“The problem is most likely caused by multiplying the ray end-points by a large scaling factor (m_FtoBScale = 253) **and may be fixed by adding some epsilon** when performing the ray pruning.

Will disable the bundleRaycast unit test for MOPPs until this issue is fixed.”
(http://jira.dub.havok.com/browse/HVK-6058) <OPEN>

“**Added small testbed to play with normalize tolerances. Bumped up tolerance as a result**”
(http://jira.dub.havok.com/browse/HVK-1767)

“**The fix is easy - add a tolerance to the check**.”
(http://jira.dub.havok.com/browse/HVK-5220) <OPEN>

“The capsule implementation is **not top quality and not epsilon safe**…
However the code is not trivial and the best thing to do is to use a mathematical software package (e.g mathematica) to simplify the equations and **remove all divisions and normalize**.”
(http://jira.dub.havok.com/browse/HVK-733) <OPEN>

“We've picked a default tolerance for our algorithm such that GSK won't break, but it lets through triangles which break sphere-triangle.”
“Add degeneracy check which will catch triangles which break closest point triangle check”
“Disadvantages: We have to pick yet another suitable eps, or maybe several.”
“If we check against 'eps' we have to pick yet another suitable eps, and isnondegenerate is called by more than just hkMeshShape::getNextKey() in the engine.”
“**After talking to Andrew, it's probably better to have the "check-everything" degeneracy check**. That way client meshes may be filtered offline.”
(http://jira.dub.havok.com/browse/HVK-2539)

“for example that a unit cube will have its vertices perturbed by values in the range ~[-1e-6,1e-6].
N.B **Hopefully this would be sufficient to "unbreak" any dodgy inputs, and NOT change noticeably any other types of input (eg. a unit cube!).** “
(http://jira.dub.havok.com/browse/HVK-169)

“The problem is that painters can introduce "degenerate" tris as part of the face that they cut into the navmesh. However, since we are keeping the internal faces (rather than discarding them like the carvers do) **we should keep "degenerate" tris created by painters**.”
(http://jira.dub.havok.com/browse/HAI-1647) <OPEN>

“The reason is that the two vertices at the endpoints of the tiny boundary edge, when projected onto the ground plane, are less than HKAI_TRIANGULATOR_EPSILON away from each other”

(http://jira.dub.havok.com/browse/HAI-890)<OPEN>

“while (clippedFrom(2) >= ep[2]) { ep[2] ++; }
becomes
while (clippedFrom(2) > ep[2] - (signRayDir[2] * HK_REAL_EPSILON * 128.f)) { ep[2] ++; }”
(http://jira.dub.havok.com/browse/HVK-5137) <OPEN>

“if(hkMath::fabs(maxDistFromPlanes) > HK_REAL_EPSILON)”
(http://jira.dub.havok.com/browse/HVK-5037)

“**Changed tolerance of assert** (exists only in fpu version, hence affects only PC (non-SIMD, and GC). Choice of new value (1e-16) ensures validity of assert as detailed in 1074.

This also fixes 1583 and 1483 (see test cases added to test/unititest/hkcollide2/agents/conservativeassertbugs.cpp.

Note that this assert is still conservative (more so on PC) - however the test case illustrates that it is unlikely that problematic vectors will arise in either algorithm as worse-case tests produce vectors 1e7 times greater than ones which could raise an unnecessary assert.”
(http://jira.dub.havok.com/browse/HVK-1583)

“This seems like a fundamental flaw; the only way to avoid this would be to check for degenerate triangles as they're being compressed, and if one is found, modify the vertices until you get something that works.”
(http://jira.dub.havok.com/browse/HVK-3873) <WONT FIX>

“Note that continuing on lets the edge connection code correctly identify that the overlap is < epsilon (hardcoded 1e-6) and so it will not connect.
Also m_minEdgeOverlap will cover this case if it is > epsilon.“
(http://jira.dub.havok.com/browse/HAI-504)

“Apart from using different epsilon values, they handle singular input differently”
(http://jira.dub.havok.com/browse/COM-1552) <OPEN>

“Enabling 'Merge Coplanar Triangles' messes up Vertex Colors: + **allow for disabling the different epsilon checks in the UI**”
(http://jira.dub.havok.com/browse/HKD-583)

"**Made epsilon correct** for hole replacement segment check."
(http://jira.dub.havok.com/browse/HAI-758)

"**I was able to get around this for the client by increasing the epsilon tolerance** for the vertical check to (fabsDelta <= HK_REAL_EPSILON * 1024.f) instead of (fabsDelta < HK_REAL_EPSILON * 256.f). The <= was needed since it seemed that the transformation actually made it equivalent to a multiple of epsilon (we noticed either 256 or 1024)."
(http://jira.dub.havok.com/browse/HVK-4933)< OPEN>

"hkaiNavMeshGenerationUtils_triangulateLoops has a hardcoded area rejection test:
if (loopsIn[0].m_area < 1e-3f)"
(http://jira.dub.havok.com/browse/HAI-889) <OPEN>

"although you may actually want tolerance*tolernace - we need to think about the units a bit.

Hopefully we'll track down some real data for this shortly."
(http://jira.dub.havok.com/browse/HVK-4196) <OPEN>

"asserted if it was "small" (less than REAL_EPSILON, which is about 1e-7)."
(http://jira.dub.havok.com/browse/COM-132) <OPEN>

"A customer has pointed out a situation when the function incorrectly returns HIT_TRIANGLE_FACE when it should return HIT_TRIANGLE_EDGE**. I tried reducing the tolerace variable by a factor of 100 and this led to the correct answer**. The numbers involved are in the ticket"
(http://jira.dub.havok.com/browse/HVK-3582) <OPEN>

"**Of course the "best" solution would use two different tolerances, but that's not going to be simple to implement I think.**"
(http://jira.dub.havok.com/browse/COM-150) <OPEN>

"Thus, they would like to be able to tweak the tolerances that get used inside the convex hull builder."
(http://jira.dub.havok.com/browse/COM-540) <OPEN>

"**Obviously this is rare, but could happen. I suggest we do the following**, with probably negligible performance difference: **+ HK_REAL_EPSILON**"
(http://jira.dub.havok.com/browse/HCL-992) <OPEN>

"**This triangle does not trigger the degenerate-triangle warning in the operator. We should increase the tolerance of this check to much greater than what it is. If there is a non-long-and-thin triangle nearby, it will give a much better influence than a thin one, so we should err on the side of caution**." (http://jira.dub.havok.com/browse/HCL-1030) <OPEN>

"And we only have 13 or 14 bits for quantization, so this gives us about a 1cm error for every vertex during cutting - quite large."
(http://jira.dub.havok.com/browse/HAI-289) <OPEN>