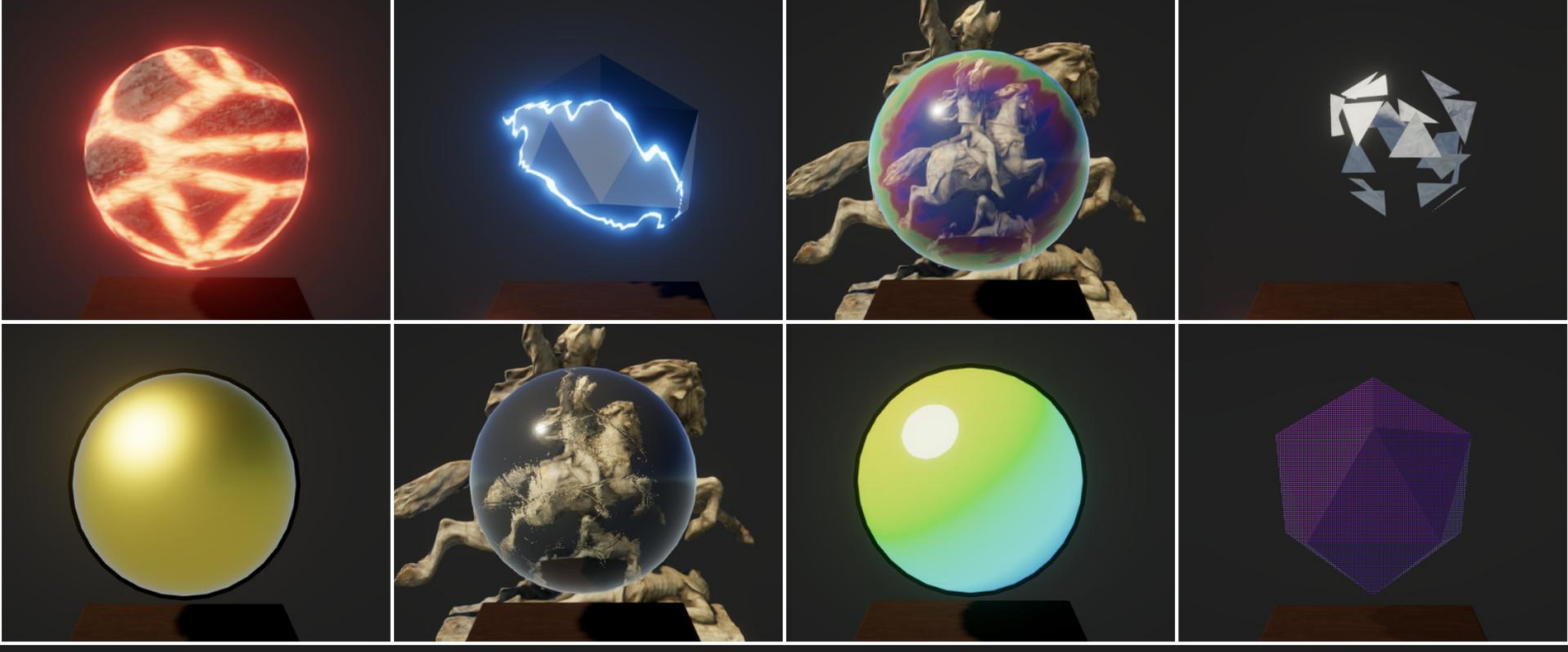

Shader Toolbox for Unity URP

WHAT IS "SHADER TOOLBOX FOR URP"?

Shader Toolbox for URP is a collection of production-ready shaders for the Universal Render Pipeline. Each shader is made with Shader Graph and modelled on the default Unity Lit shader, complete with the same kinds of material options – opaque and transparent, metallic and specular, alpha clipping, shadow receiving, and front or back face culling.

This pack contains photorealistic and stylized effects including **Glass**, **Bubble**, **Voronoi Lava**, **Dissolve**, and **Heat Distortion**. It also comes with a library of subgraphs including a normal mapping helper, better Voronoi support, stochastic texture sampling, refraction, and a whole host of lighting- and color-related subgraphs.

The latest version of this asset has been tested with the following Unity versions:

- **Unity 2022.3, 6.0, 6.1, 6.2, 6.3, 6.4**

SUPPORT

Sometimes things break! If you've encountered an error and this README doesn't have the answers (or if you have feature requests), then pop me an email at **danielilett+shadertoolbox@gmail.com** and I'll try to sort you out. Please:

- **Clearly** describe the problem you are having and what steps I can take to reproduce the error.
- Include the **Unity version you are using.**
- Also include the **package version** you are using (you can find this in the Package Manager).

- Include the **render path** (i.e. Forward, Forward+, Deferred) – this option is on your URP settings asset.
- A**ttach images or short videos** where necessary to describe your problem.

Following these steps will help me fix your issue as quickly as possible!

SETUP

Folder Structure: Upon installing the pack, all the assets will be contained in the "Shader Toolbox for URP" folder. All demo scenes are included under "Shader Toolbox for URP /Demo", containing an example scene with materials configured to work with this asset pack. Most of the shaders and subgraphs are included inside "Shader Toolbox for URP/Shaders".

Warning: If you encounter errors when you import this pack related to shader variants, please go into *Preferences -> Shader Graph* (before Unity 6) or *Project Settings -> Shader Graph* (Unity 6 and after) to increase the shader variant limit – this might need to be at least 512.

CAMERA TRANSPARENT TEXTURE

When creating effects which read from the screen, URP provides the Scene Color node, which in turn reads the `_CameraOpaqueTexture`. This means you must run such effects in the Transparent queue, but it also means you won't see transparent objects in this texture.

This asset pack contains an experimental `_CameraTransparentTexture` which contains the camera state after all transparent objects have been rendered. With this texture, if you can render some objects after the transparent queue, then you will be able to read transparent camera data for those objects. Please note that using this functionality will incur a performance cost.

This shader pack uses **Universal Render Pipeline**'s **ScriptableRenderFeature** for the `_CameraTransparentTexture` functionality. The Unity documentation will outline the basics of URP if you're not familiar with how to create custom renderers.

Please follow these steps to read transparent camera data in your scene:

- Find your **Universal Renderer Data** and add the effect(s) you wish to use in the **Renderer Features** section at the bottom.
    - This is most commonly found in the Assets/Settings folder if you created a new project using the URP template from the Unity Hub.
    - This asset will be named something like "UniversalRP-HighQuality" (Unity versions 2022.3 and prior) or "PC_RPAsset" (Unity 6) by default.
    - *Shader Toolbox for URP* also includes a ready-made asset named "**ShaderToolbox-Renderer**" in the Demo folder, which has the Outline effect pre-added.
- Use the `_CameraTransparentTexture` when rendering any object after rendering transparents. To do this:
    - Add these objects to a separate layer.
    - Find your Universal Renderer Data again and remove this layer from the Transparent Layer Mask at the top.
    - Add a new Render Objects feature to your Universal Renderer Data at the bottom.

- Set the Event setting to *AfterRenderingTransparents*.
- Set the *Filters -> Queue* to *Transparent*, and the *Filters -> Layer Mask* to the layer you originally added the objects to.
- Apply a material using a shader from Shader Toolbox for URP which has the option to read from `_CameraTransparentTexture`. Or, use the `SampleTransparentTexture` subgraph to sample it in your own effects.

RENDER GRAPH SUPPORT (UNITY 6)

Shader Toolbox for URP supports **Render Graph** and non-Render Graph workflows. Effects which use the Renderer Feature API will require URP Render Graph in the future.

You can disable Render Graph via *Project Settings -> Graphics -> Pipeline Specific Settings -> URP*. You will find a checkbox to disable Render Graph near the bottom of the window. However, this shouldn't be necessary unless your project uses other external effects which rely on the non-Render Graph workflow.

The non-Render Graph workflow was hidden behind a compiler preprocessor in Unity 6.3, and was removed entirely in Unity 6.4.

ADDITIONAL WARNINGS

These shaders are designed for **linear color space**, so you may encounter issues in gamma space. To swap between color spaces, go to *Project Settings->Player->Other Settings* and find the **Color Space** dropdown option.

