Potions with Liquid Simulation

V3.0 - July 2018

Index

New Features in v3.0

- Improved pouring location accuracy - users can now use a mesh at the bottle opening to simulate pouring locations with vertices generated at run time
- New absorber object with greater accuracy/ visual integrity on liquid surface in V2 potions
- New user asset outline scene with every asset laid out clearly for testing/ preview
- Mobile (legacy) versions of each bottle added to prefab folder to save setup time
- General project tidying for further clarity in naming conventions/ directory
- Updated documentation to provide greater first time use guidance, clarity on functionality and breakdowns of updated content

Potions with Liquid Simulation - Overview

This asset pack has been developed to allow you to easily integrate breakable potions with liquid simulation into your Unity project. Taking the prefabs we have developed you will be able to quickly drag and drop these potions into your project as environmental props, or interactable items for players to collect.

This pack is also designed to allow you to easily create your own potions making use of the shaders and scripts included.

Assets included in this pack:

- 8 potion bottle designs, set up for your use and to demonstrate functionality of pack
- Easy to use standard and mobile liquid shader - allows users to define liquid level possesses a texture channel to add detail to liquid surfaces as well as emission, smoothness and metallic sliders/ colour pickers
- Mobile friendly shader for liquid meshes
- Mobile friendly glass shader
- Glass shader with metallic, gloss, texture and normal channels
- A ‘liquid volume animator’ script allowing users to simulate approximate liquid physics and define the behavior of the liquid within the predefined parameters
- A cork mesh and texture
- Bottle smash particle effects and broken glass meshes for each bottle
- A bottle smash script that allows users to manage a bottle smash event, determining which particle effects play at the time of breakage and how the liquid puddle renders
- A workshop demo scene filled with assets

What can you do with this pack?

- Stylized liquid physics simulation
    - Control liquid ‘viscosity’, gravity direction and other simulation properties
- Pouring functionality
    - Pouring flow rate, number of liquid particles, color mixing properties and absorption method
- Animated liquid surface textures (foam, ripples, etc…)
- Bottle and liquid smash event simulation (on impact/ at max velocity)

Scene Overview

In this pack you will find a number of scenes. The purpose of these scenes is to provide examples for you in case you are having trouble with setting up your own potions. Please see the scene overview descriptions below to find the scenes that are most relevant to your needs.

Workshop Demo
An interactive scene that demonstrates how the bottles can work within a game environment. Pick up potions, drop them around the scene and watch them smash against the furniture. Sloshing potions around will also demonstrate the liquid ‘physics’.

Potion Demo
The potion demo allows users to demonstrate the liquid ‘physics’ and smashing interactions close up. This is the best place to see the pack in its entirety.

Transparency Demo
The transparency demo displays how users can set liquid transparency and distortion/ what that looks like when the liquid volume is looked through.

Pouring Demo
The pouring demo shows how one liquid volume can be tipped and poured into another. It also demonstrated how liquids can blend together in one volume, mixing colour.

Legacy Demo (mobile potion shader/ glass shader)
The legacy demo is a static scene that demonstrates how the legacy shader looks and contains a premade legacy bottle for users to look at the setup of.

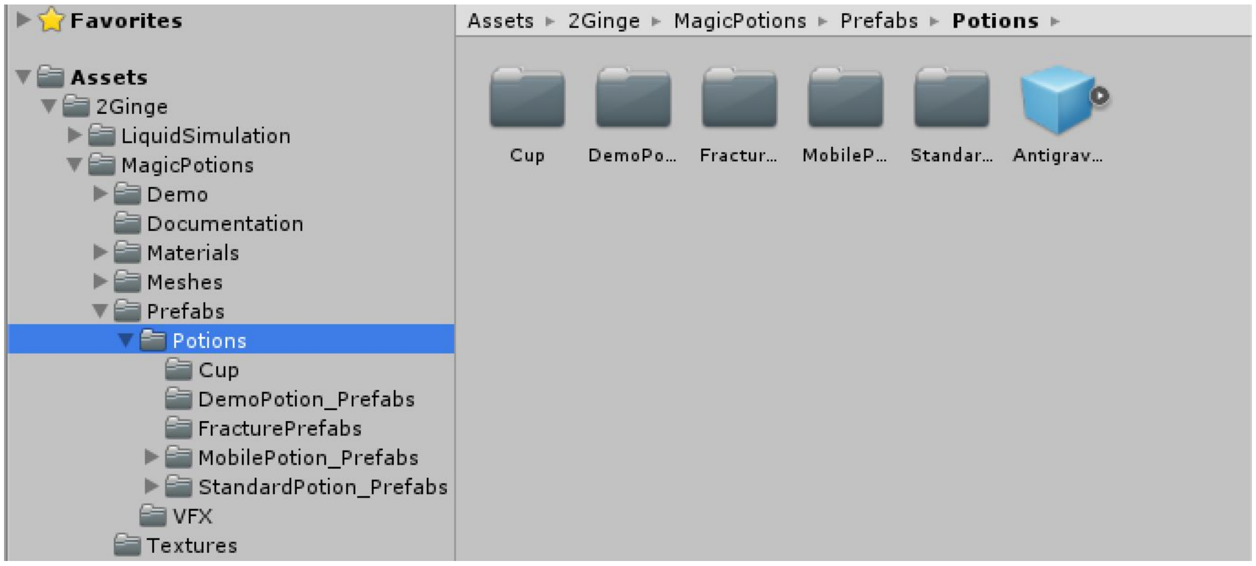
Animated Texture Demo
The animated texture demo shows users how they can set up an animated texture on the surface of liquids using coffee foam as an example.

