View Independent Environment Mapping with Dual Paraboliod Maps

Mark J. Kilgard
NVIDIA Corporation

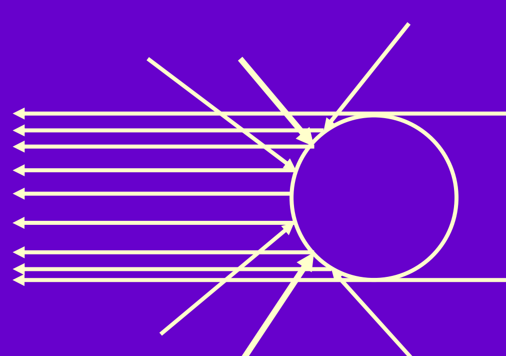
Environment Map Problem

- Sphere mapping is view dependent
- Requests constant regeneration of sphere map texture for changing viewpoints
- Must be a better way!

There is!

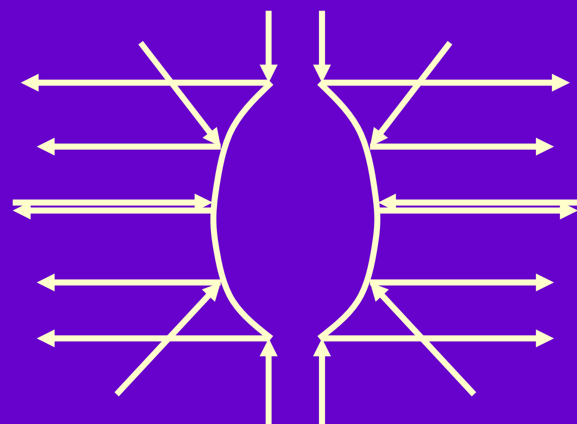
- Wolfgang Heidrich & Hans-Peter Seidel have proposed a view independent environment mapping scheme
- No special hardware requirements
- Uses two environment textures, each with a parabolic basis

Sphere maps work like this



Sphere collects environment image

Dual paraboloid approach



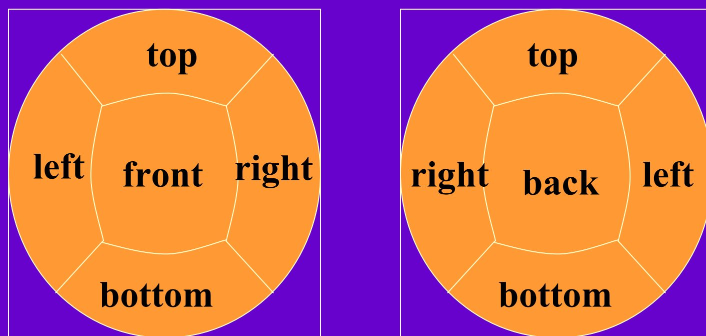
Parabolic basis (not spherical)

Requires two texture images

Advantage

- Dual paraboloid maps capture complete environment with no singularities
- Linear basis
- Well suited for dual-texture hardware such as the RIVA TNT
- View independent!

Dual Paraboloid Map

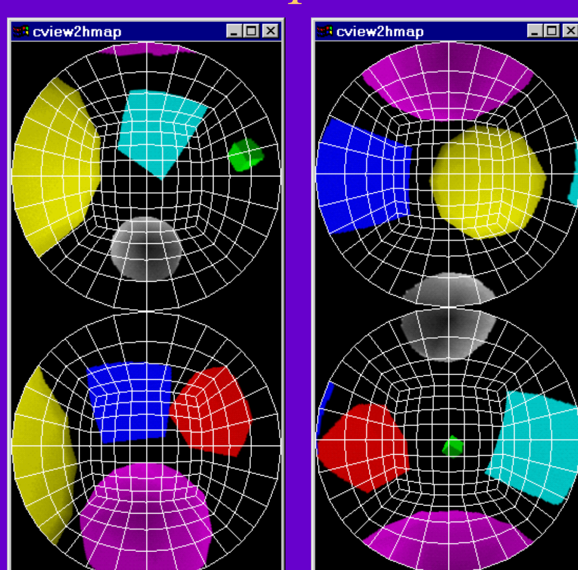


front texture

back texture

alpha=1.0 inside circle,
alpha=0.0 outside circle

In practice



meshes added for clarity

Issues

- Must have way to "pick" environment from the correct texture of the two for a given pixel
  - Alpha testing works in two passes
  - The blend texture environment works will for ARB_multitexture
- Rotation needed to supply view independence & projection needed
- Think texture matrix and perspective correct textures!

