USER GUIDE

Developed by Jan Bögemann
Published by TheGameCreators Ltd

Introduction

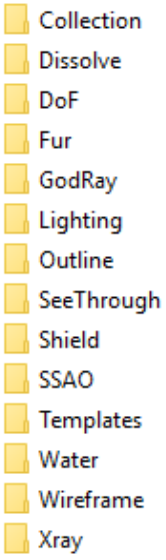
Welcome to the *AppGameKit Shader Pack* User Guide. The *AppGameKit Shader Pack* is a collection of shader effects contained in an easy to use library of source code. You can incorporate the effects into your games using the Tier1 AppGameKit script language. With just a few commands you can call up cool graphical shader effects in your projects.

Installing AppGameKit Shader Pack

If you purchased the AppGameKit Shader Pack as a download direct from TheGameCreators then you'll need to:

1. Download the file **AppGameKit_ShaderPack1_0.zip** from your TheGameCreators product page: https://www.thegamecreators.com/account/product/list

2. Once the zip file is downloaded, unzip the files inside to a location of your choice. The files you will see are:



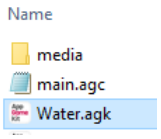
If you purchased the AppGameKit Shader Pack from Steam:

1. First you will need to activate the software key if it's not already added to your Steam library.

2. The default install location of the ShaderPack is:
   `C:\Program Files\Steam\steamapps\common\App Game Kit 2\DLC\ShaderPack`

The Files List

In all but one of the folders is a self-contained AppGameKit project demo that shows off an effect from the Shader Kit. Let's take the Water folder as an example, here are the files inside the water folder:



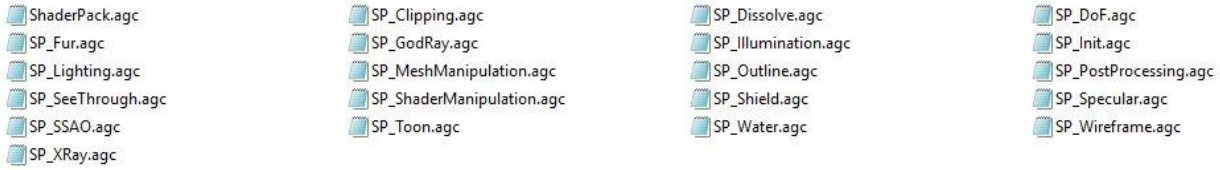
The media folder stores the images and shaders that are used to create the 3D scene and the water shader effects. *Water.agk* is the AppGameKit project file and *main.agc* is the AppGameKit main source code file for this project.

The Templates Folder

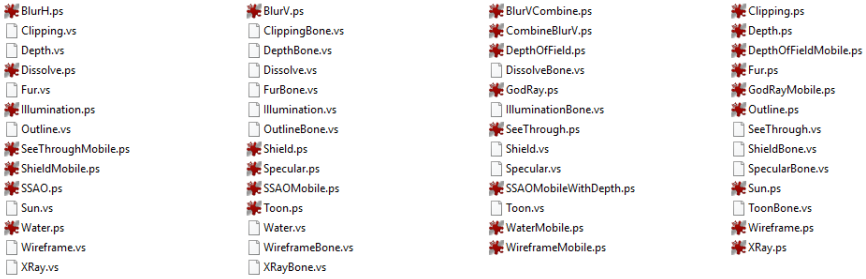
This is an important folder and one that you need to use whenever you want to make use of the Shader Pack. Deeper down the file structure can be found these two important folders:

- Templates/ShaderPack/Includes
- Templates/ShaderPack/Shader

Templates/ShaderPack/Includes: This folder stores all the source code files that make up the whole ShaderKit:

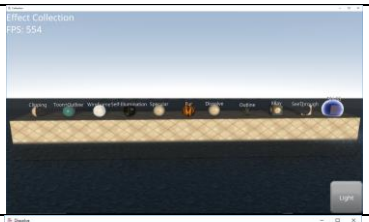
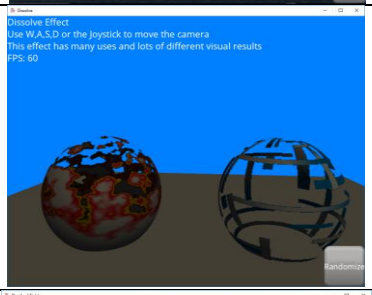
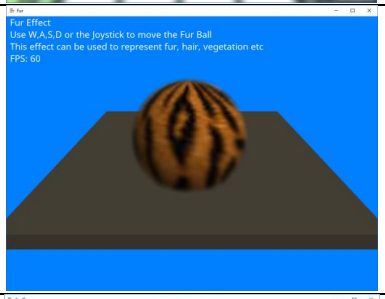


