Bullet 3DS Max Plugin User's guide

August 9, 2010

Original document by NVIDIA Corporation, modified for Bullet by Erwin Coumans

Table of Contents

Introduction

The document describes how to use the Bullet Physics Plugin to edit objects in 3ds Max with physics properties.

The Plugin helps user to create Bullet rigid bodies without write any codes. Then the actors can be exported out and then be used in other applications like demos, games.

Source code

The source code of the Bullet 3ds Max plugin is located at http://dynamica.googlecode.com together with the Dynamica Bullet Maya plugin. You can compile the source code using Microsoft Visual Studio 2008 using the Dynamica\Extras\3dsmax\compiler\vc8win32\Plugin281.sln solution for 3ds Max 2008 and later.

License

This worked is derived from the PhysX 3ds Max plugin, released by NVIDIA on sourceforge.net under the MIT license. Our modifications to replace PhysX by Bullet are also released under the MIT license.

Installation

Installation of the latest version should be easy, simply download the installer and install it into the 3ds max folder. If you are not using the installer, then the following files should be installed for full functionality:

3dsmax\plugins\pxplugin.dlm

3dsmax\scripts\startup\startup.ms

3dsmax\scripts\physx\*.ms

The plugin links statically to the Bullet Physics SDK, so there are no other dependencies.

General

The most important panel to understand, when working with the Bullet Physics Plugin, is found under the "utilities" toolbar and is called "Bullet control panel". It should have automatically started when 3ds Max started or else its absence is an indication that the installation of the plugin did not go as planned. One way to see if the C++ part of the plugin has been correctly installed and loaded is to open the maxscript listener and type "PX" (without the quotes), that is the plugin object, through which the script parts of the plugin communicates with the Bullet Physics SDK.

Typing "px.debugPrint()" will give you an indication of what objects are currently added to the Bullet dynamics world. This list is not totally complete but it can function as a help when debugging problems when starting to learn the plugin. e.g., it would be possible to see if you managed to make the plugin use the same referenced triangle mesh for several Bullet rigid bodies by looking at the number of triangle meshes currently in the simulation. For large scenes it is important that you manage to share triangle meshes (if there are instances of the same mesh). Please note that now you should not use triangle meshes for dynamic objects.

One more, VERY IMPORTANT point is that you should never use scaling on any transforms involved in Bullet objects. If you want to change the size of e.g., a sphere, then you need to do so by changing the radius of the sphere, not by scaling it. Although the Bullet SDK handles scaling, the original plugin was written for PhysX so no scaling is supported. If you scale objects, things might look good at first but then you will see generally strange physical behavior when simulating and you might end up with asserts because there will be infinite number or divisions by zero after a few seconds of simulation.

Supported Bullet Features

The following part introduces the ways to create Bullet objects. Here the document only covers how to create those objects in Max. User is encouraged to read Bullet Physics SDK documentation to understand Bullet objects (or rigid bodies) better.

