Triangle mesh
tangent space calculation

Martin Mittring
Object space normal maps

- 3d vector encoded as colour (colourful)
- Simple math
- Reuse limited to translation / scale and per object mirror / Rotate
Tangent space normal maps

- 3d vector encoded as colour (blueish)
- Relative to the surface (in tangent space)
- Reuse: Arbitrary
- Texture compression
- Hard to avoid artefacts and seams

-> good tangent space calculation helps
tangent space is a useful mathematical tool

- \((\text{tangent, binormal, normal}) = 3 \times 3 \text{ matrix}\)
- Computations in tangent space can be more efficient (cheaper pixel shader)
- Storing data in tangent space decouples the data from its local surface orientation which allows arbitrary reuse and efficient storage
- Applications: normal maps, horizon maps, POM, PTM, …
Requirements

- Easy to integrate (source, 3dsmax/maya)
- Efficient
- No magic
- Support for mirroring
- Minimal vertex splits
- Tiling textures
- Documented
- Tested and proven
- Tessellation independent result (L Shape)
L shape problem
Step 1/3: TS per triangle

- Compute 3x3 matrix that transforms 3 given points in UV space to 3 points in world space – ignoring the translation
- Weight by the UV triangle size to avoid domination of small triangles
Tangent space per triangle
Step 2/3: Normal per vertex

- Accumulate neighbour triangle normals per vertex (if edge [between vertex triangle and neighbour triangle] is smooth)
- Weighted by angle to get tessellation independent result (L shape problem)
Normal per vertex
Step 3/3: TS per vertex

- Accumulate neighbour triangle u and v per vertex (if edge [between vertex triangle and neighbour triangle] is smooth)
- Split vertices in case of mirroring (matrix party) or heavy rotations (90 degree)
- Weighted by angle to get tessellation independent result (L shape problem)
TS per vertex
Compressing the tangent space matrix

- Normalize \( u \) and \( v \)
- Store \( u \) and \( v \) in 8 or 16bit per component
- \( n = \text{normalize}(\text{cross}(u,v)) \times k \)
- \( k = \{-1;1\} \) is required for mirroring
- Storing \( n \) and reconstructing \( u \) or \( v \) does not cope well with shearing
Tips to get best quality

- The same TS computation everywhere
- Store T or T⁻¹
- Artist can hide seams
- Reorthogonalize? [Engel05]
- Avoid shearing in the input data
- Check with reference tangent space texture
- Do shading in world space
- Decoding with *2-1 doesn’t support (0,0,1), *128/255-1 does
Triangle mesh tangent space calculation

- Thanks to Ivo Herzeg and Crytek
- Free source can be found in the free Far cry MOD SDK
- Source and more details can be found in ShaderX4 book [Engel05]

References: