Baking Overview

In order for all the features of the lava lamp shader to work properly, it needs some additional information about your avatar's model beyond what Unity provides. You'll need to use the **Lava Lamp Baker** prefab in "LavaLamp/Baking/Prefabs/" to bake out this information.

This prefab has two scripts on it, which each bake out a different set of information:

SDF Baker: This takes the triangle meshes which describe the avatar's surface, and converts them into a "Signed Distance Field" which describes the avatar as a 3D volume. The SDF will be exported to a 3D texture, which the shader can use to figure out the thickness of the avatar at any point.

Bind Data Baker: This will bake additional information about the starting position and orientation of the mesh's vertices, which the shader can use to have the lava move with the avatar's body parts as it animates. This also lets you set up **Subregions**, which let lava in different parts of the avatar be lit differently or flow in different directions. The bind data will be exported as a set of 3 textures.

Tips for getting a good bake:

- Make sure the avatar is in a relaxed pose without much deformation. (99% of the time this means leaving it in the default T-pose.)
- Make sure the overall avatar is sealed, with a clearly defined inside and outside and no large holes in it.
- Make sure the avatar is not clipping into itself anywhere that isn't supposed to be physically connected. If it is, adjust the pose slightly until the two parts have a gap between them.
- Some avatars use blend shapes for more customization. (Such as allowing you to adjust the amount of body fat for instance.) The shader won't be aware if blend shapes change after the bake, which can cause artifacts. It's recommended to just set the blend shapes to what you want to use before baking, and then don't adjust them at runtime. (Or just don't put the material on parts of the avatar where blend shapes will change.)
- If the avatar is very large or very small, it's recommended that you rescale it to a more normal size (0.5m - 3m) for the bake. When you scale the avatar back to the original size you will need to set the **World Rescale** setting appropriately. (check the "Using The Material" section for more details.)
- It's recommended to remove any extremely thin objects which have both sides exposed from the avatar during the SDF bake (such as skirts or capes.) This isn't necessary, but they may cause some artifacts in spots where they touch other parts of the avatar.

Signed Distance Field Baking

1. Drag the **Lava Lamp Baker** prefab into the scene with your avatar.

2. Configure the the bake settings:

    - **SDF Pixel Size:** How large the pixels of the SDF texture will be. Increasing this will lower the resolution of the texture, improving performance and memory usage. You should try to set this as high as possible, while still making sure the SDF surface roughly matches the avatar's.

    - **SDF Padding:** How much extra space around the edges of the avatar should the SDF cover. Set this to be a couple pixels sizes higher than the **SDF Shrink Wrap Radius.**

    - **SDF Shrink Wrap Radius:** Concave areas smaller than this value will be filled in. Can be useful for smoothing over the surface of the model if it has lots of small divots.

3. Set the avatar's root GameObject as the **Target Object,** a menu will appear allowing you to choose which renderers (and which submeshes on those renderers) you want to include in the SDF.
   If the avatar is made of multiple meshes, anything you want to be connected together into a single volume that lava can flow between should be baked together. If you bake SDF textures for each mesh individually, there may be seams in the lava where the different parts meet. A single SDF texture can be shared across multiple materials.

4. Hit **Start Bake** to begin the baking process. If this takes longer than a minute you probably need to increase the **SDF Pixel Size** setting. Baking is done in four stages:

    - Combining all the geometry for all selected renderers/submeshes together.

    - Generating an unsigned distance field from the geometry.

    - Converting the unsigned distance field to a signed distance field, which distinguishes between the inside and the outside of the avatar.

    - Shrink wrapping, which removes unnecessary internal detail to improve performance, and optionally, smooths over concave areas.

5. If you're happy with how the SDF looks in the scene debug visualization (it should be roughly the same shape as the original model), export it as a texture by clicking **Save SDF Texture.** Otherwise you can reset the baker and adjust any settings as needed.