Asset Scene
This scene is simply a space used to demonstrate all potion types included in the project. It shows mobile and standard potion variants utilising the initial pouring and absorption method and the revised version 2 pouring and absorption method for greater accuracy.

First Time Users

Where to find the potion prefabs?

You can quickly import prefab versions of all the potions in the pack from the 'potions' folder under the 'prefabs' folder in the 'MagicPotions' parent folder.



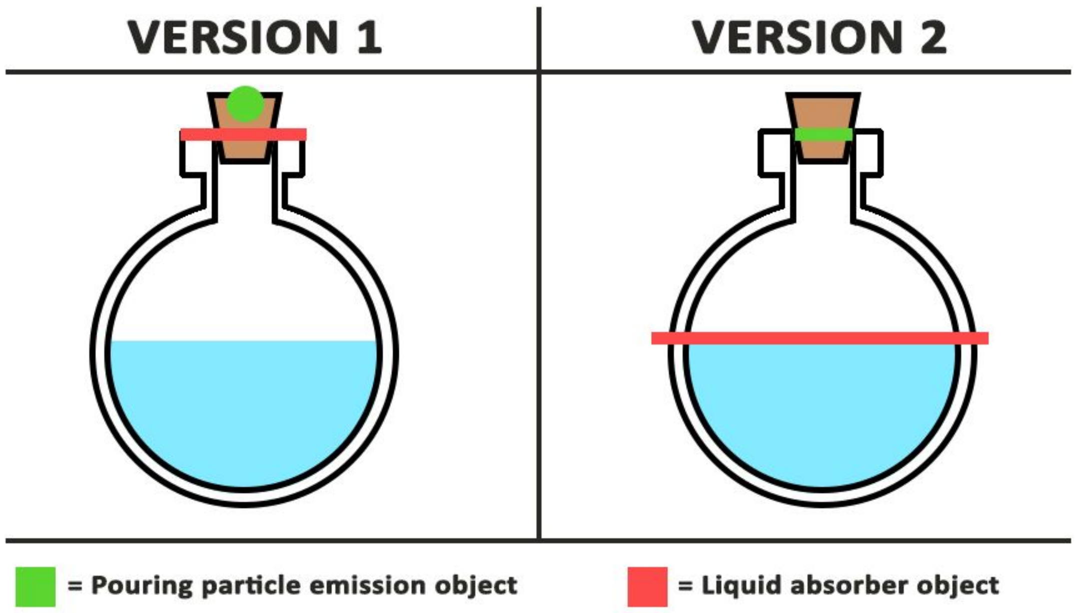
Here you will find prefabbed versions of the mobile and standard (PC) potions with pouring v2 and v1 variants for each. You will also find versions with and without potion labels.

Mobile potion prefabs make use of a simplified liquid shader, legacy glass shader with a texture channel and mobile diffuse shader for the cork.

Standard potion prefabs make use of the fully featured 'liquid_potion' shader with a greater number of display controls, such as emission, volume fade settings, distortion settings and metallic and smoothness properties. Standard potions also use the 'Glass_Potion' shader, which has fresnel, metallic and gloss properties and allows for texture and normal channels..

Version 1 or version 2 potions?

There are two versions of potion prefabs in this pack. Their differences may appear subtle, but there are significant functionality differences. Essentially, the version 2 potions are an improved version of their earlier counterparts. First time users are advised to use the version 2 prefabs as templates, but existing version 1 users may wish to continue using the earlier potions for consistency. See the table below for an outline of the core differences. Version 1 potions are slightly more performant as they require less runtime calculation during their pouring and absorption events.

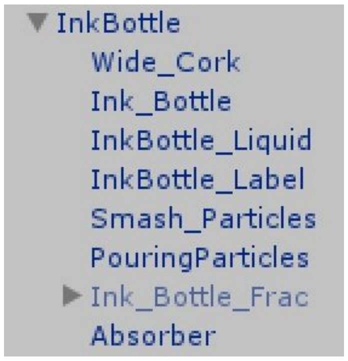

VERSION 1
VERSION 2
🟩 = Pouring particle emission object
🟥 = Liquid absorber object

• Pouring emission object is a simple particle emitter and does not take into account the angle of the bottle • Absorber is a static box collider that sits at the neck of the bottle • Particles cannot enter bottle volume and bounce off unrealistically • Potion parent contains rigidbody component to handle physics	• Pouring emission object is a mesh that generates vertices from which particles emit at runtime based on bottle rotation (more realistic) • Absorber is dynamic mesh that scales and moves with liquid surface level • Particles enter bottle neck and collide with absorber at surface level • No rigidbody on parent object, rather uses script to generate collider and rigidbody at runtime

Creating your own potions

When creating your own bottles, it is important that you identify which version of the potion functionality you would like to use. We suggest using a reference prefab from your chosen version to help you get started, but have included a short guide below to help you identify which elements you will need when creating your own potions.