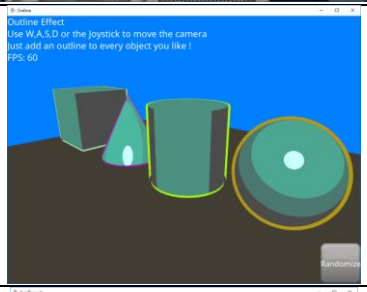
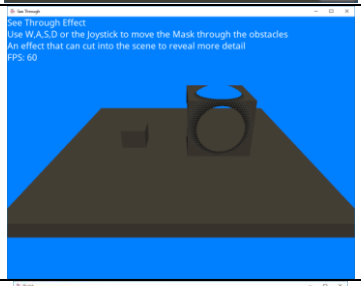
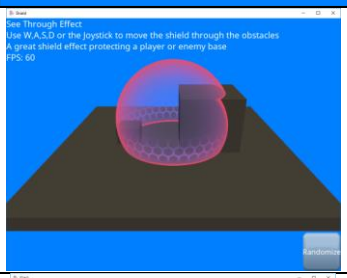
Templates/ShaderPack/Shader: This folder stores all the shader files:

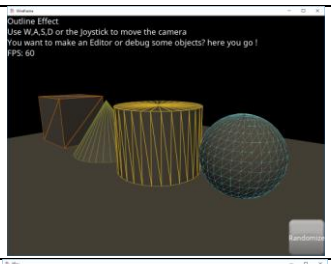
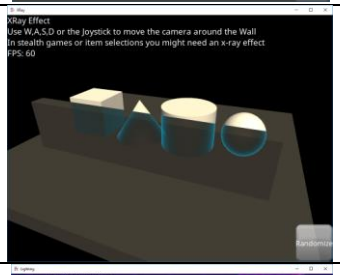
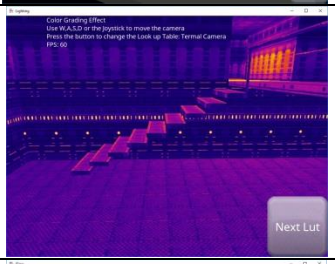
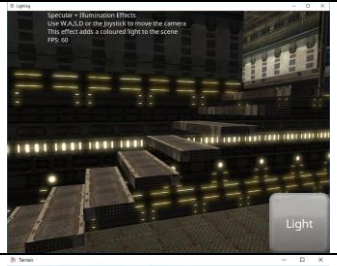
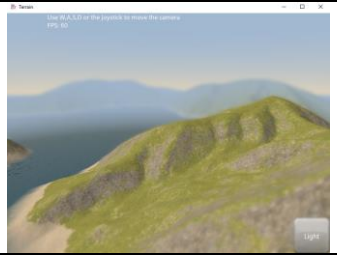


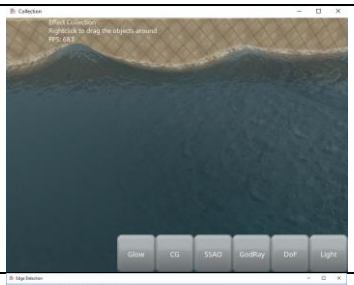
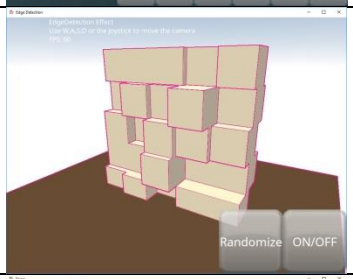
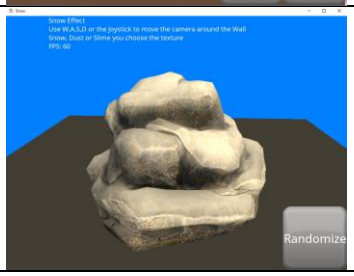
The Demo Projects

Every shader has its own demo project that you can load up and run. There's also a collection demo that shows all the effects running at the same time. While most desktop devices will run these effects super-fast there will be some mobile devices that might struggle to run some of them.

Collection		This demo shows all the effects from the pack running in a 3D scene. You can move around with the virtual joystick and touch the screen to change the camera angle.
Dissolve		In this demo a dissolve effect is applied to two sphere objects. You can keep pressing the Randomise button to change the effects.
DoF		DoF stands for Depth of Field. As you move the camera view around, notice how anything around the cross hair is kept in focus while everything else has a blur effect added to it. This effect is used in many games during aiming a weapon or bringing the attention of the player to a point of interest.
Fur		The sphere in this demo takes on a strange furry look. It's an effective visual that could be used on animals in a game to give them a more realistic look.
GodRay		GodRay effects are light rays that break out around objects that are between the camera and the sun. The effect can make a scene look much more impressive.

Lighting		In this demo you can randomly add light colour to the scene by pressing the button. Also notice that the wall is glowing with a pulsating effect from the Glow and Illumination shader.
Outline		You might want to highlight a 3D object in one of your projects. If you use this shader you can wrap the object with a coloured outline. This demo also shows the Toon shader being used to make objects look like they are drawn in a cartoon style.
SeeThrough		Using this shader you can cut into and hide parts of a 3D scene. Move the sphere around and see how it effects the cubes in the scene.
Shield		Many games will have shield effects to show the player or other game elements are shielded. Press the button to randomise how the shield will look.
SSAO		SSAO stands for Screen Space Ambient Occlusion and this demo shows how it can improve the shadows of the 3D model.
Water		Here the water demo shows how you can make a scene look realistic and detailed with this effect.

