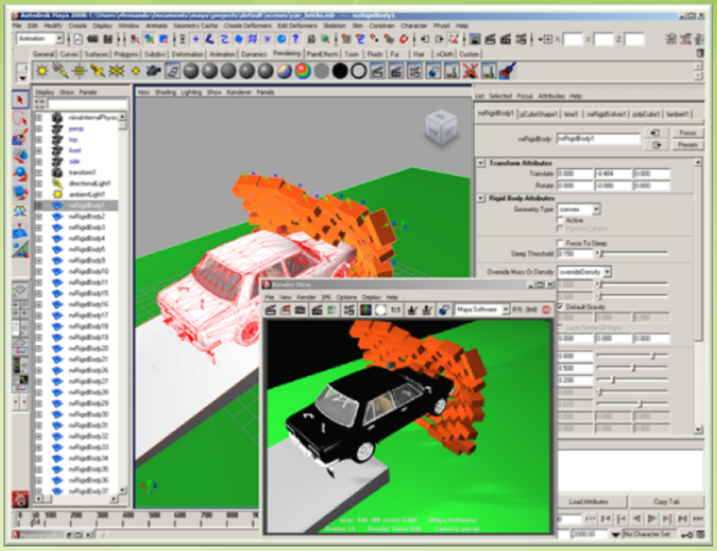
User's Guide



PhysX Plug-In for Maya

October 2008

Table of Contents

Introduction

The PhysX plug-in for Autodesk Maya allows you to conveniently work with rigid body dynamics, constraints, rag dolls, and cloth within Maya. In addition to prototyping physics simulations, you can export physical scenes for use in your game engine.

Design Philosophy

To minimize the learning effort, the plug-in was designed to behave as closely as possible to Maya's native rigid dynamics. For example, the naming and design of the nodes and attributes were kept similar to Maya's native, with the "nx" prefix generally added to node names and scripts.

Still, PhysX has many features and a few limitations that are not matched by Maya's native dynamics. For example, the plug-in's rigid constraint is based on the PhysX D6 joint (rigid constraint with 6 parameterizable degrees of freedom), which is much more powerful and complex than Maya's native rigid constraint model. Therefore, the attribute editor template for a rigid constraint is quite different. Future versions of the plug-in will simplify the user interface.

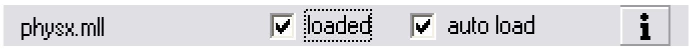
Supported Features at a Glance

The plug-in supports the following major features:

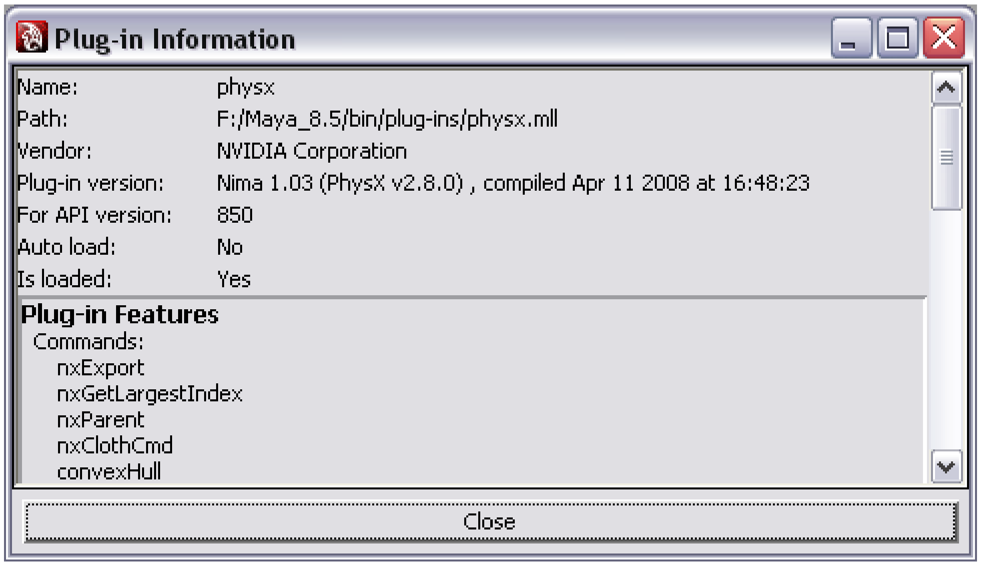
- **Rigid bodies** (implemented through NxActor in the PhysX API), containing one or more convex or non-convex meshes or physics shapes (i.e., capsules, boxes, and spheres).
- **Mesh collision shapes** can be added or removed interactively, and they can be positioned and oriented interactively using the regular Maya manipulators.
- **Physics shapes** (e.g. capsules) have a special manipulator to position the end-points, adjust the radius, etc.
- **Rigid constraints** can be created to link rigid bodies together. Almost all features of the PhysX D6 joint (NxD6Joint) are exposed through this interface (e.g. breakable force and torque, local position/orientation, etc.). Some support for motors is also provided.
- **Animation.** Almost every parameter of rigid bodies and can be keyframed and manipulated interactively, including in *interactive playback* mode. Shapes can be sculpted and the corresponding convex or non-convex collision mesh will be automatically updated.
- **Force fields** are partially supported.
- **Ragdolls** can automatically be created from skinned characters. (*Note: due to an outstanding parentConstraint bug, the resulting skinned character may not display properly.*)

Configuring the Plug-In

To run the plug-in, load the plug-in manager from Maya's menu, through Window→Settings/Preferences→Plug-in manager. Load the "physx.mll" plug-in.



To verify that you have the expected version of the plug-in loaded, you can press the **i** icon and check the version information, as shown:


Plug-in Information

Name:	physx
Path:	F:/Maya_8.5/bin/plug-ins/physx.mll
Vendor:	NVIDIA Corporation
Plug-in version:	Nima 1.03 (PhysX v2.8.0) , compiled Apr 11 2008 at 16:48:23
For API version:	850
Auto load:	No
Is loaded:	Yes

Plug-in Features

Commands:

```
nxExport
nxGetLargestIndex
nxParent
nxClothCmd
convexHull
```

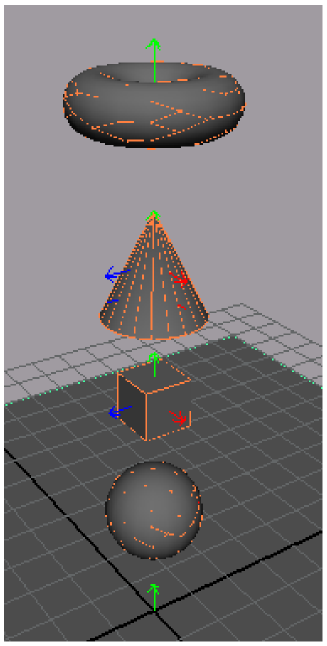
If the plug-in has been loaded properly, a "PhysX" menu should appear in Maya, next to the Help menu set. (Note that the menu may look slightly different depending on the version.) If it doesn't show up, it's likely because the latest PhysX runtime is not properly installed.

Testing the Plug-in

After installing the plug-in, you may want to do some quick tests to ensure that the PhysX.mll plug-in is loaded properly. You can run the following tests by entering their name in the MEL command line.

nxTest1

This test creates some shapes that will bounce on each other and on a box at the bottom. Pressing the "Play animation" button will cause the four central objects to fall down and bounce on each other.



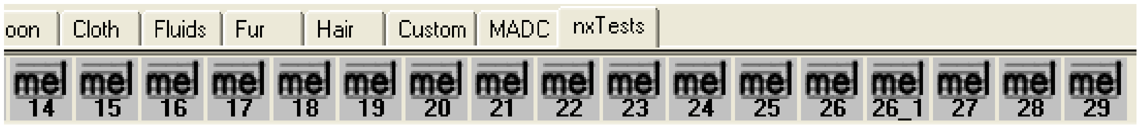
nxTest2

Similar to test 1, with keyframed kinematic ground box.

nxTest3

Similar to test2, except that the ground box and torus are grouped, and the group has some keyframed translation and rotation.

There are dozens of additional tests. To access them easily, copy the shelf_nxTests.mel file in your shelf folder, typically located at C:\Documents and Settings*User Name*\My Documents\maya\Version\prefs\shelves.



Sample Scenes

The plug-in also comes with several sample scenes, which are described in a later chapter.

Using the Plug-In

Rigid Body Concepts

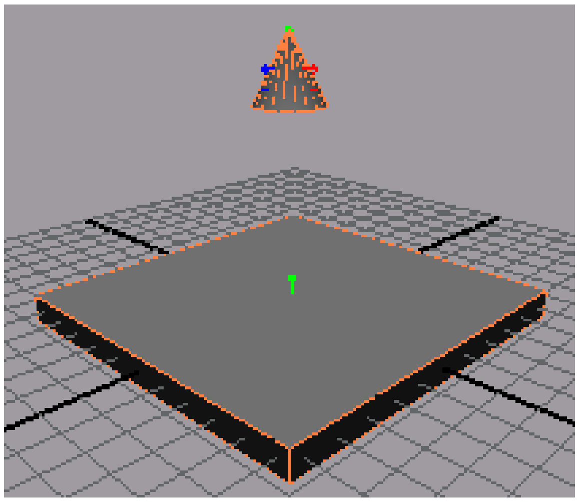
This first section will cover the basics of the plug-in, focusing on the simple aspects of rigid bodies and the rigid solver. Later sections will cover physics shapes, rigid constraints, and other technical details.

Creating a Rigid Body

PhysX rigid bodies can be created through PhysX→Create Active Rigid Body or PhysX→Create Passive Rigid Body, after selecting some shapes or groups of shapes. Only polygonal mesh shapes and "physics shapes" can be used for collision detection and for computing the density of the resulting rigid body. Other shape types (e.g., NURBS and subdivision surfaces) will be ignored for collision, but still displayed.

Example: Ground box and Bouncing Cone.

1. Create→Polygon Primitives→Cube. Press the 'r' key to resize the cube similar to the picture below. Make a ground box that is wide but not high.
2. Create→Polygon Primitives→Cone. Press the 'w' key to move the cone up.
3. Once you're satisfied with the relative position of the objects, select the cone and then use PhysX→Create Active Rigid Body. Press the playback button. You should see the cone falling down through the box.
4. Select the box, then use PhysX→Create Passive Rigid Body. Press the playback button again. You should see the cone bouncing up and down on the box.



Deleting a Rigid Body

Deleting a rigid body can be achieved by selecting it, then pressing the 'del' key, like any other Maya node.

Active/Dynamic vs. Passive/Kinematic

Note that the "active" versus "passive" terminology follows the old convention from Maya's native rigid body dynamics. The PhysX equivalent would be "dynamic" versus "kinematic." An active/dynamic rigid body will be animated by the physics solver, so it can fall down according to gravity and other forces, bounce around and eventually fall asleep if there is no movement for many frames. A passive/kinematic rigid body is not affected by the physics engine or by physical forces. Instead, it affects the dynamic rigid bodies around it through collisions or constraints, for example. The pose (position and orientation) of a passive/kinematic rigid body can be controlled through non-physical means, such as keyframed animations or interactively by the user.

Once created, a rigid body's active/passive state can be modified through the rigid body's active attribute, as described in section XYZ.

Collision Shapes

Polygonal mesh shapes which are part of a rigid body will generally be converted into a convex mesh for collision detection purposes. A rigid body's geometryType attribute controls whether the collision meshes will be made convex or not. *Note that collision detection between two non-convex meshes is not currently supported.*

Physics shapes, which include capsules, boxes and spheres, will be covered in detail in a later section. We recommend them because they allow highly efficient collision detection.

Once added to a rigid body, a shape can still be changed and sculpted. The conversion to a convex mesh, if specified, will occur automatically. The vertex positions can even be animated or deformed (although this will have an adverse impact on performance).

Convex meshes are displayed by default, in wireframe, when the *Visualize Collision Shapes* flag is enabled in the Rigid Solver's display options.

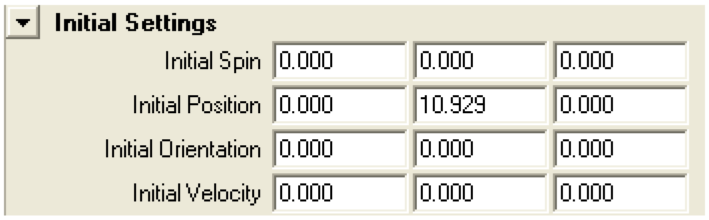
A Rigid Body is a Transform that Contains Collision Shapes

Unlike Maya's native rigid bodies, which are considered as shape nodes that affect their sibling shapes, plug-in rigid bodies are transforms that affect all children shapes. This approach has one downside: it is often necessary to go up to hierarchy to select the parent rigid body. This can be done easily by "pick-walking" – that is, pressing the up/down hotkey when the shape is selected, to walk up and down the transform hierarchy.

