



©Copyright 1999-2014 Telekinesys Research Ltd. (t/a Havok). All rights reserved. ¹

¹ Havok.com and the Havok buzzsaw logo are trademarks of Havok. All other trademarks contained herein are the properties of their respective owners.

This document is protected under copyright law. The contents of this document may not be reproduced or transmitted in any form, in whole or in part, or by any means, mechanical or electronic, without the express written consent of Havok. This document is supplied as a manual for the Havok game dynamics software development kit. Reasonable care has been taken in preparing the information it contains. However, this document may contain omissions, technical inaccuracies, or typographical errors. Havok does not accept any responsibility of any kind for losses due to the use of this document. The information in this document is subject to change without notice.

Contents

1	Havok 2014.1.0 Animation, Physics_2012 and Common Migration Guide	4
1.1	Migrating to Animation, Physics_2012 and Common 2014.1	5
1.1.1	Common	5
1.1.2	Content Tools	5
1.1.3	Physics 2012	5
1.1.4	Animation	5
1.2	Migrating to Animation, Physics_2012 and Common 2013.3	6
1.2.1	Common	6
1.2.2	Content Tools	7
1.2.3	Physics 2012	7
1.2.4	Animation	7
1.3	Migrating to Animation, Physics_2012 and Common 2013.2	7
1.3.1	Common	7
1.3.2	Content Tools	7
1.3.3	Physics 2012	7
1.3.4	Animation	8
1.4	Migrating to Animation, Physics_2012 and Common 2013.1	8
1.4.1	Common	8
1.4.2	Content Tools	8
1.4.3	Physics	8
1.4.4	Animation	9
1.5	Migrating to Animation, Physics_2012 and Common 2012.2	10
1.5.1	Common	10
1.5.2	Content Tools	10
1.5.3	Physics	11
1.5.4	Animation	12
1.6	Migrating to Animation, Physics_2012 and Common 2012.1	12
1.6.1	Common	12
1.6.2	Content Tools	13
1.6.3	Physics	13
1.6.4	Animation	15
1.7	Migrating to Animation, Physics_2012 and Common 2011.3	17
1.7.1	Common	17
1.7.2	Content Tools	18
1.7.3	Physics	18
1.7.4	Animation	21
1.8	Migrating to Animation, Physics_2012 and Common 2011.2	21
1.8.1	Common	21
1.8.2	Content Tools	21
1.8.3	Physics	21
1.8.4	Animation	25
1.9	Migrating to Animation, Physics_2012 and Common 2011.1	25

1.9.1	Common	25
1.9.2	Content Tools	26
1.9.3	Physics	26
1.9.4	Animation	28
1.10	Migrating to Animation, Physics_2012 and Common 2010.2	29
1.10.1	Common	29
1.10.2	Content Tools	31
1.10.3	Physics	31
1.10.4	Animation	33
1.11	Migrating to Animation, Physics_2012 and Common 2010.1	34
1.11.1	Common	34
1.11.2	Content Tools	40
1.11.3	Physics	40
1.11.4	Animation	43

Chapter 1

Havok 2014.1.0 Animation, Physics_2012 and Common Migration Guide

1.1 Migrating to Animation, Physics_2012 and Common 2014.1

1.1.1 Common

There are no migration notes for Havok Common for this release. Please refer to the release notes for details of any specific issues that were resolved.

1.1.2 Content Tools

There are no migration notes for the Havok Content Tools for this release. Please refer to the release notes for details of any specific issues that were resolved.

1.1.3 Physics 2012

There are no migration notes for Havok Physics 2012 for this release. Please refer to the release notes for details of any specific issues that were resolved.

1.1.4 Animation

1.1.4.1 HKA-1613 : Additive animations with animated bone translations are incorrect

An error in additive animation calculation/application has been fixed. The previous method of calculation/application could cause visibly incorrect results and sub-optimal compression in cases where bone translations were animated. All new additive animations will be created in the new format. Old assets will still work as expected in Animation-only pipelines (see Animation Studio notes for caveats specific to that product).

If you want to create a deprecated format additive animation, you can still do so through the `hkaAdditiveAnimationUtility`. To make the deprecated format the default format (eg. for the Create Additive Animations filter), the Havok SDK/HCT libraries must be rebuilt with `HKA_USE_ADDITIVE_DEPRECATED` enabled (see `hkAnimation.h`).

1.1.4.2 New Ragdoll Controller Demo

There is a new Animation demo which serves as a good starting point for tweaking ragdoll behavior interactively. It is called RagdollControllerDemo for Physics2012 or NpRagdollControllerDemo for Physics.

1.2 Migrating to Animation, Physics_2012 and Common 2013.3

1.2.1 Common

1.2.1.1 hkReferencedObject now uses atomic operations instead of a critical section

The access to `hkReferencedObject`'s internal state (i.e. reference count, memory size, and flags) is no longer protected by a critical section. Multi-threaded coherency is now ensured via atomic primitives (i.e. compare and swap). `hkReferencedObject::addReferenceLockUnchecked()` and `removeReferenceLockUnchecked()` methods have been removed.

COM-2148	Change <code>hkReferencedObject</code> to use atomic operations instead of a critical section.
----------	--

1.2.1.2 Unified `hkJobThreadPool` and `hkThreadPool`

Previously, an `hkJobThreadPool` was used to process an `hkJobQueue`, and an `hkThreadPool` was used to process an `hkTaskQueue` via the `processWorkload()` methods. `hkThreadPool` can now process both with `processJobQueue()` and `processTaskQueue()` respectively. `hkJobThreadPool` and its subclasses, `hkCpuJobThreadPool` and `hkSpuJobThreadPool`, have been removed; you should use `hkCpuThreadPool` and `hkSpuThreadPool` instead.

The following `#includes` must be updated:

- `<Common/Base/Thread/Job/ThreadPool/hkJobThreadPool.h>` is now `<Common/Base/Thread/Pool/hkThreadPool.h>`
- `<Common/Base/Thread/Job/ThreadPool/Cpu/hkCpuJobThreadPool.h>` is now `<Common/Base/Thread/Pool/hkCpuThreadPool.h>`

COM-2471	Remove <code>hkJobThreadPool</code> , allow <code>hkThreadPool</code> to work with <code>hkTaskQueue</code> and <code>hkJobQueue</code> .
----------	---

1.2.1.3 Added a new library for image processing

`hkImageUtilities` is a new library that implements image operations. It exposes an abstract `hkImage` interface for manipulating images as collections of color channels. The `hkMeshTexture::Sampler` interface has been extended to allow access to a texture's pixel data through `hkImages`. Projects previously linking against `hkGeometryUtilities` require the addition of the new `hkImageUtilities` library to their linker input.

1.2.1.4 Miscellaneous Behavior Changes

Please refer to the specified items in the release notes for more details.

COM-2249	Enable MSAA by default in demo framework
COM-2402	hkAabb::isEmpty() should return true for flat AABBs.

1.2.2 Content Tools

There are no migration notes for the Havok Content Tools for this release. Please refer to the release notes for details of specific any issues that were resolved.

1.2.3 Physics 2012

There are no migration notes for Havok Physics 2012 for this release. Please refer to the release notes for details of specific any issues that were resolved.

1.2.4 Animation

There are no migration notes for Havok Animation for this release. Please refer to the release notes for details of specific any issues that were resolved.

1.3 Migrating to Animation, Physics_2012 and Common 2013.2

1.3.1 Common

There are no migration notes for Havok Common for this release. Please refer to the release notes for details of specific any issues that were resolved.

1.3.2 Content Tools

There are no migration notes for the Havok Content Tools for this release. Please refer to the release notes for details of specific any issues that were resolved.

1.3.3 Physics 2012

There are no migration notes for Havok Physics 2012 for this release. Please refer to the release notes for details of specific any issues that were resolved.

1.3.4 Animation

There are no migration notes for Havok Animation for this release. Please refer to the release notes for details of specific any issues that were resolved.

1.4 Migrating to Animation, Physics_2012 and Common 2013.1

1.4.1 Common

Note : The `hkMath` vector classes are undergoing a redesign at the moment. This release has no interface changes to the vector types, but the old interface which is already marked deprecated since release 2011.1 will be removed in the next release. Please prepare any code using `hkVector4`, `hkSimdReal`, `hkTransform`, etc. to not use the old methods anymore.

The deprecated methods are removed from the math interfaces by defining `HK_DISABLE_OLD_VECTOR4_INTERFACE`. Put this define before any includes of Havok libs and you will get compile errors in places where the old interface is used.

1.4.1.1 Single and double precision vectors

This release prepares for the introduction of simultaneous use of single and double precision vector math. Both types are SIMD accelerated on platforms supporting it. The next release will expose the corresponding interfaces and offer conversion tools. For this release, no changes in client code using `hkVector4`, `hkSimdReal` etc. are required.

Note : If you have custom conversion operators in `hkVector4`, `hkSimdReal`, `hkMatrix3`, `hkMatrix4`, `hkTransform`, `hkQuaternion`, `hkQTransform`, `hkQsTransform` you need to put your operator implementations in the files for the single precision implementations. These are identified by the letter 'f' in the name, ie. `hkVector4f.h` `hkTransformf.h`

1.4.2 Content Tools

There are no migration notes for the Havok Content Tools for this release. Please refer to the release notes for details of specific any issues that were resolved.

1.4.3 Physics

Havok 2013.1 brings major changes to Havok Physics. Please read this section carefully.

1.4.3.1 Existing Physics SDK renamed to "Physics 2012"

The renaming of "Physics" to "Physics 2012" has been done through the product, including: packages, documentation, tools, demos and some source code. For the core Physics 2012 libraries, only their top

level folders have been renamed. This requires you to change you #include paths as follows:

Old Library	New Library
Source/Physics/Collide (hkpCollide.lib)	Source/Physics 2012 /Collide (hkpCollide.lib)
Source/Physics/Dynamics (hkpDynamics.lib)	Source/Physics 2012 /Dynamics (hkpDynamics.lib)
Source/Physics/Internal (hkpInternal.lib)	Source/Physics 2012 /Internal (hkpInternal.lib)
Source/Physics/Utilities (hkpUtilities.lib)	Source/Physics 2012 /Utilities (hkpUtilities.lib)
Source/Physics/Vehicle (hkpVehicle.lib)	Source/Physics 2012 /Vehicle (hkpVehicle.lib)
Source/Physics/Spu/...	Source/Physics 2012 /Spu/...
Source/Physics/Constraint (hkpConstraint.lib)	Source/Physics/Constraint (hkpConstraint.lib) (No changes)
Source/Physics/ConstraintSolver (hkpConstraint-Solver.lib)	Source/Physics/ConstraintSolver (hkpConstraintSolver.lib) (No changes)

Table 1.1: Physics 2012 Path Changes

Notes:

- There have been no changes to the Physics 2012 library names, file names or structure names. Only the paths have changed.
- The two constraint libraries (hkpConstraint.lib and hkpConstraintSolver.lib) are used by both physics engines, so their paths have not changed.