6. You can automatically apply the new SDF to any lava lamp materials by dragging them into the **Auto Update Materials** list and hitting the **Update Materials** button. (You can also do this before saving the texture to preview it, but the texture will disappear next time you enter play mode if you don't save it afterwards.)
   To manually apply the SDF, go to the **Volume Data** tab on the material, drag in the texture to the **SDF Texture** property, and copy the **SDF Pixel Size, SDF Lower Corner,** and **SDF Size** values from the baker's results window to the material. (Unity's shaders don't accept Vector3s so **SDF Lower Corner,** and **SDF Size** will have an unused W parameter.)

Bind Data Baking

1. Make sure the avatar is in the exact same position for this bake as it was for the SDF bake, and set the avatar's root GameObject as the **Target Object.**

2. Choose which renderer you want to bake the bind data for. Unlike the SDF, where multiple meshes can be combined, you need to bake a separate set of bind textures for each renderer in the avatar that you're going to use the material on. This also means that each renderer needs to have its own version of the material, with different bind textures set.

3. If you want to define subregions on the avatar with different lighting or flow directions for the lava, you can supply a **Subregion Mask Texture** to specify what parts of the model belong to which subregion. (See the example on the next page for reference.)

    - This texture never actually gets put on the avatar directly, but will be applied using the mesh's normal UV mapping during the baking process.

    - Each subregion on the **Subregion Mask Texture** should be colored in with a solid color. Which color should correspond to which subregion is controlled by the **Subregion 0-15 Mask Color** settings.

    - There is also the **Excluded Region Mask Color,** which you can use to specify areas of the model that shouldn't be rendered at all. This can potentially improve performance by not doing work in areas where the lava lamp wouldn't be seen anyway.

    - For consistent results, try to choose subregion colors that are distinct. If the colors in the texture don't perfectly match any of the **Subregion Mask Colors,** the baker will try to pick the closest one.

    - Subregions are chosen on a triangle-by-triangle basis. If a triangle has multiple mask colors on it, the subregion it will be set to is essentially chosen randomly.

    - If the mask color changes right at the edge of a triangle, the wrong color might be chosen by accident. Paint a border around triangles that you need consistent results on.

    - If any corner of a triangle is covered by the **Excluded Region Mask Color,** the entire triangle won't be rendered regardless of what's on the rest of it.

    - If no **Subregion Mask Texture** is set, the entire model will be set to subregion 0 by default.

4. Hit **Bake** to generate the 3 bind textures. This will also display a scene debug visualization showing which triangles of the mesh belong to which subregion.

5. If you're happy with the subregions, click **Save Bind Data Textures** to export the 3 texture assets. They will be saved with the specified filename with "Positions", "Normals" and "Tangents" appended to the end. If you need to make any adjustments you can reset the baker instead.

6. Like the SDF texture, you can apply these textures automatically to any materials by adding them to the **Auto Update Materials** list, and clicking **Update Materials.** To manually apply the bind textures, go to the **Mesh Data** tab on the material and drag the textures into the **Bind Positions Texture, Bind Normals Texture,** and **Bind Tangents Texture** fields respectively.

Example of Subregion Setup:

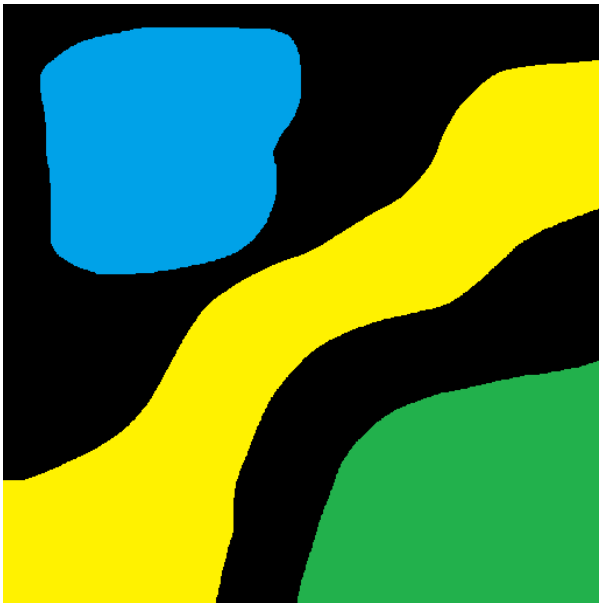

Fig 1: An example **Subregion Mask Texture**

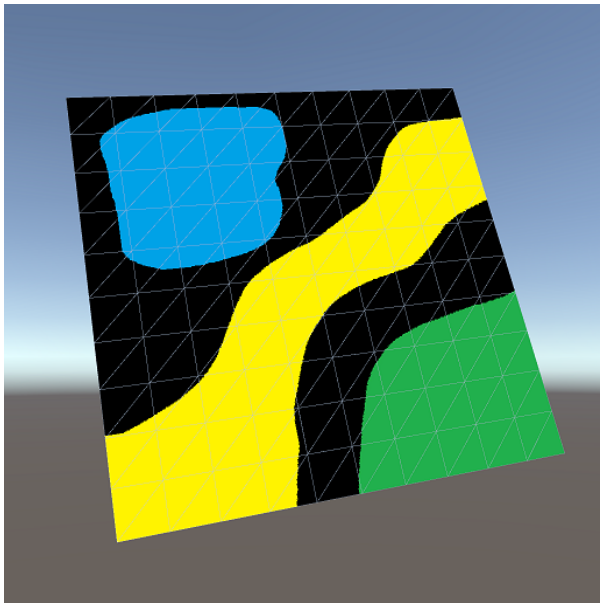

Fig 2: The mask texture applied on a mesh with an unlit material (note: this is purely for visualization purposes, you don't need to put the mask texture on the mesh itself)

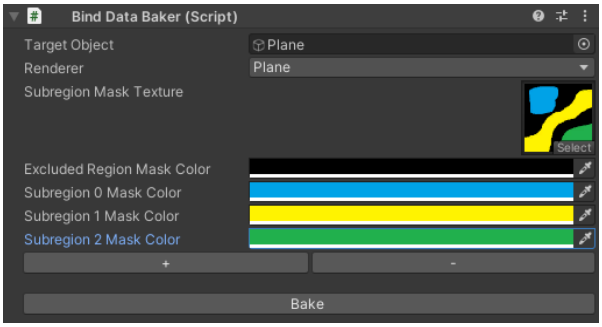

Fig 3: Example **Bind Data Baker** settings using this mask texture

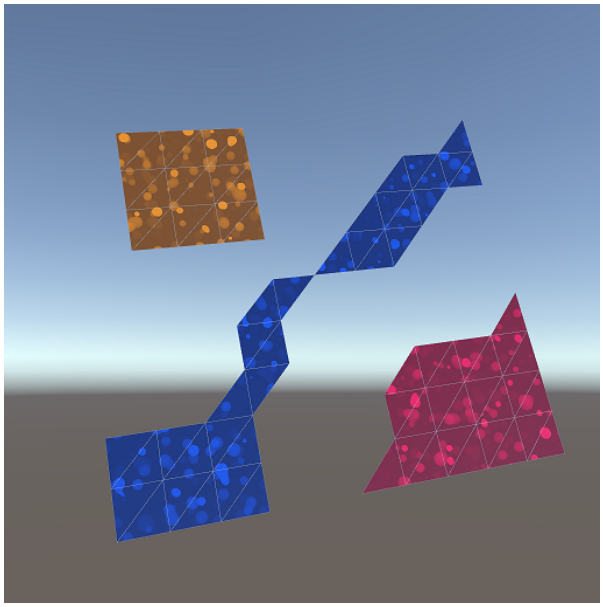

Fig 4: The resulting subregions after baking

Using The Material

Putting the Material on Your Avatar: You'll likely only want to have the lava lamp material on certain parts of your avatar, and have other parts rendered with a more conventional material. Ideally, if you have some experience with modeling software, you should do this by cutting the model into multiple different submeshes. Otherwise, the easiest way to do this is to duplicate the renderer, put the lava lamp material on the copy, and use alpha clipping on the original material to expose the lava lamp parts underneath. You can use the **Depth Offset** parameter on the lava lamp material to prevent z-fighting between the two renderers.

Deformation: The lava lamp shader can only handle so much deformation of the model due to animation before things start rendering incorrectly. For this reason, it's recommended that you cover the joints of the avatar with a different material.

Multiple Renderers: When using this material on an avatar with multiple renderers, you will need to bake the **SDF** and **Bind Data** while all the renderers are in their appropriate places on the model, so that the material can understand where each renderer is relative to each other. You will also need to make a separate copy of the material for each renderer, even if all their parameters are the same, because while the **SDF** can be shared between multiple renderers, each renderer will have its own **Bind Data** which needs to be set on its respective material.

Subregions: The material can have between 1 and 16 lava subregions. Each of these subregions can have different flow directions, lava and water colors, lighting, blob sizes, and top and bottom lava reservoir heights. This lets you, for instance, have the lava in the avatar's torso look different from the lava in the forearm. The number of subregions is controlled by a slider at the top of the **Lava Subregions** tab on the material. Changing this value recompiles the shader so it can't be changed at runtime.

Important Settings

World Rescale: If you change the size of the avatar after baking, certain features won't render correctly (specifically **Refraction** and **Depth Intersection.**) To fix this you can use the **World Rescale** parameter, found under the **Mesh Data** tab on the material. Set this to whatever the scale of the avatar is relative to what it was when you baked it. So if you tripled the size of your avatar, set the **World Rescale** to 3.0. If you baked your avatar at 0.8x scale, then shrunk it to 0.4x scale, set the **World Rescale** to 0.5. Non-uniform rescaling is not supported.

Min Thickness: If you want to render lava inside of something that is physically too thin to actually have lava inside of it, you can use this parameter, found under the **Volume Data** tab, to set a minimum distance that lava will be rendered to. If you set this setting beyond the actual maximum thickness of the avatar, you can just remove the **SDF Texture** from the material to improve performance. If you are using a **SDF Texture**, it's recommended you set this to 0.

Padding: This parameter is found at the top of the **Lava Shared Properties** tab, and inserts some space between lava blobs. Increasing this setting will improve performance and increase this distance that the lava can render to, at the cost of making the lava blobs smaller.

Glass

Reflectiveness: Controls how reflective the glass is.

Roughness: PBR roughness. If a roughness texture is set, you can specify a minimum and maximum roughness range for the glass. To use a smoothness texture instead of roughness, flip the minimum and maximum values. If no texture is set, the slider will let you set a uniform roughness for the entire surface directly.

Normals: Allows you to specify a normal map for the glass, with a slider to control its strength.

Tint: Allows you to tint the color of the glass using a texture and/or a uniform tint, which are multiplied together.

Refractive Index: Controls the index of refraction of the glass, which will distort the lava, as well as the background behind it if the **Transparent Background** setting is enabled.

Background Color: If the **Transparent Background** is enabled, this color will be alpha blended over the background. If the lava lamp material is opaque, this color is used for the background without alpha blending.

Reflection Probe Override: Enabling this setting will replace the standard cubemap reflections from the current scene with the specified custom cubemap. When using custom cubemaps, be sure to set the cubemap's convolution type to "Specular (Glossy Reflection)" and enable mipmaps on it.

Lava Shared Properties

Padding: (see the entry under the Important Settings heading)

Smoothing: Controls how much the blobs of lava blend into each other when they touch. At lower values the blobs will be larger, but more rigid. At higher values the blobs will appear smaller but can stretch and melt into each other more. At high values this setting may cause visual artifacts so it is recommended to leave it at its default value.

Vertical Spacing: Controls the average spacing between blobs in the direction of movement.

Invisible Blob Chance: Controls what percentage of lava blobs should be skipped and not rendered. Useful for preventing distribution of blobs from appearing unnaturally uniform.

