчанию 20 итераций.

real = OdeManagerStep(delta as real);
Совершает шаг интеграции размером в delta.
Используется метод, установленный функцией OdeManagerSetSolver.

real = OdeManagerSetGravity(x,y,z as real);
Устанавливает вектор гравитации менеджера. Вектор гравитации Земли будет (0,-0.981,0), предполагая, что ось +Y направлена вверх. По умолчанию гравитации нет, т.е. (0,0,0).

real = OdeManagerSetMaxContacts(max as real);
<Сведения отсутствуют>

real = OdeManagerSetVisible(mode as boolean);
Переключает режим видимости геометрии тел. По умолчанию видимость выключена.

real = OdeManagerSetGeomColor(color as real);
Устанавливает цвет отображения линий геометрии тел. По умолчанию цвет красный (RGB 255,0,0).

real = OdeManagerGetNumContactJoints();
<Сведения отсутствуют>

Параметры мира ODE
(только в Xtreme3D v2.0.2.0)

real = OdeWorldSetAutoDisableFlag(flag as boolean);
real = OdeWorldSetAutoDisableLinearThreshold(velocity as real);
real = OdeWorldSetAutoDisableAngularThreshold(velocity as real);
real = OdeWorldSetAutoDisableSteps(steps as real);
real = OdeWorldSetAutoDisableTime(time as real);
Устанавливают параметры авто-выключения (auto-disable) для всех тел. Это требует пояснения.
Каждое тело может быть включено (enabled) или выключено (disabled). Включенные тела участвуют в симуляции, в то время как выключенные тела деактивированы и их состояния не обновляются во время шагов симуляции (simulation step). Новые тела всегда создаются во включенном состоянии. Выключенные тела, которые соединены сочленениями (joint) с включенными телами, автоматически включаются на следующем шаге симуляции.
Выключенные тела не тратят время процессора, поэтому неподвижные тела надо отключать. Это может быть сделано автоматически при помощи возможностей авто-выключения (auto-disable).
Если флаг авто-выключения установлен, то тела будут автоматически выключаться, когда:
1. Они будут бездействовать заданное количество шагов симуляции.
2. Они будут бездействовать заданное время симуляции.
Тело считается бездействующим если величина линейной и угловых скоростей меньше заданного порога.
Таким образом, каждое тело имеет пять параметров авто-выключения: флаг включения (flag), количество шагов бездействия (steps), время бездействия (time), пороги линейной (linear threshold) и угловой (angular threshold) скорости. Только что созданные тела берут эти параметры из менеджера.

Параметры по умолчанию:
AutoDisableFlag=0
AutoDisableLinearThreshold=0.01
AutoDisableAngularThreshold=0.01
AutoDisableSteps=10
AutoDisableTime=0

Функции твердых тел

Основные функции динамичного тела:

real = OdeDynamicCreate(obj as real);
Создает динамическое тело и возвращает его ID. Объект obj выступает в роли объекта привязки - то есть, для того, чтобы тело было видимо, необходимо создать соответствующий объект Xtreme3D. Другими словами, функция создает физическую модель в ODE для объекта Xtreme3D.

real = OdeDynamicAlign(object, obj as real);
Применяет к уже созданному телу объект Xtreme3D, как описано выше.

real = OdeDynamicEnable(obj,mode as boolean);
Включает или выключает тело. Включенные тела участвуют в симуляции, в то время как выключенные тела деактивированы и их состояния не обновляются. Новые тела всегда создаются во включенном состоянии. Выключенные тела, которые соединены сочленениями с включенными телами, автоматически включаются на следующем шаге симуляции. Поэтому, если требуется выключить такие тела, необходимо выключить и их сочленения.

real = OdeDynamicCalibrateCenterOfMass(obj as real);
<Сведения отсутствуют>

Функции приложения сил:

real = OdeDynamicAddForce(obj,x,y,z as real);
real = OdeDynamicAddForceAtPos(obj,x,y,z,px,py,pz as real);
real = OdeDynamicAddForceAtRelPos(obj,x,y,z,px,py,pz as real);
real = OdeDynamicAddRelForce(obj,x,y,z as real);
real = OdeDynamicAddRelForceAtPos(obj,x,y,z,px,py,pz as real);
real = OdeDynamicAddRelForceAtRelPos(obj,x,y,z,px,py,pz as real);
real = OdeDynamicAddTorque(obj,x,y,z as real);
real = OdeDynamicAddRelTorque(obj,x,y,z as real);
Прикладывают силы к телам (в абсолютных или относительных координатах). Каждое тело накапливает приложенные к нему силы в аккумуляторе силы (force accumulators), после каждого шага времени (time step) накопленные силы обнуляются.
Функции ...Force... прилагают силы к определенной точке тела, а функции ...Torque... - вокруг оси вращения тела (так называемый вращающий момент).
Функции ...RelForce и ...RelTorque принимают в качестве параметров вектора в локальной системе координат этого тела.
Функции ...ForceAtPos и ...ForceAtRelPos принимают вектор с позицией точки приложения силы (в глобальной или локальной системе координат соответственно). Все другие функции прилагают силу к центру масс.

Функции контакта:

real = OdeDynamicGetContactCount(obj as real);
Возвращает количество точек контакта тела, определенных проверкой столкновения.

real = OdeDynamicGetContact(obj,index as real);
<Сведения отсутствуют>

Функции авто-выключения (только в Xtreme3D v2.0.2.0):

real = OdeDynamicSetAutoDisableFlag(obj,flag as boolean);
Устанавливает флаг авто-выключения тела.

real = OdeDynamicSetAutoDisableLinearThreshold(obj,velocity as real);
Устанавливает порог линейной скорости для автоматического отключения. Величина линейной скорости тела должна быть ниже заданного порога, чтобы тело считалось бездействующим.

real = OdeDynamicSetAutoDisableAngularThreshold(obj,flag as real);
Устанавливает порог угловой скорости для автоматического отключения. Величина угловой скорости тела должна быть ниже заданного порога, чтобы тело считалось бездействующим.

real = OdeDynamicSetAutoDisableSteps(obj,steps as real);
Устанавливает количество шагов симуляции, в течение которых тело должно бездействовать, чтобы быть автоматически отключенным. Установка этого значения в ноль исключает шаги из рассмотрения.

real = OdeDynamicSetAutoDisableTime(obj,time as real);
Устанавливает время симуляции в течении которого тело должно бездействовать чтобы быть автоматически отключенным. Установка этого значения в ноль исключает время симуляции из рассмотрения.

Функции статичного тела:

real = OdeStaticCreate(obj as real);
Создает статичное тело и возвращает его ID. Объект obj выступает в роли объекта привязки - то есть, для того, чтобы тело было видимо, необходимо создать соответствующий объект Xtreme3D. Другими словами, функция создает физическую модель в ODE для объекта Xtreme3D.

real = OdeStaticGetContactCount(obj as real);
Возвращает количество точек контакта тела, определенных проверкой столкновения.

real = OdeStaticGetContact(obj,index as real);
<Сведения отсутствуют>

Геометрия

real = OdeAddSphere(obj,r as real);
Создает геометрию сфера (sphere) заданного радиуса (r) и возвращает его ID. Точкой расположения является центр сферы.

real = OdeAddBox(obj,w,h,d as real);
Создает геометрию прямоугольный параллелепипед (box) с длинами сторон (w,h,d) и возвращает его ID. Точкой расположения является центр прямоугольного параллелепипеда.

real = OdeAddPlane(obj as real);
Создает геометрию плоскость (plane) и возвращает ее ID. Плоскостm - не перемещаемая геометрия. Это значит, что у плоскости нет позиции и вращения. Другими словами, предполагается, что плоскость является частью статичного тела и не связана с каким либо подвижным объектом.

real = OdeAddCone(obj,l,r as real);
Создает геометрию конус (cone) с заданными длиной (l) и радиусом (r) и возвращает его ID. Точкой расположения является центр конуса.

real = OdeAddCylinder(obj,l,r as real);
Создает геометрию цилиндр (cylinder) с заданными длиной (l) и радиусом (r) и возвращает его ID. Предполагается, что ось цилиндра располагается вдоль локальной оси геометрии Z.

real = OdeAddCapsule(obj,l,r as real);
Создает геометрию цилиндр с верхушкой, или капсула (capped cylinder - capsule) с заданными длиной (l) и радиусом (r) и возвращает его ID. Цилиндр с верхушкой похож на обычный цилиндр, за исключением того, что на концах цилиндра расположено по полусфере. Эта возможность делает внутренний код определения столкновения более точным и быстрым. В длину цилиндра верхушки не входят. Предполагается, что ось цилиндра располагается вдоль локальной оси геометрии Z. Радиус верхушек равен радиусу самого цилиндра и задается с помощью параметра r.

real = OdeAddTriMesh(obj as real);
Создает геометрию набор треугольников (triangle mesh - TriMesh). Система столкновений набора треугольников имеет следующие возможности:
- Может быть представлен любой “суп” из треугольников – т.е. треугольники не обязательно должны представлять собой ленту (strip), веер (fan) или сетку (grid).
- Работает хорошо для относительно больших треугольников.
- Используется временная связность (temporal coherence) для ускорения проверки столкновений. Когда происходит проверка столкновения геометрии с одним TriMesh’ем, данные располагаются внутри Tri-Mesh’a.