Create your own version 1 Potions


InkBottle
Wide_Cork
Ink_Bottle
InkBottle_Liquid
InkBottle_Label
Smash_Particles
PouringParticles
Ink_Bottle_Frac
Absorber

1. Starting with the parent object, create an empty gameobject and add the ‘BottleSmash.cs’ script to it. This script is a management script that keeps track of all the bottle elements. After this add a rigidbody and the ‘PourLiquid.cs’ script too.

2. Next add your bottle mesh and zero out its position below the parent object. It is best to ensure that your bottle mesh has a perfectly centered pivot when you export it from your 3D modelling software. Add a mesh collider to your bottle mesh. This collider is used to detect impacts when the bottle is dropped. Once you’ve done the above, assign the object to the ‘container’ layer.

3. Now add your liquid mesh. We suggest duplicating your bottle mesh, removing the outside faces and shrinking the inside faces ever so slightly so it fits inside your bottle mesh. Align the center of your liquid mesh to the same position as your bottle mesh in your 3D modelling software. Once your liquid mesh is present and a child of the overall bottle object, add the ‘LiquidVolumeAnimator.cs’ script to it. Here you can also add the ‘LiquidColor.cs’ script, which will help unify the color of your liquid volume when it is mixed with another liquid. Once you’ve done the above, assign the object to the ‘liquid’ layer.

4. Next add your shattered bottle meshes. This is simply a bunch of broken glass pieces with mesh colliders and rigidbodies that will be turned on when the bottle break event takes place. We suggest creating a parent object to hold all the meshes, then adding each mesh within the bottle volume with their accompanying rigidbodies and colliders. Once you’ve set these objects up, turn off the parent object, as the ‘BottleSmash.cs’ will handle turning it on when it is needed.

5. Now it is time to add the absorber object. This is simply a mesh placed at the opening of your bottle with a collider attached that will detect when liquid particles are colliding with it. For this reason, it will need a mesh collider, a rigidbody and you will need to add the ‘LiquidAbsorption.cs’ script. Once you’re done, assign this object to the ‘absorber’ layer.

6. The next most important object to the overall functionality of the potion is the pouring particles. These are the particles that stand in for real liquid when the bottle is tipped without the cork in place. We suggest placing the particle system right at the opening of the bottle, above any colliders that may cause issues however, such as the absorber or bottle collider. You can add whichever properties you like to the actual particle system itself, but for the pouring and absorption to work correctly, the particles will need to have collisions enabled. Some particle properties will be overridden by scripts attached to other objects (flow rate, color, etc…), but you don't need to worry about that here. The last step is to attach the 'ParticleColor.cs' script.

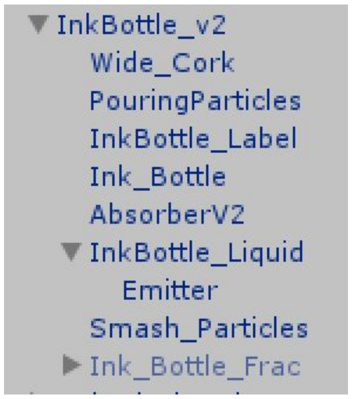
7. This step is optional, but we recommend including a particle system to simulate liquid splashing about during the bottle smash event. We call this particle system Smash_Particles. For color unification reasons, add the 'PS_Color.cs' script to the particle system.

8. When the bottle smashes, if you would like a 'puddle' to render you will need to create a 'splat' prefab. We created a simple splat looking mesh that spawns parallel with a 'ground' object. You will need to place the 'liquidpool.cs' script on this object and set the properties for the scale and time of splat display. Any objects you want to render the splat on must be set to the ground layer.

9. Finally you should add a cork object, which if present, will stop the flow of the pouring particles if the bottle is rotated with liquid in it. Add a convex mesh collider, but turn it off for now, as it will be turned back on during the smash event by the smash script. A rigidbody will also be useful so the cork can manage its own collisions once the smash event takes place.

If you've made it this far, now it is time to go through and hook up all of your potion elements to the relevant scripts. The way the components are laid out is pretty straightforward, so in most cases you should be able to see where to drag and drop the appropriate meshes, transforms and prefabs that the components require. If you are stuck, use one of our prefab potions to guide you.

Create your own version 2 Potions



V2 potions are structured in a similar way to the v1 potions, however there are significant differences in the way the absorbing, pouring and collision detection works. As such, you can follow the V1 instructions for set up, but when it comes to the parent object set up (step 1), pouring particles (step 6), and absorber (step 5) set up stages, please refer to this section instead.

Absorber
The main differences between the v2 and v1 absorber are outlined in the section above titled 'Version 1 or version 2 potions?'.

To set up a v2 absorber from scratch you will need a plane with a mesh collider, rigidbody set to 'is kinematic' and the 'LiquidAbsorptionV2.cs' script attached. Turn off the mesh renderer and assign the object to the 'absorber' layer.