Size: Lets you control the range of possible sizes a lava blob can be, and adjust the curve of the distribution of between those sizes. When the distribution is 0, all sizes are equally likely. A higher distribution value will bias the average size towards larger blobs, while a lower value will increase the probability of smaller blobs. The maximum possible size of a blob is also limited by the **Padding** and **Smoothing** settings.

Speed: Controls the range of speeds that blobs will travel in the **Flow Direction.** Half of the blobs will flow in the opposite direction, but at the same speed.

Drift Speed: This setting controls the range of speeds that blobs will drift back and forth perpendicularly to the **Flow Direction**. The larger the blob is, the less it will drift. The drift speed is a multiple of a blob's normal **Speed.** (i.e. at a drift speed of 1, small blobs will drift horizontally as fast as they are moving vertically.)

Rendering: Controls the global appearance of the lava material (other than color and lighting.) Reflectiveness and Roughness allow you to set the PBR parameters for rendering specular highlights. Soft Depth Intersection Size controls how wide the transparent fadeout should be when lava hits the far side of the glass, or when an object is inside of the lava and **Lava Depth Intersection** is enabled. Touching Side Blend Size controls how sharp the edge where the lava meets the near side of the glass should be, with larger values making it look smoother. Increasing Touching Side Blend Size too much can negatively affect the lighting quality of blobs near the side of the glass.

Lava Subregions

Scale: Adjusts the overall scale of the lava blobs for this subregion.

Lava Reservoirs: Controls the position of the large reservoirs of lava at the top and bottom of the lamp that blobs of lava come out of and meld into. The Height is relative to the root position of the avatar when bind data is baked, or relative to the pivot of the mesh when it isn't.

Lava Color: Controls the diffuse color of the lava. Core Color is the color the lava will be at the thickest points, such as in the center of large blobs. Edge Color is the color lava will be at thinner points, such as smaller blobs and the edges of large blobs. Transition Thickness Scale controls how thick the lava needs to be to transition to the Core Color. Higher values will cause the Core Color to appear more strongly in thinner parts of the lava. The reservoirs will always be the Core Color.

Water Haze: Simulates the cloudy appearance of fine particles being suspended in the water. The haze will be lit by the top and bottom lights, just like the lava. Color will control the albedo color of particles, while Density will control their concentration.

Water Tint: Simulates the appearance of colored dye in the water. Color controls the color of the dye (with white being clear), while Strength controls its concentration.

Lights: Controls the internal lighting for this subregion. The Top Light shines light down from above (relative to the flow direction) while the Bottom Light shines light up from below. The Height settings control where the light starts for the purposes of distance falloff, while the Light Falloff Scale controls how quickly the lights fall off from their start points. Anything above the Top Light Height or below the Bottom Light Height will always be lit by the respective light at maximum intensity. A Light Falloff Scale of 0 means no falloff. These lights are not actual Unity lights, they won't affect anything outside of the material and don't have the associated performance cost of adding real lights to a scene.

Flow Direction: Controls the direction the lava flows in this subregion as a 3D XYZ vector pointing in the direction of movement. (The W component is unused and only exists because Unity does not support Vector3 parameters for shaders.) The direction is relative to the model's orientation when the **Bind Data** was baked. If there is no **Bind Data,** then the direction is in the object's local space. The direction vector does not have to be normalized.

Advanced Options

Culling Mode: This lets you choose to render with triangle backface culling, frontface culling, or no culling.

Depth Offset: This parameter allows you to move the surface ever so slightly closer or further away from the camera in order to prevent z-fighting. Setting this to -1 pulls the surface closer to the camera, and +1 pushes it further away.

Max Specular Highlight Brightness: Sets an artificial limit on the brightness of specular highlights. While normally this wouldn't be desirable, in some worlds with poor lighting configurations very bright highlights might cause flickering or excessive lens flares. If you need to use this setting, it's recommended that you leave the value above 1 (and ideally as high as possible), otherwise specular highlights may appear unnaturally faint in scenes with properly configured lighting. Setting this to a negative value will disable the brightness limiting entirely.

