Tex Gen, the Texture Matrix, and Projected Textures

D. Sim Dietrich Jr
NVIDIA Corporation
sim.dietrich@nvidia.com
Texture Coordinate Generation

- New To Dx7
- Many developers have never used ‘TexGen’
- Usage similar to OpenGL
- TexGen is a way for the hardware or API to automatically generate texture coordinates for you
- Can happen in HW on T&L Devices (such as GeForce 256)
TexGen, Texture Matrix, Projected Textures Diagram

- Camera Space Vertices & Normals
- Texture Coordinate Generation “TexGen”
- 4x4 Texture Matrix
- Texture Projection
There are three built-in TexGen modes in Dx7:

- D3DTSS_TCI_CAMERASPACEPOSITION
- D3DTSS_TCI_CAMERASPACENORMAL
- D3DTSS_TCI_CAMERASPACEREFLECTIONVECTOR
Terminology

- D3D doesn’t have a consistent terminology for the 3\textsuperscript{rd} and 4\textsuperscript{th} texture coordinate, so I will use the popular S,T,R,Q for texture coordinates 0 through 3
- S corresponds to U
- T corresponds to V
TexGen and the FVF

• If you use Texgen for a particular set of texture coordinates, you don’t need to fill them in yourself

• You do need to tell D3D how many coordinates there are for each texture stage
  • Use the D3DFVF_TEXCOORDSIZEEx macros
  • D3DFVF_TEXCOORDSIZE3( 1 ) means texture stage 1 has 3 texture coordinates
  • D3DFVF_TEXCOORDSIZE1( 2 ) means texture stage 2 has 1 texture coordinate
Camera Space Position
D3DTSS_TCI_CAMERASPACEPOSITION

- In this mode, D3D takes the X,Y,Z position in view (or camera) space and uses them as the S,T,R texture coordinates

  X -> S
  Y -> T
  Z -> R
Camera Space Normal
D3DTSS_TCI_CAMERASPACENORMAL

- In this mode, D3D takes the Vertex normal in view (or camera) space and its 3 components as the S,T,R texture coordinates:
  - N.X -> S
  - N.Y -> T
  - N.Z -> R
- This is the same vertex normal used for lighting calculations.
In this mode, D3D takes the Vertex normal in view (or camera) space and the local viewer position (0, 0, 0) and calculates the reflection vector R to use for the S,T,R texture coordinates:

- R.X -> S
- R.Y -> T
- R.Z -> R

This is the same reflection vector used for a specular light with a local viewer.
How TexGen Fits in the T&L Pipe

- Vector Position & Normal
- Model->World Matrix
- World->View Matrix
- TexGen
  - Tex Coords
  - Texture Matrix
- View->Projection Matrix
- Rasterization
What are they used for?

- Camera Space Position is useful for projected shadows, spotlights, aligning screen space textures, distance functions such as point light attenuation and texture fog.

- Camera Space Normal is useful for Toon Rendering, Diffuse lighting, Diffuse Environment mapping, and simulating refraction through a surface.
What are they used for? (cont)

- Camera Space Reflection Vector is used primarily for Reflective Environment maps and pre-computed Specular Lighting.
What is the Texture Matrix?

- D3D allows each Texture Stage (2 for most cards) to have its own texture matrix

- The texture matrix takes from 1 to 4 coordinates and multiplies them by a 4x4 matrix

- This is analogous to the 4x4 view or model transform

- These can be accelerated in HW on T&L Devices, such as GeForce 256
What is the Texture Matrix used for?

- A simple example would be to scroll or rotate a texture around a surface, by letting the matrix change the texture coordinates, rather than adjusting the coordinates on the CPU.
- A more advanced usage would be to translate to another part of a texture without having to touch the geometry:
  - The left half of a texture could contain a raised button.
  - The right half has a depressed button.
  - Update the texture matrix to translate from the original coordinates pointing to left half to the right half.
What is the Texture Matrix used for?

- Another usage is to rotate a cube map

- Cube maps are considered to be defined in view space
  - Apps often want them in world space to represent the reflected environment
  - Put the inverse of the world-\(\to\)view matrix in the texture matrix to align the cube map to the world
Using the Texture Matrix with TexGen

- TexGen always generates coordinates relative to view (camera) space
- You may want them generated from a different reference point or plane
- For projected shadows or lights, you want them generated from the light plane instead
Using the Texture Matrix to Make Custom TexGen Functions

- You can’t really change the TexGen function, but because the texture matrix comes after TexGen, you can rotate, translate and scale the generated coordinates to be relative to any coordinate system.

- Just rotate from the view plane→light plane to get projected light or shadow coordinates.
Setting the Texture Matrices

- Set just like the view or world matrix

- SetTransform( D3DTRANSFORMSTATE_TEXTURE0, &textureMatrix );
  - To set the texture matrix for stage 0
Enabling the Texture Matrices

To enable a 3x3 matrix for stage 1, call:

```c
SetTextureStageState( 1, D3DTSS_TEXTURETRANSFORMFLAGS,
                      D3DTTFF_COUNT3 );
```

For a 2x2 matrix for stage 0, call:

```c
SetTextureStageState( 0, D3DTSS_TEXTURETRANSFORMFLAGS,
                      D3DTTFF_COUNT2 );
```
Texture Projection

- Assuming we are using normal 2D textures, and not a Cube Map, why would we want more than 2 texture coordinates?
  
  To allow us to project textures, just as we project points from view space to the screen.

- If we start out with 3 texture coordinates, S, T & R, we only need 2 coordinates to look up into a 2D texture, we go from 3 coordinates to 2 via a projection, or a divide.
Texture Projection

- If texture projection is enabled, the last coordinate is divided through all other coordinates, yielding one less coordinate.

- For 3D -> 2D coordinates:

  \[ S' = \frac{S}{R} \]
  \[ T' = \frac{T}{R} \]
  \[ R' = \frac{R}{R} = 1 \]
If texture projection is enabled, the last coordinate is divided through all other coordinates

For 2D -> 1D coordinates:
- \( S' = \frac{S}{T} \)
- \( T' = \frac{T}{T} = 1 \)
Texture Projection

- This is analogous to the homogeneous divide by W when projecting from view space to screen space

- To enable texture projection, add the D3DTFF_PROJECTED flag to your

  - `SetTextureStageState( 0, D3DTSS_TEXTURETRANSFORMFLAGS, x )` call.
Enabling Texture Projection

- `SetTextureStageState( 0, D3DTSS_TEXTURETRANSFORMFLAGS | D3DTTFF_COUNT3 | D3DTFF_PROJECTED );`

- D3D Always uses the last coordinate as the divisor, i.e. with 3D texture coordinates, R is the divisor, with 2D texture coordinates, T is the divisor.

- If you want different behavior, use the texture matrix to swap the S, T, R or Q coordinates.
Texture Projection

- Any ‘extra’ coordinates are ignored
- In other words, if you have a 1D texture, and you are using a 3x3 matrix, only the S coordinate is used
- If you have projection enabled, then \( S' = S / T \) is calculated, and just \( S' \) is used to index the 1D texture
Questions?