ZIBRA AI

Zibra Effects User Guide

ZIBRA EFFECTS IS A SOLUTION FOR THE UNITY ENGINE. IT ALLOWS THE USE OF GPU ACCELERATED REAL-TIME SIMULATED INTERACTIVE EFFECTS POWERED BY AI BASED OBJECT APPROXIMATION.

VERSION 2.1.0

Version comparison

Feature	Zibra Effects	Zibra Liquid	Zibra Smoke & Fire
VR support	✅	☒	☒
Liquid simulation	✅	✅	☒
Smoke & Fire simulation	✅	☒	✅

Note that you can import Zibra Liquid and Zibra Smoke & Fire into the same project, and get both simulations without VR support.
But you can’t mix Zibra Effects with Zibra Liquid or Zibra Smoke & Fire.

Table of contents

System Requirements

Required Unity version: **2021.3** or later (latest patch version is recommended)

Supported Render Pipelines: Built-in RP (BRP), URP, HDRP

VR Support:

- VR is only supported in Zibra Effects
- Liquids - Supported only in Unity Render mode
- Smoke & Fire - Supported only in Single Pass Instanced mode, except for the Effect Particles feature

Editor platforms:

- **Windows**
  - Supported Graphic APIs: DX11, DX12, Vulkan
  - Supported CPU Architectures: x64
- **macOS**
  - Supported APIs: Metal
    - Smoke & Fire requires a device with MTLReadWriteTextureTier2 support. Intel Macs without dedicated GPU will likely not satisfy that. All Apple Silicon macs do satisfy this requirement.
  - Supported CPU Architectures: x64, arm64

Please note that when using Zibra Effects in Editor, it verifies license on each Editor launch, which requires Internet connection.
If you are not connected to the Internet, you may not be able to use certain features in Editor. You can disconnect from the internet after license verification.
Player builds don’t have any license verification and will always work after build.

Build platforms:

- **Windows**
  - Supported Graphic APIs: DX11, DX12, Vulkan
  - Supported CPU Architectures: x64
- **UWP**
  - Supported Graphic APIs: DX11, DX12
  - Supported CPU Architectures: x64
- **macOS**
  - Supported Graphic APIs: Metal
  - Supported CPU Architectures: x64, arm64
- **iOS**
  - Supported Graphic APIs: Metal
  - Supported CPU Architectures: arm64
  - Supported OS: iOS 12 or later
  - Supported Devices: iPhone 6s or newer (Metal GPU Family Apple3 or higher)
  - Supported Xcode versions: 14.2 or newer
  - iOS Simulator is unsupported
    - You can run iOS build on Apple Silicon instead
- **Android**
  - Supported Graphic APIs: OpenGL ES 3.2, Vulkan
    - Smoke & Fire doesn’t support OpenGL, only Liquid will work when targeting OpenGL
      - OpenGL support for Smoke will be added in future release
  - Supported CPU Architectures: armv7, arm64
  - Supported OS: Android 7.0 or later
  - Meta Quest 2 & Quest Pro are supported in Zibra Effects Subscription