Rigid Bodies

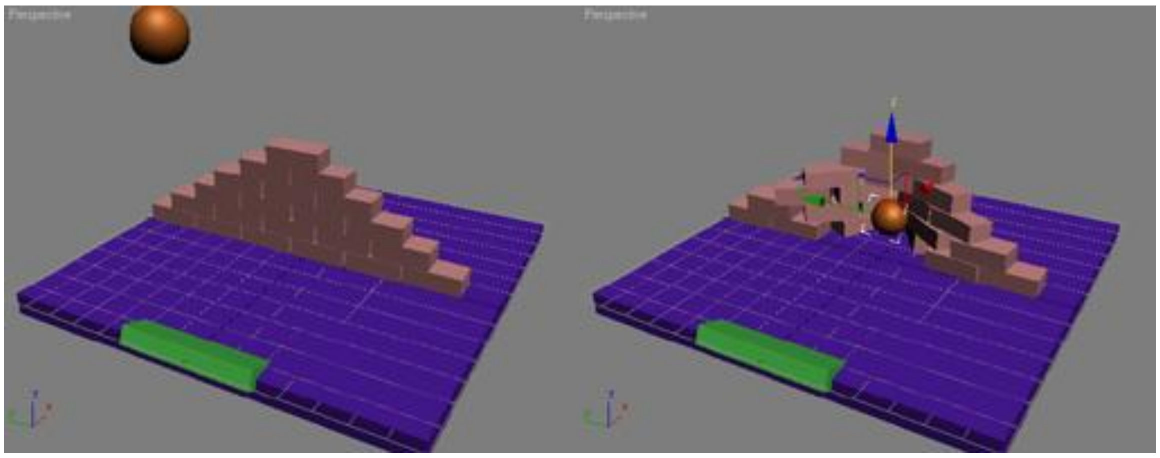


Figure 1 Rigid Body

The building block object of a physics simulation is the rigid body. The Bullet Physics SDK refers to this as a btRigidBody. This is a single physical unit that does not deform and has no inner moving parts. Every part of a rigid-body moves in unison. A dynamic rigid body will collide with other rigid bodies in the scene. A rigid body can be specified by any type geometry created in Max.

A node by itself can be a rigid-body or a number of nodes can be combined together to make a single rigid body. This is done by putting them together in a group. To do this, simply select all the relevant objects then go to the Group menu and select the first item “Group”.

The geometry types that the Bullet 3ds Max Plugin supports directly includes sphere, capsule, box, convex mesh and general triangle meshes. Other types of geometry, such as cylinder or cone, are approximated using convex or general triangle meshes.

For most physics engines it is most efficient to use convex pieces to define the collision volumes of rigid bodies. Therefore, it can be a good idea to break up a concave object into convex pieces and group them into a compound rigid body.

Grouping works within an articulated skeleton to simplify its physics model. For example, you may not need/want to physically simulate the fingers of a hand. Simply select the hand and its children (fingers) and group it. This grouping can also work with the internals of a skeleton. There may be a series of vertebrae bones used for skinning and to animate subtle gestures. These could be grouped into a single rigid body for dynamics purposes.

Rigid bodies have a number of physical properties that can be specified. Details are provided in the descriptions of the utilities in later sections of this document.

First Trial

Please follow the steps to try a simple RB creation process.

1. Create a sphere
2. Go to "Utilities" and then "Bullet control panel"
3. Select the sphere
4. Press the button named "Make physical / edit"
5. Now that button will change into a drop down and a new panel will emerge
6. In the dropdown, you can specify the type of object that you are creating. This will default to "dynamic rigid body".
7. Make sure that the dropdown really says "dynamic rigid body"
8. In the new panel, you can change properties such as mass and friction
9. When you are satisfied with the properties of the rigid body you are creating, press the "add selected" button
10. Now there should have been a rigid body created.
11. Open the maxscript listener to see if there were any error messages or warnings
12. Try writing px.debugPrint() in the maxscript listener to list the contents of the Bullet Physics SDK
13. Go back to the Bullet panel and press "continue"
14. Now the sphere should start falling "down", default gravity direction can be set in the "Bullet parameters" panel

Before you go on, you can press the "remove all" button, which will clear the Bullet dynamics world and let you add new objects to the simulation.

If you got through all those steps, well, then you have just created a dynamic, single shape rigid body (btRigidBody). If you want to create a compound actor (a rigid body with several shapes), then you do like following steps:

1. Create two spheres (close to each other but not in each other's center)
2. Select both and group them, then select the group
3. Press "make physical / edit" in the Bullet panel and make sure that it's a "dynamic rigid body"
4. Press "add all physical", that will add all physical objects to the simulation, not just the ones you have selected
5. Press "continue". The two spheres should not interact with each other and stay fixed relative to each other

If the two sphere push each other apart, then you did not manage to make the actor compound but if they don't or if you have created the spheres at a distance to each other, then you can't really tell if they are in the same actor or not, until they hit another object. Let's also create an object to simulate the ground, so that actors can fall down and come to rest on it.

First press "reset", so that objects are reset at the position where they were created. If you "add all" again after having simulated a bit, the "original" position of the objects will be the one they had when last pressing the "add all" or "add selected" buttons.

1. Create a large box "under" the dynamic objects
2. Make this box a "static rigid body"
3. Add to simulation
4. Start *continue* simulation and watch the objects interact.