Wireframe		Here we show how objects can be viewed with their wireframes showing. All 3D objects are made up from polygon shapes and this effect exposes those polygons. It's a useful effect that can be used to highlight an object to a player.
Xray		This X-Ray effect could be used in some sci-fi style game when the player wants to scan the scene with some kind of x-ray device.
ColorGrading		The Color-Grading Effect is often used for final color corrections at the end of a project, but it can also be used for game mechanics like the thermal camera effect or a death screen.
Glass		Here we demonstrate the reflection and refraction of glass.
Glow		The Glow Effect is demonstrated inside the Lighting Demo and cooperates with the Self-Illumination Effect
Terrain		This effect uses a splatmap to draw grass, dirt or roads on a bumpy terrain

Ocean		It’s like the water shader but moves vertices like real waves and adds a foam effect to it. You can alter everything from the foam texture to the wave frequency.
Edge Detection		This effect highlights the edges of every object in the scene in strength and color of your choice.
Snow		You can make a custom layer of Snow, Slime or whatever you like on top of your object, just by using this Effect and a second texture.

How to use the Shader Pack

You will need to have had some moderate experience coding with AppGameKit before you can really start to understand how to use this pack of effects. It's essential you can code and know how to use functions and commands like #include. A basic understanding of the 3D commands will prove highly useful. If you are not so experienced in these areas then please read up about them in the online help. The new Official Tutorial Guide Volume 2 has some great chapters covering the 3D commands.

Volume 2 includes detailed 3D chapters

We also advice that you to explore the demo projects by experimenting with the commands and seeing the results. You will soon pick up the logic on how they work.

The Shader Pack source code is in Tier1 AppGameKit Script and you will need to ***#include*** the .agc code files into your projects. The first thing you need to do is copy the files it uses into the same project folder of your own project(s).

Let's assume you store your projects in this location:

```
C:\Users\John\Documents\AGK2 Projects\
```

1. Copy the **Templates** folder from the Shader Kit file list into the above folder location.
2. Any old or new projects you start within this folder can now reference the shaders and the shader kit source code by including this code:

   ```
   #include ".\..\Templates\ShaderPack\Includes\ShaderPack.agc"
   ```

3. Your projects need three calls to make use of the shaders in the kit:
   a. At the beginning you initialize the library with **SP_Init()**
   b. The desired effect command for example: **SP_Specular_AddObject()**
   c. In the game loop you call **SP_Sync()** instead of the built in sync() command
4. If you release your Application to the public please make sure to **Encrypt** all shader files.

Calling the Shader Effects

Each effect is called using this function format: **SP_Effect_Command()**

- SP stands for Shader Pack
- Effect = The Shader Effect you want to call
- Action = What you want the Effect to do

Differentiate each section with the "_" character, for example: **SP_Normal_AddObject()**

Effects

Everything is handled by source code libraries. Shaders are complex and coded by advanced programmers who have studied this area of games development. Using the functions in the Shader Pack makes it much easier for most developers. If you want to get more involved with the shaders then read on where you'll find more detail on each function and the parameters you can use with them. The Shader code is encoded by default, but you can use the raw Shader code by calling the command **SP_SetReadRawFiles()** at the beginning or encode custom Shader files with the Encryption program.

Community Help

Our online forums are a great place to ask questions and share your experience with AppGameKit Shader Pack. Please join in or read what other developers are saying in the AppGameKit Shader Pack Forum Thread.

Shader Details

Clipping

The Clipping shader is mostly needed for the water shader to clip away the top part of an object if the refraction is rendered and to clip away the bottom part when the reflection part is rendered. Clipping code is generated and inserted for all other effects to work with the water shader.

If you want to clip things manually, the shader takes:

- Diffuse texture - in stage 0.
- Vec4 clippingPlane - coordinates where .xyz is the direction and .w is the distance of the clipping plane.

Specular

The Specular shader uses the built in normal mapping and adds specular highlights and reflections to it. The maximal LightID is hardcoded in the shader for compatibility reasons and is 20 by default. The Specular power is also hardcoded in the pixel shader and is 30.0 by default.

It takes:

- Diffuse texture - in stage 0.
- Front Environment texture - in stage 1.
- Normal texture - in stage 2 because stage 1 is reserved for shadow mapping by AppGameKit.
- Specular texture - in stage 3.
- Back Environment texture - in stage 4.
- Vec3 lightPosition[lightMaxID]
- Vec3 lightColor [lightMaxID]
- Float lightSize [lightMaxID]
- Float lightCount - although the maximum limit of lights is 20, you can set the light count between 0 and 20 for better performance.
- Float normalSize – to scale the bumpiness of the normal map.
- Float specularPower – to adjust the specularity power.

Self-Illumination

The Self illumination shader uses the built in normal mapping + specular mapping and adds Self-illumination to it.