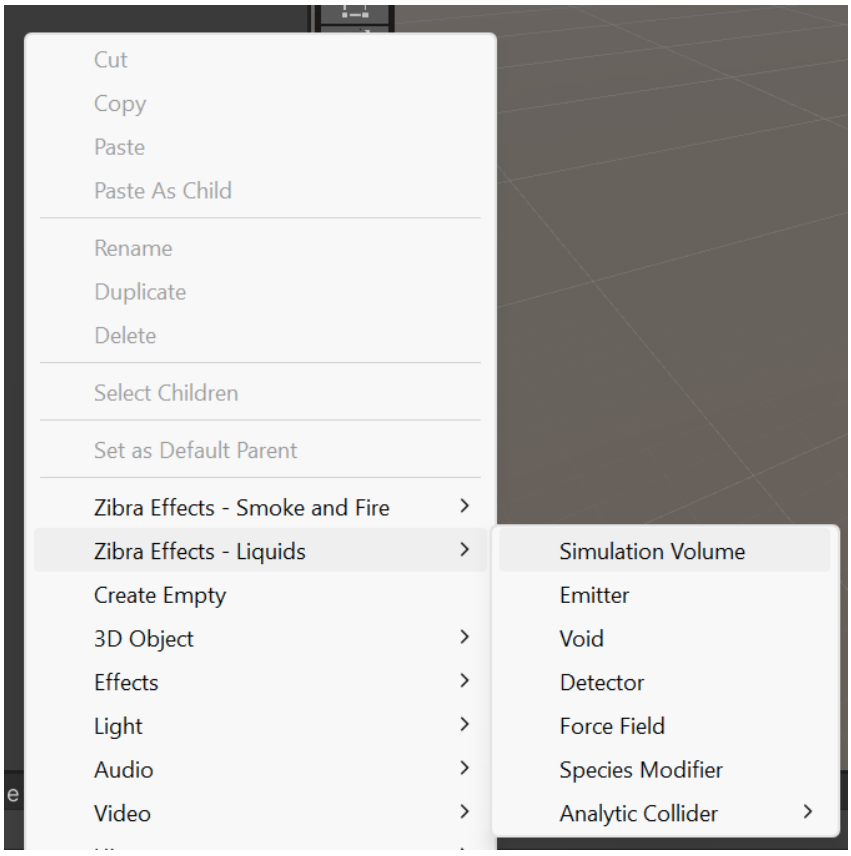
Liquid Setup

The first thing you need to do is to register your plugin.
After that you can start setting up liquid in your scene.