Other notable changes include, but are not limited to:

- Bridge libraries with other Havok products have been updated. Most products now have both a "Physics 2012" bridge and a "Physics" bridge. See the migration guides for particular products below for more details.

Please see the release notes for a list of other Physics 2012 improvements and fixes in Havok 2013.1.

1.4.4 Animation

1.4.4.1 New physics engine support and ragdoll rename

With the introduction of the new version of Havok Physics, the interface library between Havok Animation and Havok Physics 2012 was renamed, and a new interface library was added between Havok Animation and Havok Physics.

Old Library	New Library	Description
Source/Animation/Ragdoll (hkaRagdoll.lib)	Source/Animation/ Physics2012Bridge (hka-Physics2012Bridge .lib)	This library is for Physics2012 ragdolls (hkaRagdollInstances).
-	Source/Animation/PhysicsBridge (hkaPhysicsBridge.lib)	This library is for new Physics ragdolls (hkn-pRagdolls).
-	Source/Animation/PhysicsMigrationUtils (hkaPhysicsMigrationUtils.lib)	This library is for converting hkaRagdollInstances to hkn-pRagdolls.

Table 1.2: Animation-Physics Bridge Library Changes

Limitations:

- EXP-2627 Preview Tool does not support ragdoll mappers for new physics ragdolls.

1.5 Migrating to Animation, Physics_2012 and Common 2012.2

1.5.1 Common

1.5.1.1 Miscellaneous Changes

COM-1970	Don't allow <code>hkMath::min2()</code> , <code>max2</code> , and <code>clamp</code> on <code>hkSimdReals</code> in SSE builds.
----------	---

1.5.2 Content Tools

1.5.2.1 Renderable Attribute (3ds Max)

The way that nodes are gathered for export and sent to the Filter Manager Pipeline has changed slightly. For meshes and splines, only those nodes which have the **Renderable** attribute checked in the object properties will be considered for export. This change means that in scenes where the **Renderable** attribute is unchecked, those nodes will no longer be exported. Please check that you have set the **Renderable** attribute appropriately for nodes that you do or do not want to be considered for export.

1.5.2.2 Frozen Objects (3ds Max)

Where objects that were frozen in the Content Tools version 2011.3 were ignored, they are no longer ignored for 2012.1 and 2012.2 versions. This means that re-exporting assets which contain frozen geometry may cause issues. One example of this is in the Create Skeleton Filter. Setting the "Build Rig" to "Automatic" will ignore frozen bones in versions 2011.3 and earlier. However, since 2012.1 these frozen bones will be included in the resultant skeleton, which will not match the corresponding animation files from 2011.3 because the skeleton will be different. If you do not want this bone in the skeleton then please set the **Renderable** attribute to false.

1.5.2.3 Filter Manager Changes

When a configuration is run in the filter manager the number of Warnings and Errors that occurred during the export process is logged and shown at the bottom. The output is either Green (Success), Orange (Success with warnings) or Red (Errors). This status is also pushed out as an exit code from the Standalone Filter Manager, which makes it easy to determine whether the export process failed or not during automatic deployment. Please refer to the items below for more information.

EXP-1331	Show summary of warnings and errors in the message log pane.
EXP-1848	Improvements in error reporting features of HCT.

1.5.2.4 Modeller Integration Changes

Modeller Toolbars

To make switching between 3ds Max and Maya easier the toolbars layout has been slightly changed so that the icons are in the same order for each respected modeller.

Latest Versions of Max and Maya Support

Full support for 3ds Max 2013 is now available, as well as support for Maya 2013, and Maya 2013.5 (Extension).

SoftImage Deprecated

SoftImage versions of the tools have been deprecated, and are no longer actively supported. It is recommended to choose either 3ds Max or Maya instead.

1.5.2.5 Vision Integration Changes

Support for Havok Vision has been improved by fixing a number of issues with the Vision export process. The Content Tools also supports multiple UV channels now which can be exported and used appropriately in Vision. Please see the Release Notes for full details.

1.5.3 Physics

2012.2 is a maintenance release for Physics. The main change in this release is the inclusion of a new constraint library, which is described next. Please refer to the release notes for details on other improvements and bug fixes in this release. There are no new features.

1.5.3.1 New Constraint Library

In order to make constraints reusable outside of Physics, most product level constraint code has been moved to a new library - **hkpConstraint.lib** . This includes constraint atoms, datas, motors, and visualization code. Internal constraint code still resides in `hkpConstraintSolver.lib`.

Users migrating to Havok Physics 2012.2 will need to make the following changes:

1. Add "hkpConstraint.lib" to the list of libraries used by the linker.
2. Make the following changes to `#include` paths:

Previously (2012.1)	Now (2012.2)
Physics/Dynamics/Constraint/hkpConstraintData.h	Physics/Constraint/Data/hkpConstraintData.h
Physics/Dynamics/Constraint/Bilateral/...	Physics/Constraint/Data/...
Physics/Dynamics/Constraint/Pulley...	Physics/Constraint/Data/Pulley/...
Physics/Dynamics/Constraint/Chain/RagdollLimits/...	Physics/Constraint/Data/RagdollLimits/...
Physics/Dynamics/Constraint/Chain/HingeLimits/...	Physics/Constraint/Data/HingeLimits/...
Physics/Dynamics/Constraint/Motor/...	Physics/Constraint/Motor/...
Physics/Utilities/Visual Debugger/Viewer/Dynamics/Constraint/...	Physics/Constraint/Visualize/...
Physics/ConstraintSolver/SimpleConstraints/hkpSimpleConstraintUtil	Physics/Internal/Solver/SimpleConstraints/hkpSimpleCon

3. Note that classes previously defined in `Physics/ConstraintSolver/Constraint/Atom/hkpConstraintAtom.h` are now split across several files:

Class	New Location
<code>hkpConstraintAtom</code>	<code>Physics/Constraint/Atom/hkpConstraintAtom.h</code>
<code>hkpBridgeConstraintAtom</code>	<code>Physics/Constraint/Atom/Bridge/hkpBridgeConstraintAtom.h</code>
<code>hkpSimpleContactConstraintAtom</code>	<code>Physics/Dynamics/Constraint/Atom/hkpSimpleConstraintAtom.h</code>

4. Some utility functions were moved from `hkpConstraintUtils` to a new `hkpConstraintDataUtils` utility class:

- `getConstraintPivots()`
- `getConstraintMotors()`
- `loosenConstraintLimits()`

5. Any references to `setMotorActive()` and `isMotorActive()` were renamed to `setMotorsEnabled()` and `isMotorEnabled()` respectively.

1.5.4 Animation

We now allow for defining Partitions in the skeleton definition file. This has required the introduction of a new file format. The old skeleton definition file format will still work, but all newly saved files will be in the new format. Any clients using Partitions and skeleton definition files are encouraged to migrate Partitions into their files to avoid having to redefine them in every filter stacks which references the skeleton definition file. See the Create Skeletons Animation Filter documentation for more information.

Layered Animations have been renamed Partitioned Animations to avoid confusion with the concept of Layering in the Behavior product.

The Parametrized Animation Filter was removed (it was not being used by the SDK). The Extracted Motion Filter creates similar data which is used by parts of the SDK.

1.6 Migrating to Animation, Physics_2012 and Common 2012.1

1.6.1 Common

1.6.1.1 The Memory Tracker must now be initialized before the Memory System

If you use the Memory Tracker, you must now initialize it by calling `hkMemoryInitUtil::initMemoryTracker()` before the memory system is started.

1.6.1.2 Some math functions have moved

Some methods in the `hkMatrix3` and `hkQuaternion` classes have been replaced by equivalent functions in the `hkMatrix3Util` and `hkQuaternionUtil` namespaces, respectively. See the files `Common/Base/Math/Matrix/hkMatrix3Util.h` and `Common/Base/Math/Quaternion/hkQuaternionUtil.h`.

1.6.1.3 WiiU Networking now uses TCP/IP by default

The Havok Networking system, used for the connection to tools such as the Visual Debugger and the Havok Script Tools will now use TCP/IP instead of HIO. The tools should now be connected directly to the console's IP address (printed at boot time) using the normal TCP port (25001 for the Visual Debugger).

The default networking profile on the console will be used to set up the connection. The default network interface of the console must be accessible from the PC running the tools.

The old HIO connection will still be used if the macro `HK_USE_DEPRECATED_HIO_CONNECTION` is defined, however this is not recommended.

1.6.2 Content Tools

There are no migration notes for the Havok Content Tools for this release. Please refer to the release notes for details of specific any issues that were resolved.

1.6.3 Physics

2012.1 is a maintenance release for Physics, with a small number of new features. Please see the release notes for a full list of changes. Highlights are listed here.

1.6.3.1 Constraint changes

`hkpConstraintData::setMaxLinearImpulse()` previously only set a threshold for firing `contactImpulseLimitBreachedCallback()`. It did not allow for simply limiting a constraint's applicable impulse. Impulse limiting has now been added as a feature independent from impulse breaching. As a result, `setMaxLinearImpulse()` has been removed and replaced with:

- **`setBreachImpulse()`** - sets the impulse above which `contactImpulseLimitBreachedCallback()` is fired. This should be used in place of any previous occurrences of `setMaxLinearImpulse()` in your code base.
- **`setMaximumLinearImpulse()`** - limits the applicable linear impulse. Does not raise callbacks. Currently only supported by the ball-and-socket and fixed constraints.
- **`setMaximumAngularImpulse()`** - limits the applicable angular impulse. Does not raise callbacks. Currently only supported by the fixed constraint.

Two new constraints have been added to the SDK:

- **`hkpFixedConstraintData`** - limits 6 degrees of freedom, and supports linear and angular impulse limiting. (HVK-6192)
- **`hkpDeformableFixedConstraintData`** - limits 6 degrees of freedom, and deforms if the linear or angular limits are exceeded. (HVK-6197)

1.6.3.2 Mesh shape changes

hkpCompressedMeshShape has been deprecated, and will be removed in a subsequent release. The shape is still available for use in the SDK and toolchain for this release, but has been removed from documentation. The source files have moved from "Source\Physics\Collide\Shape\Compound\Collection\CompressedMesh" to "Source\Physics\Collide\Shape\Deprecated\CompressedMesh". (HVK-6175)

hkpBvCompessedMeshShape should be used in place of **hkpCompressedMeshShape**. This shape, introduced in 2011.2, offers reduced memory footprint and improved performance. Improvements for this release include:

- now supported by the Havok Content Tools "Create Rigid Bodies" filter. (HVK-6165)
- now supports tagging of primitives with strings. (HVK-6221)

Note that previous versions of **hkpBvCompessedMeshShape** had some issues with the building process which cannot be corrected during versioning (HVK-6166, HVK-6200). Therefore it is highly recommended that any instances of this shapes be recreated using 2012.1. Any old versions found during deserializing and versioning will cause a warning to be raised. (HVK-6226)

Please see the user guide for further information on how to use **hkpBvCompessedMeshShape**.

1.6.3.3 Coarse grid heightfield raycasting on SPU

hkpSampledHeightFieldShape introduced a bounding volume for raycast optimization in 2010.2, known as a "coarse grid". This is now also supported on SPU. (HVK-6078)

The default behavior of the ray cast query ELF is to link both the original DDA algorithm and the new coarse grid algorithm, and use the coarse grid algorithm when the height field has coarse grid available. The new algorithm increases the code size of the raycast query SPU ELF compared with previous releases. If this is an issue for your use case, the ELF can be recompiled using any subset of these algorithms in order to reduce code size - see **hkpSampledHeightField_register*RayCastFunction()**.

In addition, the DDA and/or coarse grid algorithms can also be stripped from any executable by #defining **HK_EXCLUDE_FEATURE_hkpSampledHeightFieldDdaRayCast** or **HK_EXCLUDE_FEATURE_hkpSampledHeightFieldCoarseTreeRayCast** before #including **hkProductFeatures.cxx**.

1.6.3.4 Generic shape key iterator

A new **hkpShapeKeyPath** utility class has been added to the SDK. This can be used to iterate a set of shape keys returned from collision callbacks or queries in a consistent manner, returning the shape at each level. The utility currently supports the following structures as inputs:

- **hkpContactPointEvent** (from collision callbacks)
- **hkpWorldRayCastOutput** (from ray casts)

- `hkpShapeRayCastOutput` (from ray casts)

`hkpContactPointEvent::getShapes()` has been removed, in favor of constructing a `hkpShapeKeyPath` from the `hkpContactPointEvent` and calling its `getShapes()` method.

1.6.3.5 Other interface changes

Notable interface changes include:

- `hkpWorldCinfo::m_BETA_useCompoundSpuElf` has been renamed to `hkpWorldCinfo::m_useCompoundSpuElf`, as `hkpBvCompessedMeshShape` and `hkpStaticCompoundShape` are no longer in beta.
- `hkpCollisionFilter::isCollisionEnabled(const hkpShapeRayCastInput& const hkpShapeContainer&, hkpShape&, hkpShapeKey)` has dropped the third (`hkpShape`) parameter. (HVK-6168)
- `hkpSimplexSolver` has been renamed `hkSimplexSolver`, and has been moved to `Source/Common/Internal`.
- `hkpMassProperties` has been renamed to `hkMassProperties`.
- `hkpMassElement` has been renamed to `hkMassElement`.

1.6.4 Animation

1.6.4.1 Interface Changes

- `hkaMirroredAnimation::NUM_EXTRA_CHUNKS` has changed from 6 to 7.
- `hkaSampleAndCombineUtils::getMaxTrackIndex()` now takes an additional optional parameter specifying whether to use partitions.

New signature:

```
static hkUInt32 HK_CALL getMaxTrackIndex(const hkInt16* trackToBoneMapping, const hkUInt8* perTrackWeights,
    , hkUInt32 numBones, hkUInt32 numTracks, hkBool hasPartitions = false);
```

- `hkaAnimationJobs::HK_MAX_NUM_DATA_CHUNKS` has changed from 10 to 11.
- `hkaSampleBlendJob::MAX_ANIMATION_CHUNKS` has changed from 10 to 11.
- `hkaSampleBlendJob::MAX_NUM_CHUNKS` has changed from 20 to 21.
- Both versions of `hkaSampleBlendJob::build()` now take an `hkaSkeleton`.

New signature:

```
void build(const hkaSkeleton& skeleton, const hkaAnimationControl& control, int& maxChunkBytesInOut);
```

```
void build(    const hkaSkeleton& skeleton,
               const hkaAnimation& anim,
               hkReal localTime,
               hkQsTransform* bonesOut,
               hkReal* floatSlotsOut,
               const hkaAnimationBinding* binding,
               const hkaSkeletonMapper* mapper,
               const hkArray<hkUInt8*> perBoneWeights,
               const hkArray<hkUInt8*> perFloatWeights,
               int& maxChunkBytesInOut );
```

- A new hkaSampleBlendJob::SampleFlags flag has been added.

```
HAS_PARTITIONS
```

- hkaSampleBlendJob::addAnimation() now takes an hkaSkeleton.

New signature:

```
void addAnimation(    const hkaSkeleton& skeleton,
                     const hkaAnimation& anim,
                     hkReal localTime,
                     hkQsTransform* bonesOut,
                     hkReal* floatsOut,
                     const hkaAnimationBinding* binding = HK_NULL,
                     const hkaSkeletonMapper* mapper = HK_NULL );
```

- hkaSampleBlendJob::m_numBones is now an hkInt16 (was int).
- Several buffer sizes in hkaSpuConfig have changed:

```
HK_SPU_TOTAL_ANIMATION_MAPPING_BUFFER_SIZE
HK_SPU_TOTAL_ANIMATION_SAMPLE_AND_COMBINE_BUFFER_SIZE
HK_SPU_TOTAL_ANIMATION_SAMPLE_AND_BLEND_BUFFER_SIZE
```

- A new flag has been added to hkaQuantizedSampleAndCombineJob::BlendFlags.

```
PARTITIONS
```

- hkaAnimationControl::m_binding is now an hkRefPtr.

- The `hkaDefaultAnimatedReferenceFrame` constructor now takes an optional `hkaReferenceFrameTypeEnum`.

New signature:

```
hkaDefaultAnimatedReferenceFrame( const MotionExtractionOptions& options, hkaReferenceFrameTypeEnum
    frameType = REFERENCE_FRAME_DEFAULT );
```

- Several functions in `hkaSkeletonMapper` now take a `startBoneIndex`.

New signature:

```
static void combinedPoseFromAdditivePoseAndReferencePose( const hkQsTransform* additivePose,
    const hkQsTransform* referencePose,
    const hkInt16* boneToTrackIndices,
    hkInt32 startBoneIndex,
    hkInt32 numBones,
    hkQsTransform* combinedPoseOut );
```

```
static void additivePoseFromCombinedPoseAndReferencePose( const hkQsTransform* combinedPose,
    const hkQsTransform* referencePose,
    const hkInt16* boneToTrackIndices,
    hkInt32 startBoneIndex,
    hkInt32 numBones,
    hkQsTransform* additivePoseOut );
```

```
static void combineSparsePoseWithFullPose( const hkQsTransform* sparsePose,
    const hkQsTransform* fullPose,
    const hkInt16* sparsePoseBoneToTrackIndices,
    hkInt32 startBoneIndex,
    hkInt32 numBones,
    hkQsTransform* sparseMulFull_sparseOut );
```

1.7 Migrating to Animation, Physics_2012 and Common 2011.3

1.7.1 Common

There are no migration notes for Havok Common for this release. Please refer to the release notes for details of specific any issues that were resolved.

1.7.2 Content Tools

1.7.2.1 Process Vision Data filter

A new filter was added to the collection of common export pipeline filters. The filter assumes the graphics meshes in the modeller file were previously exported as Vision .model files. Upon execution, it will try to locate the Vision model file for each graphics node in the scene and embed it as a BLOB in the exported asset file. Note that currently the only filter that uses and updates the embedded Vision meshes is the Destruction filter.

HKD-756	Destruction Integration with Vision.
---------	--------------------------------------

1.7.3 Physics

2011.3 is a maintenance release for Physics, with a focus on continued improvements around collision shapes. Please see the release notes for a full list of changes.

1.7.3.1 Refactor of base shape classes

The base classes for all **hkpShapes** have been refactored (HVK-6071). Two new base classes have been introduced. The **hkpShape** inheritance chart is now: **hkReferencedObject** > **hkcdShape** > **hkpShapeBase** > **hkpShape**

- **hkcdShape** : This contains the shape type and other core identifiers, and is shared across all Havok products. This class will grow in future as some shape functionality is moved from the Physics product into the shared **hkcdCollide** library. Consequences of this change include:
 - All shape type enums have changed from (e.g.) "HK_SHAPE_LIST" to "hkcdShapeType::LIST". These will need to be changed in user code.
In order to help with this, a mapping of legacy shape types to new shape types is provided by <Physics/Collide/Shape/hkpLegacyShapeType.h>, which defines the legacy shape types if this header is included by the user. Note that this is provided only to help users during migration - it is not used by the SDK and will be removed in future releases. It is advised that users rename all occurrences of legacy shape types rather than relying on this mapping.
 - The new "hkcdCollide" library now needs to be included by the linker.
- **hkpShapeBase** : This contains all virtual shape functions which may be called on SPU, with stub implementations. This class works in conjunction with the new **hkpShapeVirtualTableUtil** which implements "partial" shapes using subsets of these virtual functions for each SPU ELF "profile". This simplifies the code for concrete shape classes which should run on SPU (previously this was done by declaring these functions as static in each of those classes on SPU). Consequences of this change include:
 - Custom SPU ELFs should now modify **hkpShapeVirtualTableUtil** in order to register the subset of virtual functions that should run on SPU for each custom shape in each profile.
 - The **HKCD_PATCH_SHAPE_VTABLE()** macro should be called to replace the virtual function table whenever any shape is fetched or constructed on SPU.

Note that the overall memory footprint of **hkpShape** has not changed.

1.7.3.2 Improvements to landscape shapes

hkpBvCompressedMeshShape and **hkpStaticCompoundShape** were introduced as BETA in the previous 2011.2 release. 2011.3 improves on these shapes with bug fixes and feature suggestions from customer feedback. Highlights include:

- [HVK-6076] Added support for convex primitives in **hkpBvCompressedMeshShape**. This means that (for example) the output of a convex decomposition can be stored as a single instance of this shape, alongside any triangulated geometry. As with triangles, the convex primitives are stored in a optimized compressed format, reducing memory use and increasing query performance over previous solutions (such as **hkpCompressedMeshShape** + **hkpMoppBvTreeShape**).
- [HVK-6062] Improved **hkpBvCompressedMeshShape** construction interface. Previously a **hkGeometry** was required in order to construct an **hkpBvCompressedMeshShape**. In 2011.3, the construction structure is abstract and can be overridden to provide the triangles and convex shapes from any source. A default construction structure is provided (**hkpDefaultBvCompressedMeshShapeCinfo**) which uses a **hkGeometry** as before.
- [HVK-6024] Improved **hkpStaticCompoundShape** linear cast performance. Linear casting against **hkpStaticCompoundShape** in 2011.2 was slow due to the default **hkpBvTreeShape** linear cast algorithm being unsuitable. A new **castAabb()** method has now been added to the **hkpBvTreeShape** interface and an optimized version of this is implemented for **hkpStaticCompoundShape**. This function is called during linear casting and can also be called directly by users.