Positioning a Rigid Body

Rigid bodies are considered as transforms, and can be moved and rotated as desired, using the regular Maya move tools, for example: 'w' for translate, 'e' for rotate. If the rigid body is moved/rotated while the current frame is set to 1, the initial position/orientation will automatically be updated.

Initial Conditions: Position, Orientation, and Velocity


Initial Settings

Initial Spin	0.000	0.000	0.000
Initial Position	0.000	10.929	0.000
Initial Orientation	0.000	0.000	0.000
Initial Velocity	0.000	0.000	0.000

The initial conditions provide information on the position, orientation and velocities of the rigid body at the first frame of an animation. When an animation is rewound (when the time1 is set to frame 0 or 1), an active/dynamic rigid body will be reset to the given initial conditions. Kinematic rigid bodies are not affected by the initial conditions.

Set Initial Pose

Select `PhysX`→`Set Initial Pose` to record the **current** pose as a starting point (when rewind to Frame0) of Rigid body simulation.

Keyframed Position or Orientation of a Kinematic Rigid Body

A kinematic rigid body can have its position and orientation keyframed, using the regular Maya mechanisms, like the channel box.

Hierarchical Grouping and Rigid Bodies

Parent transforms of a rigid body affect it differently depending on whether the rigid body is active/dynamic or passive/kinematic. If the rigid body is dynamic, parent transforms will only affect the initial position and orientation. If the rigid body is kinematic, the parent transforms will affect every frame during which that body remains kinematic. It is therefore possible to keyframe parent transforms, as long as those transformations are limited to translations and rotations.

Note that it is not possible for a rigid body to contain another rigid body.

Limitations

Please note that the following limitations exist in this version of the plug-in:

- Scaling is not supported.
- Shapes only track their direct parent transform. Each shape in the rigid body can only keep track of its direct parent. If the shape is located several levels below the rigid body, only the transform of its direct parent will be used. Other intermediate parents will be treated as if they had an identity/null transform.
- Rewind before "freezing transformations" or moving pivots. Operations that affect the "rotate pivot" and "rotate pivot translation" attributes of rigid bodies should only be performed at frame 1. If you accidentally do them at a different frame, your rigid body may move in an inconsistent position when you rewind. To correct the situation, move or rotate the rigid body back in place.

Rigid Solver Concept

When creating a rigid body, a rigid solver node will automatically be created if one doesn't already exist. The rigid solver node is a centralized location for global parameters that affect the entire simulation. For example, this includes default gravity control and options to display some physical concepts, such as collision shapes and forces.

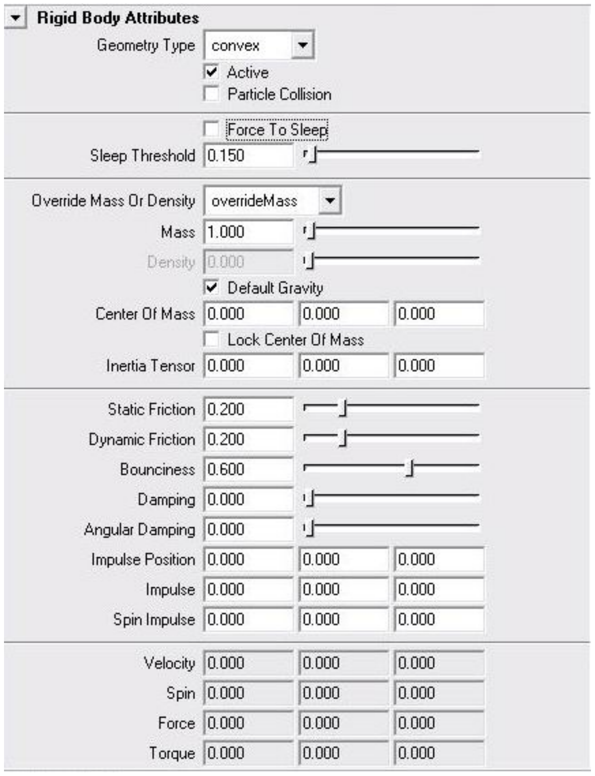
The plug-in also uses the rigid solver node to keep track of which rigid bodies and constraints exist in the scene, and to support automatic update of their parameters. If you use Maya's Hypergraph, you may notice that each rigid body and rigid constraint is connected directly to the solver.

(Note that the way these connections are set-up is currently a performance bottleneck, since Maya dependency graph evaluations incur a noticeable overhead. We plan to address this in a future plug-in version.)

PhysX supports the concept of multiple physics scenes, which can sometime optimize the performance of collision detection or allow special effects. This concept is not currently supported by the plug-in. Only one physics scene, corresponding to the rigid solver, is created at a time.

Rigid Body Attributes

Rigid bodies have dozen of attributes controlling their physical behavior. Many of them are obvious and will require only a brief mention, while we have devoted entire sections for the most complicated ones. Due to a lack of time, some of these attributes are not yet supported. In the coming sections we'll explore these attributes, starting from the top of the attribute editor.



Rigid Body Attributes

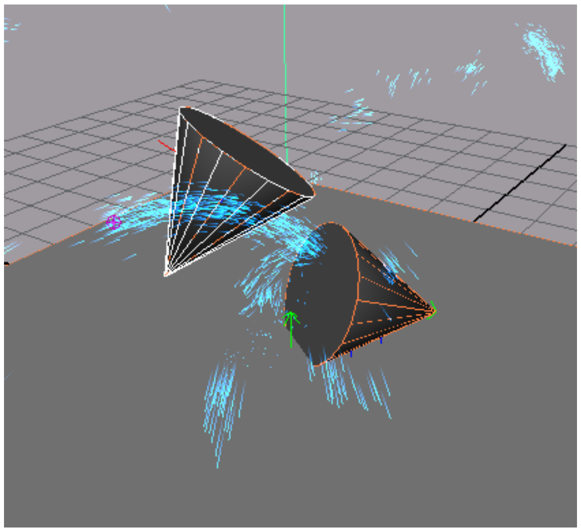
Geometry Type	convex		
	☑ Active		
	☐ Particle Collision		
	☐ Force To Sleep		
Sleep Threshold	0.150		
Override Mass Or Density	overrideMass		
Mass	1.000		
Density	0.000		
	☑ Default Gravity		
Center Of Mass	0.000	0.000	0.000
	☐ Lock Center Of Mass		
Inertia Tensor	0.000	0.000	0.000
Static Friction	0.200		
Dynamic Friction	0.200		
Bounciness	0.600		
Damping	0.000		
Angular Damping	0.000		
Impulse Position	0.000	0.000	0.000
Impulse	0.000	0.000	0.000
Spin Impulse	0.000	0.000	0.000
Velocity	0.000	0.000	0.000
Spin	0.000	0.000	0.000
Force	0.000	0.000	0.000
Torque	0.000	0.000	0.000

Active/Dynamic Attributes

The active attribute can be changed to specify whether the rigid body is active/dynamic or passive/kinematic. This flag is also accessible from the channel box, where it can be animated.

Particle Collision

This flag is currently ignored. It is possible to have particles collide with PhysX rigid bodies, but this functionality is quite limited. Use the Particles→Make Collide command to connect regular shapes to a particle system. This will only work with regular shapes (i.e., not physics shapes like capsules). The particles will collide as expected with the polygon meshes, but they will not induce forces or otherwise affect the PhysX simulation.



Sleeping

When a rigid body does not move for a certain number of frames, it is assumed that it will not move in the future. This state is called sleeping. In order to save time, sleeping objects will no longer be simulated by the PhysX solver. A rigid body in sleep will be woken up if it is moved or rotated to a new position, or any external force (excluding gravity) is applied on it, or any other rigid body collides with it. The plug-in automatically puts a rigid body to sleep when it is moving and rotating slower than the “Sleep Threshold”. You can also put it to sleep by turning on “Force To Sleep”, or turning the option off to wake up the object.

Sleeping can also help with stability even if your scene is not physically built in a stable way. For example, if you do not want the bricks in a loosely constructed brick wall to start moving before a car hits the wall, you should force the bricks to sleep at beginning of the simulation.

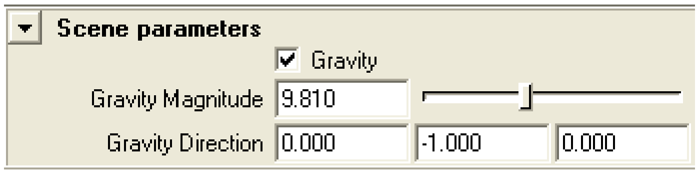
Mass and Density

It is possible to specify either the mass or the density of a rigid body. Internally, the PhysX engine always computes a final mass. If the density is specified, the sum of convex volumes is computed and multiplied by the mass to come up with a final mass value.

Please note that overriding the center of mass of a rigid body is not yet supported by the plug-in even though it is exposed in the PhysX SDK.

Default Gravity

Active rigid bodies can be affected by external forces. The most common force is a simplified version of gravity, so that it points in a given direction. By default, gravity points down (in the negative Y direction) with a magnitude of 9.81. This default value can be modified through the rigid solver’s attribute editor:


Scene parameters

☑ Gravity

Gravity Magnitude 9.810

Gravity Direction 0.000 -1.000 0.000

Friction

The static and dynamic friction coefficients can be specified using two attributes. To be physically valid, the values should range from 0 to 1 and the dynamic friction coefficient should be lower than or equal to the static friction coefficient.

A friction coefficient of 0 means that an object will slide without slowing down if in contact with another object. A friction value of 1 means that the object will tend to stop moving immediately if in contact with another object.

Please note that anisotropic friction (friction that is dependent on the direction of movement), while supported by PhysX, isn't yet exposed in the plug-in.

Bounciness

Bounciness, also called coefficient of restitution or elasticity, is a factor that influences how much energy will be retained to move objects in the opposite direction during a collision. A value of 0 means that an object will tend not to bounce if it collides with another object. A value of 1 would theoretically mean that the object would not lose any energy and would therefore bounce back with a speed equal and opposite to what it had preceding the collision. Due to numerical inaccuracies, it is possible for an object to bounce back even if bounciness is set to 0, and it is possible for an object to lose kinetic energy even if the bounciness is set to 1. In practice, it can be desirable to sometimes set this coefficient to values greater than 1 for special effects, such as creating a jumping bean or a bouncing ball.

Damping and Angular Damping

Damping is a coefficient that has much in common with friction. It tends to slow down the linear movement of a rigid body. Like friction, theoretical values should range from 0 (no air friction) to 1 (friction equal and opposite to the current speed). Like bounciness, there are numerical inaccuracies that cause damping and angular damping to not behave as theory would suggest. For example, a damping coefficient of 10 can be valid and useful for parachutes. Angular damping works similarly, slowing down the rotation of a rigid body.

Solver Iteration Count

The solver iteration count determines how accurately joints and contacts are resolved. For rigid bodies that are not connected through rigid constraints, you can safely leave the default value of 4 unchanged. For ragdolls or long chains of rigid bodies connected by constraints, you should increase this number on each rigid body in the chain, to increase stability. For example, the orc ragdoll is much more stable with a value of 64 or 128.

In Maya's native support, impulse can be used to influence the movement of a rigid body. The plug-in doesn't yet support this concept.

Interactive playback

Interactive playback is a relatively little-known feature of Maya and can be particularly effective when used in conjunction with the plug-in. But first, let's consider normal playback (the playback you get when you press the "play" icon on the right side of the timeline).

In regular playback, Maya tries to play the animated scene as fast as it can. One way to improve performance is to eliminate unnecessary dependency graph evaluations. For example, while the attribute editor is visible, if you move frames manually, you may notice that keyframed attributes are updated. During playback, these updates are temporarily disabled until the stop button is pressed. One big disadvantage of this approach is that the user cannot change these attribute values while regular playback is occurring.

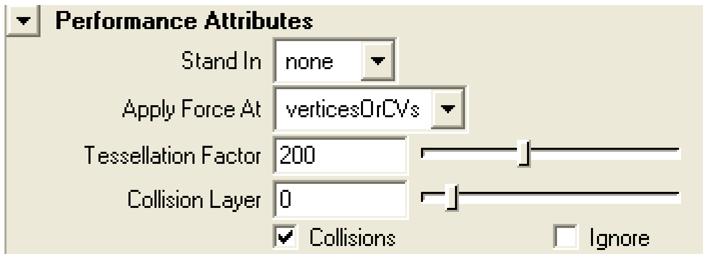
Interactive playback was designed to lift most of these limitations. For convenience, it is available in both the Solvers and PhysX menus. Interactive playback allows the user to keep changing attributes while the playback is occurring. For example, it is possible to change the mass, friction, or damping attributes of a rigid body during interactive playback. This allows you to experiment with values until you're satisfied with the final configuration. One of the best uses of interactive playback is to move and rotate passive/kinematic rigid bodies interactively. This can be quite entertaining, especially if there are many objects to bounce around or when rigid constraints are involved.