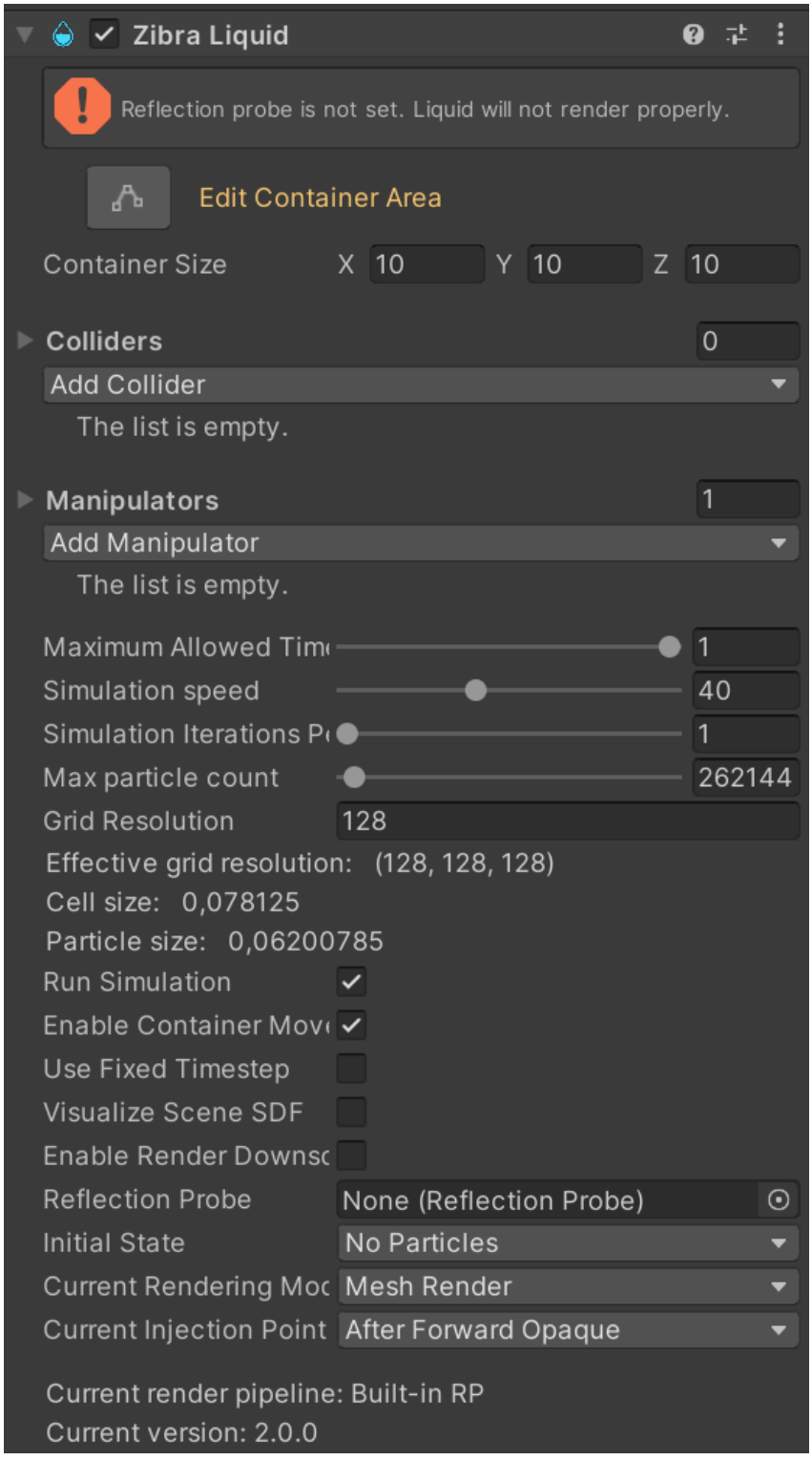
Creating Liquid Simulation Volume

To create a Liquid instance:

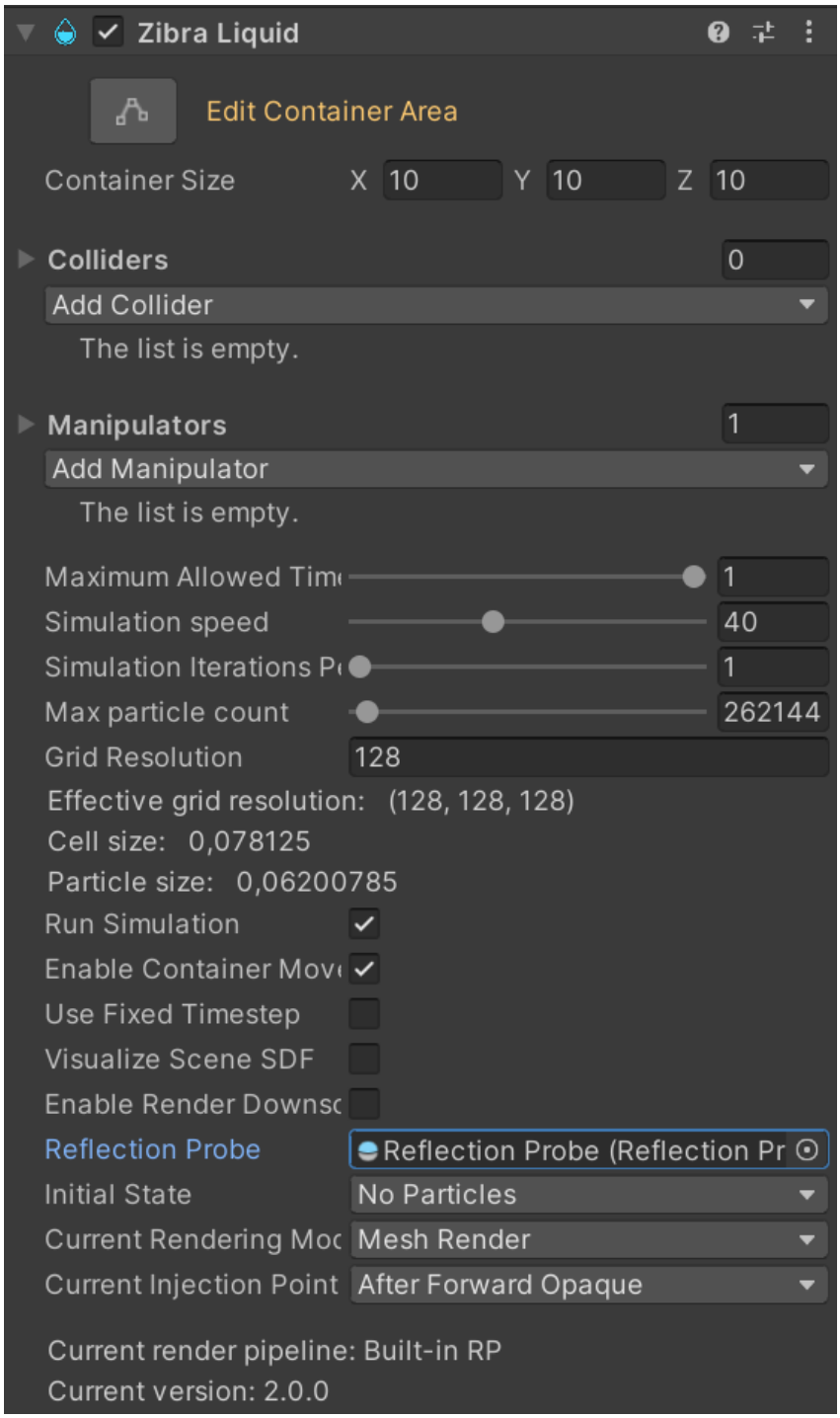
1. Right click in the Hierarchy window and select “Zibra Effects - Liquids → Simulation Volume”.