It takes the same as the Specular effect plus a fourth texture and variable to scale the illumination lights:

- Diffuse texture - in stage 0.

- Selfillumination texture - in stage 1 because stage 4-7 is taken by shadowmode(3).
- Normal texture - in stage 2 because stage 1 is reserved for shadowmapping by AGK.
- Specular texture - in stage 3.
- Vec3 lightPosition[lightMaxID]
- Vec3 lightColor [lightMaxID]
- Float lightSize [lightMaxID]
- Float lightCount - although the maximum limit of lights is 20, you can set the light count between 0 and 20 for better pervormance.
- Float illuminationGlow – to scale the area or power if the light.
- Float normalSize – to scale the bumpiness of the normal map.
- Float specularPower – to adjust the specularity power.

Wireframe

For the Wireframe Effect we have to create a mobile/webgl version too . This is because the shader language used by those platforms don't support some functions, therefore we have to make a workaround. We also need to manipulate the vertex attributes to add barycentric coordinates so we could calculate the distance to the three edges of a polygon.

It takes:

- Diffuse texture - in stage 0.
- Vec4 wireframeColor
- Float wireframeThikness
- Float wireframeSmoothness

Fur

The Fur shader creates several layers of the object and scales them up. By texturing it with a transparent texture and fading the alpha value till the last layer we can create the illusion of grass, fur or other similar materials.

It takes:

- Diffuse texture - in stage 0.
- Fur-Alpha texture - this is the texture that describes the thickness and density of the fur.
- Float FurLength - how far the fur extends the object i.e. the fur length.
- Float FurLayer - defines how many layer it has.
- Vec3 FurForce - you can give the fur a swaying effect with it.

Dissolve

The Dissolve shader can create many different texture based animations using a mask and a 1D ramp texture.

It takes:

- Diffuse texture - in stage 0.
- Mask texture - in stage 2.
- Ramp texture - in stage 3.

Outline

The Outline Effect creates a second layer of the object and scales it up. It then only draws the back face and colors it so you get the illusion of an outline. You can combine the Outline effect with other effects, but you must call the Add function after the base Effect.

It takes:

- Float outlineSize
- You can color the outline using SetObjectColor()

Toon

The maximal LightID is hard coded in the shader for compatibility reasons and is 20 by default. The Specular power is also hard coded in the pixel shader and is 30.0 by default.

It takes:

- Diffuse texture - in stage 0.
- Vec3 sunDir
- Vec3 sunColor
- Vec3 ambientColor
- Vec3 lightPosition[lightMaxID]
- Vec3 lightColor [lightMaxID]
- Float lightSize [lightMaxID]
- Float lightCount - although the maximum limit of lights is 20, you can set the light count between 0 and 20 for better performance.
- Float toonMaxLevel - defines the amount of light shades
- Float toonSpecularPower

XRay

The XRay Shader is one of three effects that you can combine with other base effects.

It takes:

- Diffuse texture - in stage 0.
- vec4 xrayColor
- Float xraySize

See Through

The See Through shader needs the depth of occluders, therefore you must render the depth to a render target first.

It takes:

- Diffuse texture - in stage 0.
- Border texture - in stage 1.
- Occluder Depth texture – in stage 2.
- Vec2 cameraRange – .x is the near value and .y is the far value of the camera range

Energy Shield

The Shield shader needs the depth of the scene.

It takes:

- Diffuse texture - in stage 0.
- Scene Depth texture - in stage 2.
- Pattern texture - in stage 3.
- Vec2 cameraRange - .x is the near value and .y is the far value of the camera range
- Vec4 shieldColor
- Vec4 shieldBorderColor
- Float shieldBorderSize

Depth of Field

The Depth of Field shader is a full screen effect and needs a quad to render to.

It takes:

- Scene Color texture – in stage 0
- Blurred Scene texture – in stage 1
- Scene Depth texture – in stage 2
- Vec2 cameraRange - .x is the near value and .y is the far value of the camera range
- vec2 dofFocus - .x is the near value and .y is the far value of the Focus.

Water

The Water shader needs the depth and a reflection and refraction render from the scene.

It Takes:

- Normal texture – in stage 1
- Scene Depth texture – in stage 2
- Reflection texture – in stage 3
- Refraction texture – in stage 4
- Vec2 cameraRange - .x is the near value and .y is the far value of the camera range
- Vec3 sunDir
- Vec3 sunColor
- The color of the water can be changed with SetObjectColor()
- Float waterDepthScale – This defines the overall scale of the water and is used to create smooth edges.
- Float waterDepthColorScale - This defines at what scale the Deep water begins.
- Float waterDepthColor – This is the deep water color.
- Float waterWaveSpeed
- Float waterWaveStrength
- Float waterShineDamper
- Float waterReflectivity

GodRay

The GodRay shader is a full screen shader and needs the scene and depth render images.

It Takes:

- Scene texture – in stage 0
- Depth texture – in stage 2
- Vec2 sunScreenPos – the position of the sun on the screen.
- Float rayWeight
- Float rayDecay
- Float rayDensity
- Float rayExposure
- Float raySampleCount – Sample count the maximum possible is 128 hardcoded in the shader.

