**水面效果的shader文件**

Posted on 2013年06月25日 by U3d / [Unity3D脚本/插件](http://www.unitymanual.com/category/script)/被围观 71 次

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 001 | Shader "FX/Water" { |
| 002 | Properties { |
| 003 | \_WaveScale ("Wave scale", Range (0.02,0.15)) = 0.063 |
| 004 | \_ReflDistort ("Reflection distort", Range (0,1.5)) = 0.44 |
| 005 | \_RefrDistort ("Refraction distort", Range (0,1.5)) = 0.40 |
| 006 | \_RefrColor ("Refraction color", COLOR) = ( .34, .85, .92, 1) |
| 007 | \_Fresnel ("Fresnel (A) ", 2D) = "gray" {} |
| 008 | \_BumpMap ("Bumpmap (RGB) ", 2D) = "bump" {} |
| 009 | WaveSpeed ("Wave speed (map1 x,y; map2 x,y)", Vector) = (19,9,-16,-7) |
| 010 | \_ReflectiveColor ("Reflective color (RGB) fresnel (A) ", 2D) = "" {} |
| 011 | \_ReflectiveColorCube ("Reflective color cube (RGB) fresnel (A)", Cube) = "" { TexGen CubeReflect } |
| 012 | \_HorizonColor ("Simple water horizon color", COLOR) = ( .172, .463, .435, 1) |
| 013 | \_MainTex ("Fallback texture", 2D) = "" {} |
| 014 | \_ReflectionTex ("Internal Reflection", 2D) = "" {} |
| 015 | \_RefractionTex ("Internal Refraction", 2D) = "" {} |
| 016 | } |
| 017 | *// -----------------------------------------------------------* |
| 018 | *// Fragment program cards* |
| 019 | Subshader { |
| 020 | Tags { "WaterMode"="Refractive" "RenderType"="Opaque" } |
| 021 | Pass { |
| 022 | CGPROGRAM |
| 023 | #pragma vertex vert |
| 024 | #pragma fragment frag |
| 025 | #pragma fragmentoption ARB\_precision\_hint\_fastest |
| 026 | #pragma fragmentoption ARB\_fog\_exp2 |
| 027 | #pragma multi\_compile WATER\_REFRACTIVE WATER\_REFLECTIVE WATER\_SIMPLE |
| 028 | #if defined WATER\_REFLECTIVE || defined WATER\_REFRACTIVE |
| 029 | #define HAS\_REFLECTION 1 |
| 030 | #endif |
| 031 | #if defined WATER\_REFRACTIVE |
| 032 | #define HAS\_REFRACTION 1 |
| 033 | #endif |
| 034 | #include "UnityCG.cginc" |
| 035 | uniform float4 \_WaveScale4; |
| 036 | uniform float4 \_WaveOffset; |
| 037 | #ifdef HAS\_REFLECTION |
| 038 | uniform **float** \_ReflDistort; |
| 039 | #endif |
| 040 | #ifdef HAS\_REFRACTION |
| 041 | uniform **float** \_RefrDistort; |
| 042 | #endif |
| 043 | **struct** appdata { |
| 044 | float4 vertex : POSITION; |
| 045 | float3 normal : NORMAL; |
| 046 | }; |
| 047 | **struct** v2f { |
| 048 | V2F\_POS\_FOG; |
| 049 | #if defined HAS\_REFLECTION || defined HAS\_REFRACTION |
| 050 | float3 **ref**; |
| 051 | #endif |
| 052 | float2 bumpuv[2]; |
| 053 | float3 viewDir; |
| 054 | }; |
| 055 | v2f vert(appdata v) |
| 056 | { |
| 057 | v2f o; |
| 058 | PositionFog( v.vertex, o.pos, o.fog ); |
| 059 | *// scroll bump waves* |
| 060 | float4 temp; |
| 061 | temp.xyzw = v.vertex.xzxz \* \_WaveScale4 + \_WaveOffset; |
| 062 | o.bumpuv[0] = temp.xy; |
| 063 | o.bumpuv[1] = temp.wz; |
| 064 | *// object space view direction (will normalize per pixel)* |
| 065 | o.viewDir.xzy = ObjSpaceViewDir(v.vertex); |
| 066 | #if defined HAS\_REFLECTION || defined HAS\_REFRACTION |
| 067 | *// calculate the reflection vector* |
| 068 | float3x4 mat = float3x4 ( |
| 069 | 0.5, 0, 0, 0.5, |
| 070 | 0, 0.5 \* \_ProjectionParams.x, 0, 0.5, |
| 071 | 0, 0, 0, 1 |
| 072 | ); |
| 073 | o.**ref** = mul (mat, o.pos); |
| 074 | #endif |
| 075 | **return** o; |
| 076 | } |
| 077 | #if defined WATER\_REFLECTIVE || defined WATER\_REFRACTIVE |
| 078 | sampler2D \_ReflectionTex; |
| 079 | #endif |
| 080 | #if defined WATER\_REFLECTIVE || defined WATER\_SIMPLE |
| 081 | sampler2D \_ReflectiveColor; |
| 082 | #endif |
| 083 | #if defined WATER\_REFRACTIVE |
| 084 | sampler2D \_Fresnel; |
| 085 | sampler2D \_RefractionTex; |
| 086 | uniform float4 \_RefrColor; |
| 087 | #endif |
| 088 | #if defined WATER\_SIMPLE |
| 089 | uniform float4 \_HorizonColor; |
| 090 | #endif |
| 091 | sampler2D \_BumpMap; |
| 092 | half4 frag( v2f i ) : COLOR |
| 093 | { |
| 094 | i.viewDir = normalize(i.viewDir); |
| 095 | *// combine two scrolling bumpmaps into one* |
| 096 | half3 bump1 = tex2D( \_BumpMap, i.bumpuv[0] ).rgb; |
| 097 | half3 bump2 = tex2D( \_BumpMap, i.bumpuv[1] ).rgb; |
| 098 | half3 bump = bump1 + bump2 - 1; |
| 099 | *// fresnel factor* |
| 100 | half fresnelFac = dot( i.viewDir, bump ); |
| 101 | *// perturb reflection/refraction UVs by bumpmap, and lookup colors* |
| 102 | #ifdef HAS\_REFLECTION |
| 103 | float3 uv1 = i.**ref**; uv1.xy += bump \* \_ReflDistort; |
| 104 | half4 refl = tex2Dproj( \_ReflectionTex, uv1 ); |
| 105 | #endif |
| 106 | #ifdef HAS\_REFRACTION |
| 107 | float3 uv2 = i.**ref**; uv2.xy -= bump \* \_RefrDistort; |
| 108 | half4 refr = tex2Dproj( \_RefractionTex, uv2 ) \* \_RefrColor; |
| 109 | #endif |
| 110 | *// final color is between refracted and reflected based on fresnel* |
| 111 | half4 color; |
| 112 | #ifdef WATER\_REFRACTIVE |
| 113 | half fresnel = tex2D( \_Fresnel, float2(fresnelFac,fresnelFac) ).a; |
| 114 | color = lerp( refr, refl, fresnel ); |
| 115 | #endif |
| 116 | #ifdef WATER\_REFLECTIVE |
| 117 | half4 water = tex2D( \_ReflectiveColor, float2(fresnelFac,fresnelFac) ); |
| 118 | color.rgb = lerp( water.rgb, refl.rgb, water.a ); |
| 119 | color.a = refl.a \* water.a; |
| 120 | #endif |
| 121 | #ifdef WATER\_SIMPLE |
| 122 | half4 water = tex2D( \_ReflectiveColor, float2(fresnelFac,fresnelFac) ); |
| 123 | color.rgb = lerp( water.rgb, \_HorizonColor.rgb, water.a ); |
| 124 | color.a = \_HorizonColor.a; |
| 125 | #endif |
| 126 | **return** color; |
| 127 | } |
| 128 | ENDCG |
| 129 | } |
| 130 | } |
| 131 | *// -----------------------------------------------------------* |
| 132 | *// Radeon 9000 cards* |
| 133 | Subshader { |
| 134 | Tags { "WaterMode"="Reflective" "RenderType"="Opaque" } |
| 135 | Pass { |
| 136 | CGPROGRAM |
| 137 | #pragma vertex vert |
| 138 | #include "UnityCG.cginc" |
| 139 | uniform float4 \_WaveScale4; |
| 140 | uniform float4 \_WaveOffset; |
| 141 | uniform **float** \_ReflDistort; |
| 142 | **struct** appdata { |
| 143 | float4 vertex : POSITION; |
| 144 | float3 normal : NORMAL; |
| 145 | }; |
| 146 | **struct** v2f { |
| 147 | V2F\_POS\_FOG; |
| 148 | float2 bumpuv[2] : TEXCOORD0; |
| 149 | float3 viewDir : TEXCOORD2; |
| 150 | float4 **ref** : TEXCOORD3; |
| 151 | }; |
| 152 | v2f vert(appdata v) |
| 153 | { |
| 154 | v2f o; |
| 155 | PositionFog( v.vertex, o.pos, o.fog ); |
| 156 | *// scroll bump waves* |
| 157 | float4 temp; |
| 158 | temp.xyzw = v.vertex.xzxz \* \_WaveScale4 + \_WaveOffset; |
| 159 | o.bumpuv[0] = temp.xy; |
| 160 | o.bumpuv[1] = temp.wz; |
| 161 | *// object space view direction* |
| 162 | o.viewDir.xzy = normalize( ObjSpaceViewDir(v.vertex) ); |
| 163 | *// calculate the reflection vector* |
| 164 | float4x4 mat = float4x4 ( |
| 165 | .5, 0, 0,.5, |
| 166 | 0,.5 \* \_ProjectionParams.x, 0,.5, |
| 167 | 0, 0,.5,.5, |
| 168 | 0, 0, 0, 1 |
| 169 | ); |
| 170 | o.**ref** = mul (mat, o.pos); |
| 171 | **return** o; |
| 172 | } *//Unity3D教程手册：www.unitymanual.com* |
| 173 | ENDCG |
| 174 | Program "" { |
| 175 | SubProgram { |
| 176 | Keywords { "WATER\_REFLECTIVE" "WATER\_REFRACTIVE" } |
| 177 | SetTexture [\_BumpMap] { 2D } |
| 178 | SetTexture [\_BumpMap] { 2D } |
| 179 | SetTexture [\_ReflectiveColor] { 2D } |
| 180 | SetTexture [\_ReflectionTex] { 2D } |
| 181 | Local 0, ([\_ReflDistort],0,0,0) |
| 182 | "!!ATIfs1.0 |
| 183 | StartConstants; |
| 184 | CONSTANT c0 = program.local[0]; |
| 185 | EndConstants; |
| 186 | StartPrelimPass; |
| 187 | PassTexCoord r3, t3.stq\_dq; # reflection vector |
| 188 | SampleMap r0, t0.str; # bump1 |
| 189 | SampleMap r1, t1.str; # bump2 |
| 190 | PassTexCoord r2, t2.str; |
| 191 | ADD r1.half, r0.bias, r1.bias; # bump = bump1 + bump2 - 1 |
| 192 | DOT3 r2, r1.2x, r2; # fresnel: dot (bump, viewer-pos) |
| 193 | # add less offset because it's purely screenspace; big ones look bad |
| 194 | MAD r3.rg, r1, c0.r, r3; # uv += bump \* strength; add less because it's not perspective |
| 195 | EndPass; |
| 196 | StartOutputPass; |
| 197 | SampleMap r3, r3.str; # reflection color |
| 198 | SampleMap r2, r2.str; # water color/fresnel |
| 199 | LERP r0.rgb, r2.a, r3, r2; # between water and reflected based on fresnel |
| 200 | MUL r0.a, r3.a, r2.a; |
| 201 | EndPass; |
| 202 | " |
| 203 | } |
| 204 | SubProgram { |
| 205 | Keywords { "WATER\_SIMPLE" } |
| 206 | SetTexture [\_BumpMap] { 2D } |
| 207 | SetTexture [\_BumpMap] { 2D } |
| 208 | SetTexture [\_ReflectiveColor] { 2D } |
| 209 | Local 0, [\_HorizonColor] |
| 210 | "!!ATIfs1.0 |
| 211 | StartConstants; |
| 212 | CONSTANT c0 = program.local[0]; |
| 213 | EndConstants; |
| 214 | StartPrelimPass; |
| 215 | SampleMap r0, t0.str; |
| 216 | SampleMap r1, t1.str; |
| 217 | PassTexCoord r2, t2.str; |
| 218 | ADD r1, r0.bias, r1.bias; # bump = bump1 + bump2 - 1 |
| 219 | DOT3 r2, r1, r2; # fresnel: dot (bump, viewer-pos) |
| 220 | EndPass; |
| 221 | StartOutputPass; |
| 222 | SampleMap r2, r2.str; |
| 223 | LERP r0.rgb, r2.a, c0, r2; # fade in reflection |
| 224 | MOV r0.a, c0.a; |
| 225 | EndPass; |
| 226 | " |
| 227 | } |
| 228 | } |
| 229 | } |
| 230 | } |
| 231 | *// -----------------------------------------------------------* |
| 232 | *// Old cards* |
| 233 | *// three texture, cubemaps* |
| 234 | Subshader { |
| 235 | Tags { "WaterMode"="Simple" "RenderType"="Opaque" } |
| 236 | Pass { |
| 237 | Color (0.5,0.5,0.5,0.5) |
| 238 | SetTexture [\_MainTex] { |
| 239 | Matrix [\_WaveMatrix] |
| 240 | combine texture \* primary |
| 241 | } |
| 242 | SetTexture [\_MainTex] { |
| 243 | Matrix [\_WaveMatrix2] |
| 244 | combine texture \* primary + previous |
| 245 | } |
| 246 | SetTexture [\_ReflectiveColorCube] { |
| 247 | combine texture +- previous, primary |
| 248 | Matrix [\_Reflection] |
| 249 | } |
| 250 | } *//Unity3D教程手册：www.unitymanual.com* |
| 251 | } |
| 252 | *// dual texture, cubemaps* |
| 253 | Subshader { |
| 254 | Tags { "WaterMode"="Simple" "RenderType"="Opaque" } |
| 255 | Pass { |
| 256 | Color (0.5,0.5,0.5,0.5) |
| 257 | SetTexture [\_MainTex] { |
| 258 | Matrix [\_WaveMatrix] |
| 259 | combine texture |
| 260 | } |
| 261 | SetTexture [\_ReflectiveColorCube] { |
| 262 | combine texture +- previous, primary |
| 263 | Matrix [\_Reflection] |
| 264 | } |
| 265 | } |
| 266 | } |
| 267 | *// single texture* |
| 268 | Subshader { |
| 269 | Tags { "WaterMode"="Simple" "RenderType"="Opaque" } |
| 270 | Pass { |
| 271 | Color (0.5,0.5,0.5,0) |
| 272 | SetTexture [\_MainTex] { |
| 273 | Matrix [\_WaveMatrix] |
| 274 | combine texture, primary |
| 275 | } |
| 276 | } |
| 277 | } |
| 278 | } |
| 279 |  |
| 280 |  |