STENCIL SUB TARGETS

This asset pack comes with new Shader Graph sub-targets. A sub-target is like a graph preset (e.g. 'Lit', 'Unlit', 'Decal', 'Fullscreen') which defines the inputs, outputs, and settings for a Shader Graph-based shader.

The new Stencil Lit and Stencil Unlit sub-targets function identically to the existing Lit and Unlit sub-targets that come with URP, except they support the stencil buffer, allowing you to create your own shaders with custom stencil support. You can create one via *Create -> Shader Graph -> URP -> Stencil {Lit, Unlit} Shader Graph*.

Once you open the graph, add nodes and properties as you normally would. When you save the graph and attach it to a material, you will see the following options under the Surface Options section:

- **Reference** - Stencil ID value to compare with the existing stencil buffer value.
- **Comparison** - Comparison test to use between Reference value and existing stencil buffer value.
- **Pass** - What to do with the existing stencil buffer value if the stencil test passes.
- **Fail** - What to do with the existing stencil buffer value if the stencil test fails.
- **Z-Fail** - What to do with the existing stencil buffer value if the stencil test passes, but the z-test fails.

The default shader GUI for the custom Stencil Lit and Stencil Unlit graph presets display the stencil settings under the Surface Options heading. If you would like to create your own custom GUI script, then you need to access these named material properties:

- `_StencilRef`
- `_StencilComp`
- `_StencilPass`
- `_StencilFail`
- `_StencilZFail`

SHADERS INCLUDED

The following assets are included in the asset pack:

The main Default Lit shader acts much like the URP default Lit shader. It acts as a useful for base for you to create your own effects.

SURFACE OPTIONS

- **Workflow Mode** – Choose between *Metallic* and *Specular* workflows. This affects how the shader calculates lighting and changes some of the options presented to you.
- **Surface Type** – Toggle between *Opaque* and *Transparent* rendering.
- **Render Face** – Choose whether to render *Front*, *Back*, or *Both* faces.
- **Alpha Clip** – Toggle whether the shader should apply the *Alpha Clip Threshold*.
- **Alpha Clip Threshold** – Pixels with final base color alpha values below this threshold will be culled if *Alpha Clip* is enabled.
- **Receive Shadows** – Toggle whether realtime shadows should be applied to the object.

LIT PROPERTIES

- **Base Color** – The albedo color of the object. The alpha channel may be used for transparency.
- **Base Texture** – Similar to Base Color, can be used to change the albedo color of the object. The tiling and offset settings used for this texture are applied to the other Lit textures supplied to the material.
- **Metallic** – Appears only in Metallic workflow mode. Controls how metallic the object is – 1 means the object is fully metallic, whereas 0 means it is completely non-metallic.
- **Specular Color** – Appears only in Specular workflow mode. Controls the color of specular highlights that appear on the object's surface.
- **Smoothness** – A value between 0 and 1 representing how smooth the surface is. Rough surfaces tend to have no specular highlights, while totally smooth surfaces tend to have small, bright highlights. The smoothness value is equal to the texture sample multiplied by the slider value.

- **Convert From Roughness** – Some 3D modelling packages output roughness textures instead of smoothness textures. Ticking this option will convert a roughness texture applied to the Smoothness slot into smoothness data.
- **Normal Map** – A texture representing normal vector directions on the object surface. The slider controls the strength of the augmented normal vector data.
- **Heightmap** – A texture which lets you simulate raised or lowered parts of the surface using UV offsets. Using this option incurs an increased performance impact.
- **Ambient Occlusion** – The amount of lighting that falls into small crevices on the object surface. 1 means the surface is fully lit, while 0 means that part of the surface is obscured (occluded).
- **Emission Color** – Applies an emissive color to the surface, which will be visible regardless of whether the object is in shadow.

The dissolve effect can cut away parts of the mesh and display a glowing edge along the cutoff boundary. This effect can cut away parts of the mesh on one side of a plane, or over a specific distance from an origin point.