SSAO (Screen Space Ambient Occlusion)

The SSAO shader is a full screen shader and needs the scene and depth render images.

It Takes:

- Scene texture – in stage 0
- Depth texture – in stage 2
- Vec2 blurDir
- Float ssaoStrength
- Float ssaoArea
- Float ssaoFalloff
- Float ssaoRadius
- Float ssaoSampleCount

ColorGrading

The ColorGrading effect let you make final changes to the overall appearance using a LUT(LookUpTable). You can also fade between two LUT's.

It Takes:

- Scene texture – in stage 0
- LookupTableA texture – in stage 1
- LookupTableB texture – in stage 2
- Float lutFading

Glow

The Glow effect utilizes the Self-Illumination effect to create a shiny glowing effect around the bright parts of the Self-Illumination effect.

It Takes:

- Scene texture – in stage 0
- Blur texture – in stage 1

Terrain

The Terrain shader uses five textures with a bump/height-map encoded in the Alpha channel to create a multitextured terrain with bump mapping.

It Takes:

- Base texture – in stage 0
- Red Channel texture – in stage 1
- Green Channel texture – in stage 2
- Blue Channel texture – in stage 3
- Alpha Channel texture – in stage 4
- Splat texture – in stage 5
- Float normalSize

Ocean

The Ocean shader needs the depth and a reflection and refraction render from the scene.

It Takes:

- Normal texture – in stage 1
- Scene Depth texture – in stage 2
- Reflection texture – in stage 3
- Refraction texture – in stage 4
- Vec2 cameraRange - .x is the near value and .y is the far value of the camera range
- Vec3 sunDir
- Vec3 sunColor
- The color of the water can be changed with SetObjectColor()
- Float OceanDepthScale – This defines the overall scale of the water and is used to create smooth edges.
- Float OceanDepthColorScale - This defines at what scale the Deep water begins.
- Float OceanDepthColor – This is the deep water color.
- Float OceanWaveStrength
- Float OceanShineDamper
- Float OceanReflectivity

- Float OceanFoamScale
- Float OceanWaveSpeed – Wave speed
- Float OceanWaveSteepness
- Float OceanMaxWaves – maximal number of waves this can't exceed six waves
- vec4 OceanWave[6] – the first two values are the direction then there is the Amplitude and Length of the Wave

Edge Detection

The Edge Detection shader is a full screen shader and needs the scene and depth render images.

It Takes:

- Scene texture – in stage 0
- Depth texture – in stage 2
- Float edgeDetectionStrength
- Vec4 edgeDetectionColor

Snow

The Snow shader lets you add a layer of snow, slime or whatever texture you choose, on top of your object.

It Takes:

- Diffuse texture – in stage 0
- Snow texture – in stage 1
- Normal texture – in stage 2
- Vec3 snowDir
- Float snowBlend
- Float snowThickness

COMMANDS

1. Init

- SP_Init()
- SP_SetClearColor(Red,Green,Blue)
- SP_RenderImage_SetSize(Width,Height)
- SP_Camera_SetRange(Near#,Far#)
- SP_Camera_GetRangeFar() float SP_Camera_Far#
- SP_Camera_GetRangeNear() float SP_Camera_Near#
- SP_SkyBox_SetVisible(Visible)
- SP_Screen_Update()
- SP_Sync()
- SP_PostEffects_Render()
- SP_FindObjectShader(ObjectID) int SP_Global[Index].ShaderID
- SP_FindMeshShader(ObjectID,MeshID) int SP_Global[Index].ShaderID
- SP_AddObject(ObjectID) int SP_ShaderID
- SP_AddMesh(ObjectID,MeshID) int SP_ShaderID
- SP_SetReadRawFiles(ReadRawFiles)

2. Lighting

- SP_Ambient_SetColor(Red,Green,Blue)
- SP_Sun_SetDirection(DirX#,DirY#,DirZ#)
- SP_Sun_SetColor(Red,Green,Blue)
- SP_Light_FindFreeID() int LightID
- SP_Light_AddPointLight(X#,Y#,Z#,Size#,Red,Green,Blue) int SP_Light.length
- SP_Light_Remove(LightID)
- SP_Light_SetPosition(LightID,X#,Y#,Z#)
- SP_Light_SetColor(LightID,Red,Green,Blue)
- SP_Light_SetSize(LightID,Size#)
- SP_Light_SetCounter()
- SP_Light_Update()
- SP_Light_GetPositionX(LightID) float SP_Light[LightID].X#
- SP_Light_GetPositionY(LightID) float SP_Light[LightID].Y#
- SP_Light_GetPositionZ(LightID) float SP_Light[LightID].Z#
- SP_Light_GetColorRed(LightID) float SP_Light[LightID].Red
- SP_Light_GetColorGreen(LightID) int SP_Light[LightID].Green
- SP_Light_GetColorBlue(LightID) int SP_Light[LightID].Blue
- SP_Light_GetColorSize(LightID) float SP_Light[LightID].Size#