Interfaces to these shapes - particularly **hkpBvCompressedMeshShape** - have changed a little in this this release, so existing users of these shapes are advised to review the header files.

1.7.3.3 MOPP ray cast results fixed

A long-standing bug - whereby the **hkpShapeKey** hierarchy returned by a MOPP ray cast hit would have an extraneous key at the beginning - has been fixed (HVK-4646). Please note that some customer code bases may include special handling of MOPP ray cast results to work around this bug, so this logic should now be removed.

This means that any shape key hierarchy can now be traversed without special cases, using:

```

// Given a ray cast output..
hkpShapeRayCastOutput output;

// ... this will extract the leaf shape that was hit
hkpShapeBuffer buffer;
const hkpShape* shape = output.m_rootCollidable->getShape();
for ( int keyIndex = 0; shape != HK_NULL; ++keyIndex )
{
    const hkpShapeContainer* container = shape->getContainer();
    if ( container )
    {
        // go to the next level
        shape = container->getChildShape( output.m_shapeKeys[keyIndex], buffer );
    }
    else
    {
        // we are at the leaf
        break;
    }
}

```

1.7.3.4 KD tree removed

2011.2 removed `hkKdTree` support from SPU. 2011.3 removes `hkKdTree` support completely from Physics (HVK-5999). This includes the removal of:

- "m_tree" member from all collision query jobs
- `hkpWorldCinfo::BROADPHASE_TYPE_SAP_AND_KD_TREE_DEPRECATED` option
- `hkpWorld` tree manager and associated functions
- `hkpSpursKdTreeBuilding` ELF

Users wishing for an alternative to KD trees should explore `hkpWorldCinfo::m_broadPhaseType`, specifically the `BROADPHASE_TYPE_TREE` or `BROADPHASE_TYPE_HYBRID` broad phases. Note that `hkpTreeBroadPhase` includes the ability to add "user objects", which are reported by broad phase queries but have no impact on simulation updates.

1.7.3.5 Other interface changes

`hkpSphereUtil::castRayUtil()` has been changed to `hkpRayCastSphere()`. Note that this is now just a wrapper around `hkcdRayCastSphere()`. It may be more appropriate to use that directly.

`hkGetWorldObject()` has been renamed to `hkpGetWorldObject()`.

Some spurious header inclusions have been removed to be consistent with the [header include policy](#). This may necessitate adding an extra `#include <Physics/Dynamics/hkpDynamics.h>` before other inclusions.

1.7.4 Animation

There are no migration notes for Havok Animation for this release. Please refer to the release notes for details of specific any issues that were resolved.

1.8 Migrating to Animation, Physics_2012 and Common 2011.2

1.8.1 Common

1.8.1.1 Libraries

The `hkcdCollide` library has been removed from the SDK, as it is no longer used by any other SDK features.

1.8.1.2 Visual Debugger

Some Visual Debugger viewers now serialize classes to the data stream; this breaks playback compatibility with previous versions of the Visual Debugger. Please make sure to use the latest version of the executable (located in `Tools\VisualDebugger`).

1.8.2 Content Tools

There are no migration notes for Havok Content Tools 2011.2. Please see the release notes for any issues that were resolved.

1.8.3 Physics

1.8.3.1 New landscape shapes

Two new Physics shapes are introduced in 2011.2, designed to represent static level geometry with flexibility and optimal performance, compared to previous approaches. There is currently no Content Tools support for the new shapes - this will be added in a future release.

Note:

`hkpCompoundShape` and the `hkcdCollide` library, introduced as beta in 2011.1, have been removed in this release. The new shapes offer much of the same functionality without the performance overhead imposed by that shape.

hkpBvCompressedMeshShape

This shape combines compressed triangulated geometry with a bounding volume tree, together in a single shape. This can replace a `hkpCompressedMeshShape` wrapped in a `hkpMoppBvTreeShape`, which was the previous best practice for landscape shapes. In comparison, `hkpBvCompressedMeshShape` offers reduced

memory footprint (around 50%-90% of the size), significantly improved ray cast times (around 2 or 3 times faster), and equivalent simulation times.

hkpBvCompressedMeshShape also allows for the following per-triangle information:

- 16 bits of welding information
- 8 bits of collision filter information*
- 8 bits of user data*

*Collision filter and user data information can represent an index into palettes of 32 values - managed by the shape - if the user wishes to store more than 8 bits for these. However in either case the maximum number of unique values is limited to 256.

This per-triangle data is stored in Run Length Encoded format, so if it is unused it will minimize the overall size of the shape in memory. Note that due to the compression of the geometry and the per-triangle data, this shape cannot be modified after construction. If users wish to disable triangles (e.g., for Destruction), this can be done by making the **hkpBvCompressedMeshShape** an instance within a **hkpStaticCompoundShape** (see below).

Please see the Physics manual and the "Physics/Api/Collide/Shapes/BvCompressedMesh" demo for an example of how to use this shape.

hkpStaticCompoundShape

This shape allows users to create a compound shape by instantiating any other **hkpShapes**, each with full transforms including (non-uniform) scale. This replaces the need to create complex shape hierarchies in order to share subshapes (e.g., MOPP of list of transforms of MOPP of mesh). In comparison to such hierarchies, **hkpStaticCompoundShape** offers minimal memory overhead, faster queries, collision filter info flexibility, and the ability to disable any shape key of any child shape even if the child shape is shared (a common use case in destructible environments).

Each "instance" of a **hkpShape** within **hkpStaticCompoundShape** uses 64 bytes (on 32-bit platforms), which includes:

- the **hkpShape** and **hkQsTransform**
- 32 bits of instance filter info
- 32 bits of filter info mask - for flexibility in combining instance filter info with child shape filter info
- 32 bits of user data

The instances are wrapped in an AABB tree when construction is finished, to provide fast queries. At that point, the subshapes and transforms can no longer be modified. However per-instance data can still be changed at any time, and shape keys and instances can be disabled or enabled.

One important detail of **hkpStaticCompoundShape** is that a single shape key directly references a particular child shape of a particular shape instance. This means that for the common case where the shape has many instances of child mesh shapes, the engine sees this shape as one (large) flat list of triangles. So only one collision agent is needed to process collisions with this shape. This means there is reduced

runtime memory use versus a traditional shape hierarchy approach. However it also means that users may need to do extra work if they wish to retrieve the "intermediate" child shapes from collision events - there are helper methods such as `hkpStaticCompoundShape::decomposeShapeKey()` that can be used for this. The number of shape key bits assigned to the instance id and to the child shape key is determined dynamically during construction of the static compound shape. Instances can only be added if their maximum child shape key fits within the maximum number of bits available.

Please see the Physics user guide and the "Physics/Api/Collide/Shapes/StaticCompound" demos for examples of how to use this shape.

Changes to `hkpConvexTransformShape`

In order to support transforms with scaling (as required by `hkpStaticCompoundShape`), `hkpConvexTransformShape` now stores a `hkQsTransform` rather than a `hkTransform`. Note that this does not currently support scaling of arbitrary convex shape types so should not currently be used for such. Importantly, convex radius is currently not scaled - this is generally not an issue for shapes used by fixed bodies as the convex radius for these is often zero.

PlayStation®3 SPU support

On PlayStation®3, two distinct SPU ELF's are provided for collision detection in 2011.2 due to code size issues:

- `hkpSpursCollide.elf` - this remains mostly unchanged from previous releases, providing support for the same set of `hkpShapes`. This excludes the new shapes listed above.
- `hkpSpursCollideStaticCompound.elf` - this has support for the new shapes (`hkpBvCompressedMeshShape`, `hkpStaticCompoundShape`), and many of the same shapes as `hkpSpursCollide.elf`, including `hkpMoppBvTreeShape`. However it does **not** include support for:
 - `hkpCompressedMeshShape` - `hkpBvCompressedMeshShape` replaces this
 - `hkpExtendedMeshShape` - `hkpStaticCompoundShape` replaces this
 - Sending collision events to PPU using `hkpSpuCollisionCallbackUtil` - this is largely unused as contact point callbacks are fired from the solver since Havok 7

By default, the original `hkpSpursCollide.elf` will be used by the engine. If you wish to use the new shapes added in 2011.2 on SPU during collision detection, you must instruct the engine to do so when constructing the world, by setting:

```
hkpWorldCinfo::m_useCompoundSpuElf = true;
```

In addition, `hkpStaticCompoundShape` vs `hkpStaticCompoundShape` collision is currently **not** supported on SPU. As such, `hkpStaticCompoundShape` should be used for fixed landscape bodies only. For dynamic bodies, please use `hkpListShape` wrapped in `hkpMoppBvTreeShape`, as in previous releases.

Note:

All other SPU ELF's (integrate, ray cast, collision query, character proxy) include support for all the original shapes plus the new shapes. No additional setup work is required for these, no matter which shapes you use.

1.8.3.2 KD tree support removed from SPU

`hkKdTree` was deprecated in 2011.1 and has now been removed from SPU in order to make room for the new shapes. While you can still create and use a `hkKdTree` in 2011.2, any ray casts or collision queries performed on it will now be forced to PPU. `hkKdTree` will be removed completely in the next release.

Users looking for an alternative to the 3 axis sweep broadphase should explore the `hkpWorldCinfo::BroadPhaseType`'s, which includes a tree based broad phase which has replaced `hkKdTree`.

1.8.3.3 Improved deactivation usability

Previously, `hkpKeyframeUtility::applyHardKeyframe()` would always call `setLinearVelocity()` and `setAngularVelocity()` even if there was no change in velocity. These cause rigid bodies to remain active. So if a rigid body was being driven using `applyHardKeyframe()` every frame, it would never get an opportunity to deactivate. To avoid this, some users wrapped `applyHardKeyframe()` with a check for whether the velocity needed to be applied or not. `hkpKeyframeUtility::applyHardKeyframe()` has now been changed to do this check internally. Rigid bodies can now deactivate even if `applyHardKeyframe()` is called on them every frame with no change in velocity.

Also, `hkpEntity::deactivate()` is misleading in that it actually tries to deactivate the entity's (split) simulation island. This was causing confusion to users who were attempting to deactivate single bodies by using this method. `hkpEntity::deactivate()` has now been deprecated and replaced with `hkpEntity::requestDeactivation()` - this sets up the entity so that the engine will treat it as inactive during the next check (rather than waiting for potentially lots of steps) but does not force an immediate deactivation. The entity's (split) simulation island is then deactivated only if all of its entities are inactive, as per usual.