2. In the Inspector window, you'll see the Zibra Liquid parameters:



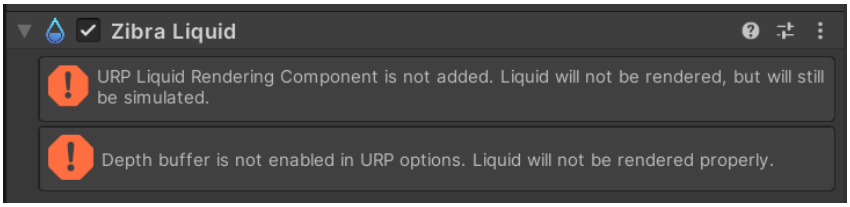
3. Set the “Reflection Probe” parameter. On BRP/URP it’s strongly recommended that you set this parameter, but is not strictly necessary. On HDRP it’s strictly necessary to set this parameter. You can check which render pipeline you are using by looking at the “Current render pipeline” message - on the screenshot the BRP is shown.



4. If you are using BRP, you are now ready to use your liquid. For URP/HDRP please proceed to Additional setup on URP or Additional setup on HDRP accordingly.

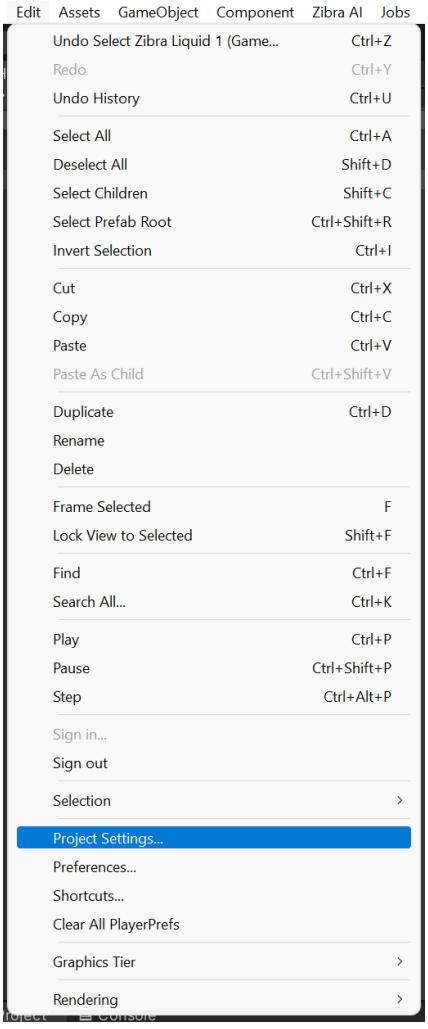
Additional setup on URP

If you are using URP you may see those errors:

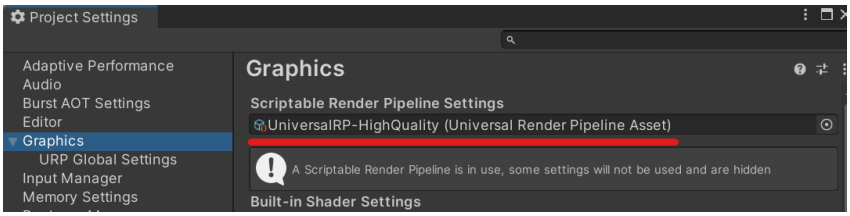


To add “URP Liquid Rendering Component” and fix first error:

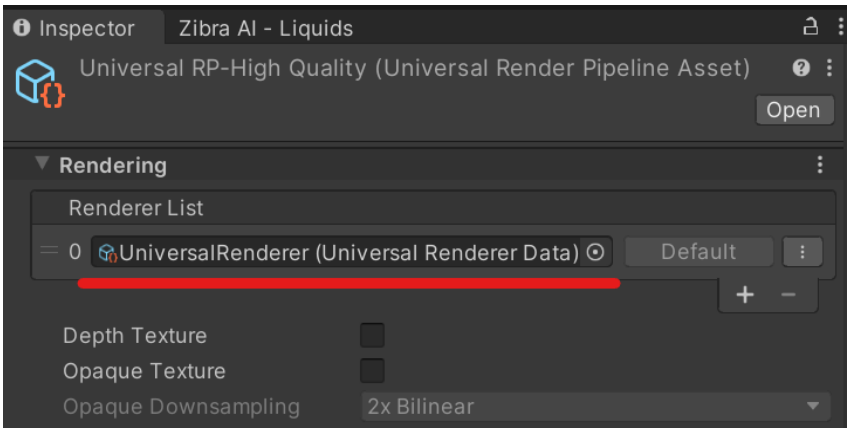
1. Navigate to “Edit → Project Settings...”