Triangle meshes

The Bullet SDK support both static concave triangles meshes (btBvhTriangleMeshShape) and concave moving triangle meshes (btGImpactMeshShape).

Currently there is no sharing of triangle meshes implemented in the Bullet 3ds Max plugin yet. Once we ported this feature from the PhysX version, here is how to enable sharing:

A quick test on how to share a trimesh:

1. Start with a clean scene, press "remove all" on the PhysX panel and then new the scene
2. Create a torus with 10 segments and 10 sides
3. Select the "move tool"

4. Drag the torus to the side and drop to make a copy but you must make the copy an "instance", otherwise mesh sharing won't work
5. Select both tori
6. Press "make physical" and make it a "static rigid body"
7. Add the object to the simulation
8. Now go to the maxscript listener and write "px.debugPrint()", if you were successful, there should only be one triangle mesh but with two references to it

Sometimes the "make physical" button stopped working although it is seldom to happen. If this happens, close the "Bullet utility" by pressing the "close" button and then issue the following command in the maxscript listener: "addRollout px_control", that will open the Bullet control panel again and now the button should be working.

D6 Joint (not supported yet)

Being able to joint rigid bodies together is an important feature in a physics engine, so this is of course also supported through the 3ds Max plugin. This support is implemented only for the "6 degrees of freedom joint", which is a single joint object that can be configured to work as most other joint types that are usually available in physics SDKs.

Steps to create a D6 Joint

1. Create a Kinematic RB object, taking a sphere as example
2. Create a Dynamic RB object, taking a box as example

   These two actors will be connected with a D6 Joint.

3. Choose ‘Helper’ in ‘Geometry’ tool as the following picture

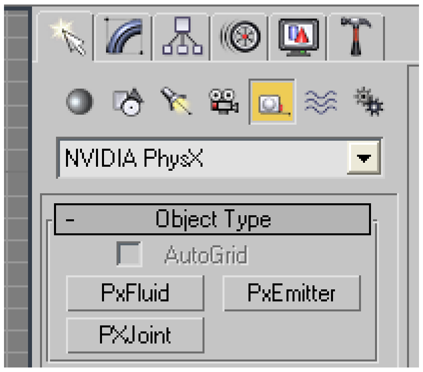
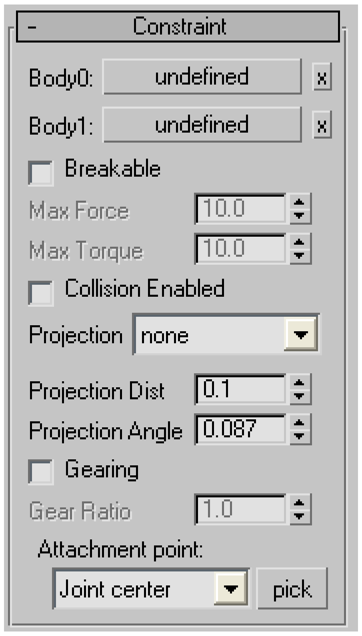


Figure 2 Helpers

4. Choose ‘NVIDIA PhysX’ inside the drop-down list. It will show a window include 3 buttons, ‘PxFluid’, ‘PxEmitter’ and ‘PxJoint’.
5. Press button ‘PxJoint’ and create the joint object.

6. Choose the Joint that you created and choose 'Modify' Tool. Then it shows the following window.



- Constraint

Body0: undefined

Body1: undefined

Breakable: [ ]

Max Force: 10.0

Max Torque: 10.0

Collision Enabled: [ ]

Projection: none

Projection Dist: 0.1

Projection Angle: 0.087

Gearing: [ ]

Gear Ratio: 1.0

Attachment point: Joint center

Figure 3 D6 Joint Constraint

7. Press button 'undefined' beside label 'Body0' and then choose the Kinematic sphere
8. Press button 'undefined' beside label 'Body1' and then choose the Dynamic box
9. Modify other parameters of the Joint like Swing, Twist etc.
10. Add all of them to physics

Please see Sample 'simpleJoint'.

Create joints using 'Joint Specification' (not supported yet)

The Joint Specification rollout specifies the joint constraint between the current object and the current object's parent. Should the object not have a parent, then the constraint is with respect to the static world. Note that not all joint-body systems can be built this way but it is a good way to build many systems such as biped rag dolls or single objects with simple constraint behavior such as a door on a hinge.