1.8.3.4 Notable interface changes

`hkpKeyFrameUtility` has moved from "Physics/Utilities/Constraint/Keyframe/hkpKeyFrameUtility.h" to "Physics/Utilities/Dynamics/KeyFrame/hkpKeyFrameUtility.h", as it relates to dynamics not constraints.

`hkRegisterAlternateShapeTypes` has been renamed to `hkpRegisterAlternateShapeTypes`, to have the correct "hkp" prefix.

`hkpListShape::ChildInfo::m_shapeSize` has been changed from `int` to `short`. The Havok Destruction Runtime required the addition of a new member, `m_shapeInfo`. The memory cost of the `hkpListShape` is not affected by these changes.

`hkpExtendedMeshShape::Subpart::m_type`, `m_materialIndexStridingType`, and `m_numMaterials` were compressed into a new member, `m_typeAndFlags`, and removed. The Havok Destruction Runtime required the addition of a new member, `m_shapeInfo`. The memory cost of the `hkpExtendedMeshShape` is not affected by these changes.

1.8.4 Animation

There are no migration notes for Havok Animation 2011.2. Please see the release notes for any issues that were resolved.

1.9 Migrating to Animation, Physics_2012 and Common 2011.1

1.9.1 Common

1.9.1.1 Memory changes

The `hkStatisticsCollector` interface, the classes derived from it in `Common/Base/DebugUtil/StatisticsCollector`, and the associated virtual function `hkReferencedObject::calcContentStatistics()` have all been removed. This approach to memory reporting was onerous to maintain as it required keeping all `calcContentStatistics()` implementations up to date. The new approach uses `hkMemoryTracker` to automatically track memory usage at the point of allocation. See the function `MenuDemo::writeSimpleMemoryStatistics()` for an example of how to collect statistics using the new interfaces.

The constructor for `hkFreeListAllocator` now takes an additional parameter of type `hkMemoryAllocator::ExtendedInterface*`, which provides some additional functionality. To get the old behavior, just pass null for this pointer. `hkMemoryInitUtil::initFreeList()` has also been changed to accept this new parameter.

The files `hkLargeBlockAllocator.h` and `hkLargeBlockAllocator.cpp` have been moved from `Common/Base/Memory/Allocator/FreeList` to `Common/Base/Memory/Allocator/LargeBlock`.

`hkMemoryInitUtil::initFreeListLargeBlock()` and `hkMemoryInitUtil::initHeapAllocator()` have been added and `hkMemoryInitUtil::initDefault()` now calls `initFreeListLargeBlock()` instead of `initFreeList()`.

1.9.1.2 Math changes

The math library has been substantially modernized. In most cases this is transparent as a mostly backwards compatible wrappers have been maintained.

There is a compile time option (preprocessor define `HK_DISABLE_IMPLICIT_SIMDREAL_FLOAT_CONVERSION`) to make expensive implicit conversions such as `hkSimdReal` to `hkReal` explicit. Callers can either pay the conversion cost with a call to `hkSimdReal::getReal()` or refactor to use `hkSimdReal` which usually results in better code. Even without this define, in some cases `getReal()` conversion calls will have to be made.

Global values for vector constants (`hkQuadReal1000`, `hkQuadRealHalf`, etc.) have been replaced with methods (e.g. `hkVector4::getConstant<HK_QUADREAL_XXXX>()` and `hkSimdReal::getConstant<HK_QUADREAL_EPS>()`).

There are several other preprocessor defines which affect the math library. These need only

be considered if making heavy use of the math library. Define the preprocessor variable `HK_DISABLE_OLD_VECTOR4_INTERFACE` to disable the old interface wrappers at compile time. The define `HK_DISABLE_MATH_CONSTRUCTORS`, which removes constructors from the basic math types, often leads to better code generation.

1.9.1.3 Convex decomposition has been moved

`hkgpConvexDecomposition` has been moved from `Common/ConvexDecomposition/hkgpConvexDecomposition.h` to `Geometry/ConvexDecomposition/hkgpConvexDecomposition.h`.

1.9.2 Content Tools

1.9.2.1 Support for 3ds Max 2012, Maya 2012 and SoftImage 2012

The Havok Content Tools now support version 2012 (both 32 and 64 bit) of 3ds Max, Maya and SoftImage.

Warning:

3ds Max 2012 introduced a new viewport renderer called Nitrous which is incompatible with some of Havok's viewport display elements such as 'RB' labels. Currently, these will only show up if using one of the other viewport renderers (OpenGL, DirectX, Heidi).

1.9.3 Physics

1.9.3.1 New hkcdCollide libraries

Core collision detection and related functionality has been moved from the physics libraries into an independent set of libraries. The new libraries to link to are `hkcdCollide.lib` and `hkcdInternal.lib`.

1.9.3.2 hkAabbTree has been removed

`hkAabbTree`, which was deprecated in 2010.2, has now been removed in favor of a more efficient replacement. This means that the `hkpWorldCinfo::m_useMultipleTree` flag is gone, as are classes related to the AABB tree such as `hkpAabbTreeWorldManager`, `hkAabbTreeRaycastUtil` and `hkpAabbTreeCollidableRaycaster`.

1.9.3.3 hkKdTree has been deprecated

`hkKdTree` has been deprecated in favor of the newer broad phase types which do not require a tree to be rebuilt each frame. The `m_useKdTree` option has been removed from `hkpWorldCinfo` which means that the KD-tree is now only used if the broad phase type is set to `hkpWorldCinfo::BROADPHASE_TYPE_SAP_AND_KD_TREE_DEPRECATED` (see below).

Also note that the KD-tree building jobs have been moved from the raycast ELF's (in `Physics/Collide/Query/Multithreaded/RayCastQuery`) to a new ELF (in

Physics/Collide/Query/Multithreaded/BuildKdTreeQuery). This requires registering the new jobs by calling `hkpBuildKdTreeQueryJobQueueUtils::registerWithJobQueue()`, renaming `hkpRayCastQueryJobHeader` to `hkpBuildKdTreeQueryJobHeader`, etc.

1.9.3.4 Broad phase changes

The interface for choosing which broad phase variant to use has been refactored. The `hkpWorldCinfo::m_useHybridBroadphase` option has been removed and `hkpWorldCinfo::m_broadPhaseType` has been added. The available options are:

- `hkpWorldCinfo::BROADPHASE_TYPE_SAP`
- `hkpWorldCinfo::BROADPHASE_TYPE_TREE`
- `hkpWorldCinfo::BROADPHASE_TYPE_HYBRID`
- `hkpWorldCinfo::BROADPHASE_TYPE_SAP_AND_KD_TREE_DEPRECATED`

Where previously you set `m_useHybridBroadphase` to true, you would now select the `BROADPHASE_TYPE_HYBRID` option; where previously you set `m_useKdTree` to true, you would now select `hkpWorldCinfo::BROADPHASE_TYPE_SAP_AND_KD_TREE_DEPRECATED`.

The `BROADPHASE_TYPE_TREE` option is new and allows a spatial tree to be used for broad phase without sweep-and-prune. See the User Guide for more information about the new broad phase types.

In addition, the `hkpHybridBroadPhase` class has been renamed to `hkpTreeBroadPhase` and `hkpBroadPhase::Enable32BitBroadphase` has been renamed to `hkpBroadPhase::Enable32BitBroadPhase`.

1.9.3.5 Other interface changes

`hkpCdVertex` (defined in `Common/Internal/Collide/Gsk/hkCdVertex.h`) has been replaced by `hkcdVertex` (defined in `Geometry/Internal/Types/hkcdVertex.h`).

`hkpConvexVerticesShape::FourVectors` and `hkKdTreeUtils::RayBundle` have both been replaced by `hkFourTransposedPoints` (defined in `Common/Base/Math/Vector/hkFourTransposedPoints.h`).

`hkpShapeCutterUtil::SubtractShapeInput::m_allowedPentration` has been renamed to `hkpShapeCutterUtil::SubtractShapeInput::m_allowedPenetration`.

`hkpTriangleCompressor` (defined in `Physics/Collide/Util/hkpTriangleCompressor.h`) has been removed.

1.9.3.6 New compound shape type

Note:

This is a beta feature.

This release introduces a new, highly-scalable shape type, `hkpCompoundShape`, that is intended to be a replacement for all other shape types except single convex objects. This provides many new features that were previously difficult to support such as instancing, non-uniform scale, addition and removal of subparts at runtime and greatly improved raycast performance. Please refer to the User Guide for more information about using `hkpCompoundShape`.

1.9.4 Animation

1.9.4.1 New predictive compression scheme

A new predictive compression scheme (`hkaPredictiveCompressedAnimation`) has been introduced that is optimized for SIMD. It compresses more compactly than spline compression and is faster on some platforms. Please see the Platform Guide for your platform for the comparative performance of the different compression types.

HKA-1364	Add a new predictive compressed animation type that is optimized for SIMD.
----------	--

1.9.4.2 Wavelet and Delta compression schemes have been removed

The wavelet and delta compression schemes are no longer available. If you upgrade to this version you will need to re-export your animation assets using one of the other compression types.

HKA-1396	Remove wavelet and delta compression schemes.
----------	---

1.9.4.3 The `hkaChunkCache` has been removed

With the removal of Wavelet and Delta compression, the `hkaChunkCache` is no longer needed and has been removed. All interfaces that required an `hkaChunkCache` no longer do.

1.9.4.4 New job type for decompressing and blending animations

A new job type `hkaSampleBlendJob` has been introduced that replaces both the `hkaSampleAndCombineJob` and the `hkaSampleOnlyJob`, both of which are now deprecated. The new `hkaSampleBlendJob` is typically faster than the old job types and supports more features. In particular, it can do animation retargeting automatically if an `hkaAnimationControl` on your `hkaAnimatedSkeleton` is configured to do retargeting. Please see the Animation Manual for more details.

HKA-1415	Introduce a new <code>hkaSampleBlendJob</code> that can decompress, blend, mirror and retarget animations.
HKA-1402	Unify retargeting with sampling and blending on the SPU.

1.9.4.5 Interface changes to how jobs notify when they are done

The semaphore and flag that can optionally be used to receive notification that an animation job is done are now nested inside a new `hkaJobDoneNotifier`:

```

/// A container for two kinds of notification for when an animation job completes.
struct hkaJobDoneNotifier
{
    HK_DECLARE_NONVIRTUAL_CLASS_ALLOCATOR(HK_MEMORY_CLASS_ANIMATION, hkaJobDoneNotifier);

    hkaJobDoneNotifier();

    /// Calls release() on the semaphore and sets *m_flag to true.
    void signal();

    /// This semaphore is released when the work is finished.
    /// It can be set to HK_NULL if you don't need to wait on a specific job.
    /// The same semaphore can be shared by multiple jobs.
    hkSemaphoreBusyWait* m_semaphore;

    /// This flag is incremented when the work is finished.
    /// It can be set to HK_NULL if you don't need to wait on a specific job.
    /// Each job must point to its own flag to avoid a race condition.
    /// This is a lightweight mechanism to determine if a job has completed.
    hkUInt32* m_flag;
};

```

Previously, animation jobs had these members:

```

hkSemaphoreBusyWait* m_jobDoneSemaphore;
hkUInt32* m_jobDoneFlag;

```

They have been replaced by this member:

```

hkaJobDoneNotifier m_jobDoneNotifier;

```

The functionality is the same as before. You just have to set the members inside `m_jobDoneNotifier` just as you would have set the old members.

