Behavior Reference Guide

Version 1.0

Published in Ireland.

Havok.com Inc.

www.havok.com

Table of Contents

Havok Xtra Behaviors 1
Creating you own Havok behaviors 1
Additional Behavior Creation Hints 2
Quick and Dirty *3*
Correct Solution *3*

Behaviors Supplied 4
Control Behavior Library 4
Move Model *4*
Apply Constant Force *4*
Push Model *5*
Apply Constant Impulse *5*
Set Gravity *5*
Antigravity *6*
Setup Behavior Library 6
Make Moveable Rigid Body *6*
Make Fixed Rigid Body *7*
Physics (HKE) *7*
Physics (no HKE) *7*
Make Spring *8*
Make Angular Dashpot *9*
Make Linear Dashpot *9*

1 Havok Xtra Behaviors

Behaviors are a powerful tool that you can use to rapidly construct Director movies. They use libraries of frequently used actions or controls. You can reduce the pain of authoring with the Havok Xtra through the use of supplied behaviors, or by creating your own for re-use. This section of this document covers creating your own behaviors, while the next section outlines the behaviors supplied with the Havok Xtra.

1.1 Creating you own Havok behaviors

You might need to create your own behaviors to extend the functionality of the physics simulation. The following example illustrates the creation of a gravity behavior, a behavior that applies a force in a downward direction as specified by the user.

Define the user-defined properties that this behavior requires which in this case is **pForce**, the gravity force to apply at each step:

```
property pForce
```

Assign some general properties to keep track of things:

```
property pSprite -------- Sprite. This sprite.
property pHavok --------- The instance of the Havok physics simulation
property pModel --------- User to iterate over the 3D models later
```

You need to get handles to the sprite and the Havok simulation at the start. You can do this in the **beginSprite** function:

```
on beginSprite(me)

 -- through the sprite get a handle to the physics system
 pSprite = sprite(me.spriteNum)
 pHavok = pSprite.pHavok

end beginSprite
```

Next is the core of the gravity behavior functionality. In this example you only need to worry about the **enterFrame** handler. This determines the force to apply to each model, depending on the scaling factor you use. It iterates over all the models, applying that force to each in turn:

```
on enterFrame(me)

 f = pForce

 repeat with j = 1 to pHavok.rigidBody.count
  rb = pHavok.rigidBody[j]
  rb.applyForce( f )
 end repeat

end enterFrame
```

Finally, add the **isOKtoAttach** and **getPropertyDescriptionList** handlers to create a drag-and-drop behavior, and the behavior is complete:

```
on isOKtoAttach(aScript, aSpriteType, aSpriteNum)

 case aSpriteType of
  #Graphic:
   case sprite(aSpriteNum).member.type of
    #shockwave3d:
```

```
    return(TRUE)

   #text:
    if sprite(aSpriteNum).member.displayMode = #mode3D then
     return(TRUE)
    else
     return(FALSE)
    end if
  end case

 #script:
  return(FALSE)
end case

return(FALSE)

end isOKtoAttach

on getPropertyDescriptionList(aScript)

 if the currentSpriteNum > 0 then

  tGPDList = [:]
  tGPDList[#pForce] = \
  [\
  #comment:"Gravity",\
  #format: #vector,\
  #default: vector( 0, 0, -9.81 )\
  ]

  tGPDList[#pScale] = \
  [\
  #comment:"Scale",\
  #format: #boolean,\
  #default: true\
  ]
  return(tGPDList)

 end if

end getPropertyDescriptionList
```

1.2 Additional Behavior Creation Hints

The gravity behavior in the previous example applies its force to all the models in the scene each time the Havok Xtra calls it by invoking the **beginSprite** function. This is fine and works well for simulations where the **substeps** property is set to zero (**havok.subSteps = 0**). However, Havok does not call the gravity behavior often enough for simulations with 1 or more substeps. The Xtra only calls behaviors for every new frame and not during internal Havok substeps.

For example, if the **substeps** property is set to four, Havok calls the behavior only once for every fourth step of the simulation. Effectively this is similar to applying 25% of the required force, so gravity would not be strong enough. In many cases this is not noticeable. You could simply increase the force. However if you attempt to apply real force values (in world scale units) then you do not get the desired result.

There are 2 ways to solve this problem. The first is quick and dirty, while the second is correct, but a more long-winded approach:

Quick and Dirty

If you know how many sub steps the Havok engine takes, scale the applied force appropriately. Modify the **onBegin** sprite handler as follows:

```
f = pForce * ( pHavok.subSteps )
```

This is a better solution than the first attempt but does not quite work correctly. Four steps of the physics simulation elapse before Havok applies a larger force. The larger force is applied less frequently, when there should be a smaller force at each step.