Pouring Particles and Emitter
The pouring particles themselves can remain the same as the set up described in the v1 guide above. For the updated pouring to work as intended, you will need to create a custom 'emitter' object to sit at the opening of the bottle. Essentially, this object provides vertices for the 'MeshLiquidEmission/.cs' script to access and generate pouring locations from based on the overall bottle rotation. To do this you can create a custom mesh, or use the circle mesh we have in the pack. As mentioned before, add the 'MeshLiquidEmission/.cs' script as a component of this 'emitter' object and hook up the relevant references, set the variables as you like and you're good to go.
Do note that the cork object is also required as a reference on the new emitter script.

Collider [PAY SPECIAL ATTENTION HERE]
The collider set up for the V2 potions is necessarily quite complex. We wanted to include functionality for pouring 'into' the V2 bottles, rather than having an absorber sit at the neck. This meant that convex colliders on the bottle would no longer be appropriate as they would block the particles reaching the surface (where collision is detected). Our solution to this has been to generate a 'ghost' collider at runtime. Please follow the steps below and refer to the FAQ section if you experience any problems.

1. Once you have your parent potion object set up, add the 'GhostPhysics.cs' component. It is very important that you do not have a rigidbody on the parent object at this stage (you may experience crashes if you do leave a rigidbody active).
2. In the liquid simulation prefabs folder you will find a prefab called 'DummyPhysics', you will need to reference this object in the 'Ghost Object' variable. This will generate a

collider at runtime which can handle all the collisions the bottle needs to experience, while avoiding collisions with the particles when pouring is happening.

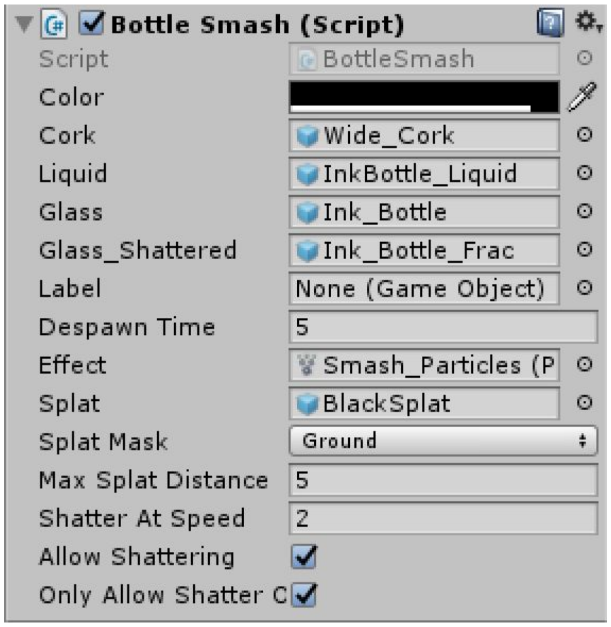
3. If you would like to quickly find the ghost physics object at runtime, turn on the 'redirect selection' bool. You will need to select the ghost physics object to move the potion at runtime. Once this is on, when you select the parent object, your selection will be directed to the ghost physics object instead. Allowing you to move the potion as normal.
4. Please note that the physics settings on your potions will be inherited from the rigidbody on the ghost object. So if you would like different settings for different potions, simply create a duplicate of the 'Dummy Physics' prefab, change the rigidbody as desired and reference that object instead.

Scripts Overview

Described below are the functions of the two core scripts within the asset pack. If you are unclear on the functionality of either of these scripts and their many components please do not hesitate to contact us for assistance. You can find our support contact details at the bottom of this document.

BottleSmash.cs

The bottle smash script handles the bottle smash event, keeping track of all the bottle elements before, during and after the smash event. 'Cork', 'Liquid', 'Glass' and 'Label' are all gameobjects to be destroyed. Cork is unique in that it will despawn after the despawn timer has completed, so be sure to have a disabled mesh collider with a rigidbody (with kinematic ticked) on it to take full effect.


Bottle Smash (Script)

Script	BottleSmash
Color	■
Cork	Wide_Cork
Liquid	InkBottle_Liquid
Glass	Ink_Bottle
Glass_Shattered	Ink_Bottle_Frac
Label	None (Game Object)
Despawn Time	5
Effect	Smash_Particles (P
Splat	BlackSplat
Splat Mask	Ground
Max Splat Distance	5
Shatter At Speed	2
Allow Shattering	✓
Only Allow Shatter C	✓

Color is a way of seeing the current overall color in the inspector, or overriding all other color settings.

Glass_Shattered is a disabled container full of the glass shards (each with their own rigidbody and mesh collider).

Despawn Time is how long parts of the mesh will stay around for (including the shattered glass).

Effect is the particle effect at the center of the bottle (splash effect essentially).
Splat is the mesh that gets instantiated on the ground.

Mask is the layer that the 'ground' exists on.

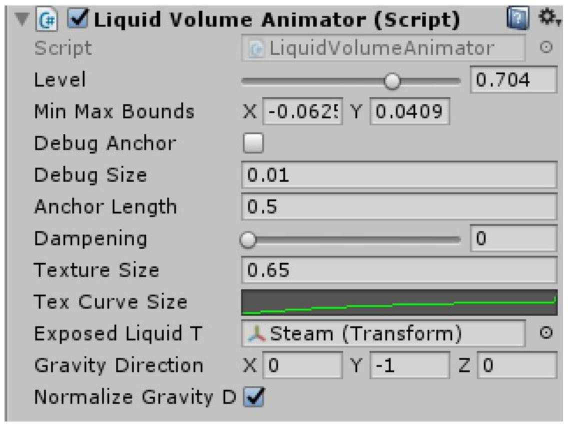
Splat Distance means that any ground that is greater distance than this, the splat will not occur.

ShatterAtSpeed is the speed at which the potion needs to be travelling in (subtracting its previous velocity) in order to 'break', think stopping very quickly or starting very quickly.

AllowShattering will disable any normal shattering logic (however it can still be shattered through code.

OnlyAllowShatterOnCollision means that there is a small window after colliding with another collider that the potion can shatter (0.2 seconds).

LiquidVolumeAnimator.cs



Liquid Volume Animator (Script)

Script	LiquidVolumeAnimator
Level	0.704
Min Max Bounds	X -0.062 Y 0.0409
Debug Anchor	☐
Debug Size	0.01
Anchor Length	0.5
Dampening	0
Texture Size	0.65
Tex Curve Size	
Exposed Liquid T	Steam (Transform)
Gravity Direction	X 0 Y -1 Z 0
Normalize Gravity D	✓

Level is the slider which fills or empties the mesh (0, being empty, 1 being full).

MinMaxBounds is used to debug if the volume is being approximated correctly.

Debug Anchor allows you to see the physicality of the liquid as a pendulum object.

Anchor Length will increase (closer to 0) and decrease (closer to positive infinity) the effect of rotation and movement on the bottle.

Dampening controls the intensity of liquid movement that will take place during the bottles rotation or transformation (1 will have no effect while 0 will have full effect).

Texture Size the size of the cutoff surface texture

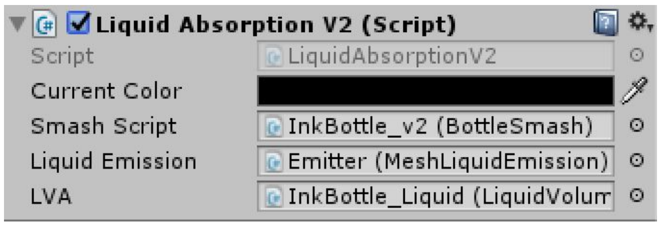
Texture Curve Size the texture curve graph allows users to set the texture size “keys” so that when the liquid volume empties and fills, the cutoff texture scales with the cutoff surface size at any level of fill. This can take some playing around with, but a good approach is set a start, middle and end key and fill in where scaling is not matching surface size where necessary.

Exposed Liquid T this is simply a place for you to provide the transform of any objects you want to sit on and stay with the surface of the liquid as it scales. For example, in the case of a steaming cup of coffee, you may attach a steam particle system transform here.. In the case of v2 potions, we use this place to provide the transform of the v2 absorber. But you can use it however you like assuming you don’t want the absorber v2 moving surface functionality.

Gravity Direction -1 = world down, 1 = world up, 0 = crazy rotation. Set the direction of “gravity” that the liquid references or more simply the direction the pendulum debug object ‘hangs’.

Normalize Gravity On/ off

LiquidAbsorptionV2.cs



Liquid Absorption V2 (Script)
Script LiquidAbsorptionV2
Current Color
Smash Script InkBottle_v2 (BottleSmash)
Liquid Emission Emitter (MeshLiquidEmission)
LVA InkBottle_Liquid (LiquidVolum

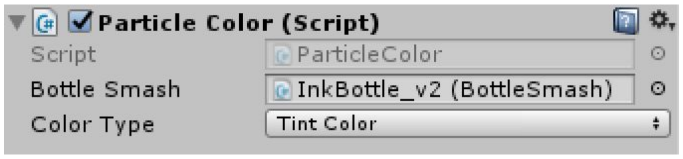
Current Color displays the current colour, which updates at runtime when liquids are mixed

Smash Script a reference to the smash script for the parent potion object

Liquid Emission a reference to the emitter object that is a child of the liquid object in v2 potions

LVA a reference to the liquid volume animator script attached to the potions liquid object

ParticleColor.cs

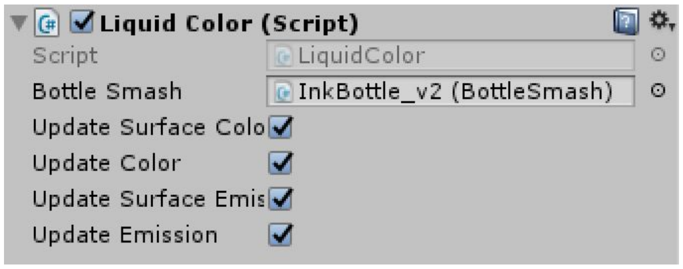


Particle Color (Script)
Script ParticleColor
Bottle Smash InkBottle_v2 (BottleSmash)
Color Type Tint Color

Bottle Smash a reference to the bottle smash script attached to the parent bottle object

Color Type tint or color

LiquidColor.cs



Liquid Color (Script)
Script LiquidColor
Bottle Smash InkBottle_v2 (BottleSmash)
Update Surface Colo ✓
Update Color ✓
Update Surface Emis ✓
Update Emission ✓

Bottle Smash a reference to the bottle smash script attached to the parent object

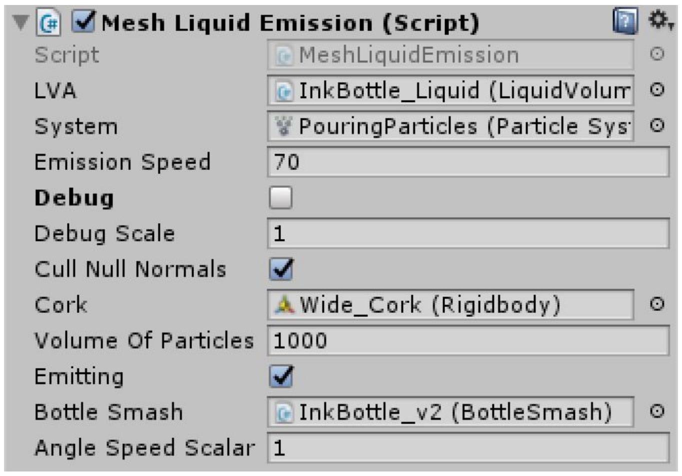
Update Surface Color when on, the liquid surface color will update when other liquids are poured into this potion

Update Color when on, liquid colour will update when other liquids are poured into this potion

Update Surface Emission when on, surface emission values will update when poured into

Update Emission when on, liquid emission values will update when poured into

MeshLiquidEmission.cs



Mesh Liquid Emission (Script)

Script	MeshLiquidEmission
LVA	InkBottle_Liquid (LiquidVolum
System	PouringParticles (Particle Sys
Emission Speed	70
Debug	☐
Debug Scale	1
Cull Null Normals	✓
Cork	Wide_Cork (Rigidbody)
Volume Of Particles	1000
Emitting	✓
Bottle Smash	InkBottle_v2 (BottleSmash)
Angle Speed Scalar	1

LVA a reference to the liquid volume animator on the liquid object

System a reference to the pouring particles

Emission Speed the rate at which the volume of particles will emit

Debug renders spheres at the vertices that are currently emitting pouring particles at runtime

Debug Scale sets the scale of the debug spheres above

Cull Null Normals if there is a candidate particle for emission and the emission mesh normal direction is 0 and the bool is checked, that emission will not take place. This prevents leaks.

Cork a reference to the cork object (if not present, liquid will flow when bottle is rotated)

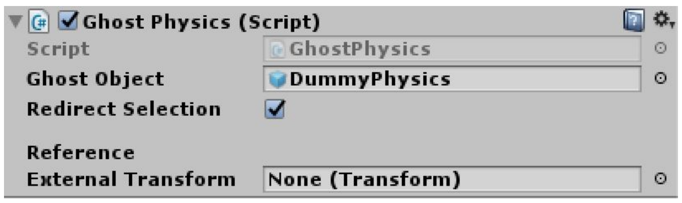
Volume of Particles the number of particles that will pour out when the bottle is rotated

Emitting are the pouring particles emitting? Yes/ No.

Bottle Smash a reference to the bottle smash script on the parent potion object

Angle Speed Scalar Scales the initial velocity of particles as they emit

GhostPhysics.cs



Ghost Physics (Script)

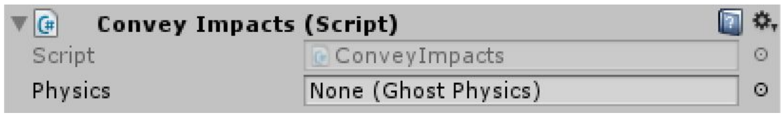
Script	GhostPhysics
Ghost Object	DummyPhysics
Redirect Selection	✓
Reference External Transform	None (Transform)

Ghost Object a reference to the prefab that is used to generate the ghost collider object at runtime

Redirect Selection turning this on ensures that when you try to select the parent potion object at runtime, you instead select the ghost physics object (prevents crash)

Reference External Transform this is filled at runtime to allow you to easily find the clone of the ghost physics object (can select while paused if redirect selection is turned on)

ConveyImpacts.cs

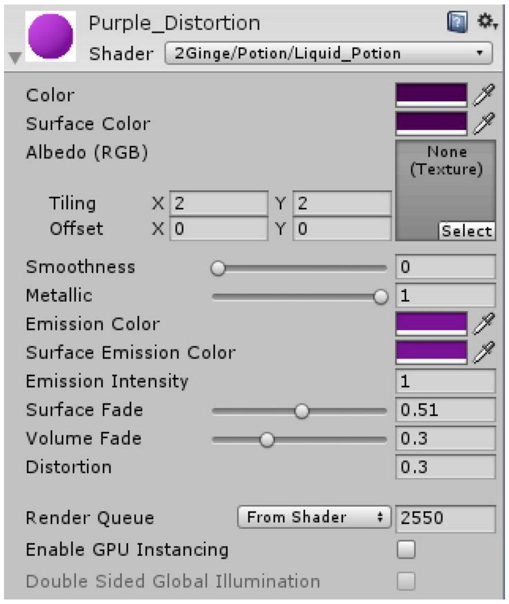

Convey Impacts (Script)

Script	ConveyImpacts
Physics	None (Ghost Physics)

Physics this is a reference back to the parent potion objects ghost physics script

