Kerbal Space Program - Bug #27401

Lines between terrain texture zoom levels (Triplanar and Atlas shader)

03/16/2021 11:21 PM - Gameslinx

Status:	New	**Start date:**	03/16/2021
Severity:	Low	**% Done:**	0%
Assignee:			
Category:	Visual FX		
Target version:			
Version:	1.11.1	**Language:**	English (US)
Platform:	Windows	**Mod Related:**	No
Expansion:	Core Game		

Description

Between the two different texture fade levels, there is a noticeable line (or seam) between the two levels. This line is evident in this video:
https://www.youtube.com/watch?v=lnOR17T4_Co

This is a problem within the shader code in a line that uses the tex2D() function to sample the terrain textures. tex2D takes 4 values: sampler_2D Texture, float2 Texture_UV_Coordinates, float2 X_Derivative, float2 Y_Derivative.
The line in the video above is produced when only the Texture and Texture UV Coordinates are specified.

When blending two textures together the X and Y derivatives must be specified to avoid this from happening.

The shader affected by this is the Triplanar shader, so this code can be referenced to help fix this issue. It's adapted from my Parallax mod which uses biplanar mapping instead of triplanar mapping, but the implementation is the same:

```
//UV Coordinates for triplanar mapping
            float2 uvX = IN.worldPos.yz;
            float2 uvY = IN.worldPos.xz;
            float2 uvZ = IN.worldPos.xy;

//Derivatives for worldPosition
            float3 dpdx = ddx(IN.worldPos);
            float3 dpdy = ddy(IN.worldPos);

//UV coordinates for adjacent X and Y positions
            float2 derivativeUVX_X = dpdx.yz;   //Derivatives for uvX
            float2 derivativeUVX_Y = dpdy.yz;
            float2 derivativeUVY_X = dpdx.xz;   //Derivatives for uvY
            float2 derivativeUVY_Y = dpdy.xz;
            float2 derivativeUVZ_X = dpdx.xy;   //Derivatives for uvZ
            float2 derivativeUVZ_Y = dpdy.xy;

//Sampling textures for zoom level 1
            fixed4 albedoZoom1_X = tex2D(_MainTex, uvX * _MainTex_ST, derivativeUVX_X * _MainTex_S
T, derivativeUVX_Y * _MainTex_ST);
            fixed4 albedoZoom1_Y = tex2D(_MainTex, uvY * _MainTex_ST, derivativeUVY_X * _MainTex_S
T, derivativeUVY_Y * _MainTex_ST);
            fixed4 albedoZoom1_Z = tex2D(_MainTex, uvZ * _MainTex_ST, derivativeUVZ_X * _MainTex_S
T, derivativeUVZ_Y * _MainTex_ST);

//Sampling textures for zoom level 2 - Make sure to scale the derivatives by the same factor as th
e UV coordinates. 14 is the scaling factor here
            fixed4 albedoZoom2_X = tex2D(_MainTex, uvX * _MainTex_ST / 14, (derivativeUVX_X * _Mai
nTex_ST) / 14, (derivativeUVX_Y * _MainTex_ST) / 14);
            fixed4 albedoZoom2_Y = tex2D(_MainTex, uvY * _MainTex_ST / 14, (derivativeUVY_X * _Mai
nTex_ST) / 14, (derivativeUVY_Y * _MainTex_ST) / 14);
            fixed4 albedoZoom2_Z = tex2D(_MainTex, uvZ * _MainTex_ST / 14, (derivativeUVZ_X * _Mai
nTex_ST) / 14, (derivativeUVZ_Y * _MainTex_ST) / 14);
```

```
//Now just blend the textures together as usual and the line will have disappeared
            // - Linx
```

This should hopefully be useful for solving this issue.