For each step where Havok does not apply the force, small errors in the motion of the model accumulate. This occurs particularly if other forces under the control of the physics engine act on the models.

Correct Solution

The correct solution is to use the Havok system's **substep callback** mechanism. If a handler is registered for this callback, Havok calls that handler for every internal step. To get the correct solution, call the gravity behavior in the substep handler. This guarantees the gravity force applies to all models for every step that the physics engine takes.

To register the handler, add the following call in **beginSprite**:

```
havok.registerStepCallback( #stepHandler, me )
```

Now add the **stepHandler** function and either add the gravity force as shown here, define a new function **addgravity** and call this in both **stepHandler** and **enterFrame**, or simply call **enterFrame**:

```
on stepHandler(me, timeStep)
 f = pForce
  repeat with j = 1 to pHavok.rigidBody.count
  rb = pHavok.rigidBody[j]
  rb.applyForce( f )
 end repeat

end stepHandler
```

2 Behaviors Supplied

The Havok Xtra includes two sets of default behaviors for use in creating Director movies. They are organized into Control and Setup behaviors. Control behaviors affect the simulation when running. You can use Setup behaviors to create physics simulations. These groupings are accessible through the **Havok Behaviors** tabbed page in the **Cast Library** dialog box.

This document describes each of these behaviors and their parameters.

2.1 Control Behavior Library

Control behaviors give control over the motion and behavior of rigid bodies and models during the simulation. Havok typically calls them for each frame of a Director movie.

Move Model

The Move Model allows you to pick a model and move it around under mouse control. You need to associate the behavior with a trigger to work, typically the left mouse trigger. This is a powerful behavior for interacting with a 3D scene. However, the move action's strength can pull objects through other objects if you apply a sufficiently strong pulling force.

Property	Description	Default
pModel	Model to move: you can limit the behavior to a single named model, or let it operate on the model picked by the mouse.	"Any Model"
pGroupName	The group to which this behavior belongs.	"Havok"
pStrength	The strength of the move behavior.	10
pDamping	The damping factor applied to the move.	1
pMassProportional	The force proportional to the mass.	True

Apply Constant Force

Applies a constant linear or angular force to the specified model at each frame

Property	Description	Default
pModel	The model to which you apply the force.	Model(1)
pForce	The force to apply at each frame.	vector(0,0,0)
pAngular	Sets whether the force applies a torque.	false
pScale	Sets whether the force should be scaled by the world-scaling factor.	true

Push Model

Applies a force of specified strength in the direction in which camera is currently looking. This behavior requires a **mouse event trigger**.

Property	Description	Default
pModel	The model to which you apply the push force. This is overridden at run-time.	"Any Model"
pGroupName	The group to which this behavior belongs.	"Havok"
pStrength	The strength of the push behavior	1000
pMassProportional	The force proportional to the mass	True

Apply Constant Impulse

This behavior applies a constant linear or angular impulse to the specified model at each frame.

Property	Description	Default
pModel	The model to which you apply the force.	Model(1)
pImpulse	The impulse that Havok applies at each frame.	vector(0,0,0)
pScale	Sets whether the force should be scaled by the world-scaling factor.	true
pAngular	Sets whether the impulse is angular or not.	false

Set Gravity

This behavior applies a gravity force to every model at each frame. This behavior cir-

cumvents the substeps problem described above by setting the **gravity** property for the Havok member.

Property	Description	Default
pForce	The model to which you apply gravity.	Vector(0,0,-9.81)
pScale	A boolean that indicates whether to scale gravity by the world scale.	true

Antigravity

This behavior applies a force equal in magnitude but opposite in direction to gravity. It applies the force to every model at each frame. Anti-Gravity uses a similar method to the Gravity to avoid any substep problems. y

Property	Description	Default
pForce	The model to which you apply antigravity.	Vector(0,0,-9.81)

2.2 Setup Behavior Library

You use setup behaviors during movie construction to speed up the process of creating physically simulated scenes, rigid bodies and specifying physical properties and behaviors.

Make Moveable Rigid Body

This behavior allows you to create a moveable rigid body associated with a specified model in the scene. You can specify the physical properties and mass of the model at this time.