Physics Shapes

The physics shapes category corresponds directly to other commands in the PhysX menu. They will be covered in a dedicated section later in this document.

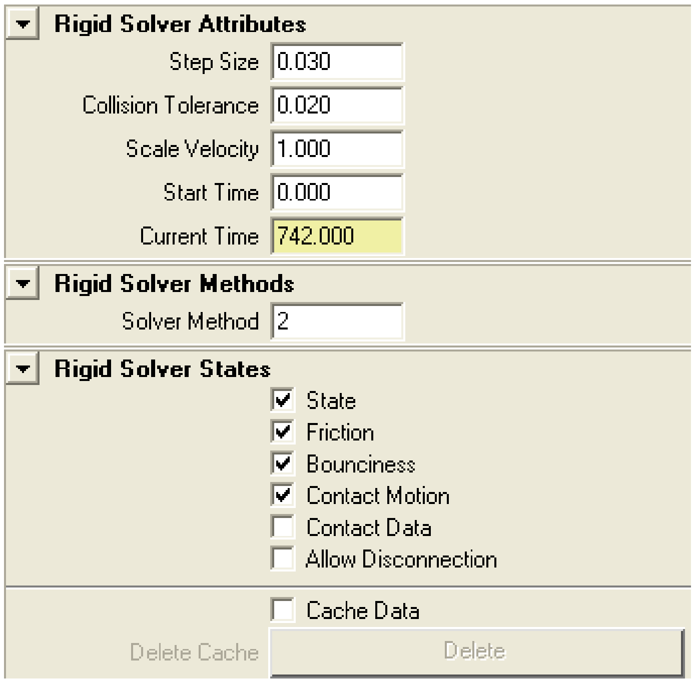
Limitation: Performance attributes are ignored

These attributes were cloned from Maya's native support and are not currently used. Some of them will eventually be replaced by more powerful PhysX-specific features.


Performance Attributes

Stand In: none
Apply Force At: verticesOrCVs
Tessellation Factor: 200
Collision Layer: 0
[✓] Collisions
[ ] Ignore

Limitation: Many rigid solver attributes not currently supported



Rigid Solver Attributes

Step Size	0.030
Collision Tolerance	0.020
Scale Velocity	1.000
Start Time	0.000
Current Time	742.000

Rigid Solver Methods

Solver Method	2

Rigid Solver States

- ✓ State
- ✓ Friction
- ✓ Bounciness
- ✓ Contact Motion
- ☐ Contact Data
- ☐ Allow Disconnection

- ☐ Cache Data

Many of the rigid solver attributes are currently not supported, and most of these attributes will be eliminated since supporting them would incur a lot of engineering effort for relatively little gain.

However, two parameters *are* supported:

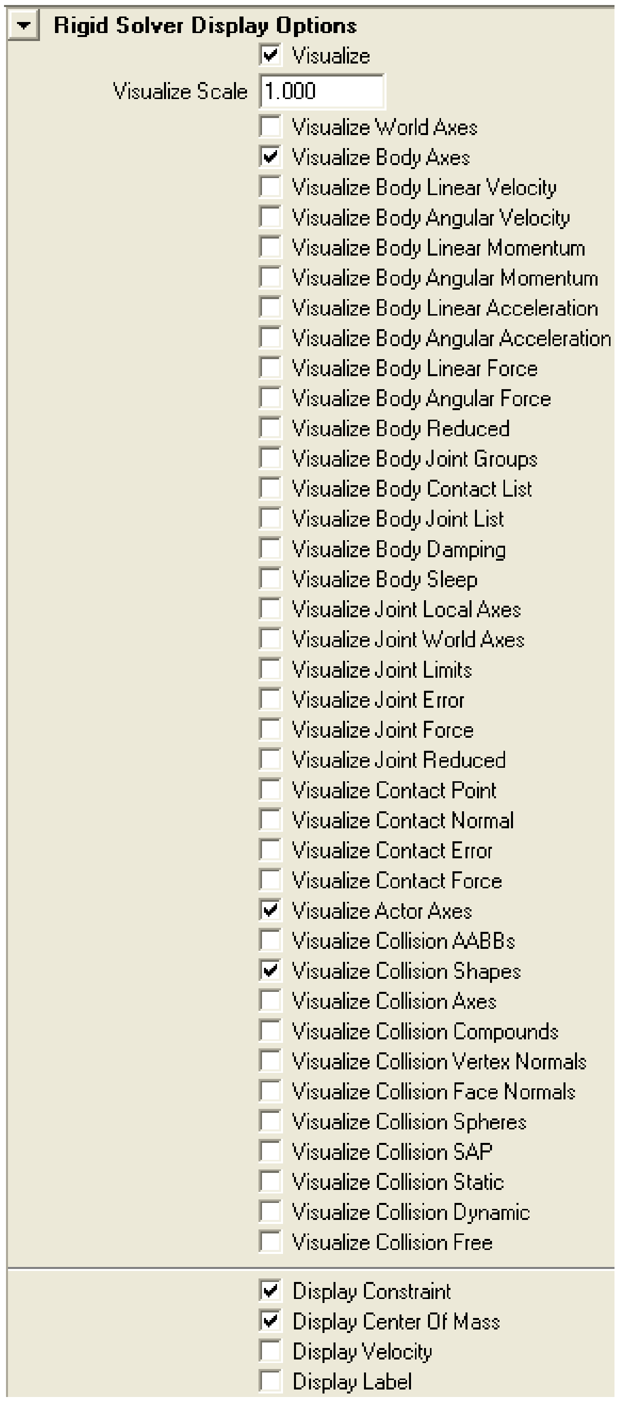
- *Step size* corresponds to the PhysX parameter of the same name (i.e. the first argument to NxScene::setTiming())
- *Collision tolerance* is the absolute value of the PhysX `NX_MIN_SEPARATION_FOR_PENALTY` PhysicsSDK parameter.

There are two sets of features which are expected to soon be supported:

- We plan to expose more PhysX-specific solver parameters, to simplify the tweaking of these values for a given scenario.
- We plan to eventually support caching of rigid body data, to allow timeline scrubbing and timing effects like "bullet time". Note that this is likely to be a post 1.0 feature.

Rigid Solver Display Options

The plug-in fully supports PhysX visualization options, which allow visualizing a wide range of physical parameters. These parameters apply to the entire scene and are exposed under their own category in the solver's attribute editor. First, let's look at the extensive list of parameters that are available:



Rigid Solver Display Options

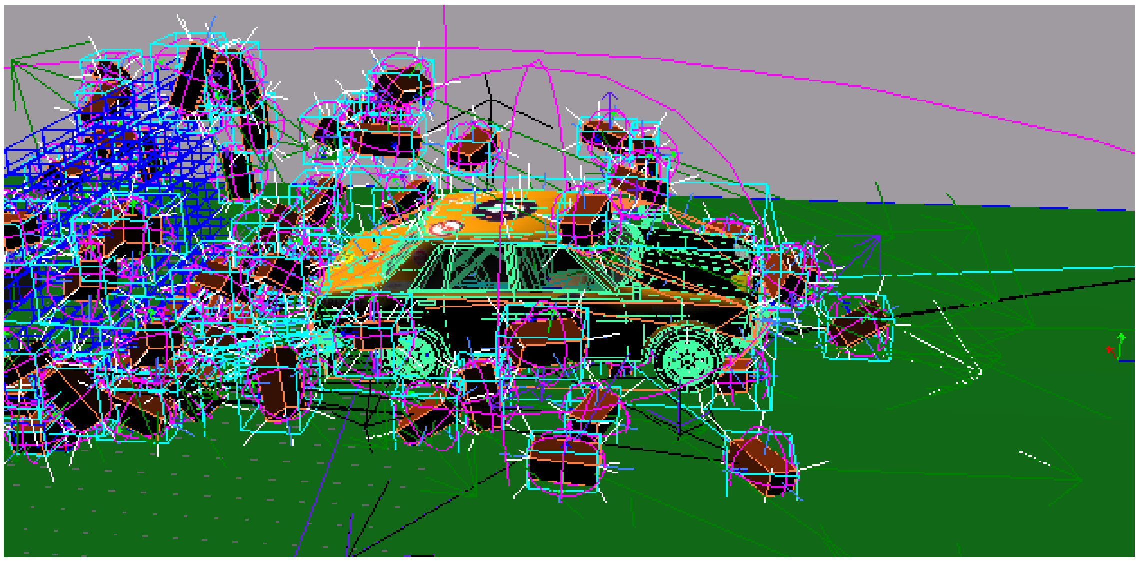
- [✓] Visualize
- Visualize Scale: 1.000
- [ ] Visualize World Axes
- [✓] Visualize Body Axes
- [ ] Visualize Body Linear Velocity
- [ ] Visualize Body Angular Velocity
- [ ] Visualize Body Linear Momentum
- [ ] Visualize Body Angular Momentum
- [ ] Visualize Body Linear Acceleration
- [ ] Visualize Body Angular Acceleration
- [ ] Visualize Body Linear Force
- [ ] Visualize Body Angular Force
- [ ] Visualize Body Reduced
- [ ] Visualize Body Joint Groups
- [ ] Visualize Body Contact List
- [ ] Visualize Body Joint List
- [ ] Visualize Body Damping
- [ ] Visualize Body Sleep
- [ ] Visualize Joint Local Axes
- [ ] Visualize Joint World Axes
- [ ] Visualize Joint Limits
- [ ] Visualize Joint Error
- [ ] Visualize Joint Force
- [ ] Visualize Joint Reduced
- [ ] Visualize Contact Point
- [ ] Visualize Contact Normal
- [ ] Visualize Contact Error
- [ ] Visualize Contact Force
- [✓] Visualize Actor Axes
- [ ] Visualize Collision AABBs
- [✓] Visualize Collision Shapes
- [ ] Visualize Collision Axes
- [ ] Visualize Collision Compounds
- [ ] Visualize Collision Vertex Normals
- [ ] Visualize Collision Face Normals
- [ ] Visualize Collision Spheres
- [ ] Visualize Collision SAP
- [ ] Visualize Collision Static
- [ ] Visualize Collision Dynamic
- [ ] Visualize Collision Free

- [✓] Display Constraint
- [✓] Display Center Of Mass
- [ ] Display Velocity
- [ ] Display Label

The parameters that start with “visualize” are all documented in the PhysX documentation. Please refer to that documentation if you want details on the different options.

The *visualize* flag can be used to enable/disable visualization. The *visualize scale* is a scaling factor, to make lines shorter or longer depending on the dimensions of your scene. Please note that visualization flags are only simulated after at least a frame of simulation has occurred. Therefore, any change should be followed by playback to see a difference. To give you an idea of the complexity and completeness of these visualization flags, here is a scene shown with all flags enabled.

Please note that the last four display options (Display Constraint, Display Center of Mass, Display Velocity, and Display Label) are not currently implemented.



Physical Units

We haven't discussed physical units until now. The reason is simple: they are left to your discretion. It is suggested that you choose meaningful and consistent length and mass units, such as meters and kilograms, and set gravity and other physical parameters to match. The Maya scene units are not currently taken in consideration.

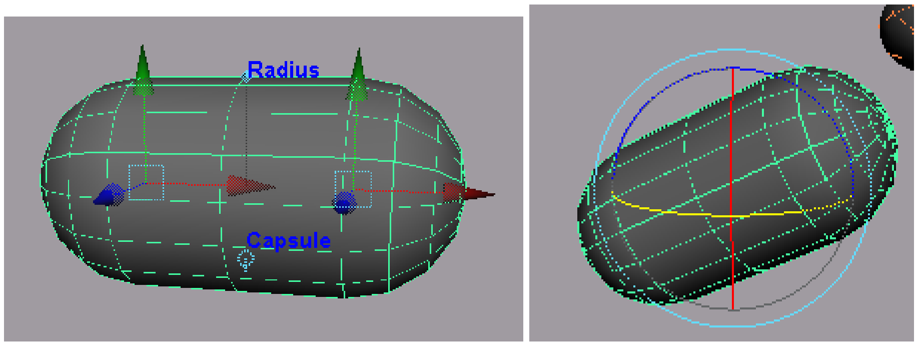
Physics Shapes

In addition to mesh collision shapes, a rigid body can contain any number of physics shapes, which are especially useful for their fast collision-detection capability. These physics shapes include spheres, boxes, and capsules. There are several ways to create such shapes:

- You can add a physics shape to an existing rigid body, by selecting the rigid body or one of its shapes, and by executing *PhysX→Create Box/Sphere/Capsule*.
- It is also possible to create them through the rigid body's attribute editor. The physics shapes that are created this way will automatically be attached to the rigid body and affects its collisions.
- You can create a physics shape which is not assigned to any rigid body, by running the above menu command. You can later create a rigid body containing that shape (or a group of shapes) by running *PhysX→Create Active/Passive Rigid Body*.
- Or you can attach the shape to an existing rigid body by parenting the shape somewhere under the rigid body, and by selecting *PhysX→Collision Shapes→ Enable Collisions*.

