Animated Grass with Pixel and Vertex Shaders

John Isidoro
ATI Research

Drew Card
ATI Research

Introduction

The availability of programmable vertex and pixel shaders has allowed programmers to re-examine how they implement passive background scene geometry. Before the advent of programmable shaders, deformation animation was only used for characters and a select few scene objects due to the CPU overhead associated with it. With the ability to transfer this processing over to the GPU, previously static background objects can now be animated, adding life and realism to the scene. In this gem, an approach is shown for realistically and inexpensively animating grass with a combination of vertex and pixel shaders.

Waving the Grass

The waving motion of the grass can be accomplished in a vertex shader. Using a traditional method the grass is rendered with randomly placed intersecting quads. The quads are texture mapped and rendered with an alpha test. In a vertex shader the top two vertices of each quad are animated using a combination of four sinusoidal waves. The waves are approximated using a Taylor Series approach laid out in Chapter [Isidoro]. This combination of sine waves using various different frequencies creates a natural waving that does not look like an animation or overly repetitious.

Lighting the Grass

When grass blades wave in the wind they also turn and change their orientation with respect to the sunlight. Because this method involves using a textured quad to represent various blades of grass it is impossible to change the lighting on individual grass blades. To simulate the change in color that would occur as the blades twisted in the wind the same sine wave that was used to animate the grass is used to oscillate between two colors. In this case, the green channel of the color was changed to make the grass color change from intense green to a more yellowish-brown.

Excerpted from ShaderX: Vertex and Pixel Shader Tips and Tricks
Animated Grass with Pixel and Vertex Shaders

Figure 1 - Grass Texture

Figure 2 - Results

Excerpted from ShaderX: Vertex and Pixel Shader Tips and Tricks
Animated Grass with Pixel and Vertex Shaders

**Vertex Shader Code**

```c
//vertex shader for grass.. 
//sinusoidal vertex motion for waving grass 
pos + sumOverI(wavedirI * texcoordy * sin( xdirI * (xpos+time)) + ydirI * (ypos+time)))
SetVertexShaderConstant 0  commonConst
SetVertexShaderConstant 1  appConst
SetVertexShaderConstant 2  worldSpaceCamPos
SetVertexShaderConstant 4  wvp
SetVertexShaderConstant 8  sin9
SetVertexShaderConstant 10  frcFixup
SetVertexShaderConstant 11  waveDistortx
SetVertexShaderConstant 12  waveDistorty
SetVertexShaderConstant 13  waveDistortz
SetVertexShaderConstant 14  waveDirz
SetVertexShaderConstant 15  waveDirx
SetVertexShaderConstant 16  waveSpeed
SetVertexShaderConstant 17  piVector
SetVertexShaderConstant 18  lightingWaveScale
SetVertexShaderConstant 19  lightingScaleBias

vs.1.1
mul r0, c14, v0.x     // use vertex pos x as inputs to sinusoidal warp
mad r0, c15, v0.y, r0 // use vertex pos y as inputs to sinusoidal warp
mov r1, c1.x          // get current time
mad r0, r1, c16, r0   // add scaled time to move bumps according to speed
frc r0.xy, r0         // take frac of all 4 components
frc r1.xy, r0.zwzw    //
mov r0.zw, r1.xyxy    //
mul r0, r0, c10.x     // multiply by fixup factor (due to inaccuracy of taylor series)
sub r0, r0, c0.y     // subtract 0.5
mul r1, r0, c17.w    // **2pi coords range from(-pi to pi)
mul r2, r1, r1       // (wave vec)^2
mul r3, r2, r1       // (wave vec)^3
mul r5, r3, r2       // (wave vec)^5
mul r7, r5, r2       // (wave vec)^7
mul r9, r7, r2       // (wave vec)^9
mad r0, r3, c8.x, r1 // (wave vec) - {(wave vec)^3}/3!
mad r0, r5, c8.y, r0 // + {(wave vec)^5}/5!
mad r0, r7, c8.z, r0 // - {(wave vec)^7}/7!
mad r0, r9, c8.w, r0 // - {(wave vec)^9}/9!
dp4 r3.x, r0, c11
dp4 r3.y, r0, c12
dp4 r3.zw, r0, c13
mul r4, v7.y, v7.y   //
mul r3, r3, v7.y     // attenuate sinusoidal warping by tex0.y^2
mov r2.w, v0         // add sinusoidal warping to grass position
add r2.xyz, r3, v0
m4x4 oPos, r2, c4    //scale and add sin waves together
dp4 r1.x, r0, c18    //scale and bias color values (green is scaled more
mad oD0, c19.xzxx, -r1.x, c19.y // than red and blue)
mov oT0, v7
```

**Pixel Shader Code**

```c
ps.1.4
tex1d r0, t0
mul_x2 r0, r0, v0
```

Excerpted from ShaderX: Vertex and Pixel Shader Tips and Tricks