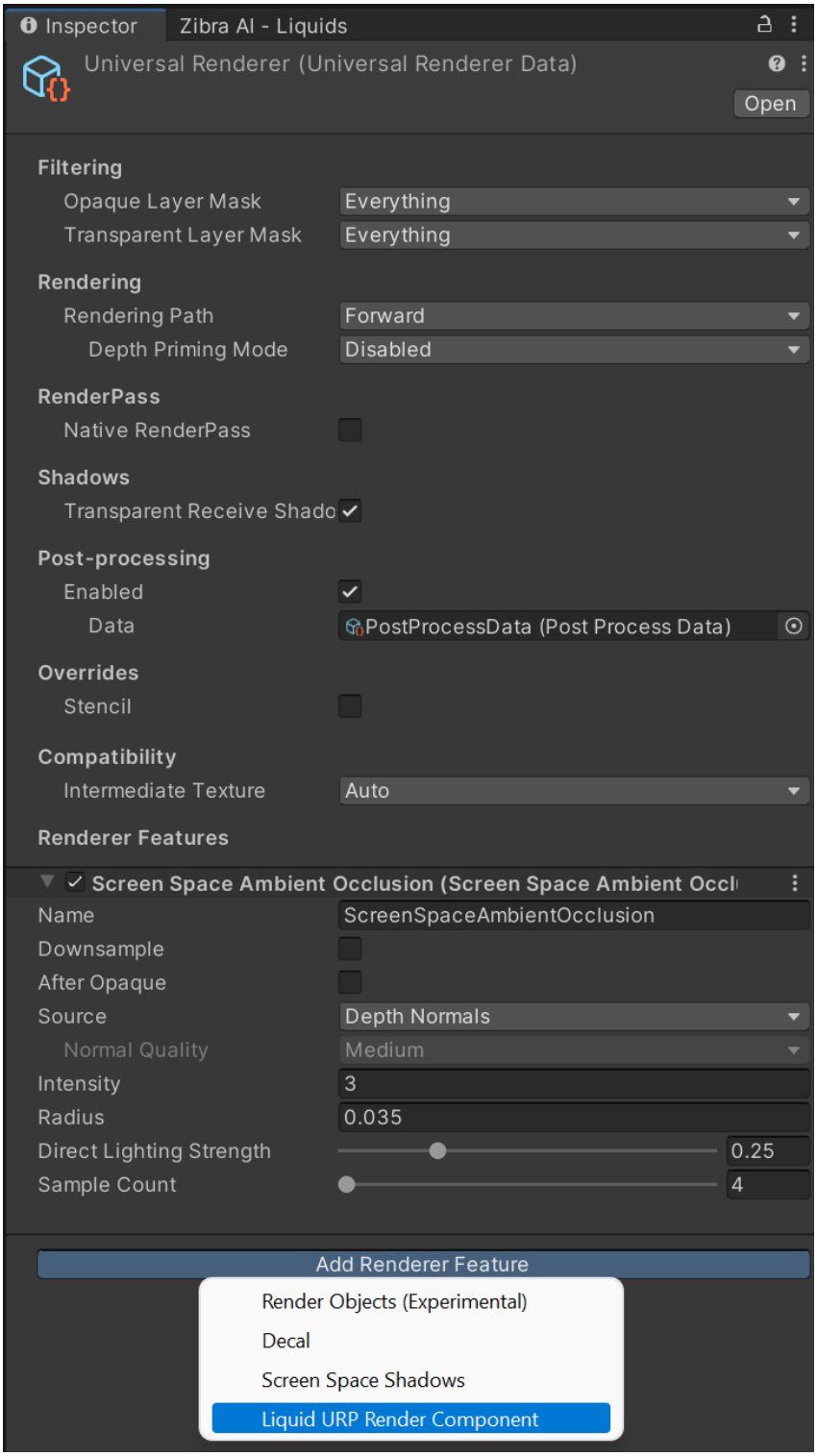
2. From there go to Graphics and open your current Scriptable Render Pipeline Settings asset in the Inspector (you can do it by double clicking it).



3. Now you can see the Renderer List. Open your default Renderer asset in the Inspector (you can do it by double clicking it). You may need to repeat following steps for non default Renderers if you intend on using liquid with them.

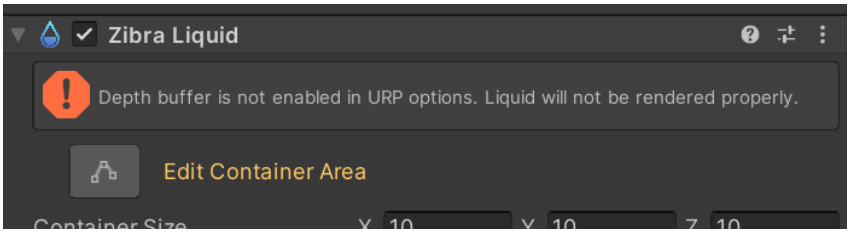


4. Add the “Liquid URP Render Component”



(Specific UI elements can vary depending on Unity version)