Surface Lighting: This toggles whether or not the glass on the lava lamp's surface will be lit by Unity's lights. Disabling this can improve performance since additional lighting passes will no longer be rendered.

Transparent Background: If this is enabled, the lamp will be transparent and refract what is behind it. This is automatically disabled when the material is not in the transparent render queue. Enabling this setting will make Unity perform a "grab pass" to copy the background to a texture, which can affect performance.

Lava Depth Intersection: If this is enabled, opaque objects will be able to intersect with the lava. Otherwise, anything inside of the lava lamp will always appear to be behind it. This is automatically disabled when the **Transparent Background** is disabled. This will also be disabled in scenes that do not have a readable depth buffer. (The depth buffer will be readable in worlds with shadowcasting directional lights, certain post processing effects, or if it is explicitly enabled by the world creator.)

Write Depth: When the material is in the transparent render queue, this controls whether or not it will write to the depth buffer. It's recommended you leave this on unless you have a specific reason to disable it.

FAQ

“How will this shader impact performance?”
This shader is pretty heavy and may cause framerate dips on lower end machines. The biggest factor affecting performance is just going to be the number of pixels getting rendered. The total number of renderers using this material isn’t going to be as important as avoiding overdraw, so try not to use the material on parts of the model where it won’t be visible anyway, or block those spots with another opaque material. Ideally, set the render queue so that the lava renders after most other opaque materials, which will prevent pixels from being rendered unnecessarily. In order to lower the cost of the pixels that are rendered, try decreasing the **Min Thickness** (if applicable) and increasing the subregion **Scale,** both of which will reduce the number of blobs of lava that need to be evaluated, improving performance. Lowering the resolution of the **SDF Texture** and increasing the lava **Padding** will also both make the algorithm used to evaluate those lava blobs more efficient. A rough guideline for the overall cost of rendering a pixel is:

((Thickness / Subregion Sale) / Padding) + (Thickness / SDF Pixel Size)

“Past a certain distance the lava degrades into a glitchy looking wall. / There are miscolored rings around the lava blobs.”
You can fix this by turning up the **Padding** parameter.

“Inside of the material, the entire thing looks like one solid color rather than there being individual blobs.”
Go to the **Lava Reservoirs** tab under the appropriate subregion properties, then raise the **Top Height** and/or lower the **Bottom Height** values.

“In certain worlds with post processing effects like bloom, the shader is extremely bright and/or flickering.”
If all of the lava is too bright, try lowering the intensity of the **Top and Bottom Light Colors** in the appropriate subregion properties. If the HDR intensity of the light colors is above 0, there may be clipping or issues in worlds with poor lighting configurations. If the problems are specifically with the specular highlights, you can use the **Max Specular Highlight Brightness** setting to artificially limit their brightness, which should alleviate these issues.

“Why can’t I see transparent lava lamp materials through each other?”
The material technically isn’t transparent, it uses a “grab pass” to get the background as a texture, which allows for refraction and tinting. Getting this texture has some performance overhead, so it’s only done once when the first transparent lava lamp material gets rendered. This means anything else transparent drawn after that won’t show up through any of the lava lamps, including the other lava lamps themselves.

“I want the lava to be a flat color instead of being lit. How can I do this?”
Under the **Lava Shared Properties** tab, set **Reflectiveness** and **Roughness** to 0. Under the appropriate subregion properties, set **Top Light Color** to white, **Bottom Light Color** to black, and **Light Falloff Scale** to 0.

“I don’t want to see any lighting or reflections on the outer surface of the lamp. How can I do this?”
Under the **Glass** tab, set **Reflectiveness** to 0 and **Roughness** to 1, or alternatively, enable the **Reflection Probe Override** and set it to an all black cubemap. Then under **Advanced Settings** uncheck **Surface Lighting.**

“Do I need to bake bind data if I just want to use this shader on a simple prop?”
If you don’t set any of the bind textures, the lava will just be oriented in the object’s local space. This should be fine for an object with a single non-animated renderer. If you want to use a SDF without baking bind textures, set the renderer’s scale to 1 and rotation to 0 before baking.