3. Postprocessing

- SP_CreateDepthRenderImage() int ImageID
- SP_RenderScene()
- SP_RenderExcludingDepth()
- SP_AddMaskingObject(ObjectID)
- SP_SetMaskingObjectsVisible(Value)
- SP_DrawMaskingObjects()
- SP_Debug_SceneVisible(Visible)

- SP_Debug_DepthVisible(Visible)

4. Shader Manipulation

- SP_LoadCustomFullscreenShader(Pixel$) int ShaderID
- SP_LoadCustomShader(Vertex$,Pixel$) int ShaderID
- SP_GetCustomVertexShader(File$) string String$
- SP_GetCustomPixelShader(File$) string String$
- SP_GetDefaultNormalVS(ObjectID,MeshID,NormalImageID) string String$

5. Specular

- SP_Specular_AddObject(ObjectID,DiffuseImageID,NormalImageID,SpecularImageID,EnvironmentID)
- SP_Specular_AddMesh(ObjectID,MeshID,DiffuseImageID,NormalImageID,SpecularImageID,EnvironmentID)
- SP_Specular_RemoveObject(ObjectID)
- SP_Specular_RemoveMesh(ObjectID,MeshID)
- SP_Specular_SetNormalSize(ObjectID,Size#)
- SP_Specular_SetSpecularPower(ObjectID,Power#)
- SP_Specular_UpdateLight(LightID,X#,Y#,Z#,Red,Green,Blue,Size#)

6. Selfillumination

- SP_Illumination_AddObject(ObjectID,DiffuseImageID,NormalImageID,SpecularImageID,IllluminationImageID)
- SP_Illumination_AddMesh(ObjectID,MeshID,DiffuseImageID,NormalImageID,SpecularImageID,IlluminationImageID)
- SP_Illumination_RemoveObject(ObjectID)
- SP_Illumination_RemoveMesh(ObjectID,MeshID)
- SP_Illumination_SetNormalSize(ObjectID,Size#)
- SP_Illumination_SetSpecularPower(ObjectID,Power#)
- SP_Illumination_SetGlow(ObjectID,Glow#)
- SP_Illumination_UpdateLight(LightID,X#,Y#,Z#,Red,Green,Blue,Size#)

7. Wireframe

- SP_Wireframe_AddObject(ObjectID,DiffuseImageID)
- SP_Wireframe_AddMesh(ObjectID,MeshID,DiffuseImageID)
- SP_Wireframe_RemoveObject(ObjectID) int Index
- SP_Wireframe_RemoveMesh(ObjectID,MeshID) int Index
- SP_Wireframe_SetColor(ObjectID,Red,Green,Blue,Alpha)
- SP_Wireframe_SetSmoothness(ObjectID,Smoothness#)
- SP_Wireframe_SetThikness(ObjectID,Thikness#)

8. Fur

- SP_Fur_AddObject(ObjectID,DiffuseImageID,MaskImageID,MaxLayer,Length#)
- SP_Fur_AddMesh(ObjectID,MeshID,DiffuseImageID,MaskImageID,MaxLayer,Length#)
- SP_Fur_RemoveObject(ObjectID)
- SP_Fur_RemoveMesh(ObjectID,MeshID)
- SP_Fur_GetMaxLayer(ObjectID) int SP_Fur[Index].MaxLayer
- SP_Fur_GetLength(ObjectID) float SP_Fur[Index].Length#
- SP_Fur_SetForce(ObjectID,ForceX#,ForceY#,ForceZ#)
- SP_Fur_Render()
- SP_Fur_CreateMaskImage(Size,Density) int ImageID

9. Dissolve

- SP_Dissolve_AddObject(ObjectID,DiffuseImageID,MaskImageID,RampImageID)
- SP_Dissolve_AddMesh(ObjectID,MeshID,DiffuseImageID,MaskImageID,RampImageID)
- SP_Dissolve_RemoveObject(ObjectID)
- SP_Dissolve_RemoveMesh(ObjectID,MeshID)
- SP_Dissolve_SetThreshold(ObjectID,Threshold#)
- SP_Dissolve_SetSize(ObjectID,Size#)

10. Outline

- SP_Outline_AddObject(ObjectID,SoftNormals)
- SP_Outline_AddMesh(ObjectID,MeshID,SoftNormals)
- SP_Outline_RemoveObject(ObjectID)
- SP_Outline_Render()
- SP_Outline_SetColor(ObjectID,Red,Green,Blue,Alpha)
- SP_Outline_SetSize(ObjectID,Size#)

11. Toon

- SP_Toon_AddObject(ObjectID,DiffuseImageID)
- SP_Toon_AddMesh(ObjectID,MeshID,DiffuseImageID)
- SP_Toon_RemoveObject(ObjectID)
- SP_Toon_RemoveMesh(ObjectID,MeshID)
- SP_Toon_SetSpecularPower(ObjectID,Power#)
- SP_Toon_SetLevel(ObjectID,level)
- SP_Toon_UpdateAmbient(Red,Green,Blue)
- SP_Toon_UpdateSun(DirX#,DirY#,DirZ#,Red,Green,Blue)
- SP_Toon_UpdateLight(LightID,X#,Y#,Z#,Red,Green,Blue,Size#)