Shader Overview

Liquid_Potion


Purple_Distortion

Shader 2Ginge/Potion/Liquid_Potion

Color		
Surface Color		
Albedo (RGB)	None (Texture)	
Tiling	X 2	Y 2
Offset	X 0	Y 0
Smoothness	0	
Metallic	1	
Emission Color		
Surface Emission Color		
Emission Intensity	1	
Surface Fade	0.51	
Volume Fade	0.3	
Distortion	0.3	
Render Queue	From Shader	2550
Enable GPU Instancing	☐	
Double Sided Global Illumination	☐	

Colour overall colour, can be overridden with code (displays current colour)

Surface Colour set the colour of the cut off surface independently from the remainder of the liquid volume

Albedo Liquid volume texture

Smoothness / Metallic set values as needed

Emission Colour if using emission, set colour for liquid volume here

Surface Emission Colour if using cutoff surface emission, set colour value here

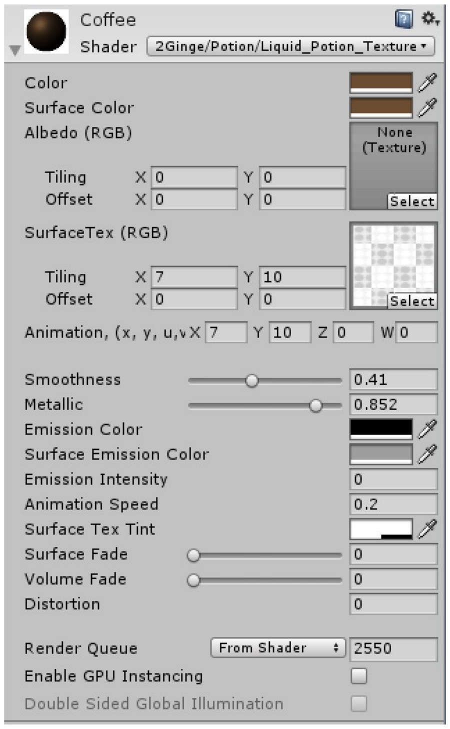
Emission Intensity set emission intensity for liquid volume total

Surface Fade set “opacity” of cutoff surface

Volume Fade set “opacity” of liquid volume (semi-transparent has subtle distortion on objects behind bottles when looking through liquid volume)

Distortion set the distortion value that determines how distorted objects appear through liquid

Liquid_Potion_Texture



Coffee		
Shader	2Ginge/Potion/Liquid_Potion_Texture	
Color		
Surface Color		
Albedo (RGB)	None (Texture)	Select
Tiling	X 0	Y 0
Offset	X 0	Y 0
SurfaceTex (RGB)		Select
Tiling	X 7	Y 10
Offset	X 0	Y 0
Animation, (x, y, u,v	X 7 Y 10 Z 0 W 0	
Smoothness	0.41	
Metallic	0.852	
Emission Color		
Surface Emission Color		
Emission Intensity	0	
Animation Speed	0.2	
Surface Tex Tint		
Surface Fade	0	
Volume Fade	0	
Distortion	0	
Render Queue	From Shader	2550
Enable GPU Instancing	☐	
Double Sided Global Illumination	☐	

This shader is effectively the same as the above, except that is possesses a channel for a cutoff surface texture, as well as animation values if the texture is animated (bubbles popping, foam, etc…). Below the unique elements of this shader are listed, see above for any standard liquid shader features.

SurfaceTex the texture to display on the cutoff surface (size set on liquid animator script)

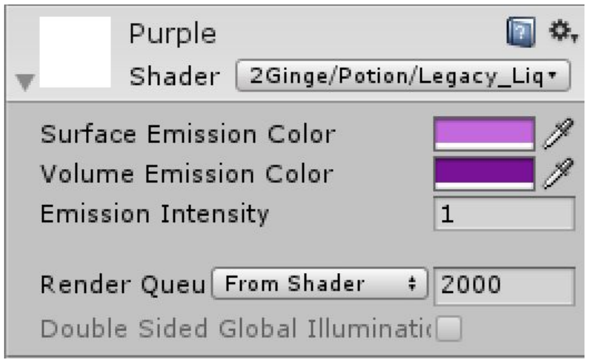
Tiling the tiling value for the cutoff surface texture

Animation the number of vertical and horizontal rows/ columns in the attached sprite sheet if texture is animated.

Animation Speed speed of cycling through animation frames

Legacy_Liquid_Potion

This shader is for use on lower end devices/ mobile and does not possess the surface texture elements, and additional maps/ variables that the more complicated 'Liquid_Potion' shader does.