Please note that you cannot scale physics shapes using the Maya Scale Tool. Instead, use the Manipulator Tool to drag and select its parameters.

Due to their specialized purpose, collision shapes have a number of limitations. In particular, they do not support non-uniform scale and any shear or deformer. To change the parameters of a shape, such as the radius of a sphere or capsule, use its manipulator, which is generally bound to the 't' hotkey. The following figure shows an example of this manipulator in action:



Physics shapes always have a direct parent transform, to allow the position and orientation of the whole shape to be manipulated, using the regular Maya move and rotate tools.

Collision Shapes – Enabling/Disabling Collisions

You can disable/enable collisions for a given regular or physics shape in a rigid body, using *PhysX→Collision Shapes→Enable collisions* and *PhysX→Collision Shapes→Disable collisions*. Make sure to select the shape and not the rigid body. Selecting the rigid body prior to running these commands will cause all shapes in that rigid body to be enabled/disabled.

***Limitation**: Currently, if a rigid body was created with a single mesh shape, the mesh shape gets parented right under the rigid body. In this case, selecting the convex shape will be equivalent to selecting the entire rigid body, and these commands will not behave as expected. This problem will be fixed in a future release.*

Rigid Constraints

Rigid constraints are used to constrain two rigid bodies along some rotational or linear axis. For example, a ball joint allows two objects to rotate relative to each other while restricting their relative translational movement. Likewise, a slider constraint allows two objects to slide relative to each other along a given axis, but it restricts the translational movement along the other axes, as well as any rotational movement.

In the plug-in, rigid constraints are implemented using the PhysX 6-DOF (degrees-of-freedom) joint (known in the SDK as an NxD6Joint). To understand every detail of the plug-in's rigid constraints, we recommend that you refer to the PhysX documentation.

Maya Native vs. Plug-In Rigid Constraints

Users of Maya's native rigid body dynamics may be interested in comparing it to the plug-in's rigid constraint. Native rigid constraints come in five types: nail, pin, hinge, spring, and barrier. The first four types are supported through the plug-in's rigid constraints. Additionally, the plug-in's rigid constraints are more flexible and powerful, allowing the user to specify such things are breakable joints, limits, and angular springs.

The barrier native type is not supported. Instead, we recommend that you use other features such as additional passive/kinematic rigid bodies, or rigid constraint linear or angular limits to achieve similar effects.

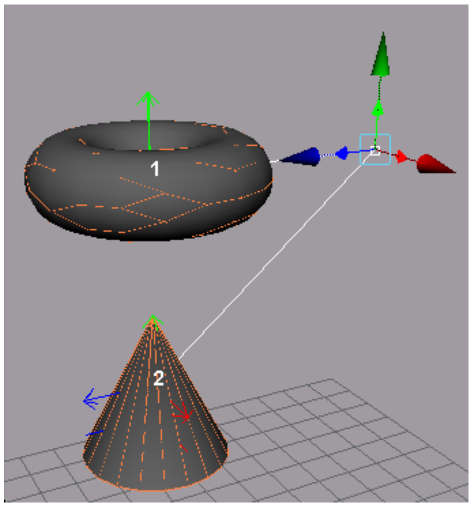
The plug-in's rigid constraint is somewhat more complex than Maya's. It may seem a bit daunting at first, but with time, we hope you will become more familiar with it and appreciate its extra flexibility.

Creating and Deleting a Rigid Constraint

PhysX rigid constraints can be created by selecting two rigid bodies (you can only select 2 nxRigidBodies to create a rigid constraint) and using *PhysX→Create Rigid Constraint*.

By default, the rigid constraint that gets created has no degrees of freedon.

Rigid constraints can be deleted using the 'del' hotkey, like any other Maya nodes.

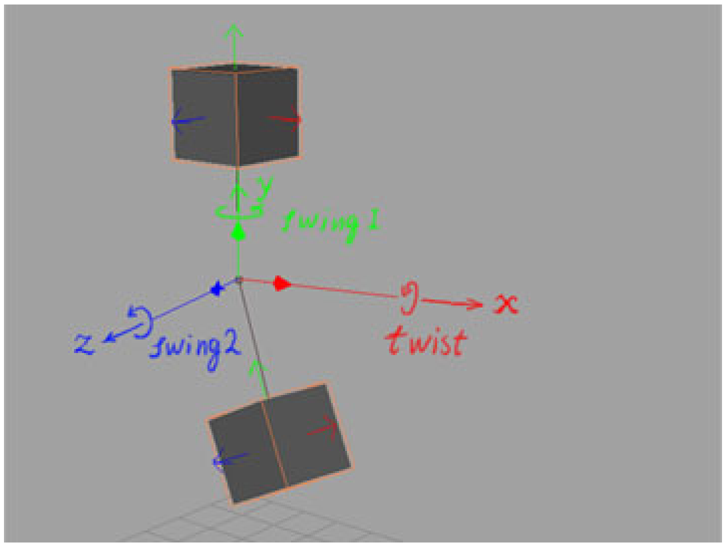


Attachment Frames

A rigid constraint has one attachment frame per rigid body. Currently, the plug-in only supports rigid constraints which have two associated rigid bodies, and therefore two attachment frames. The small red-green-blue arrow triad that can be seen in the figure above corresponds to both attachment frames. These are used to visualize the position and orientation of each rigid body relative to the other.

Rigid constraints keep track of the attachment frames in the local space of each rigid body. The role of these local space attachment frames will become clear in the next section.

Degrees of Freedom (DOF)

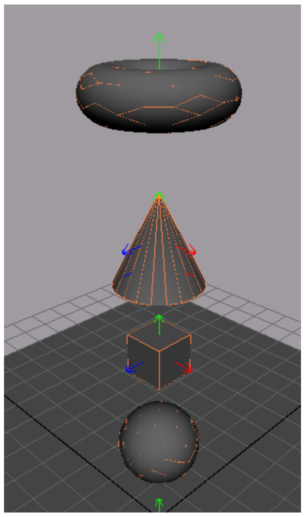


In a rigid constraint, each axis of the attachment frame has two potential degrees-of-freedom. A degree of freedom (DOF) is specific to an axis and can be angular or linear, in which case it indicates whether the second rigid body is free to rotate or slide along the given axis. Since we are operating in 3 dimensions and there are two rigid bodies, this gives a total of 6 possible degrees of freedom (one rotational and one translational along each axis).

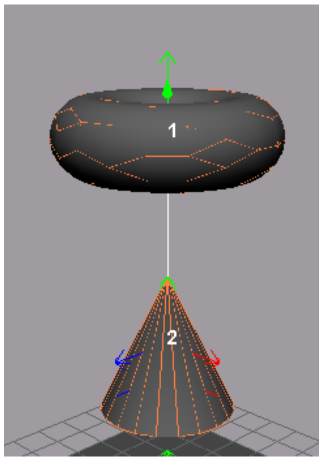
By default, a rigid constraint is created with no degrees of freedom. This means that the two objects are “glued” together, and cannot move relative to each other in any way.

Rigid Constraint Examples: Hinges, Ball Joints, Sliders

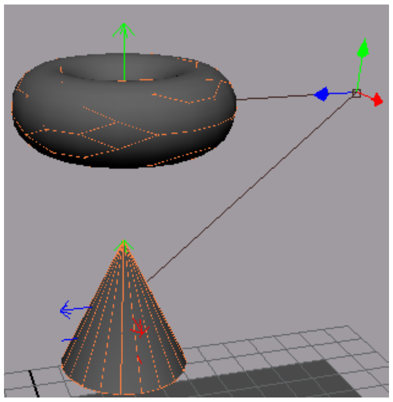
To better understand the DOF concept, it is worth trying out some common configurations: hinge, ball joint and slider. First, run the sample scene from the *nxTest1* test script (by typing “nxTest1” into the MEL command line). You should get the following scene:



Next, connect the two top rigid bodies with a default constraint, by selecting them and then using *PhysX→Create Rigid Constraint*. Make sure to select the top object first, then the second. You should see that the newly created constraint, when selected, will provide a visual indication of which rigid body is first and second, using labels that read ‘1’ and ‘2’:

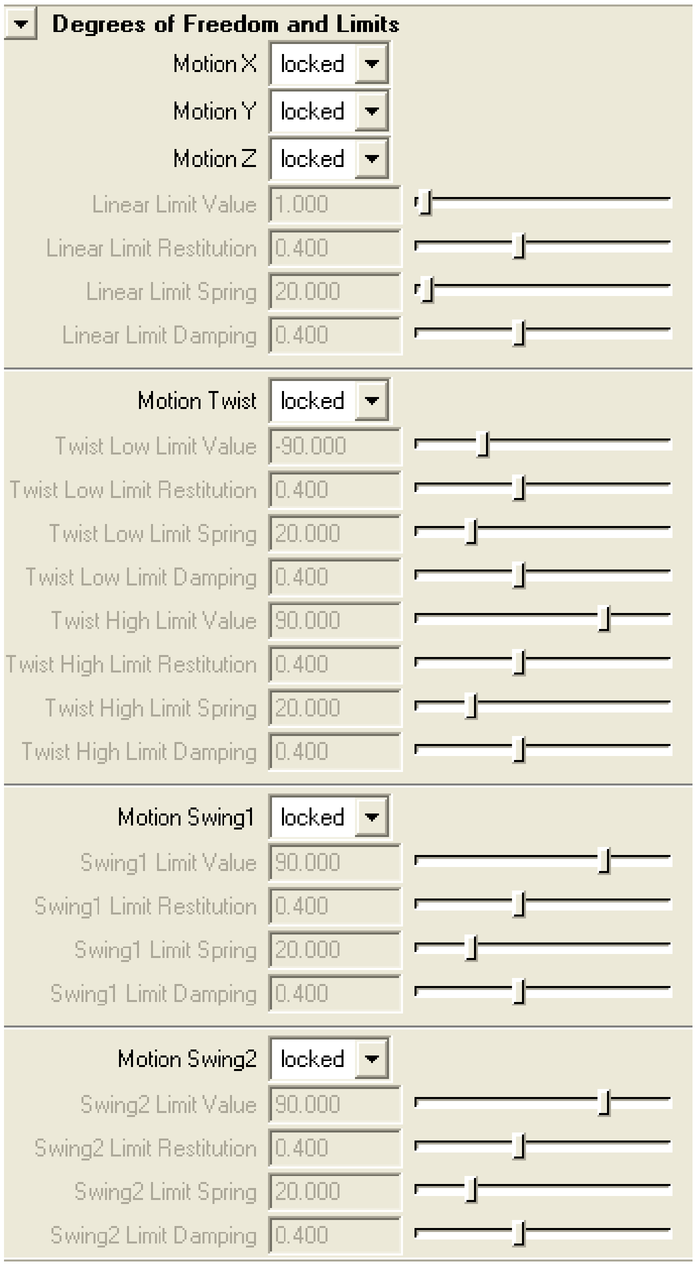


To better see the attachment frames, use the Move tool (‘w’) and move it along –Z:



You could also use the rotate tool (‘e’) to orient the attachment frames. By default, these tools change the two attachment frames so they match in world space.