In Plane mode, the plane is represented using a point on the surface of the plane and the normal direction of the plane. The included **`DissolvePlane.cs`** script can automatically send data from a plane to the material.

SURFACE OPTIONS

As above.

LIT PROPERTIES

As above.

DISSOLVE PROPERTIES

- **Noise Scale** – Scale of the noise pattern used for the edge lip of the dissolve effect.
- **Noise Strength** – How strongly the noise pattern distorts the edge lip shape.
- **Origin Type** – Whether the effect uses Plane mode or Point mode.
- **Cutoff Point** – In Point mode, this is the origin point from which the dissolve emanates. In Plane mode, this is a point that lies on the plane.
- **Cutoff Direction** – Only visible in Plane mode. Represents the normal vector direction of the plane.
- **Cutoff Height** – How far from the plane or point that the dissolve starts (or ends).
- **Flip Direction** – When enabled, the dissolve will instead occur the opposite direction of the plane (Plane mode), or at negative distances from the point (Point mode).
- **Glow Color** – Color of the glowing lip. Can be applied as emissive or base light.
- **Glow Thickness** – Width of the glowing lip edge.

- **Use World Space** – When enabled, world space positions are used for all quantities in the shader. Otherwise, object space is used.
- **Use Emission** – Apply the glowing lip to the graph Emission output. Otherwise, it is applied to Base Color.

The dither transparency effect cuts away pixels using alpha clipping using a dither pattern. This pattern means that the object appears semi-transparent when viewed on a screen with sufficient pixel density.

SURFACE OPTIONS

As above.

LIT PROPERTIES

As above.

DITHERING PROPERTIES

- **Use Dither Texture** – Should the effect use a texture for its dither thresholds? Otherwise, a Bayer matrix is generated inside the shader.
- **Dither Texture** – The texture containing the dither thresholds. The red channel contains the thresholds.
- **Dither Scale** – The size of the dither 'pixels'. Integer values garner the best results.
- **Opacity** – A multiplier that is applied to the output alpha.

The mesh explosion effect expands individual triangles of the mesh away from some origin point. These triangles can have gravity applied over time.

Note that a `BakeFaceColors.cs` script is included that will bake each triangle center point into the vertex colors of the mesh. This is useful for drawing the exploded mesh.

SURFACE OPTIONS

As above.

LIT PROPERTIES

As above.

EXPLOSION PROPERTIES

- **Expansion Mode** – How should the mesh expand outwards? In Normal mode, the mesh expands along vertex normals, which can result in the triangles getting larger. In Offset

mode, the mesh expands away from an origin point in 3D space (per vertex). In Colors mode, the mesh expands away from the origin point using a position vector baked into the vertex colors (this allows you to have per-triangle offsets).

- **Explosion Origin Point** – Only appears in Offset or Colors mode. Determines the origin point in space from which the explosion effect expands.
- **Explosion Distance** – How far the explosion has travelled from the origin.
- **Debris Shrink Speed** – How swiftly the triangles get smaller as the distance increases.
- **Gravity** – How quickly the triangles fall along the y-axis.
- **Random Offset Range** – A random modifier applied to the explosion distance for each vertex. High values might result in high levels of distortion.

Inverted-hull outlines can be used to render the mesh a second time, invert the normal direction of each triangle, and expand the mesh along vertex normal vectors. By coloring the mesh, you can achieve a cheap outline effect. However, the effect works best on rounded objects.

HULL OUTLINE PROPERTIES

- **Outline Color** – Base color to use for the entire outline.
- **Outline Thickness** – How far the vertices expand along vertex normals.

A copy of the Default Lit shader, but each texture is sampled using random UV offset to help break up texture tiling. This effect is more expensive, as three texture samples are used per texture.

SURFACE OPTIONS

As above.

LIT PROPERTIES

As above.

The glass effect uses the camera texture to operate on the visuals of objects behind the glass. A refraction effect can be applied to distort those visuals.

This effect can be used together with the custom `_CameraTransparentTexture` included in *Shader Toolbox for URP*. See the section above for setup tips.

SURFACE OPTIONS

As above.