1.10 Migrating to Animation, Physics_2012 and Common 2010.2

1.10.1 Common

1.10.1.1 Transition from GCC-XML to LLVM

The GCC-XML header parser has been replaced with a custom parser based on the LLVM compiler project. The new parser should be downloaded, unzipped, and the environment variable `HAVOK_PARSER_ROOT` set to the directory in which it was unzipped. This should be the root of the installation (i.e. the directory containing the `bin/` and `lib/` directories). `reflectionSettings.cache` files from 2010.1 will work as before, and the interface to `generateReflections.py` is unchanged.

The parser will normally work without configuration, however if configuration is required for custom projects, a file named `havokParserConfig.txt` may be created, in the same directory as the parser executable. Each line in this file is inserted into the command line of the parser. The parser command line is essentially that of the LLVM/CLANG compiler. In general, only include paths (`-I/path/to/dir`) and macros (`-DMACRO`) will need to be set.

Files may now be excluded from parsing by adding `//HK_REFLECTION_PARSER_EXCLUDE` anywhere in the file. Sections of a file may be excluded using `#ifdef __HAVOK_PARSER__ / #endif`. (`__GCCXML__` is still supported for compatability). The `addParserExcludeDir` list in `reflectionSettings.cache` can be used to exclude an entire directory from parsing. One of these methods should be used to exclude any code that is not parsed correctly and does not require reflection or memory tracking.

The format of reflected files and classes has not changed.

1.10.1.2 Version patch changes

Additional patch functions are now available to set the defaults for a class in the patch. This should be used in addition to setting the default in the reflected class, and allows the default to be properly viewed by the patching and reflection system in all cases. Patches Version Verify should report any new patches required, as before. It will now check that the default set in the most recent patch matches that set in the reflected class.

The format of the vector / transform patch macros has changed, `HK_PATCH_MEMBER_ADDED_VECTOR4` and `HK_PATCH_MEMBER_ADDED_QUATERNION` are renamed to `HK_PATCH_MEMBER_ADDED_VEC_4`, `HK_PATCH_MEMBER_ADDED_TRANSFORM` becomes `HK_PATCH_MEMBER_ADDED_VEC_12` and `HK_PATCH_MEMBER_ADDED_QSTRANSFORM` is renamed as `HK_PATCH_MEMBER_ADDED_VEC_16`

C String and pointer members may also be added using the simple patch macros, `HK_PATCH_MEMBER_ADDED_CSTRRING` and `HK_PATCH_MEMBER_ADDED_POINTER`

1.10.1.3 hkSerializeUtil, hkHavokSnapshot, hkTagfileWriter interface changes

All of these classes now have Options structures instead of passing boolean options directly. In every case, the default behaviour has not changed.

`hkSerializeUtil` now has options to:

- save in a text format (`hkXmlTagfileWriter`)
- save members not normally serialized (for debugging)
- save human readable copies of hex floats
- load only data which does not require versioning

1.10.1.4 XML Tagfile format created

A new XML file format has been introduced which fixes some fundamental issues with the old XML packfile format.

Floating point values are now written as their big endian hex value by default to avoid loss of precision on round tripping. There are options to use the normal human readable format or to add them as comments. Also, the hex floating point format is independent of locale settings.

1.10.1.5 Compression Features Added

A lightweight LZ style compressor and stream wrappers have been added. See `hkCompression`, `hkCompressedStreamReader` and `hkCompressedStreamWriter`. Decompression uses no memory and is similar in speed to a naive `memcpy`. Binary tagfiles typically compress by roughly 40%.

1.10.1.6 Improved Memory Leak Report

The memory leak report now searches leaked blocks for pointers to other leaked blocks. Thus it can generally give much more accurate reports of the cause of the root leak and suppresses the incidental details. This is particularly noticeable when a leak involves a chain or tree of referenced objects.

1.10.2 Content Tools

There are no major migration issues for Havok Content Tools 2010.2. Please refer to the release notes for details of any minor interface and behavior changes, as well as new features, bug fixes etc.

1.10.3 Physics

1.10.3.1 `hkAabbTree` is deprecated

`hkAabbTree` and the option `hkpWorldCinfo::m_useMultipleTree` which maintains an AABB tree for fast raycasting of inactive objects has been deprecated. Newer APIs provide better performance and memory usage. Other options are `KDTree` and the new hybrid broad phase.

1.10.3.2 New Hybrid BroadPhase

A new method to accelerate world queries (ray-cast, linear-cast, etc) is available and can be enabled with settings `hkpWorldCinfo::m_useHybridBroadphase` to `true`. The user guide has more detailed information. See the section "Using `hkpHybridBroadphase` to improve world queries performance" in the Asynchronous Queries collision detection chapter.

In addition, this new broadphase allow for convex queries, see `demo: Demo/Physics/Api/Collide/Culling`.

1.10.3.3 Constraint Stability Improvements

A new solver for constraints based on the `hkpBallSocketConstraintAtom` is available for the ball-and-socket, hinge, limited hinge and ragdoll constraints. The solver is capable of simulating previously unstable configurations (i.e. long chains, pairs of constrained bodies with disproportionate inertias or long force arms) in a more stable manner. It is disabled by default, and provided utility functions (see `hkpConstraintStabilizationUtil`) enable it for all constraints attached to a rigid body / in a physics system / physics world.

For certain configurations, the new solver may still fail to reach a stable solution. To prevent that from happening, one should use either one of the methods provided by the `hkpConstraintStabilizationUtil` or the *Create Constraints* asset export filter UI, and pre-stabilize the simulated system.

HVK-5740	Improve stability of ball-and-socket constraints.
HVK-4682	Create constraints filter should use smart logic to scale inertia tensor / angular damping to make systems stable by default.
HVK-4605	Ball and socket constraint gains energy in a very simple situation.

1.10.3.4 Per Object Slow Motion Feature

Two new methods, `hkpMotion::getTimeFactor()` and `hkpMotion::setTimeFactor()`, allow for controlling the time scale on a per motion basis.

1.10.3.5 Collision Detection Fixes

PSI collisions are now processed at every step of an asynchronous simulation (HVK-5701), and not only when the PSI step is reached. This only affects customers who carried out multiple asynchronous steps per PSI step.

Constraint Violated callbacks are now generated for singlethreaded and asynchronous simulations, and not only multithreaded as before (HVK-5687)

The linear cast agent now handles welding correctly. Previously, welded contact normals could cause the agent to return incorrect results (HVK-5642) The normal returned by the agent is still welded if welding is enabled.

1.10.3.6 `hkpCdPoint.m_contact` has been made protected

In order to access the contact point, use `getContact()` and the various `setContact()` methods.

1.10.4 Animation

1.10.4.1 Cropping of extracted motion has been fixed

The animation system now correctly crops the extracted motion. To fix this required some interface changes to be made. The interfaces affected are not commonly used, but you may encounter them if you are doing something out of the ordinary or if you have written custom subclasses of `hkaAnimation` or `hkaAnimatedReferenceFrame`, in which case you will have to update the interfaces on your custom subclasses.

The virtual method `hkaAnimation::getExtractedMotionDeltaReferenceFrame()` has additional arguments. The old interface was:

```
virtual void hkaAnimation::getExtractedMotionDeltaReferenceFrame( hkReal time, hkReal nextTime, int loops,
    hkQsTransform& deltaMotionOut ) const;
```

The new interface is:

```
virtual void hkaAnimation::getExtractedMotionDeltaReferenceFrame( hkReal time, hkReal nextTime, int loops,
    hkQsTransform& deltaMotionOut, hkReal cropStartAmount, hkReal cropEndAmount ) const;
```

The virtual method `hkaAnimatedReferenceFrame::getDeltaReferenceFrame()` also has additional arguments. The old interface was:

```
virtual void hkaAnimatedReferenceFrame::getDeltaReferenceFrame( hkReal time, hkReal nextTime, int loops,
    hkQsTransform& deltaMotionOut ) const = 0;
```

The new interface is:

```
virtual void hkaAnimatedReferenceFrame::getDeltaReferenceFrame( hkReal time, hkReal nextTime, int loops,
    hkQsTransform& deltaMotionOut, hkReal cropStartAmount, hkReal cropEndAmount ) const = 0;
```

HKA-1372

The cropping feature in the `hkaDefaultAnimationControl` does not correctly crop the extracted motion when looping.

1.11 Migrating to Animation, Physics_2012 and Common 2010.1

1.11.1 Common

1.11.1.1 Keycode Change

All Client and Evaluation keycodes have changed for Havok 2010.1. You should have received a new keycode with your download details. If you have not please contact your account manager for a new keycode. PcXS and Ps3XS keycodes have not changed.

1.11.1.2 Product Feature Deadstripping

It is now easier to strip out unused parts of the SDK without rebuilding the libraries. Product features are only linked in if explicitly requested. This release supports some of the most common targets for removal, more will be added in future releases.

It is now required that you include `hkProductFeatures.cxx` in your project. Typically this replaces your include for `hkBuiltinTypeRegistry.cxx` Usually you will want to do this in conjunction with your existing keycode include because the feature registration depends on which keycode values are set.

```
#include <Common/Base/keycode.cxx>
#include <Common/Base/Config/hkProductFeatures.cxx>
```

The `hkProductFeatures.cxx` ties together the optional features of the SDK.

- `hkClass` registration
- Versioning patch registration
- Memory tracker registration (see memory improvements section)
- Newly optional features such as the inertia tensor computer, heightfields etc.

By default, all features for your product combination are enabled. To disable features, place the relevant preprocessor macros between the two includes. For example if there is a keycode value for animation, but the application only needs physics we could use:

```
#include <Common/Base/keycode.cxx>
#undef HK_FEATURE_PRODUCT_ANIMATION // was defined by the keycode include
#define HK_EXCLUDE_LIBRARY_hkpVehicle // not using hkpVehicle library
#define HK_EXCLUDE_FEATURE_SerializeDeprecatedPre700 // not using old serialization
#define HK_EXCLUDE_FEATURE_RegisterReflectedClasses // if we register the classes elsewhere (using
    hkBuiltinTypeRegistry.cxx)
#define HK_EXCLUDE_FEATURE_MemoryTracker // not using memory tracker
#include <Common/Base/Config/hkProductFeatures.cxx>
```

See `Common/Base/Config/hkProductFeatures.inl` for a list of switches and descriptions. Some features have been replaced by this process. For example, using the default options, one no longer needs to call `registerPatches()`, `#include hkBuiltinTypeRegistry.cxx` nor call `hkSerializeDeprecated::initDeprecated()`.

Other macros which affect the operation are `HK_CLASSES_FILE` (for a custom class list) and `HK_COMPAT_FILE` to cut down on the number of old versions supported.

As part of this refactoring `hkSerializeDeprecated::initDeprecated()` has been renamed to `hkFeature_serializeDeprecatedPre700()`.

COM-1059	Create a config file that allows customers to dead-strip pieces of code they aren't using
----------	---

Note:

Versioning packfiles at runtime has been deprecated in version 7.0.0. To version packfiles at runtime (even those created *after* 7.0.0), `SerializeDeprecatedPre700` must not be excluded. In the case where only files from a few previous versions need to be supported, the class lists may be trimmed, but this feature will still cost several hundred Kb.

1.11.1.3 Memory Improvements

Garbage Collection

The `hkMemorySystem` interface has been extended for easier freeing of cached memory. `hkMemorySystem::garbageCollect()` will now garbage collect the calling threads local cache before collecting the shared cache. Each thread may call `garbageCollect()` though this may lead to duplication of work as the shared area will be collected several times. There are new methods which allow finer grained control (`garbageCollectThread()` and `garbageCollectShared()`)

COM-973	Garbage collection should call <code>hkThreadMemory::releaseCachedMemory()</code>
---------	---

Solver Buffer Size

`hkMemoryInitUtil::initDefault()` now requires you specify an allocator and default size for the solver buffer. See the 'Solver Buffer Size' of the user guide for more info.

COM-960	Solver buffer size must be set explicitly
---------	---

Memory Tracking and Reporting System

Note:

This is a beta feature.

The memory tracker system is a new system that aims to provide information about memory usage largely automatically. The tracker enables creating a 'snapshot' of memory usage which can then be processed by a variety of reports to provide information about memory usage. The snapshot captures memory allocations, as well as a description of how memory is used. For more details on what the memory tracker system is and how it works, please look at the section in the main Havok Guide.

The key points in using the memory tracker system are as follows..

- You must be using a `hkMemorySystem` that implements `memoryWalk`.
 - `hkCheckingMemorySystem` and `hkFreeListMemorySystem` do
 - In demos you can get the checking memory system by giving `-c` option on the command line
- You must have a library build compiled with `HK_MEMORY_TRACKER_ENABLE` (currently enabled if `HK_DEBUG` is enabled)
- You must explicitly enable memory tracking. (`hkMemoryInitUtil::initMemoryTracker`)

To try out memory tracking in the Havok demos use the options `'-c -me'` on the command line. `'-me'` will cause the demo system to output a collection of memory tracker reports once a demo is exited. `'-c'` makes the Havok demos use the `hkCheckedMemorySystem` – which provides stack traces for all allocations as well as implementing `memoryWalk` – a feature required for memory tracking operation. If the demo uses a lot of memory it can take a fair amount of time (perhaps minutes) to dump out all of the reports.

To incorporate memory tracking into your own project – a good starting point would be to check out `hkMemoryInitUtil` which has two methods `initMemoryTracker` and `quitMemoryTracker`. The tracker should be initialized just after the memory system and before `hkBaseSystem::init`, otherwise some allocations will be missed. Similarly the tracker should be quit after `hkBaseSystem::quit()` and before the memory system shutdown.

The VDB has been extended to grab a snapshot report in machine friendly plain text which may be analysed offline. A sample tool (`MemoryAnalyzer.py`) is supplied to visualize the data.

To include custom classes in the report, it must contain a memory allocator declaration (`HK_DECLARE_CLASS_ALLOCATOR` or `HK_DECLARE_NONVIRTUAL_CLASS_ALLOCATOR`). Additionally for the most accurate data, information about contained pointers is generated by GccXML parsing. If no such data is available, the blocks are scanned for values which look like pointers.

A variety of different memory reports are available and can be found in the `'Source/Common/Base/Memory/Tracker/Report'` directory.

COM-934	Automatic memory tracking
---------	---------------------------

1.11.1.4 Serialization Improvements

When patching fails (usually due to the default patches not being registered), an informative error is now given.

Arrays of 64-bit values will now be serialized correctly to tagfiles. Previously they were silently truncated to 32-bits when serialized. Any assignment or read to a `hkDataObject` int array that evaluates to a 64-bit data type or value will cause the array to become 64-bit, and sign extension will occur. A warning is output in this case. If the array is only accessed using `int/hkInt32`, it will remain as a 32-bit array.

COM-1017	Patching can fail without a helpful error message
COM-1049	Arrays of 64-bit ints not supported

1.11.1.5 Reflection Parser

A new reflection parsing system is used for 2010.1 . This uses **GCC-XML** to generate reflected class files and memory tracker definitions. The old script `regenerateXml.py` is replaced by `generateReflections.py` . The key differences are:







































- The new system operates on one project at a time, not one file. Reflected classes are output into the **Classes** directory in each project. Old `*Class.cpp` files will be automatically renamed to avoid conflicts.
- It compiles each file using a version of **GCC**. Each header file must therefore contain the required includes and include paths must be configured correctly
- `reflectionSettings.cache` is used to set extra include paths for the build
- A cache of reflected files, called `reflections.db` is stored in each project to speed up rebuilds. This file and the contents of the **Classes** directory may safely be deleted if a completely clean build is required.

Installation

See section ‘The GCC-XML parser’ in the user guide for more detailed installation instructions, however in general:

- Install **GCC-XML** from a packaged distribution for Mac or Linux builds, or from <http://www.language-binding.net/pygccxml/download.html> for windows.
- Install `pygccxml` from the above location. Python 2.5 or 2.6 is required (`cygwin python` will not work correctly)
- Configure `generateReflections.py` to find **GCC-XML**. This requires editing the file `Tools/Serialize/reflectionDatabase/gccxmlSettings.py`. Normally only setting `HK_GCCXML_PATH` is required.

Both packages must be installed – **GCC-XML** and `pygccxml`. The location of the two packages at the time of release, at the above download location is shown below:

Browse Files for C++ Python language bindings						
File/Folder Name	Platform	Size	Date ↓	Downloads	Notes/Subscribe	
Newest Files						
 pydsc-0.4.zip		9.5 KB	2009-02-15	939		
All Files						
▼  pydsc		52.0 KB	2009-02-15	1,565	 	
▶  pydsc-0.4		9.5 KB	2009-02-15	939	 	
▶  pydsc-0.2		42.5 KB	2006-08-24	626	 	
▶  ctypes code generator		3.0 MB	2009-01-19	373	 	
▼  gccxml-setup		5.5 MB	2009-01-07	1,323	 	
▶  gccxml-7-JAN-2009-rev1.127		2.7 MB	2009-01-07	1,100	 	
▶  gccxml-22-DEC-2008		2.7 MB	2008-12-23	223	 	
▼  pygccxml		37.4 MB	2008-10-20	5,714	 	
▶  pygccxml-1.0		21.6 MB	2008-10-20	3,850	 	
▶  pygccxml-0.9.5		15.7 MB	2008-02-04	1,864	 	
▶  pyplusplus		14.3 MB	2008-10-20	4,173	 	
▶  pyboost		11.4 MB	2006-06-07	2,352	 	

More detailed information about installation and troubleshooting is found in the user guide. This parser also generates memory tracker files for the Havok libraries.

New reflection class layout

The old reflection parser generated one reflected classfile per header file. The new reflection parser generates one file per project, containing all of the reflected classes for that project. This is located in the **Classes** directory at the root of the project. The **hkClass** format has not changed so existing reflected class files do not need to be regenerated, however new or updated reflected classes should be generated using the new reflection system. The single reflection class file will contain all reflection information for the project, so it will conflict with any old classfiles remaining in the project. For this reason, old ***Class.cpp** files that are encountered during the reflection build are automatically renamed with a **.cpp.0** extension. These files should be removed from the project.

Directory parsing rules

When a directory name is passed to **generateReflections.py**, the following rules are used to determine what is parsed:

- If the specified directory contains a **reflectionSettings.cache** file, it and its subdirectories are parsed as a single project
- If **reflectionSettings.cache** files are found in subdirectories of the specified directory, each directory containing a **reflectionSettings.cache** file and its subdirectories are parsed as a single project

- If no `reflectionSettings.cache` files are found, the specified directory and its subdirectories are parsed as a single project. The default include paths are used.

Managed class manifest generation

Managed class manifests, as used by Havok Behavior, are generated using the reflection parser. The parser is run internally by the `genManagedClassManifests.py` script. The parser must be installed and functioning correctly to rebuild `HavokAssembly` using custom managed classes. In particular, a `reflectionSettings.cache` file may be necessary to set custom include directories.

Custom Project Configuration – `reflectionSettings.cache`

The parser compiles each file and must have enough information to access all required header files. Custom configuration is carried out by creating a file called `reflectionSettings.cache` in the root directory of the project. This file contains a list of settings, in the form of a Python dictionary. There are three configuration values that can be set from this file:

- **addIncludePath** – This is a list of extra directories to add to the search path while searching for header files
- **setPchFile** – Set a precompiled header file. While precompiled headers are not used by the parser, this file is automatically included for each file in the project
- **setPrefix** – Set the prefix to prepend to any generated files. This may be used to distinguish between files from different projects

A sample `reflectionSettings.cache` file is shown below:

```
{'addIncludePath': ['C:/Havok/hk2010_1_0/Source', 'C:/Code/Src'], 'setPchFile': ['stdafx.h'], 'setPrefix': ['hkb']}
```

The file must take this format. The brackets and quotes are all required syntax. Extra includes are added inside the square brackets after **addIncludePath**, as a comma separated list.

Once one `reflectionSettings.cache` file is encountered while searching a directory, the build system assumes that all projects contain this file and it will ignore directories that do not.

Checklist for custom projects

The procedure to convert an old custom project to use the new system for serialized or managed classes is therefore:

- Install and setup GCC-XML and `pygccxml`.
- Create a `reflectionSettings.cache` file in the root directory of the custom project, containing any extra includes required to compile correctly. If more than one project is used, create one for each
- Run the parser, either directly as `generateReflections.py` from the command line or via `genManagedClassManifests.py`.
- Remove any old classfiles from the project (usually generated files ending in `Class.cpp` . These will have been renamed to avoid conflicts)

- Add the newly generated `Classes/projectNameReflections.cpp` to the project

1.11.2 Content Tools

1.11.2.1 Support for 3ds Max 2011, Maya 2011 and SoftImage 2011

The Havok Content Tools now support version 2011 (both 32 and 64 bit) of 3ds Max, Maya and SoftImage.