Поверхность

real = OdeSurfaceEnableRollingFrictionCoeff(obj,mode as boolean);
Включает или выключает трение качения для поверхности. Если включено, коэффициент трения должен быть установлен функцией OdeSurfaceSetRollingFrictionCoeff.

real = OdeSurfaceSetRollingFrictionCoeff(obj,rfc as real);
Устанавливает коэффициент трения качения для поверхности. Значение должно лежать в промежутке 0..1. 0 означает отсутствие трения (тела будут катиться бесконечно), 1 - максимальное трение (тела не катятся).

real = OdeSurfaceSetMode(obj,Mu2,FDir1,Bounce,SoftERP,SoftCFM,Motion1,Motion2,Slip1,Slip2 as boolean);
Устанавливает флаги поверхности. Это комбинация одного или более следующих флагов:
Mu2 - если не установлен, то используется mu для обоих направлений трения. Если установлен, то mu используется для первого направления трения, а mu2 для направления трения 2.
FDir1 - если установлен, то FDir1 берется направлением трения 1, в противном случае FDir1 рассчитывается автоматически как перпендикуляр к нормали контакта (в этом случае результирующие направление не предсказуемо).
Bounce - если установлен, то поверхность считается упругой; другими словами, тела будут пружинить друг от друга. Точное значение упругости определяется функцией OdeSurfaceSetBounce.
SoftERP - если установлен, то параметр уменьшения ошибки нормали контакта контролируется функцией OdeSurfaceSetSoftERP. Этот нужно, чтобы сделать поверхность мягче.
SoftCFM - если установлен, то смешивающая сила соединения нормали контакта контролируется функцией OdeSurfaceSetSoftCFM. Этот нужно, чтобы сделать поверхность мягче.
Motion1 - если установлен, то предполагается, что поверхность движется независимо от тел. Если этот флаг установлен, то Motion1 определяет скорость поверхности в направлении трения 1.
Motion2 - то же что и выше, но в направлении трения 2.
Slip1 - скольжение, зависящее от силы (FDS) в направлении трения 1.
Slip2 - скольжение, зависящие от силы (FDS) в направлении трения 2.
Вектор направления трения 2 (второго направления трения) вычисляется как перпендикуляр к нормали и FDir1.
Направление трения 1 - это вектор первого направления трения, который определяет направление, вдоль которого должна быть приложена сила трения. Он должен быть в единицах масштаба (unit length) и перпендикулярен нормали контакта (т.е. быть касательной к поверхности контакта). Он должен быть определен только в том случае, если установлен флаг FDir1.

real = OdeSurfaceSetMu(obj,mu as real);
Устанавливает mu - коэффициент трения Coulomb. Значение 0 дает отсутствие трения в поверхности. Учтите, что отсутствие трения в контакте требует гораздо меньше времени на вычисления. Это значение всегда должно быть определено.

real = OdeSurfaceSetMu2(obj,mu as real);
Устанавливает mu2 - необязательный коэффициент трения Coulomb для направления трения 2. Должен быть определен только в случае установки соответствующего флага в OdeSurfaceSetMode.

real = OdeSurfaceSetBounce(obj,b as real);
Устанавливает параметр восстановления состояния, или упругости (restitution parameter). Значение 0 значит, что поверхность абсолютно не упруга, 1 - максимальная упругость. Должен быть определен только в случае установки соответствующего флага в OdeSurfaceSetMode.

real = OdeSurfaceSetBounceVel(obj,b as real);
Устанавливает минимальную скорость, необходимую для упругости (в м/с). Скорость ниже этого порога определяет параметр упругости равным 0. Должен быть определен только в случае установки соответствующего флага в OdeSurfaceSetMode.

real = OdeSurfaceSetSoftERP(obj,erp as real);
Устанавливает параметр “мягкости” нормали контакта. Должен быть определен только в случае установки соответствующего флага в OdeSurfaceSetMode.