To modify the degrees of freedom, open the attribute editor while the rigid constraint is selected. There is a category that concerns DOFs:

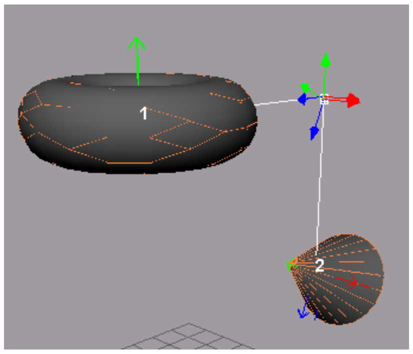


Degrees of Freedom and Limits

Motion X	locked
Motion Y	locked
Motion Z	locked
Linear Limit Value	1.000
Linear Limit Restitution	0.400
Linear Limit Spring	20.000
Linear Limit Damping	0.400
Motion Twist	locked
Twist Low Limit Value	-90.000
Twist Low Limit Restitution	0.400
Twist Low Limit Spring	20.000
Twist Low Limit Damping	0.400
Twist High Limit Value	90.000
Twist High Limit Restitution	0.400
Twist High Limit Spring	20.000
Twist High Limit Damping	0.400
Motion Swing1	locked
Swing1 Limit Value	90.000
Swing1 Limit Restitution	0.400
Swing1 Limit Spring	20.000
Swing1 Limit Damping	0.400
Motion Swing2	locked
Swing2 Limit Value	90.000
Swing2 Limit Restitution	0.400
Swing2 Limit Spring	20.000
Swing2 Limit Damping	0.400

As mentioned earlier, by default all DOFs are locked so the rigid bodies are stuck together. In this special configuration, regardless of how you position or orient the attachment frames, the two rigid bodies will fall together. To make it easier to demonstrate the degrees of freedom, make the top rigid body passive/kinematic, so that it is fixed in space and no longer obeys gravity. The two rigid bodies should now remain fixed in space.

To demonstrate a simple hinge-style constraint, with only one rotational degree of freedom, set the constraint's *Motion Twist* to *free*. The twist corresponds to the rotational degree of freedom along the local X axis (red arrow), therefore the rigid body should start rotating around that axis, as in the following figure:



For the first time, we see that there are really two attachment frames, which each associated with one rigid body.

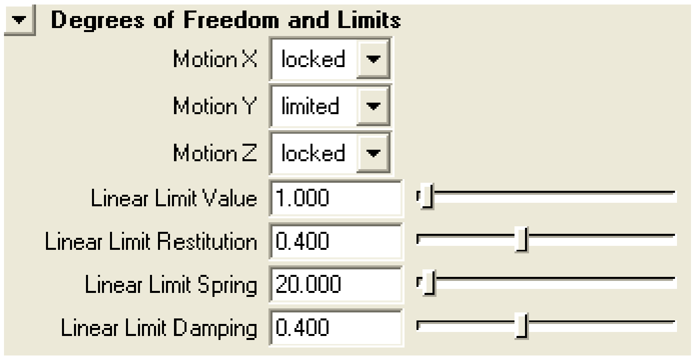
After selecting the top rigid body, you can use PhysX→Interactive Playback to move the rigid body around and get a feeling of the hinge's direction. Also, rotate the attachment frame's red axis to see that the hinge follows that direction at all time. If during your experimentation you notice that the two axes get out of sync, stop the playback, rewind and use the rotate tool to bring them back in sync. (We'll see later how to avoid this default behavior to sync the attachment frames.)

Now, to demonstrate a ball joint behavior, simply set the *Motion Swing1* and *Motion Swing2* to *free* too. That way the second rigid body is free to rotate along all axes. The difference is especially noticeable during interactive playback.

Finally, we can demonstrate a slider behavior. First, set all rotation DOFs (*Motion Twist*, *Motion Swing1* and *Motion Swing2*) to *locked*. Next, set the *Motion Y* to *free*. Select the first rigid body and do an interactive playback. You should notice that the object falls down along the Y axis, but it doesn't rotate, or translate along the X and Z axis.

Limits for DOFs

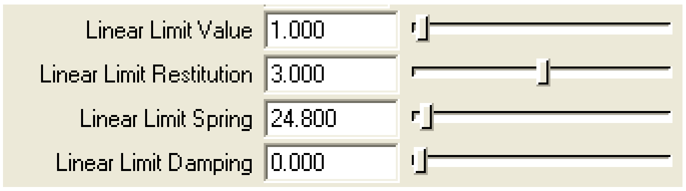
One of the key features of the plug-in's rigid constraints, compared to Maya's native rigid constraints, is the possibility to set limits and spring coefficients along different axes. To demonstrate this feature, use the previous slider example with two rigid bodies connected by one constraint. Instead of setting the *Motion Y* to *free*, set it to *limited*. Make sure that the following linear limit attributes are set as in the figure below:



Degrees of Freedom and Limits

Motion X	locked
Motion Y	limited
Motion Z	locked
Linear Limit Value	1.000
Linear Limit Restitution	0.400
Linear Limit Spring	20.000
Linear Limit Damping	0.400

By running an interactive playback, you should notice that the second rigid body falls down a short distance (in theory, 1.0) before slowing down and stopping in mid-air. By changing the above parameters, you could make it bounce back and forth, as if a spring was involved:


Linear Limit Value 1.000
Linear Limit Restitution 3.000
Linear Limit Spring 24.800
Linear Limit Damping 0.000

To change the maximum distance, modify the *linear limit value* to 1.0. This corresponds to the maximum allowed distance between the two attachment frames. The *restitution* parameter indicates how much energy is preserved when reaching the limit. Since the object is pulled by the gravity, a value greater than 1 is necessary for the object to bounce back to its original value. Be careful not to set this value too high, in which case the rigid body's kinetic energy will keep increasing exponentially, causing what is commonly called "explosions" where parts of the physical system gets out of control and bounces to infinity. If that occurs, reduce the restitution to a value closer to 0.

The *limit spring* parameter is a spring parameter; you can think of it as a way to control the overall speed of the bounce. The *restitution* and *spring* parameters are interrelated and you might want to play with both of them to get a better feeling of how they interact.

Finally, the *damping* parameter is similar to the rigid body's attribute of the same name. It tends to reduce overall velocity along the limited axes.

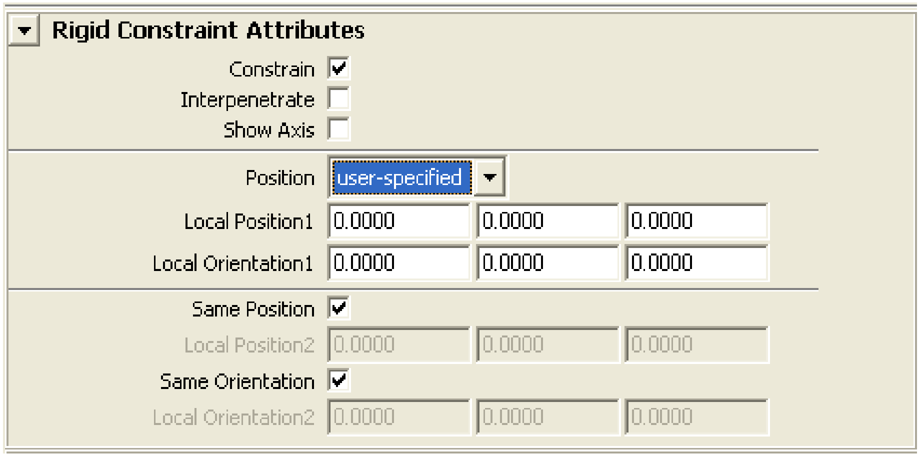
For now all 3 linear axes are controlled using a same limit parameter. We will provide independent limit to control 3 linear axes in the future.

Angular Limits

Angular limits can be set on each axis, and are similar to linear limits: they affect angles and orientations instead of linear distance.

Notably, the *Motion Twist* has two sets of limit parameters, defining minimum and maximum limits along with separate restitution, spring and damping parameters for each direction.

The Constrain Flag


Rigid Constraint Attributes
Constrain ✓
Interpenetrate
Show Axis
Position user-specified
Local Position1 0.0000 0.0000 0.0000
Local Orientation1 0.0000 0.0000 0.0000
Same Position ✓
Local Position2 0.0000 0.0000 0.0000
Same Orientation ✓
Local Orientation2 0.0000 0.0000 0.0000

The *constrain* flag is used to enable/disable the entire rigid constraint. When disabled, a constraint has no influence on the rigid bodies that it connects.

While this attribute can be animated (for example, through keyframes) enable constraints with caution. For example, if the rigid constraint is a hinge and its two attachment frames are separated and far away from each other, the solver will try to bring the two attachment frames in the same position, which will result in large forces being applied for a short period of time. This can make the overall physical model unstable. In practice, this problem can often be avoided by moving the rigid bodies closer to each other or by setting linear limits and springs such that the two bodies move toward each other in a controlled manner.

The Interpenetrate Flag

The interpenetrate flag indicates whether the two rigid bodies that are connected by a constraint are allowed to interpenetrate. This effectively ignores collisions between the shapes of the two rigid bodies.

Local position2/orientation2

Internally, the attachment frames are processed in the local space of each rigid body. The translate/rotate attributes of the rigid constraint is expressed in terms of the first rigid body's local space. The localPosition2 and localOrientation2 are similar, except that they are expressed in terms of the second rigid body's local space.

It is possible to see and modify the positions and orientations of the rigid bodies' attachment frames through these two set of attributes.

Specifying a Second Attachment Frame

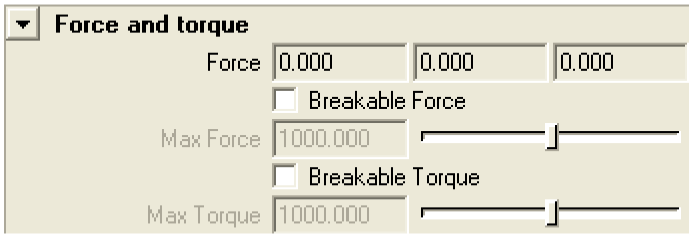
Until now, we have always specified the position and orientation of both attachment frames simultaneously. It is also possible to specify the second attachment frame separately. To do so, uncheck the *same position* and/or *same orientation* attributes. When unchecked, the move/rotate tool only affects the first attachment frame.

It is therefore possible to position and orient the second attachment frame while the *same position* and/or *same orientation* flag is enabled (effectively affecting both attachment frames), then uncheck the flags to modify only the first attachment frame.

This workflow will be improved in a future version.

Current Force/Torque

Please note that the plug-in does not report the current force or torque exercised on a given rigid constraint. This will be addressed in a future version.


Force and torque
Force 0.000 0.000 0.000
☐ Breakable Force
Max Force 1000.000
☐ Breakable Torque
Max Torque 1000.000

Breakable Force/Torque

The plug-in supports the concept of a breakable force or torque. It is therefore possible to have a rigid constraint break (i.e., become disabled) when the force or torque that it sustains becomes larger than a user-specified value.

Note that a broken rigid constraint will automatically be repaired when the *constrain* flag is switched from unchecked to checked. Additionally, broken rigid constraints are automatically repaired when rewinding the scene.

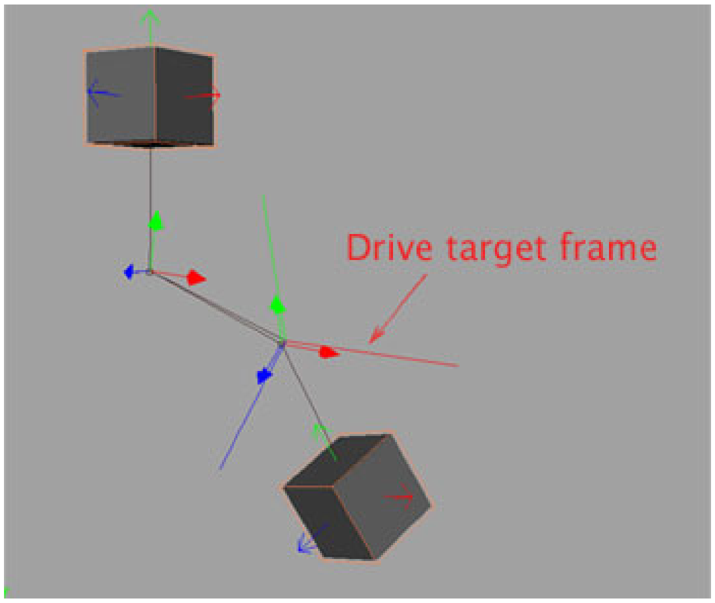
Motors

In a rigid constraint, the second attachment frame can be pushed to specific position and/or rotated to specific orientation by *motors*. Motors add a “force” to influence constrained rigid bodies, similar to how a car engine tries to make wheels roll or how an elevator can attempt to lift passengers. If the force is not sufficient (e.g. because the people in the elevators are too heavy), the elevator will stall or possibly even fall down.