Joint Specification is not shown directly after running Max. First click the MAXScript button in Utility page (Hammer). Then in the MAXScript rollout, choose 'Joint Specification' in the combo box. See pictures:

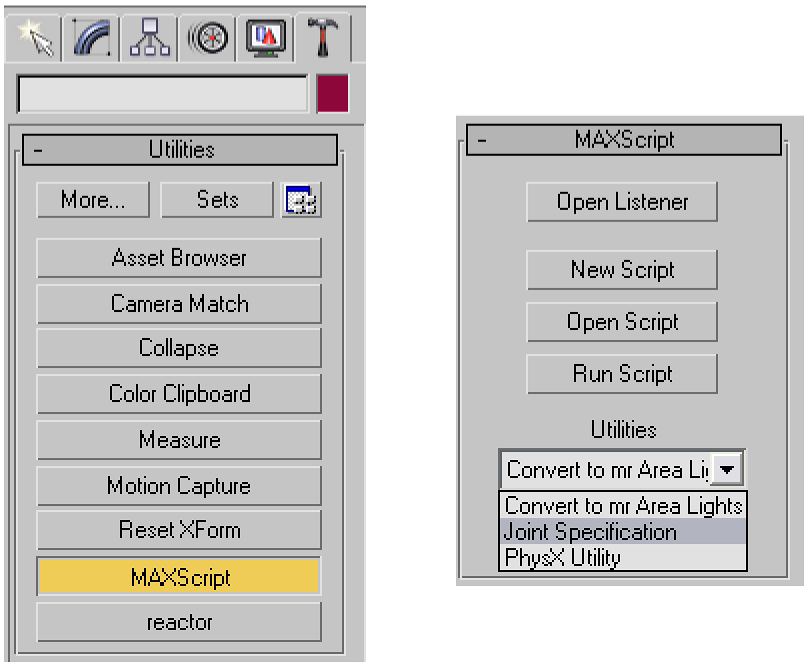


Figure 4 Run 'Joint Specification'

Deriving Joints from Animation Data

If the movement characteristics of an object are simple enough then the easiest way to specify this is for an artist to simply animate the object through its range of motion. In other words, the way something moves in Max is how it should move in the game.

Right now there are two options for calculating a joint from animation. If you know that you have a hinge joint (1 degree of freedom) then click on the "Calc Hinge Joint From Anim" button. If the joint requires more maneuverability then calculate a sphere joint from it. These routines cycle through the key frames and try to determine the joint characteristics. Single 1 degree of freedom joints (hinges) are easiest to derive, even for an arbitrary axis of rotation. So you don't even have to try to reorient the pivot to align it with the hinge. Just animate the object – that's all. More complicated rotational joints present more of a mathematical challenge due to the non-orthogonal nature of orientation space. Calculating a sphere joint is useful for computing the limits of motion for that type of joint.

Until our tools are more robust please try to keep your pivots aligned with the node's coordinate frame. For any geometry that is part of a group (compound rigid body) please keep the pivot at the node's origin. There is no benefit to having an offset pivot to something that can't move.

To speed up artist productivity, the joint editor can be applied to multiple selections. For example you can select the forearms and calves and calculate a hinge joint from the animations, then select the upper-arms and thighs and calculate sphere joints.

Editing Joint Parameters Directly

The individual joint parameter can be modified using a collection of UI elements that map directly to the joint specification part of the PhysX SDK data model.

Exporting

The plugin can export the Bullet objects in the binary .bullet format. The .bullet files can be imported into the Bullet SDK using the btBulletWorldImporter class.

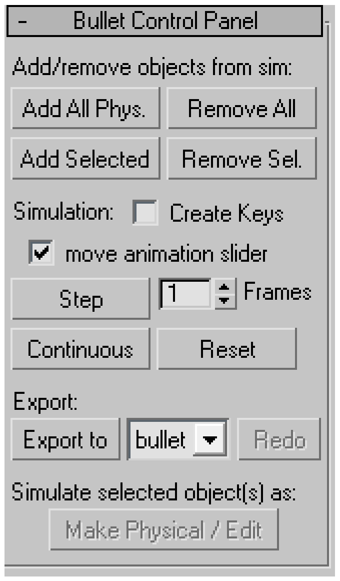


Figure 5 bullet export

Samples

Ballbounce

It shows three balls with different properties which cause them different behaviors after falling. The higher the Restitution is, the longer the ball bounces.