LIT PROPERTIES

As above.

GLASS PROPERTIES

- **Refractive Index** – The ratio between the speed of light in the glass and in the air next to the glass. Higher values result in stronger visual distortion.
- **Glass Strength** – What proportion of the final base color is made from the refracted camera texture.
- **Fresnel Power** – Power applied to the Fresnel calculations. Higher values mean thinner Fresnel highlights around the rim of the object.
- **Fresnel Color** – Color multiplier for the Fresnel lighting.
- **Use Emission** – Should the Fresnel layer be output to Emission instead of Base Color?
- **Camera Texture Mode** – Should the shader sample the `_CameraOpaqueTexture` (default) or `_CameraTransparentTexture`?

Gooch shading is a non-photorealistic technique that is often used in computer-aided design programs. Objects are drawn with basic colors, where lighting highlights are drawn with warm colors (e.g. yellow) and shadowed areas with cold colors (e.g. blue). This version applies an extra specular highlight on top.

GOOCH PROPERTIES

- **Warm Color** – Color to use for lit areas of the mesh.
- **Cold Color** – Color to use for shadowed areas of the mesh.
- **Temperature Offset** – A skew value to move the distribution of warm and cold colors.
- **Specular Power** – Power applied to the specular highlights. Higher values result in a smaller highlight.
- **Specular Color** – Controls the color of the specular highlights that appear on the mesh surface.
- **Use HCL Color Space** – Converts the color to the Hue-Chroma-Luminance color space before performing the warm-cold mapping. Results in a nicer transition from warm to cold than when using RGB color space.

A soapy bubble effect with iridescent colors. The colors are visible in the Fresnel light on the object.

This effect can be used together with the custom `_CameraTransparentTexture` included in *Shader Toolbox for URP*. See the section above for setup tips.

SURFACE OPTIONS

As above.

LIT PROPERTIES

As above.

BUBBLE PROPERTIES

- **Color Ramp Texture** – A texture encoding the transition of colors along the edge of the bubble.
- **Fresnel Power** – Power applied to the Fresnel lighting equation.
- **Fresnel Noise Strength** – How strongly the noise offsets the color ramp.
- **Fresnel Noise Scale** – Scale of the noise offset applied to the color ramp.
- **Iridescent Strength** – How strongly the color ramp is applied to the Fresnel light.
- **Iridescent Flow Direction** – Which direction, in world space, the flow map scrolls in.
- **Refractive Index** - The ratio between the speed of light in the bubble and in the air next to the bubble. Higher values result in stronger visual distortion.
- **Use Emission** – Should the Fresnel light be output to Emission, or Base Color?
- **Camera Texture Mode** – Should the shader sample the `_CameraOpaqueTexture` (default) or `_CameraTransparentTexture`?

The heat distortion effect uses a flow map to distort the camera texture, simulating a heat haze effect.

This effect can be used together with the custom `_CameraTransparentTexture` included in *Shader Toolbox for URP*. See the section above for setup tips.

HEAT DISTORTION PROPERTIES

- **Heat Origin Point** – World space point where the heat emanates from.
- **Heat Falloff** – How quickly the heat falls off as you get further from the origin.
- **Distortion Map** – Flow map which controls how the heat haze distorts the image.
- **Distortion Strength** – How far the distortion map pushes away each pixel in the camera texture.
- **Distortion Velocity** – How quickly the distortion map scrolls across the screen.
- **Distortion Tiling** – How many times the distortion map is scaled.
- **Camera Texture Mode** – Should the shader sample the `_CameraOpaqueTexture` (default) or `_CameraTransparentTexture`?

A Voronoi-based lava effect which separates two layers of textures according to their distance from the edges between Voronoi cells.

SURFACE OPTIONS

As above.

VORONOI PROPERTIES

- **Voronoi Density** – How tightly packed the Voronoi cells are.
- **Voronoi Angle Offset** – A parameter for creating the Voronoi cells. Use values above zero.
- **Voronoi Thickness** – Width of the edges between Voronoi cell shapes. This controls the distribution of Layer 1 and Layer 2 pixels.
- **Voronoi Falloff** – Smooth falloff region between both texture layers.