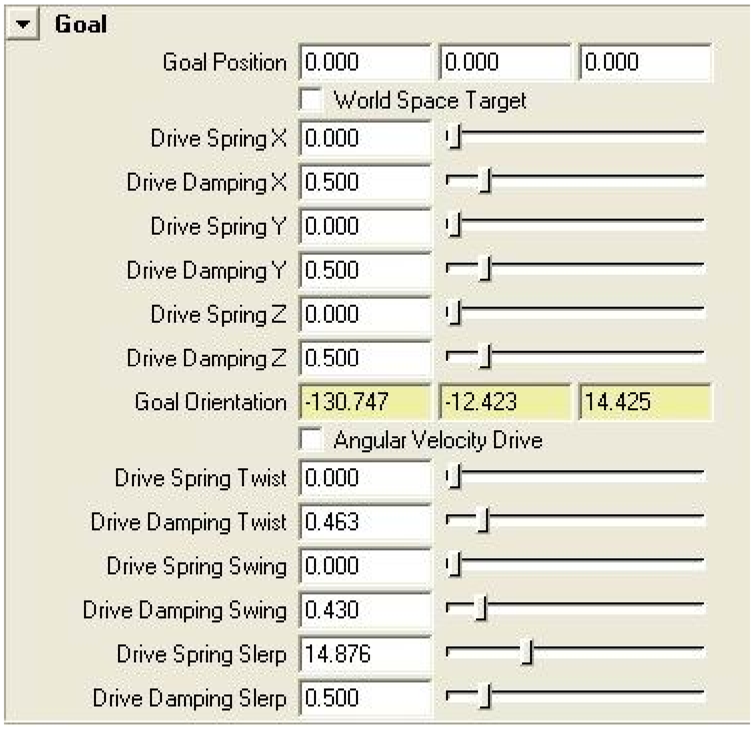
Using motors is also a robust way to blend key framed animation with physics simulation. You can pose your character during simulation by specifying motion for the joint drive target frames.

A rigid constraint has 3 linear drives (drives X, Y, Z) to move the joint along XYZ axes, and 3 angular drives: drive Twist rotates the joint around X-axis, drive Swing rotates the joint around the Y-axis, drive Slerp rotates the joint along the shortest spherical arc to the destination orientation. The Slerp drive only has an effect when all the three angular motions are not locked when the twist and swing drives are ignored.

Angular drives can also work in velocity mode that rotates the joint along the axis at a desired velocity. The frame is drawn in as the visualization for the target position and orientation of the joint motor.



Joint Motor Attributes

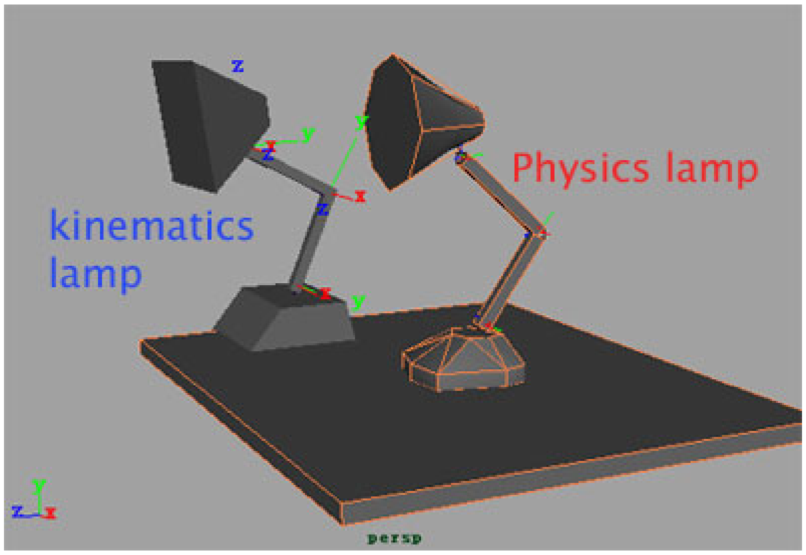

Goal

Goal Position	0.000	0.000	0.000
	☐ World Space Target		
Drive Spring X	0.000		
Drive Damping X	0.500		
Drive Spring Y	0.000		
Drive Damping Y	0.500		
Drive Spring Z	0.000		
Drive Damping Z	0.500		
Goal Orientation	-130.747	-12.423	14.425
	☐ Angular Velocity Drive		
Drive Spring Twist	0.000		
Drive Damping Twist	0.463		
Drive Spring Swing	0.000		
Drive Damping Swing	0.430		
Drive Spring Slerp	14.876		
Drive Damping Slerp	0.500		

1. `Goal Position`: target position of the linear drive.
2. `World Space Target`: specifies the goal position relative to the world space if checked. Otherwise, the target position will be defined in the local coordinate of the first attachment frame.
3. `Goal Orientation`: target orientation of the angular drive. It is always in the local coordinate of the first attachment frame.
4. `Angular Velocity Drive`: drive angular motions in velocity mode if checked.
5. `Drive Spring X, Y, Z, Twist, Swing`: magnitude coefficient for drive relevant to the xyz axis and twist and swing1 rotation. No force will be generated of set to zero.
6. `Drive Damping X, Y, Z, Twist, Swing`: damping coefficient for drive relevant to the xyz axis and twist and swing1 rotation. Larger values will stop the motion of drive more rapidly.

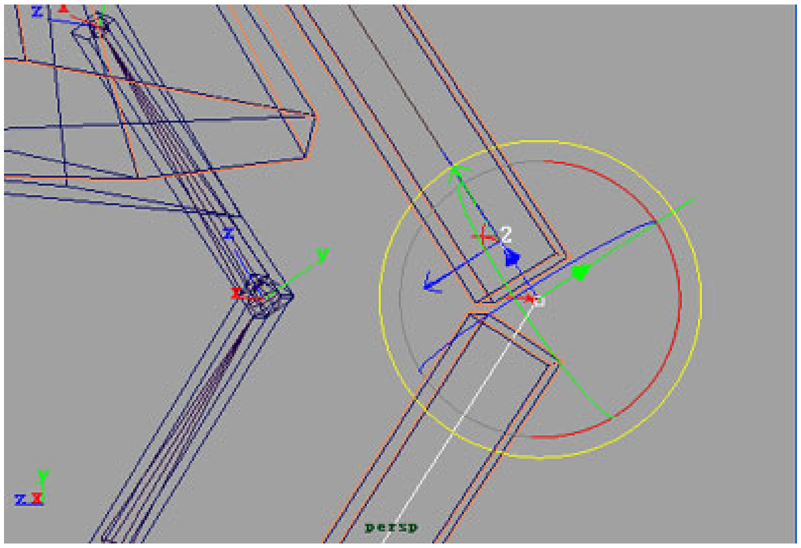
Motor Limits

A motor will have no effect if the relevant degree of freedom is locked. The motion of the constrained rigid bodies is also constrained by the linear and angular DOF limits.



Joint Motor Example: Lamp Animation

In this example, the lamp to the right consists of four rigid bodies constrained together, and the left one is animated by kinematics. The motion of the right lamp is a blend of the overall pose derived from the left one and all the detailed bouncing and swinging movement generated by the physics simulation.



In each rigid constraint, linear motions are locked and the goal orientation attribute is connected to the rotation of a skeleton bone. The constraints are rotated to align the local rotation axes of the connected bone.

Limitation: Rigid Constraint with Only One rigid body Not Yet Supported

The support for a rigid constraint linking a rigid body and a transform is not yet completed. This will be addressed in a future version.

Force Fields

The plug-in provides some limited support for force fields to influence rigid bodies.

Follow this procedure to assign one or more rigid bodies to a force field:

1. First make sure that no rigid body or shape is selected.
2. Create the force field as usual (for example, through *Fields→Newton*).
3. Select the force field(s) and rigid bodies you want to connect, and choose *PhysX→Connect to fields*. All force fields, including field plug-ins, can be supported in this way.

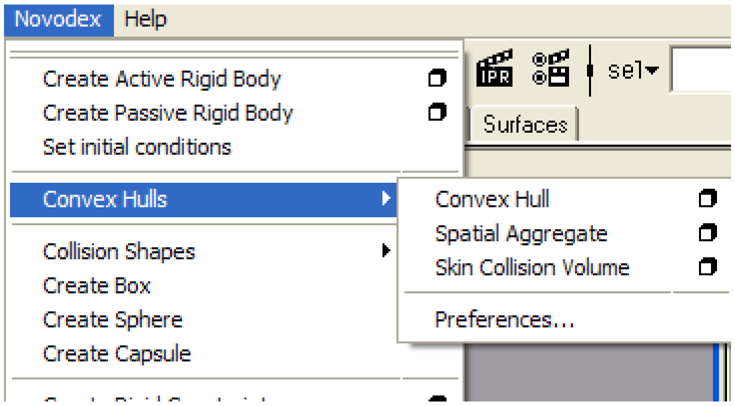
Note that care must be taken not to use other commands from the *Fields* menu. These standard Maya commands do not understand the node architecture used by the plug-in, and could corrupt the scene. This is an issue that we hope to address in the future.

Currently, the forces are only applied on the center of mass of each rigid body; consequently they do not induce torque.

Convex Hulls

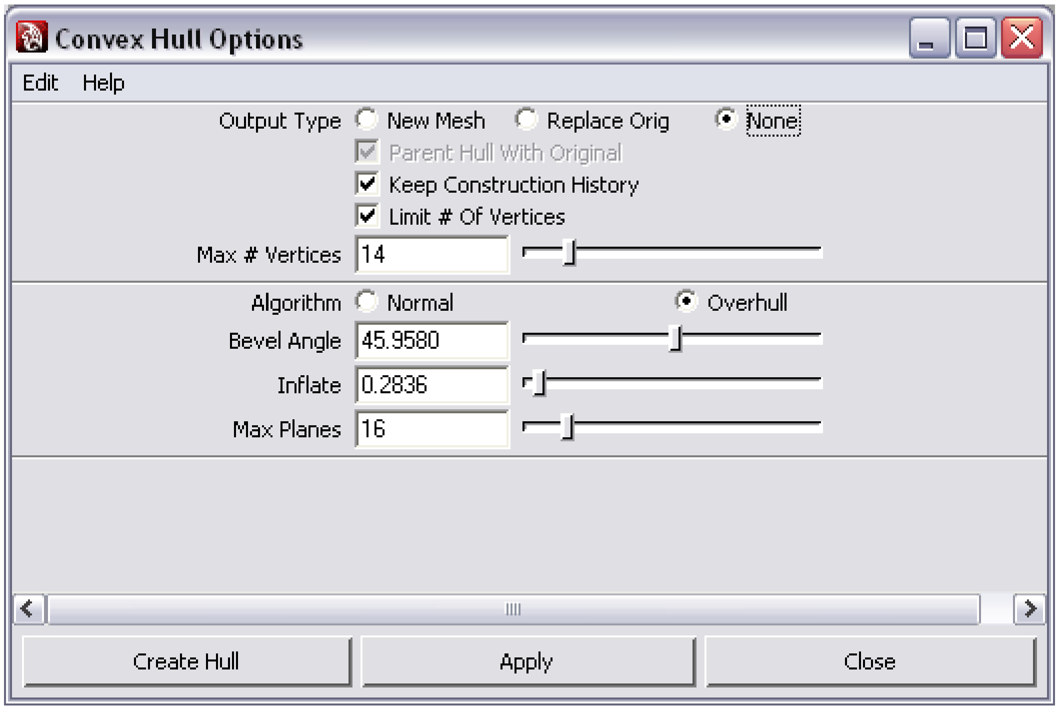
Three commands are available to create convex meshes:

- *Convex hull* creates a single convex mesh from a single input mesh.
- *Spatial aggregate* creates N convex meshes from a single input mesh, where N is selected by the user.
- *Skin collision volume* creates N convex meshes from a skinned input mesh, where N is the number of joints in the skin skeleton.



Convex Hull

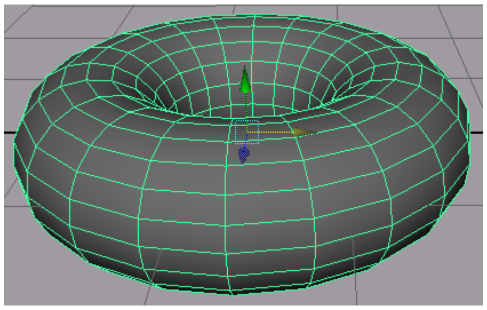
To use the convex hull tool, first select an input polygon mesh, and then use *PhysX→Convex Hulls→Convex Hull*.



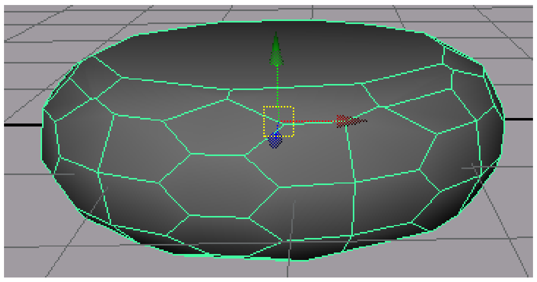
Output Type ○ New Mesh ○ Replace Orig [✓ None]
[✓ Parent Hull With Original]
[✓ Keep Construction History]
[✓ Limit # Of Vertices]
Max # Vertices 14
Algorithm ○ Normal [✓ Overhull]
Bevel Angle 45.9580
Inflate 0.2836
Max Planes 16

Many parameters are provided to control the resulting surface quality, in terms of number of vertices and polygons. Two algorithms are provided: Normal and Overhull. Here is a demonstration of how to use the convex hull operation on a torus:

Create a polygonal torus and select it, as shown below.