real = OdeSurfaceSetSoftCFM(obj,cfm as real);
Устанавливает параметр “мягкости” нормали контакта. Должен быть определен только в случае установки соответствующего флага в OdeSurfaceSetMode.

real = OdeSurfaceSetMotion1(obj,m as real);
real = OdeSurfaceSetMotion2(obj,m as real);
Устанавливают скорости поверхности в направлении 1 и 2 (в м/с). Должны быть определены только в случае установки соответствующих флагов в OdeSurfaceSetMode.

real = OdeSurfaceSetSlip1(obj,s as real);
real = OdeSurfaceSetSlip2(obj,s as real);
Устанавливают коэффициенты скольжения, зависящего от силы (FDS) для направлений трения 1 и 2. Должны быть определены только в случае установки соответствующих флагов в OdeSurfaceSetMode.
FDS - это эффект, который вынуждает контактирующие поверхности скользить относительно друг друга со скоростью, пропорциональной силе, приложенной по касательной к поверхности.
Учтите, что FDS не имеет отношения к эффекту трения Coulomb – оба этих режима могут быть использованы одновременно в одной точке контакта.

Сочленения

Типы сочленений:
real = OdeAddJointBall();
real = OdeAddJointHinge();
real = OdeAddJointSlider();
real = OdeAddJointUniversal();
real = OdeAddJointHinge2();
real = OdeAddJointFixed();
Создают новые сочленения заданного типа (шарик-в-разъеме, сгибание, скольжение, универсальное, сгибание-2, фиксированное) и возвращают их ID. В начале сочленение находится в неопределенном состоянии (т.е. не влияет на симуляцию), потому что не присоединено ни к какому телу.

- Сочленение шарик-в-разъеме (ball-and-socket) позволяет телам свободо вращаться вокруг точки соединения (anchor). “Шарик” одного тела совпадает с расположением “разъема” другого тела. Это больше всего напоминает шарнирное соединение.
- Сочленение сгибание (hinge) позволяет телам сгибаться только в одном месте по оси сгиба. Напоминает соединение в дверных петлях.
- Сочленение скольжение (slider) не позволяет телам менять ориентацию относительно друг друга, разрешая только движение тел вдоль одной линии. Похоже на структуру поршня.
- Универсальное сочленение (universal) имеет более сложную структуру. Оно похоже на сочленение шарик-в-разъеме, у которого ограничена степень свободы вращения. Определив ось 1 для тела 1 и ось 2 для тела 2 перпендикулярно друг другу, их перпендикулярность будет сохраняться. Другими словами, можно сказать, что тела будут сохранять перпендикулярную ориентацию относительно своих осей. Универсальные сочленения проявляют себя в автомобилях, а именно при соединении двигателя с валом, вращающимся вдоль оси машины.
- Сочленение сгибание-2 (hinge-2) похоже на два сочленения, соединенных последовательно, с разными осями сгибания. Например, это может быть колесо машины, где одна ось позволяет колесу поворачиваться а вторая вращаться. Сочленение сгибание-2 имеет точку соединения (anchor) и две оси сгибания. Ось 1 определяется относительно тела 1 (может быть осью поворота, если тело 1 - рама). Ось 2 определяется относительно тела 2 (может быть осью вращения колеса, если тело 2 - колесо).
- Фиксированное сочленение (fixed) сохраняет фиксированную позицию и ориентацию тел, или тела и статического окружения. Использование такого сочленения не является хорошей идеей, за исключением случаев отладки. Если вам надо использовать два тела, склеенных вместе, то лучше представить их как одно тело.

Основные функции сочленения:

real = OdeJointSetObjects(joint,obj1,obj2 as real);
Присоединяет сочленение к телам. Если сочленение уже куда то присоединено, то сначала оно будет отсоединено от старых тел. Для того чтобы присоединить сочленение к одному телу надо установить obj1 или obj2 в ноль - ноль значит статическое окружение. Установка обоих этих параметров в ноль переведет сочленение в неопределенное состояние и оно не будет принимать участия в симуляции.
Чтобы работать, некоторые сочленения, такие как сгибание-2 (hinge-2), обязательно должны быть присоединены к двум телам.

real = OdeJointEnable(joint as boolean);
Включает или выключает сочленение. Включенные сочленения участвуют в симуляции, в то время как выключенные - деактивированы и их состояния не обновляются. Новые сочленения всегда создаются во включенном состоянии.