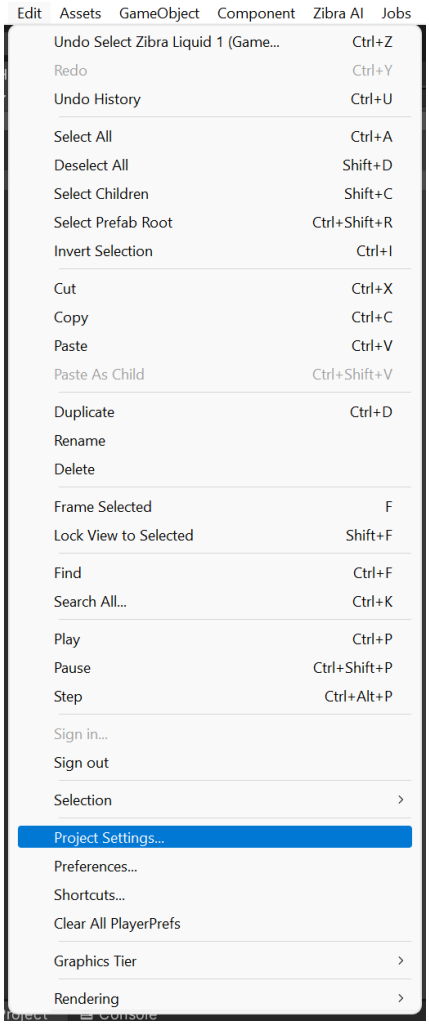
5. If you did everything correctly, that error in Zibra Liquid will disappear:



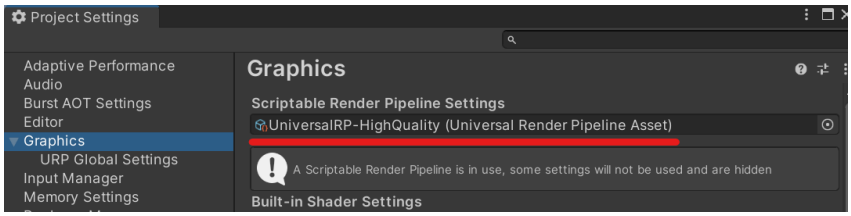
Notes:
Adding “URP Liquid Rendering Component” is project wide, and is only needed once.
You may have separate settings for desktop and mobile platforms. In that case, please make sure to add Liquid URP Render Component to each setting you intend to use with Liquids.

To enable depth buffer and fix second error:

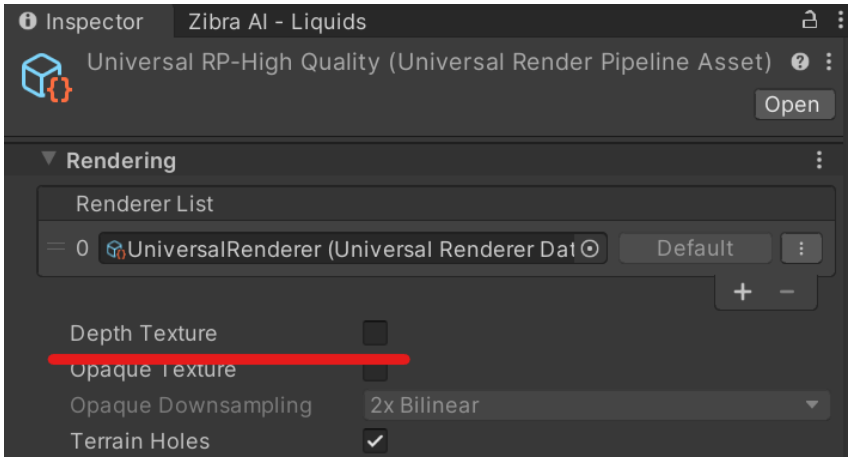
1. Navigate to “Edit → Project Settings...”



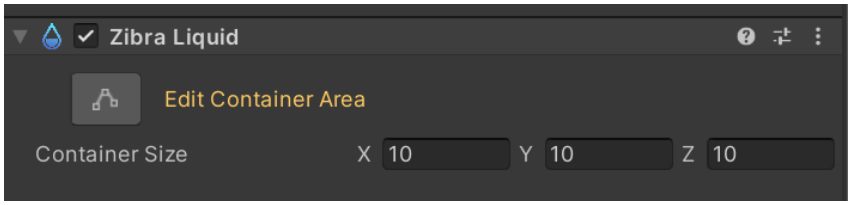
2. From there go to Graphics and open your current Scriptable Render Pipeline Settings asset in the Inspector (you can do it by double clicking it).