12. XRay

- SP_XRay_AddObject(ObjectID)
- SP_XRay_AddMesh(ObjectID,MeshID)
- SP_XRay_RemoveObject(ObjectID) int Index
- SP_XRay_RemoveMesh(ObjectID,MeshID) int Index
- SP_XRay_Render()
- SP_XRay_SetColor(ObjectID,Red,Green,Blue,Alpha)
- SP_XRay_SetSize(ObjectID,Size#)

13. See Through

- SP_SeeThrough_AddObject(ObjectID,DiffuseImageID,PatternImageID)
- SP_SeeThrough_AddMesh(ObjectID,MeshID,DiffuseImageID,PatternImageID)
- SP_SeeThrough_RemoveObject(ObjectID)
- SP_SeeThrough_RemoveMesh(ObjectID,MeshID)
- SP_SeeThrough_SetCameraRange(Near#,Far#)

14. Energy Shield

- SP_Shield_AddObject(ObjectID,PatternImageID,MaskImageID)
- SP_Shield_AddMesh(ObjectID,MeshID,PatternImageID,MaskImageID)
- SP_Shield_RemoveObject(ObjectID)
- SP_Shield_RemoveMeshID(ObjectID,MeshID)
- SP_Shield_SetCameraRange(Near#,Far#)
- SP_Shield_SetColor(ObjectID,Red,Green,Blue,Alpha)
- SP_Shield_SetBorderColor(ObjectID,Red,Green,Blue,Alpha)
- SP_Shield_SetBorderSize(ObjectID,Size#)

15. Depth of Field

- SP_DoF_SetActive(Active)
- SP_DoF_SetBlurRenderImageSize(Width,Height)
- SP_Dof_SetScreenFocus(ScreenX#,ScreenY#,Steps#)
- SP_DoF_SetCameraRange(Near#,Far#)
- SP_DoF_SetFocus(Distance#,Range#)
- SP_DoF_SetBlur(Horizontal#,Vertical#)
- SP_DoF_Render(SceneImageID)
- SP_DoF_RenderBlur()
- SP_DoF_Debug_BlurVisible(Visible)

16. Water

- SP_Water_AddObject(ObjectID ,NormalImageID) int Index
- SP_Water_RemoveObject(ObjectID)
- SP_Water_SetReflectionRenderImageSize(Width,Height)
- SP_Water_SetRefractionRenderImageSize(Width,Height)
- SP_Water_RenderReflection()
- SP_Water_SetCameraRange(Near#,Far#)
- SP_Water_SetHeight(Height#)
- SP_Water_SetSunDir(ObjectID,DirX#,DirY#,DirZ#)
- SP_Water_SetSunColor(ObjectID,Red,Green,Blue)
- SP_Water_SetReflectivity(ObjectID,Reflectivity#)
- SP_Water_SetShineDamper(ObjectID,ShineDamper#)
- SP_Water_SetWaveStrength(ObjectID,WaveStrength#)
- SP_Water_SetWaveSpeed(ObjectID,WaveSpeed#)
- SP_Water_SetColor(ObjectID,Red,Green,Blue)
- SP_Water_SetDepthColor(ObjectID,Red,Green,Blue)
- SP_Water_SetDepthScale(ObjectID,DepthScale#)
- SP_Water_SetDepthColorScale(ObjectID,DepthScale#)
- SP_Water_UpdateSun(DirX#,DirY#,DirZ#,Red,Green,Blue)
- SP_Water_Debug_ReflectionVisible(Visible)

17. GodRay

- SP_GodRay_SetActive(Active)
- SP_GodRay_Render(SceneImageID)
- SP_GodRay_UpdateSun(DirX#,DirY#,DirZ#)
- SP_GodRay_SetWeight(Weight#)
- SP_GodRay_SetDecay(Decay#)
- SP_GodRay_SetDensity(Density#)
- SP_GodRay_SetExposure(Exposure#)
- SP_GodRay_SetSamples(SampleCount)
- SP_GodRay_SetSunSize(SunSize#,HaloSize#)
- SP_GodRay_SetSunColor(Red,Green,Blue)

18. SSAO

- SP_SSAO_SetActive(Active)
- SP_SSAO_SetRenderImageSize(Width,Height)
- SP_SSAO_SetBlurRenderImageSize(Width,Height)
- SP_SSAO_Render(SceneImageID)
- SP_SSAO_SetCameraRange(Near#,Far#)
- SP_SSAO_SetStrength(Strength#)
- SP_SSAO_SetBase(Base#)
- SP_SSAO_SetArea(Area#)
- SP_SSAO_SetFalloff(Falloff#)
- SP_SSAO_SetRadius(Radius#)
- SP_SSAO_SetSamples(SampleCount)
- SP_SSAO_SetBlur(Horizontal#,Vertical#)
- SP_SSAO_Debug_SSAOVisible