real = OdeJointInitialize(joint as real);
<Сведения отсутствуют>

real = OdeJointSetAnchor(joint,x,y,z as real);
Устанавливает точку соединения, или якорь (anchor) сочленения. Сочленение будет пытаться удержать два тела относительно этой точки. Входное значение дается в абсолютных координатах.

real = OdeJointSetAnchorAtObject(joint,obj as real);
<Сведения отсутствуют>

real = OdeJointSetAxis1(joint,x,y,z as real);
Устанавливает вектор первой оси сочленения (axis-1). Функция может быть применена к следующим типам сочленений: сгибание (hinge), скольжение (slider), универсальное (universal), сгибание-2 (hinge-2).

real = OdeJointSetAxis2(joint,x,y,z as real);
Устанавливает вектор второй оси сочленения (axis-2). Функция может быть применена к следующим типам сочленений: универсальное (universal), сгибание-2 (hinge-2).

Двигатели и остановки:

Когда сочленение впервые создано, ничто не препятствует его движению в любом направлении. Например, сгибание может происходить на любой угол, а скольжение на любую длину.
Этот диапазон движений может быть ограничен установкой остановок (stops) в сочленении. Угол сочленения (или позиция) будет предохранен от достижения меньше заданного минимального значения или превышения заданного верхнего значения. Учтите, что угол сочленения (или позиция) считается нулевым в начальной позиции тел.
Так же как и остановки, многие типы сочленений могут иметь двигатели(motors). Двигатели прилаживают вращающий момент (или силу) к сочленениям для достижения желаемой скорости в точке вращения (или скольжения). Двигатели имеют ограничения силы, это значит, что они могут прилагать не более заданного значения силы/вращающего момента в сочленении.
Двигатели имеют два параметра: желаемая скорость и максимальная сила, с помощью которой может быть достигнута эта скорость. Это очень упрощенная модель двигателя из реальной жизни. Тем не менее, этого достаточно для моделирования двигателя, управляемого коробкой передач, подключаемого к сочленению. Такие устройства часто контролируют установку желаемой скорости и могут генерировать только максимальное количество мощности для достижения этой скорости (сообщая сочленению определенное количество силы).

real = OdeJointSetBounce(joint,axis,val as real);
Устанавливает упругость остановок. Параметр восстановления лежит в диапазоне 0..1. 0 значит, что остановка совсем не упруга, 1 - максимальная упругость.

real = OdeJointSetCFM(joint,axis,val as real);
Устанавливает значение смешивающей силы соединения (CFM), используется, когда нет остановки.

real = OdeJointSetFMax(joint,axis,val as real);
Максимальная сила или вращающий момент, который двигатель будет использовать для достижения желаемой скорости. Это значение всегда должно быть больше или равно нулю. Установка этого значения в ноль (значение по умолчанию) выключает двигатель.

real = OdeJointSetFudgeFactor(joint,axis,val as real);
В текущей реализации остановка/двигатель есть маленькая проблема: когда сочленение имеет остановку и двигатель, который пытается не допустить остановки, слишком большая сила может быть приложена за один шаг времени, вызывая “скачки” движения. Для масштабирования избыточной силы используется надстроечный показатель (fudge factor). Значение должно лежать между нулем и единицей (значение по умолчанию). Если в сочленении видны скачки движения, значение должно быть уменьшено. Делая это значение меньше, можно уберечь двигатель от движения сочленения из положения остановки.

real = OdeJointSetHiStop(joint,axis,val as real);
Устанавливает верхний предел угла или позиции. Для вращающихся сочленений это значение должно быть меньше числа pi, чтобы был эффект. Если верхний предел меньше нижнего предела, то оба предела не дают никакого эффекта.

real = OdeJointSetLoStop(joint,axis,val as real);
Устанавливает нижний предел угла или позиции. Для вращающихся сочленений, это значение должно быть больше числа pi, чтобы был эффект.

real = OdeJointSetStopCFM(joint,axis,val as real);
Устанавливает значение смешивающей силы соединения (CFM), используемой остановками. Вместе со значением ERP может быть использовано для получения более мягких остановок. Учтите, предполагается что сочленения не достигают лимита силы, сочленения, достигшие лимита силы, будут работать некорректно.

real = OdeJointSetStopERP(joint,axis,val as real);
Устанавливает параметр уменьшения ошибки (ERP), используемый остановками.

real = OdeJointSetVel(joint,axis,val as real);
Устанавливает желаемую скорость двигателя (здесь должна быть угловая или линейная скорость).