**Appendix B - Shader Cheat Sheet**

Lambert and BlinnPhong

1. struct SurfaceOutput
2. {
3. fixed3 Albedo; // diffuse color
4. fixed3 Normal; // tangent space normal, if written
5. fixed3 Emission;
6. half Specular; // specular power in 0..1 range
7. fixed Gloss; // specular intensity
8. fixed Alpha; // alpha for transparencies
9. };

Standard

1. struct SurfaceOutputStandard
2. {
3. fixed3 Albedo; // base (diffuse or specular) color
4. fixed3 Normal; // tangent space normal, if written
5. half3 Emission;
6. half Metallic; // 0=non-metal, 1=metal
7. half Smoothness; // 0=rough, 1=smooth
8. half Occlusion; // occlusion (default 1)
9. fixed Alpha; // alpha for transparencies
10. };

Standard Specular

1. struct SurfaceOutputStandardSpecular
2. {
3. fixed3 Albedo; // diffuse color
4. fixed3 Specular; // specular color
5. fixed3 Normal; // tangent space normal, if written
6. half3 Emission;
7. half Smoothness; // 0=rough, 1=smooth
8. half Occlusion; // occlusion (default 1)
9. fixed Alpha; // alpha for transparencies
10. };

Vertex/Fragment Structures

**AppData**

1. struct appdata\_full {
2. float4 vertex : POSITION; //vertex xyz position
3. float4 tangent : TANGENT;
4. float3 normal : NORMAL;
5. float4 texcoord : TEXCOORD0; //uv coordinate for first set of UVs
6. float4 texcoord1 : TEXCOORD1; //uv coordinate for second set of UVs
7. float4 texcoord2 : TEXCOORD2; //uv coordinate for third set of UVs
8. float4 texcoord3 : TEXCOORD3; //uv coordinate for fourth set of UVs
9. fixed4 color : COLOR; //per-vertex colour
10. };
11. struct v2f
12. {
13. float4 pos : SV\_POSITION; //The position of the vertex in clipping space.
14. float3 normal : NORMAL; //The normal of the vertex in clipping space.
15. float4 uv : TEXCOORD0; //UV from first UV set.
16. float4 textcoord1 : TEXCOORD1; //UV from second UV set.
17. float4 tangent : TANGENT; //A vector that runs at right angles to a normal.
18. float4 diff : COLOR0; //Diffuse vertex colour.
19. float4 spec : COLOR1; //Specular vertex colour.
20. }

Multipass Shader Format

1. Shader "MultipassShader"
2. {
3. Properties //PROPERTIES BLOCK
4. {
5. \_Color ("Main Color", Color) = (1,1,1,1)
6. \_MainTex ("Base (RGB)", 2D) = "white" {}
7. }
9. SubShader //ENCLOSING SHADER BLOCK
10. {
11. //FIRST PASS - SURFACE SHADER DOES NOT REQUIRE PASS BLOCK
12. Tags { "Queue" = "Geometry+1" }
14. CGPROGRAM
15. #pragma surface surf BlinnPhong
17. float4 \_Color;
18. struct Input
19. {
20. };
22. void surf (Input IN, inout SurfaceOutput o)
23. {
24. }
25. ENDCG
27. //SECOND PASS - ANOTHER SURFACE SHADER, NO PASS BLOCK REQUIRED
28. ZWrite Off
29. Blend DstColor Zero
30. CGPROGRAM
31. #pragma surface surf BlinnPhong
32. float4 \_Color;
33. struct Input
34. {
35. };
37. void surf (Input IN, inout SurfaceOutput o)
38. {
39. }
40. ENDCG
42. //THIRD PASS - VERT/FRAG NEEDS TO BE ENCLOSED IN PASS
43. Pass
44. {
45. Tags { "LightMode" = "Always" }
46. ZWrite Off
47. CGPROGRAM
48. #pragma vertex vert
49. #pragma fragment frag
50. #include "UnityCG.cginc"
51. sampler2D \_MainTex;
53. struct v2f
54. {
55. };
56. v2f vert (appdata\_full v)
57. {
58. }
60. half4 frag( v2f i ) : COLOR
61. {
62. }
63. ENDCG
64. }
66. //FOURTH PASS - SIMPLE SHADER LAB FUNCTIONS
67. Pass
68. {
69. Tags { "LightMode" = "Always" }
70. ZWrite Off
71. SetTexture [\_MainTex]
72. {
73. combine constant\* texture
74. }
75. }
76. }
77. Fallback "Diffuse
79. }