19. Glow

- SP_Glow_SetActive(Active)
- SP_Glow_Render(SceneImageID)
- SP_Glow_RenderMask()
- SP_Glow_Debug_MaskVisible(Visible)

20. Environment

- SP_Environment_AddPoint(X#,Y#,Z#,Width#,Height#)
- SP_Environment_AddCamera(X#,Y#,Z#,Width#,Height#)
- SP_Environment_FixToObject(ObjectID,EnvironmentID)
- SP_Environment_Update()
- SP_Environment_DPRender(EnvironmentID)
- SP_Environment_SSRenderReflection(EnvironmentID)
- SP_Environment_SSRenderRefraction(EnvironmentID)
- SP_Environment_SetPosition(EnvironmentID,X#,Y#,Z#)
- SP_Environment_GetPositionX(EnvironmentID)
- SP_Environment_GetPositionY(EnvironmentID)
- SP_Environment_GetPositionZ(EnvironmentID)
- SP_Environment_Debug_DPVisible(EnvironmentID,Visible)

21. Glass

- SP_Glass_AddObject(ObjectID,NormalImageID,EnvironmentID)
- SP_Glass_AddMesh(ObjectID,MeshID,NormalImageID,EnvironmentID)
- SP_Glass_RemoveObject(ObjectID)
- SP_Glass_RemoveMesh(ObjectID,MeshID)
- SP_Glass_SetNormalSize(ObjectID,Size#)
- SP_Glass_SetEta(ObjectID,Eta#)

22. GolorGrading

- SP_ColorGrading_SetActive(Active)
- SP_ColorGrading_Render(SceneImageID)
- SP_ColorGrading_SetLookUpTable(Lookup1ImageID,Lookup2ImageID)
- SP_ColorGrading_SetFade(Fade#)
- SP_ColorGrading_Debug_LookupVisible(Visible)

23. Terrain

- SP_Terrain_AddObject(ObjectID,BaseImageID,RedImageID,GreenImageID,BlueImageID,AlphaImageID,SplatImageID)
- SP_Terrain_AddMesh(ObjectID,MeshID,BaseImageID,RedImageID,GreenImageID,BlueImageID,AlphaImageID,SplatImageID)
- SP_Terrain_RemoveObject(ObjectID)
- SP_Terrain_RemoveMesh(ObjectID,MeshID)
- SP_Terrain_SetNormalSize(ObjectID,Size#)

24. Ocean

- SP_Ocean_AddObject(ObjectID ,NormalImageID) int Index
- SP_Ocean_RemoveObject(ObjectID)
- SP_Ocean_SetReflectionRenderImageSize(Width,Height)
- SP_Ocean_SetRefractionRenderImageSize(Width,Height)
- SP_Ocean_RenderReflection()
- SP_Ocean_SetCameraRange(Near#,Far#)
- SP_Ocean_SetHeight(Height#)
- SP_Ocean_SetSunDir(ObjectID,DirX#,DirY#,DirZ#)
- SP_Ocean_SetSunColor(ObjectID,Red,Green,Blue)
- SP_Ocean_SetReflectivity(ObjectID,Reflectivity#)
- SP_Ocean_SetShineDamper(ObjectID,ShineDamper#)
- SP_Ocean_SetWaveStrength(ObjectID,WaveStrength#)
- SP_Ocean_SetWaveSpeed(ObjectID,WaveSpeed#)
- SP_Ocean_SetColor(ObjectID,Red,Green,Blue)
- SP_Ocean_SetDepthColor(ObjectID,Red,Green,Blue)
- SP_Ocean_SetDepthScale(ObjectID,DepthScale#)
- SP_Ocean_SetDepthColorScale(ObjectID,DepthScale#)
- SP_Ocean_UpdateSun(DirX#,DirY#,DirZ#,Red,Green,Blue)
- SP_Ocean_Debug_ReflectionVisible(Visible)
- SP_Ocean_SetWaveSpeed(OceanID,WaveSpeed#)
- SP_Ocean_SetWaveSteepness(OceanID,WaveSteepness#)
- SP_Ocean_SetMaxWaves(OceanID,MaxWaves)

25. Edge Detection

- SP_EdgeDetection_SetActive (Active)
- SP_EdgeDetection_Render (SceneImageID)
- SP_EdgeDetection_SetCameraRange(Near#,Far#)
- SP_EdgeDetection_SetStrength(Strength#)
- SP_EdgeDetection_SetColor(Red,Green,Blue,Alpha)

26. Snow

- SP_Snow_AddObject (ObjectID ,DiffuseImageID,NormalImageID,SnowImageID)
- SP_Snow_AddMesh (ObjectID,MeshID,DiffuseImageID,NormalImageID,SnowImageID)
- SP_Snow_RemoveObject(ObjectID)
- SP_Snow_RemoveMesh(ObjectID,MeshID)
- SP_Snow_SetDir(ObjectID,DirX#,DirY#,DirZ#)
- SP_Snow_SetBlend(ObjectID,Blend#)
- SP_Snow_SetThickness(ObjectID,SnowThickness#)