FIRST LAYER PROPERTIES

Same as the Lit Properties above.

SECOND LAYER PROPERTIES

Same as the Lit Properties above.

Overlays a Voronoi-based pattern of small fragments across the surface of the object which glint when viewed at different angles.

SURFACE OPTIONS

As above.

LIT PROPERTIES

As above.

GLITTER PROPERTIES

- **Noise Scale** – How frequently the Voronoi noise pattern is tiled across the surface of the mesh
- **Noise Rotation Speed** – How quickly the random values increase over time, rotating the random vector direction that is used for glint calculations.
- **Glitter Offset** – A modifier which determines how likely is is that a glitter particle actually reflects light.
- **Sparkliness** – Higher values mean glitter particles reflect with a narrower angle range, causing the object to look 'more sparkly' when moving around it.
- **Glitter Color** – Color of the glitter particles at one end of the generated noise range.
- **Glitter Color 2** – Color of the glitter particles at the other end of the generated noise range.
- **Fresnel Power** – Power applied to the Fresnel calculations. Higher values mean thinner Fresnel highlights around the rim of the object.
- **Fresnel Color** – Color multiplier for the Fresnel lighting.
- **Spot Offset** – An angle offset applied to the Voronoi pattern generator. A value of zero results in an even grid of glitter particles.

- **Spot Thresholds** – Smoothstep thresholds values applied to the glitter particles. The second value should be at least as large as the first. Values above zero typically result in circle shapes, whereas values near zero will give you jagged glitter particle shapes.

portal card

Fades colors based on depth-distance from scene elements, and fades alpha based on camera distance. Based on a concept from the game *Abzû.*

- **Fade Thresholds** – Blend the color based on the distance between the mesh and the depth of the last rendered object at the same screen coordinate. Pixels with distance below the x-value get tinted with **Fog Start Color**, and pixels with distance over the y-value get tinted with **Fog End Color**, with a smooth transition in between.
- **Distance Thresholds** – Fades the alpha of the mesh based on the distance between the camera and the mesh pivot point. The mesh becomes invisible at distances over the y-value, and is at full alpha at distances below the x-value.
- **Edge Fade** – Fades the edges of the mesh based on UV corodinates, with separate values for the U- and V-axes.
- **Fog Start Color** – Color of the portal fog close to the mesh.
- **Fog End Color** – Color of the portal fog far away from the mesh.

silhouette-pom lit

Uses the Silhouette Parallax Occlusion Mapping technique to improve the object's silhouette (edges) over regular parallax occlusion mapping.

SURFACE OPTIONS

As above.

LIT PROPERTIES

As above.

SPOM PROPERTIES

- **Steps** - How many iterations the algorithm should use to calculate parallax.
- **LOD** - Which mip level to use when sampling the heightmap texture.

lcd screen

Overlays red, green, and blue subpixels onto an object when the camera approaches it.

SURFACE OPTIONS

As above.

LIT PROPERTIES

As above.

LCD PROPERTIES

- **Distance Fade** - Distances used to fade between the regular texture and the LCD subpixel version.
    - The first parameter controls the camera distance at which the LCD is at full strength.
    - The second parameter controls the camera distance at which the LCD starts to fade in.
- **Resolution** - Resolution of the base texture, and therefore the LCD subpixels.
    - Note that the base texture should have mipmaps enabled.
- **RGB Smoothing** - Amount of smoothing to apply around the individual RGB subpixels.
- **LCD Boost Color** - A color multiplier to use when the LCD panel is visible.
    - It's recommended to use an HDR white color so that the object does not get darker when zoomed in.
- **Use Screen Space** - Should the LCD subpixels be mapped in UV space, or screen space?

The Stencil Lit effect works just like the Base Lit shader, with added support for the stencil buffer.

SURFACE OPTIONS

As above, with an extra Stencil section.