Frictionramp

It shows RB behaviors under different friction values.

Support

Forum and feedback

You can provide feedback in the discussion forum at http://bulletphysics.org When you can reproduce a problem or if you want to contribute improvements to the plugin, you can use the issue tracker at the googlecode repository at http://dynamica.googlecode.com

Questions and Answers

Why don't I see my node hierarchy in the exported file?
A collection of rigid bodies might be represented in a tree hierarchy in max. For example, the (fattened) bones of a biped will be become the rigid bodies for a rag doll. In this 3ds max context it makes most sense to think of positions and orientations of a node in the parent's frame (coordinate system). This parent-child relationship is not meaningful in a dynamics setting. The relationships between rigid bodies are nothing more than physical constraints such as joints and body-body collision rules. Therefore rigid bodies are exported in model or global coordinates and hierarchy information is not reflected in the physics model.
There are there multiple ways to define physics properties, which method works?
The best approach is to use the PhysX Properties Editor. You can also specify properties such as the mass of a shape using Reactor's dialog or the user property editor.
What if someone else, without the PhysX plugins and dlls, edits and saves my max file, will the physics specifications be lost?
No worries. Edit away. That's why we used the user property facilities of max instead of relying on modifier variables.
What about other modeling packages such as Maya or Softimage XSI?
PhysX is built into version 5 of XSI Essentials and Advanced.
For Maya we also provide a plug-in on our developer support website.
When we move a file from one modeling package to another we sometimes sacrifice information, usually the higher level information about our models. What will happen to the physics data?
We are working with the Collada consortium to standardize the physics representation. The objective was that by using Collada, the physics information will not be lost when using different applications to edit the same models. However this is a long term goal.
Does our rendering tool chain have to migrate over to Collada in order to use the Physics exporter?
No, this isn't a requirement. Within a Collada file, the physics data sits separate-from but parallel-to the rendering data. The physics data will easily work with your existing graphics data in any format. There are also other export options including PML and ASCII/Binary Serializations as well as the ability to write custom exporters by traversing 3DS Max's nodes.
Why didn't you create subclasses instead of using user props?

In addition to the problems with competing versions and sharing .max files with others, using subclasses would limit the types of objects that can become physics objects. Its better to allow any type of geometry be used as a rigid body.

Why is the user property "Ellasticity" misspelled?

This is in order to interface seamlessly with Reactor. When data is exported, this is corrected to properly fit the physics data schema.

I changed the user property "mass" but my change didn't show up. What happened?

Be careful here. Max is usually case insensitive. However, the user property hash strings are case sensitive. Look for "Mass" instead of "mass".

I like to use Max's groups for collections, other than compound rigid bodies, to help clean up and manage my scene. Can I do that without turning all my groups into single rigid bodies?

Unfortunately, we don't support that right now.

Why does setting mass to 0 make something static?

Mathematically speaking the mass should be set to infinite. However, the user interface elements of max (such as spinners) don't support that. Since zero mass is meaningless, it is a convenient way to specify something is static. The best thing to do is to specify 'static', 'dynamic', or 'kinematic' using the radio selection on the PhysX properties editor.

How come I can't change the mass of a group (compound rigid body)?

The mass of a group is a derived value. Mass is specified in each of the shapes that make up the compound rigid body. A physics engine has to also know how the weight is distributed within an object in order to compute other important inertial properties that affect how an object rotates.

How can I make the name of a rigid body different than the shape inside used to describe it?

Simply Group the node. Yes, you can have a group of one. The name of the group is the name of the rigid body, the name of the initial node determines the shape name.

What does 'SW' button (inside 'Geometry Tool') for?

It uses the current SDK skin width to set the skin width. If current SDK skin width is not available, then it sets the value as 0.025.

Known Issues

- Sometimes user must press 'reset' several times to make all objects go back