Overview

- Background & Motivation
- Implementation
- Results
Anisotropic Lighting
Bump Mapping
Frame Mapping
Implementation in Sh

- Normal & Tangent Maps
- Quaternion Map
Constructing the Normal Map

- treat the input image as a height field
- use forward or central differencing to find the image gradients
- map image to the xz-plane, so that +y is the up direction
- normal is then defined by the gradient and a scalar factor
Constructing the Tangent Map

- not sure what we really want...?
- projecting the gradient onto the tangent plane is a good bet
Code Snippets in Sh

```
ShUnclamped< ShTextureRect<ShVector3f> > normalMap(w, h);
ShUnclamped< ShTextureRect<ShVector3f> > tangentMap(w, h);

fsh = SH_BEGIN_FRAGMENT_PROGRAM {
    ShVector3f zmap = tangentMap(tc);
    ShVector3f ymap = normalMap(tc);

    normal = (tanToView | ymap);
    tan1 = (tanToView | zmap);
    tan2 = cross(normal, tan1);
}
SH_END_PROGRAM;
```
Constructing the Quaternion Map

- construct normal and tangents as before
- construct rotation/orthogonal matrix mapping the standard frame to new frame
- construct quaternion from the matrix
More Code Snippets

```c
ShUnclamped< ShTextureRect<ShVector4f> > quaternionMap(w, h);

fsh = SH_BEGIN_FRAGMENT_PROGRAM {

    ShQuatertionf frame(quaternionMap(tc));
    ShVector3f ymap = ShVector3f(0,1,0);
    ymap.setUnit(true);
    ShVector3f zmap = ShVector3f(0,0,1);
    zmap.setUnit(true);
    ShMatrix4x4f mat = frame.getMatrix();
    ymap = mat | ymap;
    zmap = mat | zmap;

    normal = (tanToView | ymap);
    tan1 = (tanToView | zmap);
    tan2 = cross(normal, tan1);

} SH_END_PROGRAM;
```
Current Issues

- space vs. time trade-off
- interpolation in textures
- use quaternions directly in a lighting model?
Pictures
Pictures
Pictures
Pictures