STENCILS

- **Reference** - Stencil ID value to compare with the existing stencil buffer value.
- **Comparison** - Comparison test to use between Reference value and existing stencil buffer value.
- **Pass** - What to do with the existing stencil buffer value if the stencil test passes.
- **Fail** - What to do with the existing stencil buffer value if the stencil test fails.
- **Z-Fail** - What to do with the existing stencil buffer value if the stencil test passes, but the z-test fails.

LIT PROPERTIES

As above.

The Stencil Unlit shader is a simple color effect, with added support for the stencil buffer.

SURFACE OPTIONS

As above.

UNLIT PROPERTIES

- **Base Color** – The albedo color of the object. The alpha channel may be used for transparency.
- **Base Texture** – Similar to Base Color, can be used to change the albedo color of the object.

The Silhouette effect tints the object based on the value inside the depth buffer at the same position on-screen, with configurable colors and distance thresholds.

SILHOUETTE PROPERTIES

- **Distance Thresholds** - Depth thresholds used for color blending.
    - Objects closer than the first distance use the Near Color.
    - Objects further than the second distance use the Far Color.
- **Near Color** - Color to use for objects at the closer distance threshold.
- **Far Color** - Color to use for objects at the further distance threshold.
- **Blend Color Space** - Which color space should be used for blending silhouette colors?
    - *RGB* is what you would probably expect to use.
    - *HSV* uses the hue-saturation-value color space for different color blends which cycle through different hues.
    - *HCL*, or hue-chroma-luminance, is similar to HSV, with a distinctly different journey through different hues.

A Matcap, short for *Material Capture*, is a technique for baking lighting and color data into a texture, and applying it in view space based on the surface normal vector at runtime. You can use matcaps to efficiently fake many points of light on an object.

MATCAP PROPERTIES

- **Matcap Texture** - Matcap texture which encodes color and lighting information for the object.
- **Matcap Color** - Color to use for the matcap.
- **Normal Texture** - Offsets the surface normal vectors to change the UVs used for sampling the matcap.
- **Normal Strength** - Controls the strength of the offsets produced by the normal texture.
- **Fresnel Color** - A color to use for the Fresnel highlights.
- **Fresnel Power** - Power to use for the Fresnel highlights. Higher values mean thinner highlights around the edges.
- **Vector Space** - Determines whether to apply the matcap texture in view space (as is usual) or world space.

The Triplanar Tint effect overlays three fake light colors onto an object, aligned to the three cardinal axes.

SURFACE OPTIONS

As above.

LIT PROPERTIES

As above.

TRIPLANAR PROPERTIES

- **Color X** - Color to use for surfaces aligned with the YZ plane.
- **Color Y** - Color to use for surfaces aligned with the XZ plane.
- **Color Z** - Color to use for surfaces aligned with the XY plane.
- **Direction Blend** - Controls the blending region where pixels are on the boundary between any two or three planes.
- **Color Blend** - How strongly to blend between the original colors and the triplanar tint colors.
- **Tint Emissive** - Should the triplanar tint also be applied to the emissive color in additional to the base color?

The Greyscale effect lets you desaturate the base and emission colors of the object.

SURFACE OPTIONS

As above.

LIT PROPERTIES

As above.

GREYSCALE PROPERTIES

- **Greyscale Blend** - How strongly should the shader blend between original colors and greyscale colors?

COLOR SUBGRAPHS

BLEND HCL

Lerp between two RGB input colors in HCL color space.

BLEND HSV

Lerp between two RGB input colors in HSV color space.

HCL TO RGB

Converts from Hue-Chroma-Luminance color space to Red-Green-Blue.

Based on work by Ian Taylor.

RGB TO HCL

Converts from Red-Green-Blue color space to Hue-Chroma-Luminance.

LCD SCREEN

Splits the input texture into red, green, and blue subpixels, with a horizontal scanline and a smoothing parameter.

LUMINANCE

Converts an RGB color into a single luminance value based on the human eye sensitivity to red, green, and blue light.

FOG SUBGRAPHS

APPLY FOG

Takes in a color and world-space position as input and outputs that color with the appropriate level of fog applied to it.

PORTAL CARD

Based on the 'portal cards' from the game *Abzû*, which blend the color of the surface based on the distance of the mesh from scene elements behind it, and fades the transparency of the mesh based on the distance of the mesh from the camera. Also fades the alpha at the edges of the mesh (based on UV coordinates).

Based on work by Giant Squid.

LIGHTING SUBGRAPHS

APPLY TRIPLANAR TINT

Applies triplanar lighting to the mesh aligned with the cardinal axes. You may specify a different color for each axis, and a blend parameter for surfaces that don't align with the axes.

GET MAIN LIGHT

Retrieve data from the main light, such as the direction (as with the Main Light Direction node), color, shadow attenuation, and distance attenuation.

GET AMBIENT LIGHT

Gets ambient light data from spherical harmonics.

LIT DEFAULT

The main subgraph which combines Lit properties and outputs values you can output to the graph output stack.

LIT DEFAULT LOD

Same as Lit Default, with added LOD parameter for selecting a mip level.

LIT SPOM

Same as Lit Default, but uses SPOM instead of regular parallax mapping for the heightmap.

LIT STOCHASTIC

A version of the Lit Default subgraph which also applies stochastic texture sampling to each texture map, disrupting obvious texture tiling.

MATCAP

Maps a texture based on the surface normal vector. The matcap texture should encode lighting and color information.

NOISE SUBGRAPHS

BETTER VORONOI

A new version of the Voronoi node which outputs pixel distance from the nearest edge between cells (as well as the distance from the cell center point, as the built-in Voronoi node does).

Based on work by Inigo Quilez.

RANDOM SUBGRAPHS

HASH {1-4D} TO {1-4D}

A series of subgraph nodes (16 in total) that map an input seed vector of between 1 and 4 elements to a random output vector of between 1 and 4 components, with the ability to set min and max values independently for each output component.

Based on work by Dave Hoskins.

TEXTURE SUBGRAPHS

APPLY NORMAL MAP

Takes in a normal map and applies its normal data to the world normal vector to produce a combined vector.

RESOLUTION LIMIT

Uses the mip maps of a texture to reduce its resolution.

SAMPLE STOCHASTIC

A variant of Sample Texture 2D which uses randomized sampling methods to reduce the prevalence of texture tiling.

SAMPLE TEXTURE 2D AA (ANTI-ALIASED)

A version of Sample Texture 2D which smooths out parts of the image where point sampling would usually snap a pixel to either side of the 'pixel grid' of the screen.

Based on work by Tyler Glaiel.

SAMPLE TEXTURE 2D GRAD

A version of Sample Texture 2D which lets you manually supply gradient vectors.

SAMPLE TRANSPARENT TEXTURE

Samples the `_CameraTransparentTexture`. If this isn't set up, then it will output a grey texture instead.

SILHOUETTE PARALLAX OCCLUSION MAPPING

Apply parallax occlusion mapping with an added step where the edges of the mesh get alpha-clipped based on parallax height.

UTILITY SUBGRAPHS

SCENE INTERSECTION

Finds the distance between the surface of the currently-drawn object from whatever object has already been drawn into the depth buffer at the same screen coordinate. Only functions properly in transparent graphs.

SMOOTHSTEP BETWEEN

Perform two smoothstep functions to return a new kind of smoothstep with both lower and higher threshold windows.

STEP BETWEEN

Perform two step functions, returns 1 whenever the input is between the lower and upper thresholds.

VECTOR SUBGRAPHS

FIND PERPENDICULAR VECTORS

Given a single vector input, this node finds two vectors which are perpendicular to that vector (and to each other), with a safety mechanism if the zero vector is input.

NORMAL DISPLACEMENT

Apply a displacement along the normal vector direction to a given position.

REFRACT

Outputs a refracted vector given an input vector, a surface normal, and a refractive index.

SHEAR

Applies a shear transformation matrix to an input 2D position vector.

SPECIAL THANKS

Many thanks to:

- ambientCG for many of the CC0 licensed textures used in the demo
- OpenGameArt for some CC0 licensed audio used in the demo
- Kenney for some smoke particle textures used in the demo
- Blender Foundation for the Suzanne monkey model used in the demo
- SoupBubbles for the Matcap textures in the pack