Two pass approach

- Adjust texture matrix for front view
- Bind to "front" paraboloid map
- Draw object with reflection map texgen, alpha test away non-unity alpha
- Adjust texture matrix for back view
- Bind to "back" paraboloid map
- Draw object same as before

Texture Matrix Setup

$$T \bullet P \bullet S \bullet M_l^{-1} \bullet \begin{bmatrix} Rx \\ Ry \\ Rz \\ 1 \end{bmatrix} = \begin{bmatrix} s \\ t \\ 1 \\ 1 \end{bmatrix}$$

M_l^{-1} is linear part of the affine modelview transformation

Texture Matrix Sub-parts

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

P is projective transform that divides by the z component

Texture Matrix Sub-parts (2)

$$S = \begin{bmatrix} -1 & 0 & 0 & dx \\ 0 & -1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

S subtracts object-space reflection vector from each paraboloid's orientation, d

d is either [0, 0, -1] or [0, 0, 1]

Texture Matrix Sub-parts (3)

$$T = \begin{bmatrix} .5 & 0 & 0 & .5 \\ 0 & .5 & 0 & .5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T maps [-1, 1] vector space range into [0, 1] texture space range

Reflection Vector Tex Coords

- Texture matrix assumes **glTexCoor3f** used to pass in eye-space reflection vector
- Alternative is to use NV_reflection_vector OpenGL extension
  - Special texture coordinate generation mode
  - Supported by RIVA TNT OpenGL and Mesa 3.1
  - GL_REFLECTION_MAP_NV generates eye-space reflection vector
  - GL_NORMAL_MAP_NV generates eye-space normal vector

Extension Usage

- Reflection map usage

```
mapMode = GL_REFLECTION_MAP_NV;
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, mapMode);
glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, mapMode);
glTexGeni(GL_R, GL_TEXTURE_GEN_MODE, mapMode);
```

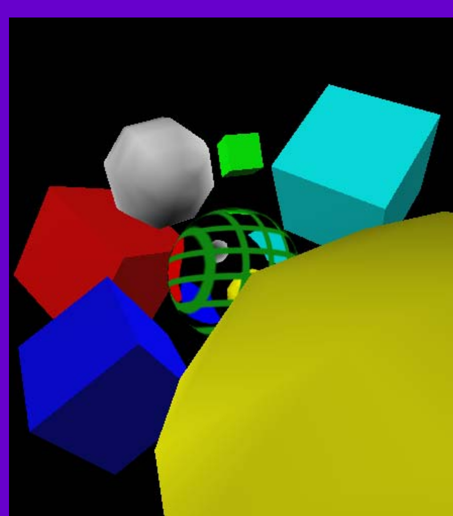
- Normal map usage

```
mapMode = GL_NORMAL_MAP_NV;
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, mapMode);
glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, mapMode);
glTexGeni(GL_R, GL_TEXTURE_GEN_MODE, mapMode);
```

Extension Enumerant Values

- #define GL_NORMAL_MAP_NV 0x8511
- #define GL_REFLECTION_MAP_NV 0x8512

Example Scene



Basis for Per-pixel Lighting

- Encode the specular directional light contributions in the scene as two dual paraboloid maps; use with reflection vector
- Encode the diffuse directional light contributions for the scene in another set of maps; use with normal vector
- Two multi-textured passes gives per-pixel specular and diffuse lighting

Dual-parabolic Mapping Math

How to map (rx,ry,rz) unit vector to
(s,t) texture coordinate in [-1,1] range.

front

s = rx / (1-rz)
t = ry / (1-rz)

back

s = - rx / (rz + 1)
t = - ry / (rz + 1)

Reverse mapping: (s,t) to (rx,ry,rz)

front

rx = 2 s / (s^2 + t^2 + 1)
ry = 2 t / (s^2 + t^2 + 1)
rz = (-1 + s^2 + t^2) / (s^2 + t^2 + 1)

back

rx = - 2 s / (s^2 + t^2 + 1)
ry = - 2 t / (s^2 + t^2 + 1)
rz = - (-1 + s^2 + t^2) / (s^2 + t^2 + 1)

References

- Heidrich & Seidel, “View-independent Environment Maps” *Eurographics Workshop on Graphics Hardware*, 1998.
- Background: Doug Voorhies & Jim Foran, “Reflection vector shading hardware,” *SIGGRAPH ‘94 Proceedings*, 1994.