Dev:Shading/Tangent Space Normal Maps

- Chinese

- Developer page
- Discussion
- View source
- History

Page

- What links here
- Related changes
- Permanent link

From BlenderWiki

Tangent Space Normal Maps

Implementation Dependent

A common misunderstanding about tangent space normal maps is that this representation is somehow asset independent. However, normals sampled/captured from a high resolution surface and then transformed into tangent space is more like an encoding. Thus to reverse the original captured field of normals the transformation used to decode would have to be the exact inverse of that which was used to encode.

This presents a problem since there is no implementation standard for tangent space generation. Every normal map baker uses a different implementation and, additionally, there is no standard for how the interpolated frame is to be used to transform the normal into tangent space.

The math error which occurs from this mismatch between the normal map baker and the pixel shader used for rendering results in shading seams. These are unwanted hard edges which become visible when the model is lit/shaded. They are a source of wasted time and frustration to an artist/designer.

Order-Dependencies

To make matters worse it is not enough to use the same tangent space generation code. Most implementations have order-dependencies which can result in different tangent spaces depending on which order faces are given in or the order of vertices to a face. Others generate different results if the index list changes (multiple/single/single with duplicate vertices). And others again produce different results if degenerate primitives are removed.

Order-dependencies also result in mirrored meshes not always getting correct mirrored tangent spaces. See for instance an example of a commercial product used to produce tangent spaces, on a mirrored model, in figure 17a on page 45 in the master's thesis Simulation of Wrinkled Surfaces Revisited (http://image.diku.dk/projects/media/morten.mikkelsen.08.pdf) by Morten S. Mikkelsen.

There are additional examples of problems with different commercial products shown on pages 44, 52-56. These problems do not just occur with mirroring. The problem is general and is mostly visible at tangent space splits in locations where the vertex normal is shared among surrounding faces. The unwanted hard edge is the result of the decoded normal deviating from the original captured normal.

The Solution

The tangent space generation code of Morten S. Mikkelsen overcomes these problems. The implementation is designed, specifically, to make the generation of tangent space as resilient as possible to a 3D model being moved from one application to another. That is generate the same tangent spaces even if there is a change in index list(s), ordering of faces/vertices of a face, and/or the removal of degenerate primitives. Both triangles and quads are supported.

The implementation was made by Morten S. Mikkelsen during the development of his master's thesis and is free for anyone to use. It is contained in the two standalone files mikktspace.h (https://svn.blender.org/svnroot/bf-blender/trunk/blender/intern/mikktspace/mikktspace.h) and mikktspace.c (https://svn.blender.org/svnroot/bf-blender/trunk/blender/intern/mikktspace/mikktspace.c) . This makes it easy for anyone to integrate the implementation into their own application and thus reproduce the same tangent spaces. This also makes the code a perfect candidate for an implementation standard. We hope the standard will be adopted by as many developers as possible.

The standard is used in Blender 2.57 and is also used by default in xNormal since version 3.17.5 in the form of a built-in tangent space plugin (binary and code).

Pixel Shader Transformation

In regards to the interpolated tangent space the baker in both Blender and the xnormal plugin will keep the vertex normal and the vertex tangent normalized in the vertex shader. However, in the pixel shader the "unnormalized" and interpolated vertex normal and tangent are used to decode the tangent space normal. The bitangent is constructed here to avoid the use of an additional interpolater and again is NOT normalized.

```
// vNt is the tangent space normal
vB = sign * cross(vN, vT);
vNout = normalize( vNt.x * vT + vNt.y * vB + vNt.z * vN );
```

The key to get flawless results is that the baker is designed to do the EXACT inverse of this very transformation allowing the pixel shader to remain fast and simple.

Note that mikktspace returns the tangent spaces in an unindexed form. Do NOT try to average these over an existing index list. It will NOT work correctly. If you need the tangent spaces to be indexed you can use a welder to create a new index list. If you do not have such an implementation you can get a free one here:

http://jbit.net/~sparky/academic/welder/

This welder works with an arbitrary number of floats per vertex.

The Quad Caveat

Eventhough mikktspace supports quads these will eventually be split into triangles before baking and pixel shading. Furthermore, though mikktspace guarantees a consistent choice of tangent frame, at the quad vertices, the interpolated tangent frame is heavily affected by which diagonal split is chosen later on. This is not a big issue when the tangent frame transitions slowly across the face. For a well-behaved low polygonal mesh this is true for most regions.

When there is a great change in the tangent frame a problem occurs when the tools pipeline (prior to pixel shading) and the baker do not choose the same diagonal split. One possible solution is to choose your split based on an order-independent strategy. For instance, split all quads by the shortest diagonal using the vertex positions. If these have the same length then split by the shortest diagonal using the texture coordinates. This will lead to an order-independent choice and works with mirrored meshes. The triangulator in xNormal has supported this since 3.17.5 and triangulation is performed after the mikktspace plugin is done. Reproducing the same splits in your own tools pipeline is trivial to do.

Another more trivial solution which any artist can perform on his/her own is to simply triangulate the model before baking and export which will also ensure consistent triangulation. However, this way the quad information is lost which may or may not be of any importance depending on the game engine and how the asset is to be used.

We stress that in most cases this is NOT a visible issue. But in the few cases where it is a problem this section explains why and how to deal with it.

Retrieved from "https://wiki.blender.org/index.php/Dev:Shading/Tangent_Space_Normal_Maps"

Contents

quick search...

Wiki

- Report a wiki bug
- Wiki Guidelines

- Special pages
- Categories
- Popular pages
- New files
- New pages
- Recent changes