Property	Description	Default
pModel	The model from which you construct the rigid body.	Model(1)
pRestitution	The bounciness of the rigid body	0.3
pFriction	The stickiness of the rigid body	0.3
pType	This is the representation you use for collision detection. Valid values are: Convex:Sphere, Convex:Box, Convex:Hull, and Concave.	"Convex:Hull"
pMass	The mass of the rigid body in kilograms.	1.0

Make Fixed Rigid Body

This behavior allows you to create a fixed rigid body associated with a specified model in a scene. You can specify the physical properties of the model at this time. This rigid body is fixed in space. It never moves, but other objects can collide with it. This is useful for creating scenery elements. Objects that do not move have no need for a mass property.

Property	Description	Default
pModel	The model from which you construct the rigid body.	Model(1)
pRestitution	The bounciness of the rigid body	0.3
pFriction	The stickiness of the rigid body	0.3
pType	This is the representation you use for collision detection. Valid values are: Convex:Sphere, Convex:Box, Convex:Hull, and Concave.	"Convex:Hull"

Physics (HKE)

This Havok physics behavior provides access to the rigid body simulation engine. This behavior's design assumes that the physics scene has been created from a **HKE** (Havok Exporter) file. The behavior automatically creates your required physics objects using information contained within the **HKE** file. So you can drag and drop a physics scene into the sprite.

Property	Description	Default
pHavok	The cast member that you attach the physics to.	pHavok
pTolerance	The global collision tolerance value used in simulation.	0.1
pTimeStep	The global timestep parameter used for each frame.	0.025
pSubSteps	The number of substeps simulated at each frame.	5

Physics (no HKE)

This Havok physics behavior provides access to the rigid body simulation engine. When you create a scene directly from Lingo, you make no reference to a **HKE** file. No physics objects are created initially. You must create and define physics objects with properties through Lingo.

Property	Description	Default
pHavok	The cast member that you attach the physics to.	pHavok
pTolerance	The global collision tolerance value used in simulation.	0.1
pTimeStep	The global timestep parameter used for each frame.	0.025
pSubSteps	The number of substeps simulated at each frame.	5
pScale	The world scaling factor that you use.	0.0254 (assumes inches

Make Spring

This behavior attaches a spring between two models. The strength of the spring defines how quickly the spring attempts to reach its rest length.

Property	Description	Default
pName	The name of the spring.	“Spring”+num
pModelA	The model at one end of the spring.	Model(1)
pPointA	The point in local space that attaches the spring to ModelA	Vector (0,0,0)
pModelB	The model at the other end of the spring.	Model (2)
pPointB	The point in local space that attaches the spring to ModelB	Vector (0,0,0)
pRestLength	The length at which the spring exerts no force.	1.0
pElasticity	The strength of the spring.	10.0
pDamping	The property that defines how quickly the spring comes to rest.	10.0
pActOnCompression	Determines how the spring exerts for if its length is smaller than its rest length.	true
pActOnExtension	Determines how the spring exerts force if its length is greater than its rest length.	true

Make Angular Dashpot

Angular dashpots are like stiff angular springs with a rest length of zero. You can use them to fix a model's orientation to a given desired orientation. Or you can use them to fix an orientation with respect to a second rigid body. If **pModelB** is set to "**none**" the orientation of **pModelA** is fixed with respect to the global reference frame (i.e. global x, y and z axes). The orientation of the dashpot is specified with an axis and angle and a zero rotation by default.

Property	Description	Default
pName	The name of the angular dashpot.	"AngularDash-pot"+num
pModelA	The model at one end of the dashpot.	Model(1)
pModelB	The model at the other end of the dashpot.	"None"
pAxis	The axis orientation that the dashpot tries to maintain.	Vector (0,0,1)
pAngle	The angular orientation that the dashpot tries to maintain.	0.0
pStrength	The strength of the dashpot.	1.0
pDamping	The property that defines how quickly the spring comes to rest.	0.1

Make Linear Dashpot

Linear dashpots are like stiff springs with a rest length of zero. You can use them to fix a model's position to a given desired position in the world, or to the position of a second rigid body. If **pModelB** is set to "**none**", the position of **pModelA** is fixed to a world co-ordinate.

Property	Description	Default
pName	The name of the linear dashpot.	"LinearDash-pot"+num
pModelA	The model at one end of the dashpot.	Model(1)
pModelB	The model at the other end of the dashpot.	"None"
pPointA	The point in local space that attaches the dashpot to modelA.	Vector (0,0,0)

Property	Description	Default
pPointB	The point in local space that attaches the dashpot to modelB.	Vector (0,0,0)
pStrength	The strength of the dashpot.	1.0
pDamping	The property that defines how quickly the spring comes to rest.	0.1