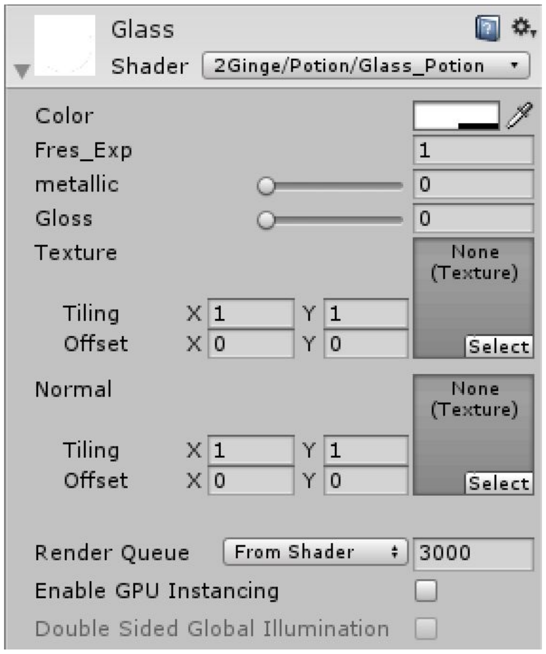
Purple		
Shader	2Ginge/Potion/Legacy_Liq	
Surface Emission Color		
Volume Emission Color		
Emission Intensity	1	
Render Queu	From Shader	2000
Double Sided Global Illuminati	☐	

Surface Emission Color simply the color of the cut-off surface emission

Volume Emission Color the color of the liquid volume emission

Emission Intensity the intensity of emission across the liquid volume and cut-off surface

Glass_Potion



Glass
Shader 2Ginge/Potion/Glass_Potion

Color
Fres_Exp 1
metallic 0
Gloss 0
Texture None (Texture) Select
Tiling X 1 Y 1
Offset X 0 Y 0
Normal None (Texture) Select
Tiling X 1 Y 1
Offset X 0 Y 0

Render Queue From Shader 3000
Enable GPU Instancing
Double Sided Global Illumination

Color Set the glass color

Fres_Exp the amount of variance in gloss/ metallic reflection based on viewing angle

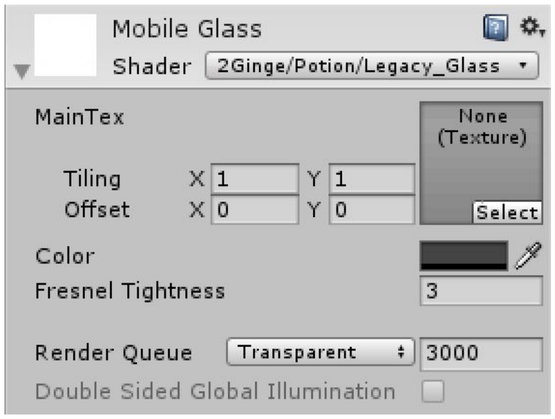
Metallic amount of metallic effect in glass

Gloss amount of gloss effect in glass

Texture texture channel for display on glass mesh

Normal normal channel for glass mesh

Legacy_Glass



Mobile Glass
Shader 2Ginge/Potion/Legacy_Glass

MainTex None (Texture) Select
Tiling X 1 Y 1
Offset X 0 Y 0
Color
Fresnel Tightness 3

Render Queue Transparent 3000
Double Sided Global Illumination

MainTex texture channel to display on glass

Color color channel for glass

Fresnel Tightness tightness of fresnel effect based on viewing angle (reflectivity)

F.A.Q

Mobile potions aren’t working correctly?

There could be many reasons for this, check the list below and get in touch if none of these apply to you:

- Ensure your build target is set to the platform you wish to build for (mobile ought to display correctly on most platforms, especially PC and mobile platforms)
- Make sure you’re actually using a potion with the ‘Legacy_Liquid’ shader applied to its ‘liquid’ object
- Make sure you’re actually using a potion with the ‘Legacy_Glass’ shader applied to the bottle/ glass mesh (same goes for shattered objects)

I want the pouring particles to go inside the bottle volume?

Try out the v2 potions if you would prefer the stylised pouring particles to collide with the surface of the liquid that is being poured into, no the bottle neck/ glass. If that’s the functionality you’re after, just make sure you use those potions, or setup your own potions following the hierarchy example set by the v2 potion prefabs.

Splat isn’t rendering after bottle smashes?

To ensure the splat renders correctly in your project, the object you would like the splat to render on needs to be tagged as ‘ground’.

Why are there two potion/ pouring/ liquid versions?

In the first two versions of this pack, the pouring, particle emission and particle detection (absorption) features were far more basic than we wanted them to be. Though they were performing fine for our users, we wanted to offer a higher quality version. V2 potions allow for greater accuracy in pouring and receiving of liquid between potions. The V2 potions are a bit more involved than the V1 potions and have a number of new elements and scripts. Please see the potion and script overview sections for more information.

Where are the V2 potion physics controls?

In V2 potions, the physics interactions are handled by a collider and rigidbody generated at runtime by the ‘GhostPhysics.cs’ script. This script references a prefab called ‘DummyPhysics’. If you would like to turn gravity simulation on, simply reference ‘DummyPhysics1’ instead, which has the gravity bool ticked.

Additional Help/ Contact

If you are having any issues in setting up your own bottles with the provided scripts and shaders, please watch the video demonstration which can be found on our Youtube channel. There you will also be able to find a video detailing the modelling practices put into place when creating the meshes required to create your own bottle assets.

Feel free to contact us with any issues you may be having via any channel. We are always happy to support our customers and will address bug fixes as soon as possible. Please do not hesitate to contact us with feature requests either! We'd love to continue to make our tools and assets better wherever possible.

Email: contact@2ginge.com
Website: www.2ginge.com
Twitter: @TwoGinge **|** @PezzSp **|** @JairMcBain

If you'd like to hear about our other projects and tools, please find our newsletter signup form at www.2ginge.com or check out our Unity Asset Store developer profile.