1.11.3 Physics

1.11.3.1 Added transform per `hkpExtendedMeshShape::TrianglesSubpart`

You can now set an `hkQsTransform` (translation, rotation and scaling) per each `hkpExtendedMeshShape::TrianglesSubpart`, which can be very useful for geometry instancing.

HVK-4156	Add transform to <code>hkpExtendedMeshShape::TrianglesSubpart</code>
----------	--

1.11.3.2 Inconsistent winding viewer

When exporting a mesh from a modeller you can check the box "Mark Inconsistent Winding Edges" in order to mark and later view the edges and triangles reported as inconsistent welding pairs. There is also a new VDB viewer called "Inconsistent Winding Viewer" which can be useful for analyzing the geometry when meshes are exported with this option turned on.

HVK-5218	Welding viewer should highlight unwelded edges
----------	--

1.11.3.3 `hkpCompressedMeshShape` improvements

The building process has changed: you have to call `addInstance` at least once for the added geometries and convex pieces in order to correctly compute the AABB of the mesh. Scaling is now supported in the transform (an `hkQsTransform`) when adding geometries and instances. Also enabled collision filtering to work both on CPU and SPU. Another improvement is the fact that degenerate triangles are being filtered now both in the builder and the `getFirstKey` and `getNextKey` functions.

HVK-5314	<code>hkpCompressedMeshShape</code> : Add a scaling ability like <code>hkpExtendedMeshShape</code> has
HVK-5354	<code>hkpCompressedMeshShape</code> does not implement <code>getCollisionFilterInfoImpl</code>
HVK-5510	<code>hkpCompressedMeshShape</code> can return degenerate triangles

1.11.3.4 Improvements to the `hkpTriggerVolume`

The `hkpTriggerVolume` utility was introduced in 7.1 to demonstrate the use of contact point callbacks in creating a body which senses contacts without having a physical effect. Much work has been done to improve its quality since then and it should now offer a useful alternative to `hkpPhantoms` and

`hkpPhantomCallbackShapes` as a way of implementing sensors. In particular, `hkpTriggerVolumes` are sensitive to continuous events.

As part of this work, the `hkpTriggerVolume`'s interface has been slightly altered. The methods `enterEvent()` and `leaveEvent()`, which could be fired multiple times for the same body, have been replaced by a single method `triggerEventCallback()` which is called at most once for each body. It describes whether the body entered during the frame, left during the frame, or both entered and left during the frame.

Additionally, support for the `hkpTriggerVolume` has been hard-wired into the `hkpCharacterProxy`, since the `hkpCharacterProxy` is responsible for its own integration. To pick up interactions between them, the `hkpTriggerVolume` fires a variant of the `triggerEventCallback()` which has a `hkpCharacterProxy` argument.

HVK-5432	<code>hkpTriggerVolume</code> should order its enter and leave events by when they occur
HVK-5423	The <code>hkpTriggerVolume</code> can give extra enter and leave events using <code>SIMULATION_TYPE_CONTINUOUS</code>
HVK-5472	<code>hkpTriggerVolume</code> can fail to raise enter and leave events if a TOI is generated but no PSI contact points
HVK-5467	Trigger Volume should reference count its <code>m_overlappingBodies</code>
HVK-5422	<code>hkpTriggerVolume</code> can give <code>hkpCharacterRigidBody</code> support planes
HVK-5415	Visual Debugger (VDB): Improve visualization of <code>hkpTriggerVolume</code> instances in the VDB
HVK-5418	Demo for comparing the <code>hkpTriggerVolume</code> , <code>hkpPhantomCallbackShape</code> and phantoms
HVK-5477	Add unit tests for <code>hkpTriggerVolume</code>

1.11.3.5 New approach to the lifetime management of some utility objects

The `hkpBreakOffPartsUtil` and the `hkpCollisionCallbackUtil` managed their own memory by adding a reference to themselves and being a `hkpWorldDeletionListener`. This was inflexible, so we've introduced a new abstraction, `hkpWorldExtension`, for this kind of object. A `hkpWorldExtension` is created and then added to the world, at which point the world holds a reference to it, similar to entities. There are changes to the usage of both classes:

- The `hkpBreakOffPartsUtil` no longer takes the `hkpWorld` as an argument to its constructor. It should be constructed and then added to the world as a `hkpWorldExtension`.
- The `hkpCollisionCallbackUtil` is no longer held in the member `hkpWorld::m_collisionCallbackUtil`. You can find it using `hkpWorld::findWorldExtension(HK_WORLD_EXTENSION_COLLISION_CALLBACK)`. To be consistent with the naming used in `hkpWorldExtension`, its `addCollisionCallbackUtil()` and `removeCollisionCallbackUtil()` methods have been renamed `requireCollisionCallbackUtil()` and `releaseCollisionCallbackUtil()`.

HVK-4781	Refactor reference counting for utility objects
----------	---

1.11.3.6 `hkpCharacterProxy` on SPU

The `hkpCharacterProxy` can now be multithreaded on SPU.

A caveat of using the `hkpCharacterProxy` on SPU is that the callback methods of `hkpCharacterProxyListener` are not fired. Hooks are provided for intercepting the same information on SPU (see `hkpSpuCharacterProxyUtil`), but issues such as code-size will proba-

bly prevent any sophisticated game-code from running there. An alternative is to override `hkpCharacterProxyUtil::handleResults()`.

Creating and running multithreaded jobs is done by using the `hkpCharacterProxyJobUtil`. This has not changed significantly although there are some extra members in the `JobData` struct.

Some internal methods of the `hkpCharacterProxy` class have had their interface changed and many internal classes involved in multithreading the `hkpCharacterProxy` have been changed substantially.

Support for the `hkpTriggerVolume` has also been hard-coded into the `hkpCharacterProxy`.

HVK-5258	Character controller on SPU
----------	-----------------------------

1.11.3.7 Changes to `hkpBreakOffPartsUtil`

The `hkpBreakOffPartsUtil::LimitContactImpulseUtil` was designed to run on SPU. However, when its default behavior was not quite as desired, having it on SPU made customizing it difficult (due, for example, to code size limitations). We have now provided an additional implementation which always runs on CPU to allow easier modification. The `hkpBreakOffPartsUtil` has a factory method, `hkpBreakOffPartsUtil::createLimitContactImpulseUtil()`, which by default creates a `LimitContactImpulseUtilDefault`. If you want a custom implementation, you should override the factory method to create an object of (your subclass of) `LimitContactImpulseUtilCpuOnly`.

As part of this change, `hkpEntity::m_limitContactImpulseUtil` has been renamed `m_limitContactImpulseUtilAndFlag` and is no longer a pointer. You should now use `hkpBreakOffPartsUtil::getLimitContactImpulseUtilPtr()` to obtain a pointer to the utility.

See the section [New approach to the lifetime management of some utility objects](#) above for another interface change to the `hkpBreakOffPartsUtil`.

HVK-5447	Support the <code>hkpBreakOffPartsUtil</code> on PPU
----------	--

1.11.3.8 Deprecation of `hkpConstrainedSystemFilter`

The `hkpConstrainedSystemFilter` has been deprecated and may be removed in a future release. It is recommended that `hkpConstraintCollisionFilter` be used to disable collisions between pairs of constrained bodies.

HVK-5303	Deprecate the <code>hkpConstrainedSystemFilter</code>
----------	---

1.11.3.9 Added `hkpMultithreadedVehicleManager`

Note:

This is a beta feature.

The `hkpMultithreadedVehicleManager` has been introduced, supporting multithreaded vehicle execution. The manager generates and executes jobs of type `hkpVehicleIntegrateJob` which calculate the set of impulses to apply to the vehicle and to other bodies, and then applies the impulses afterwards

in a deterministic manner. The `hkpVehicleIntegrateJob` is currently **not supported on SPU** . The `hkpVehicleIntegrateJob` offers up to a 40% speed-up versus the `hkpRayCastBatchingManager` or `hkpLinearCastBatchingManager` for a group of vehicles. These only support multithreaded execution of the wheel collision step of vehicle simulation.

HVK-5248	Multithread the Vehicle
----------	-------------------------

1.11.4 Animation

1.11.4.1 Significant Speed Improvements

New animation decompression and blending modules particularly appropriate for SIMD platforms have been introduced in this release. These new modules consist of the following classes:

- `hkaQuantizedAnimation` - A new, lightweight, compression type optimized for speed of decompression on SIMD platforms.
- `hkaBlender` - A new blend framework optimized for blend speed on SIMD platforms.
- `hkaQuantizedSampleAndCombineJob` - A new job description for sampling and blending which is designed to be general and used on all platforms in single or multithreaded modes.

For optimal performance, `hkaQuantizedAnimation` should only be blended with other instances of `hkaQuantizedAnimation` using the `hkaQuantizedSampleAndCombineJob` job description, or by using an `hkaAnimatedSkeleton` containing only `hkaQuantizedAnimation` instances. Mixing `hkaQuantizedAnimation` instances with other animation types (such as `hkaSplineCompressedAnimation`) will not use the new `hkaBlender` framework and will not provide the best possible performance.

The `hkaQuantizedSampleAndCombineJob` job description contains a flat list of animations to be blended, along with per-job, per-animation and per-bone blending options. The `hkaQuantizedSampleAndCombineJob` is designed to be filled out by users manually, or automatically through the `hkaAnimatedSkeleton` class. Please see *Playback of hkaQuantizedAnimation* section of the *Animation Runtime* chapter of the user guide for further documentation. See also the `Animation\Api\Multithreading\SampleAndBlend\SampleAndBlendMultithreadingDemo` in the Demo Framework for source code examples.

HKA-1339	New SIMD Optimized Sample And Blend Pipeline
----------	--

Important:

`hkaQuantizedAnimation` is available in Havok Behavior. `hkaBlender`, however, is not. This means that `hkaQuantizedAnimations` can be decompressed but cannot be blended using the SIMD blender. Therefore `hkaQuantizedAnimation` inside of Havok Behavior will not yield the same performance gains as when used in Havok Animation when blending many animations. Future work will be done to integrate `hkaQuantizedAnimation` and `hkaBlender` into Havok Behavior.

1.11.4.2 Deprecation of `hkaWaveletCompressedAnimation` and `hkaDeltaCompressedAnimation` compressed animation types.

`hkaWaveletCompressedAnimation` and `hkaDeltaCompressedAnimation` compressed animation types are now deprecated and may be removed in a future release. Prefer `hkaQuantizedAnimation` for maximum decompression speed, or `hkaSplineCompressedAnimation` for minimum compressed asset size.

HKA-1340	Deprecation of <code>hkaWaveletCompressedAnimation</code> and <code>hkaDeltaCompressedAnimation</code> compressed animation types.
----------	--