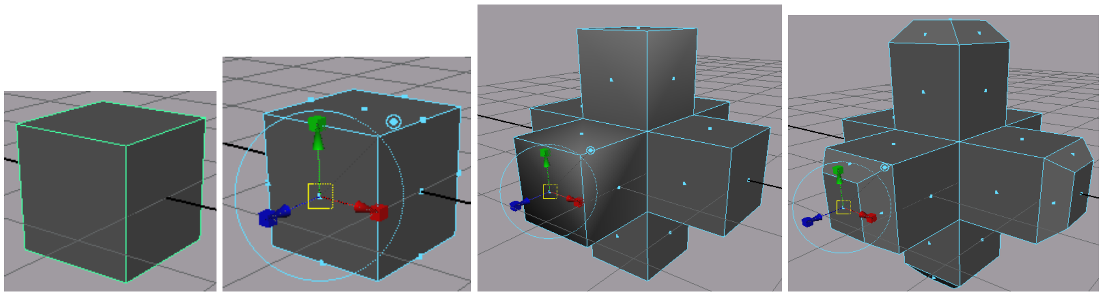
Run the convex hull operation with default parameters, except for the construction history, which should be disabled:



The resulting mesh can then be further modified, triangulated, converted into a rigid body, etc.

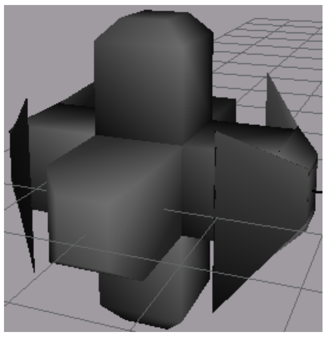
Spatial Aggregate

The spatial aggregate tool can be used to create a group of convex meshes that closely approximates some input mesh. This is especially useful for architectural models such as cities or buildings. To demonstrate this tool, we will first create an interesting shape that can be found in the figure below.

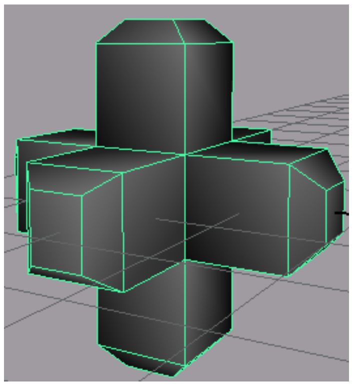


Try the following steps:

1. Create a polygonal cube.
2. Keep the whole cube selected then select *Edit Polygons→Extrude Face*.
3. Translate the selected extruded face along the normal; this will automatically update every other face of the cube.
4. Repeat the extrude face operations on the same faces and scale them smaller to produce beveled edges.
5. Finally, press F8 to get out of component mode and to select the whole mesh.
6. Do a *PhysX→Convex hull→Spatial aggregate*. This will produce a series of convex meshes, centered at the origin:



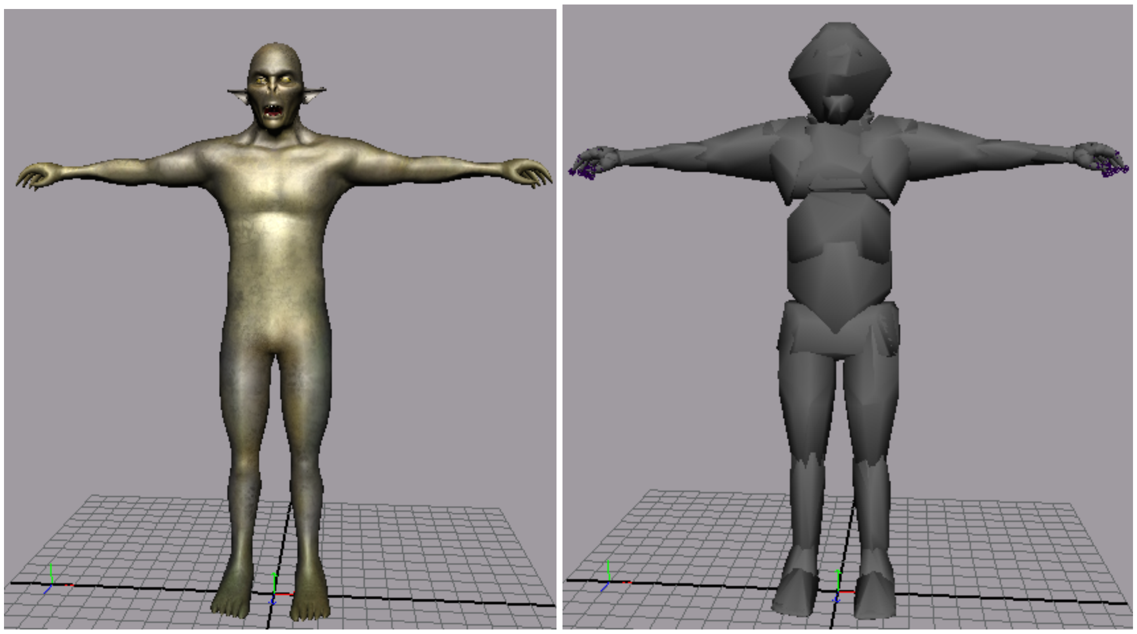
Note that in some cases, like this one, the spatial aggregate produces some unnecessary meshes can be safely deleted. Delete any excessive mesh to get a result like this one:



The resulting five convex meshes are optimal. They can be grouped and converted into a rigid body that will reproduce the concavity of the original mesh. Another possibility would be to hold them together using breakable rigid constraints, so the object breaks on impact.

Skin Collision Volumes

The skin collision volume tool takes an input skinned mesh and generates one convex mesh for each rigid joint in the associated skeleton. Read the next section, on ragdoll construction, to learn more about this tool.



Ragdoll Tools

Several scripts have been provided to ease the creation of ragdolls from skinned characters. Although it would have been possible to provide a single script that does the entire ragdoll creation in one shot, this solution would not have been practical, since it is often desirable to tweak shapes and parameters to produce optimal results.

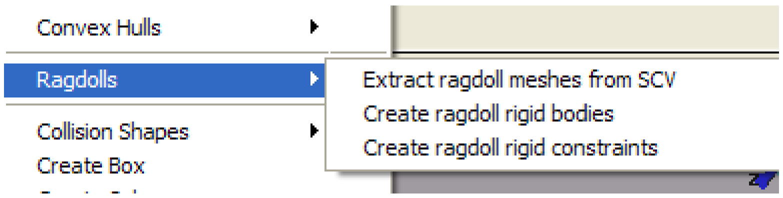
Please note: Sometimes, the skin of a ragdoll character may show knots, due to a limitation of the standard Maya parentConstraint not handling flipping correctly. We hope to address this problem in a future release.

Steps to Create a Ragdoll

To create a ragdoll, start from a skinned character (such as the orc sample model) and go through 4 steps:

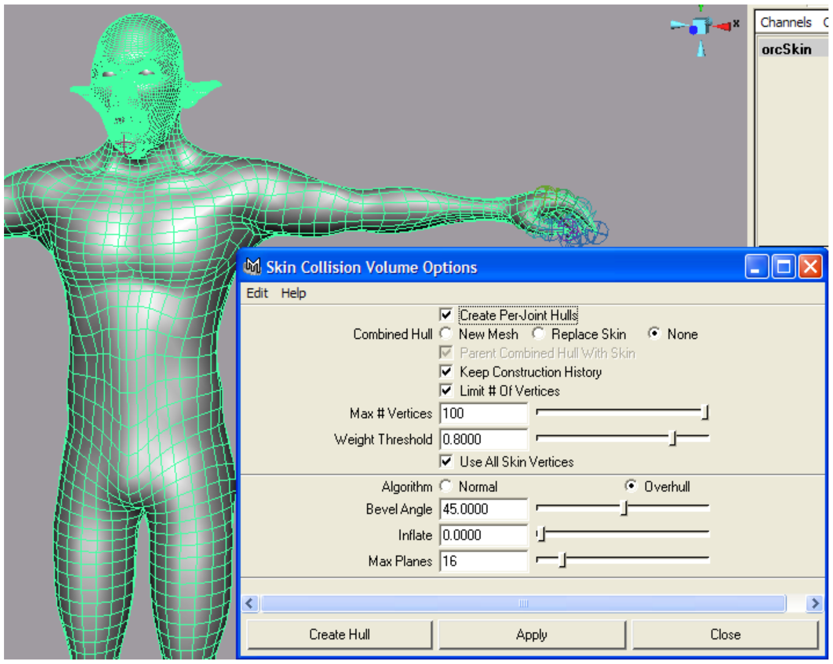
1. Convex Hulls → Skin Collision Volume (SCV),
2. Ragdolls → Extract the ragdoll meshes from SCV
3. Ragdolls → Create ragdoll rigid bodies
4. Ragdolls → Creating ragdoll rigid constraints.

After each step, we recommend that you examine and tweak the intermediate results to ensure that the final ragdoll has the desired level of detail and quality of animation. Finally, once the constraints are created using the provided scripts, angular limits can be set manually for each constraint.


Convex Hulls
Ragdolls
Collision Shapes
Create Box
Extract ragdoll meshes from SCV
Create ragdoll rigid bodies
Create ragdoll rigid constraints

Skin Collision Volume (SCV)

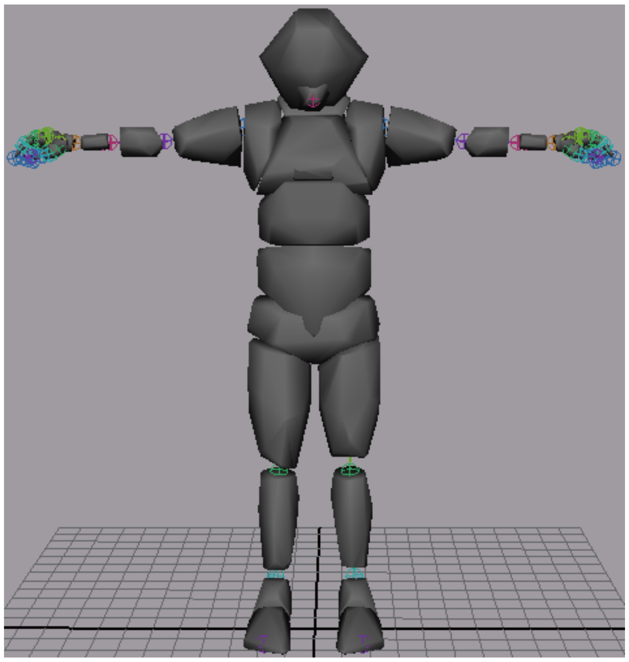
Select the skinned mesh of a skinned character, and then load the Skin Collision Volume options. The next figure shows the options displayed for the orc character:



[✓] Create Per-Joint Hulls
Combined Hull: New Mesh / Replace Skin / [✓ None]
[✓] Parent Combined Hull With Skin
[✓] Keep Construction History
[✓] Limit # Of Vertices
Max # Vertices: 100
Weight Threshold: 0.8000
[✓] Use All Skin Vertices
Algorithm: Normal / [✓ Overhull]
Bevel Angle: 45.0000
Inflate: 0.0000
Max Planes: 16

We recommend that you keep the construction history enabled to facilitate tweaking of these parameters. Since the recomputation of the SCV is expensive, the construction history will automatically be deleted during the next step (extraction of ragdoll parts).

Now press the *Create Hull* button. This may take several seconds depending on the complexity of your model. Once the initial computation is done, you should see that many shapes (typically one per joint) are now approximating the original skin. Hide the skin, so we can see these collision volumes better:



To get good results, follow these guidelines:

- It is acceptable for contiguous body parts to interpenetrate, since they will be connected through rigid constraints. Larger interpenetration should be avoided if possible.
- The *weight threshold* is the most useful parameter. Play with different values until your character is closely approximated, without excessive interpenetration.
- Do not move or rotate the shapes at this point… but you can safely sculpt vertex positions during later steps.

Extracting Ragdoll Meshes

The skin collision volumes are created in the skeleton of the skinned character, next to each joint that they correspond to. This step extracts the ragdoll parts and groups them in a flat list, in a ragdoll group. It also eliminates the construction history of the ragdoll parts.

After this step is done, you can safely sculpt the shapes to better approximate the skin or to reduce interpenetration.

PhysX, like most game physics engines, is quite sensitive to mass ratios. If your ragdoll has many small parts, like the fingers in this example, it is recommended that you get remove the smallest parts to reduce the computational load. For example, you can delete the fingers and sculpt the wrist shape to cover the entire hand. Be careful to only delete parts that are extremeties. Deleting intermediate body parts (for example, the abdomen or the neck) will probably cause the ragdoll to become disconnected. If you find that you have too many useless parts, you should probably go back and reduce the number of joints of your skinned character.

Creating the Ragdoll Rigid Bodies

This script takes each shape in the selected ragdoll group and replaces it by a rigid body. After this step is done, you can adjust the mass or density of each rigid body (for example, set them all to the same density), the friction, bounciness, and so on.

Playing the animation at this stage will often produce explosive results, due to the interpenetration of the rigid bodies. The next step, adding the constraints, will generally reduce or eliminate this problem.

Creating the Ragdoll Rigid Constraints

This step attaches contiguous body parts using rigid constraints. By default, the rigid constraints are very stiff; they have no angular or linear degrees of freedom. After this step, you should establish the angular limits for each rigid constraint. Watch the ragdoll construction movies for details. Here are some guidelines:

- If the constraint corresponds to an anatomical hinge (such as an elbow or knee), you should rotate the constraint so that the X axis (called twist in PhysX, shown in red) is aligned with the hinge. The twist axis has an upper and lower limit, which makes it convenient for body parts like the elbow that should only be allowed to rotate in one direction.
- When setting hard angular limits, you should set the spring values to 0 to ensure that they are considered as *hard limits* by the PhysX solver. Otherwise the limit will be *soft* and the ragdoll will often exceed it during fast movement.
- You will generally want to increase the solver iteration count on the ragdoll rigid bodies, to ensure that the ragdoll behaves as expected. Values of 32 to 64 are common. Values that are too low will cause the ragdoll to sag and stretch, while values that are too high may cause jumpiness.

Using Cloth

Creating a Cloth

PhysX cloth objects can be created through *PhysX→Cloth→Create Cloth*, after selecting a polygonal mesh. A new mesh will be duplicated and act as a cloth object. Behind the mesh, there is a cloth node creates and controls the simulation. The original mesh will be recorded as the initial pose of the cloth simulation. Every time the animation is rewound (by setting the timeline to frame 1), the cloth will be reset to its initial pose.

PhysX supports collision detection between cloth and active or passive rigid bodies. Cloth objects can collide with each other too. Cloth self-collision is also supported.

Deleting a Cloth

Select the cloth mesh and press the 'Delete' key to remove it, along with the related cloth node.

Editing Cloth

A cloth object can be translated, rotated, and tweaked after being created. When the mesh and/or mesh vertices of the cloth are moved to a new position, by hand or through simulation, select `PhysX→Cloth→Set Initial Pose` to set the current state as the starting point of cloth simulation. This change only has an effect after replaying the animation.

Attach Cloth Points

Select some vertices of the cloth and ctrl-select the rigid body as target of attachment, pick `PhysX→Cloth→Create Constraint` to create the attachment, which is a locator node drawing a cross hair on each vertex being constrained. This change only has an effect after replaying the animation.

A Selection Tip

Since a physical cloth is associated with a graphical representation, it can be difficult to select individual vertices on the physical cloth in their initial position (where they overlap). One way around this problem is to advance the animation by a few frames, so that the physical cloth and graphical cloth are more separate.

Remove Cloth Attachment

Select the locator node of the constraint and press 'Delete' key to free the cloth points it is attached to. This change only has an effect after replaying the animation.

Cloth Attributes

The physical cloth has the following attributes:

- **Thickness:** thickness of the cloth
- **Bending Stiffness:** stiffness against bending, only has effect when bending is enabled
- **Stretching Stiffness:** stiffness against stretching

- **Damping Coefficient:** spring damping of the cloth, only has an effect when damping is enabled
- **Friction:** coefficient of damping velocity of the cloth points in contact
- **Pressure:** how much the cloth volume will expand, comparing to the size of resting volume, only has an effect when pressure is enabled
- **Bounciness:** bouncing factor when the cloth pushes rigid bodies, only has an effect when two way collision is enabled
- **Tear Factor:** how long the cloth can be stretched before being torn apart, comparing to rest length, only has an effect when tearing is enabled
- **Current Time**
- **Static:** disable simulation and make the cloth static
- **Enable Bending**
- **Enable Damping**
- **Two Way Collision**
- **Enable Pressure**

Limitations

- Cloth simulation will stop if new vertices or edges are added to the mesh.
- A cloth constraint must have a target rigid body. Turn on the 'World Space' attribute of the locator if you want to attach cloth points to world positions.
- Self-collision on cloth isn't working in the plug-in, even though it is supported by the PhysX SDK. This will be fixed in a future version.
- No cloth visualizations appear.
- Two-way collision with passive rigid bodies is buggy.
- No texture coordinates available for a tearing cloth.
- Tearing isn't supporte

Note

To workaround cloth wrong behavior, a rigid body named nimaHelperRB is added. Don't worry, it will never affect your simulation, its size is 0.001*0.001*0.001 and is moved to coordinate (10000, 10000, 10000).

We don't support Tearing factor for now for the error caused by large memory consumption.

Baking a Simulation

The *PhysX→Bake All* and *PhysX→Bake Selected* commands are used to convert a simulated active rigid body into a passive rigid body whose position and orientation are keyframed for each frame.

Prior to running these commands, you should select the rigid bodies of interest (in the case of *Bake Selected*) and make sure that the time line covers the animation range that you want to bake. Note that Maya will be unresponsive while the baking occurs.

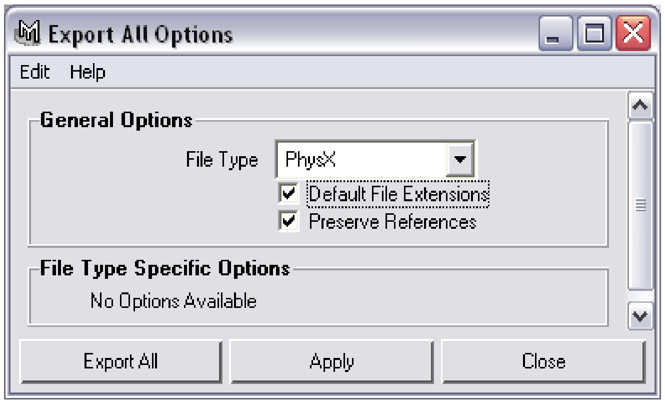
We recommend that you make a backup copy of your scene prior to baking.

Exporting PhysX Assets

It may be desirable to export physical assets to be further processed outside of Maya. For example, game studios can author rag dolls and game levels in Maya, and then export the results into their game engine. Several approaches exist, but currently the two most common ways are to do a PhysX export (in ASCII, binary, or a subset of COLLADA), or export a common-profile through ColladaMaya.

PhysX exporting leverages Ageia's nxuStream2 utility library to export the PhysX state in one of three formats: xml, binary or a subset of COLLADA 1.4.1 Physics. It is then easy, through nxuStream2, to import these assets in a game engine or viewer. This approach is simple but still has some limitations. For example, because PhysX doesn't have a concept of key-frame animation, any kinematic animation is lost in the process.

The PhysX export is exposed through two mechanisms. First, you can access it through the *File→Export All* menu:


Export All Options

General Options

File Type: PhysX

[✓] Default File Extensions

[✓] Preserve References

File Type Specific Options

No Options Available

After selecting the *PhysX* file type, press *Export All* or *Apply* to bring up a file dialog. When specifying the filename, make sure to append a recognized extension at the end.

The recognized extensions are:

Extension	Format
.nxb	PhysX binary serialization Note: not recommended for anything but final production content because it is not forwards or backwards compatible.
.xml	PhysX xml serialization
.dae, .collada	COLLADA-like serialization. Only saves the physics portion of the data. The graphics portion is not generated.

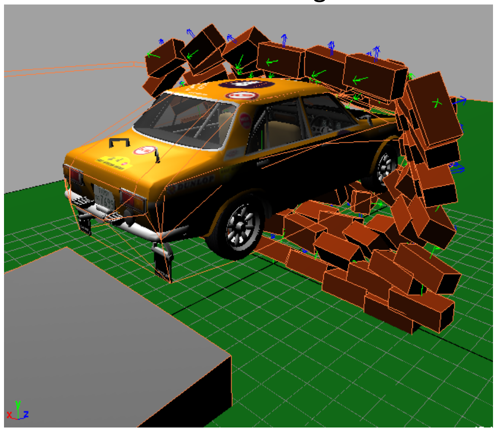
The plug-in also exposes the *nxExport* command to allow exporting automatically from MEL or from the command line.

ColladaMaya Export

ColladaMaya is a full-featured COLLADA translator for Maya, developed by Feeling Software. ColladaMaya recognizes plug-in-created nodes and can export the physics assets of a physical scene using the Common physics profile. The export includes only rigid body and constraint, no cloth or force-field.

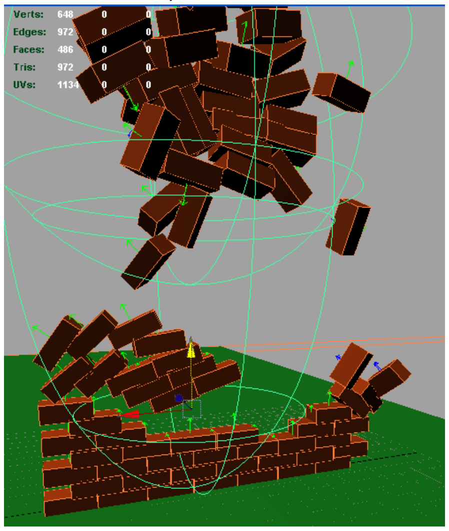
Car Crash

This scene demonstrates collisions between convex and non-convex rigid bodies. A non-convex rigid body car slides along a ground plane and ramp. In front of it, a wall of bricks waits eagerly. The car then breaks through the brick wall.



Tornado

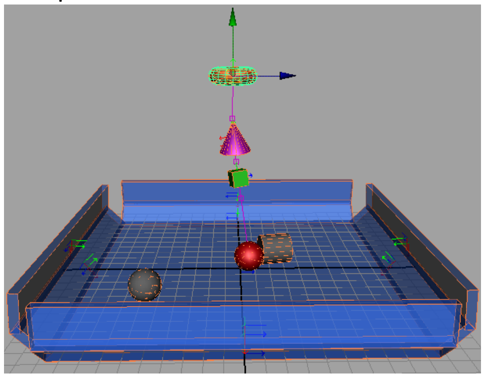
The same brick wall is now affected by a tornado made from two force fields (an attraction field in the bottom, and a vortex field on top). Use the interactive playback to try it out.



```
Verts:   648    0    0
Edges:   972    0    0
Faces:   486    0    0
Tris:    972    0    0
UVs:     1134   0    0
```

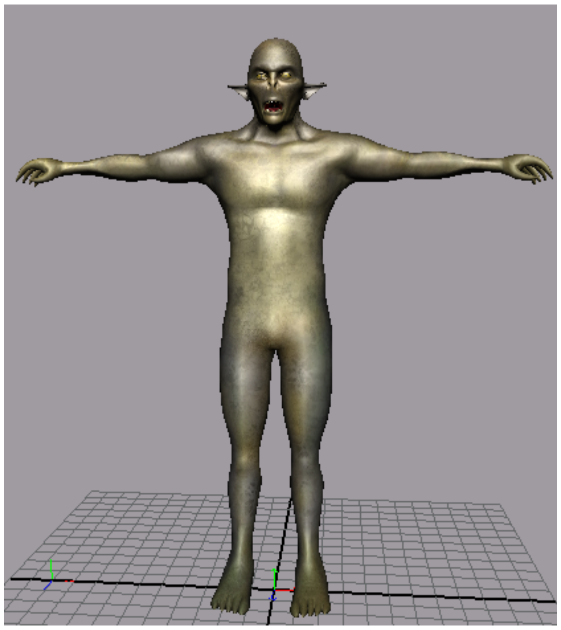
6dof_constraint

This is a quick demonstration of 6 degrees of freedom rigid constraint. A bunch of rigid bodies are connected in a chain. The top rigid body (torus) is passive and can therefore be interactively manipulated to demonstrate constraints.



Orc

This free model was adapted from Blacksmith3d.com. It was downloaded for free from TurboSquid. Two versions of this scene are provided: one is skinned and ready to be converted into a ragdoll. The other is already rigged as a ragdoll.



NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com