3. Enable the Depth Texture option



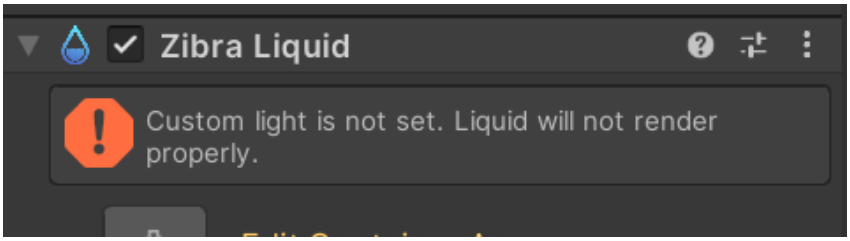
4. If you have any additional Universal Render Pipeline assets used in your project (e.g. for use on Mobile devices), please enable Depth Texture
5. If you did everything correctly, that error in Zibra Liquid will disappear:



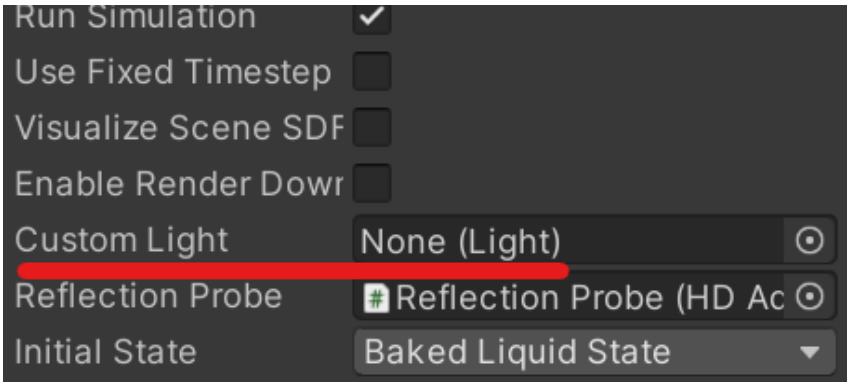
Notes:
Enabling depth texture is project wide, and is only needed once.
You may have separate settings for desktop and mobile platforms. In that case, please make sure to enable depth texture in each setting you intend to use with Liquids.

Additional setup on HDRP

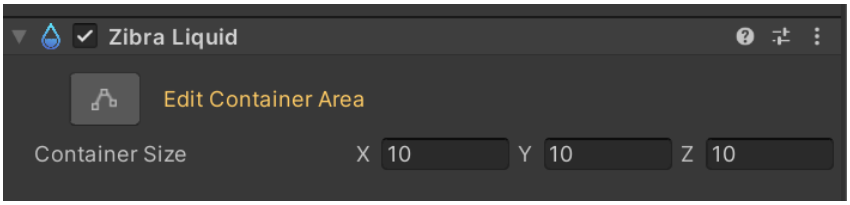
If you are using HDRP you'll see this message:



On HDRP you also need to set the Custom Light parameter.
This parameter sets which light will be used for the liquid lighting.
Currently, only 1 light can be used for lighting the liquid.



After setting the Custom Light parameter the error message will disappear:



Configure Zibra Liquid

Main Liquid parameters

- **Container size**
  The Container size parameter changes how large the simulation volume of liquid is.
  You can press “Edit Container Area'' to enable resizing gizmos.
  This parameter cannot be changed on the “live” liquid instance.
- **Maximum allowed timestep**
  This parameter specifies what the highest delta time that may be used in simulation is, when the FPS drops. If that limit is hit, liquid simulation will slow down.
  Higher values correspond to less stable simulation during FPS drops, but will result in less cases when the liquid slows down.
- **Simulation speed**
  This parameter controls the speed of the liquid simulation. You can slow down or speed up the liquid with this parameter.
- **Iterations per frame**
  Controls how many physic steps will be calculated each Update or FixedUpdate. This affects performance, especially on mobile, but results in better simulation quality. In most cases you will want to set it to 1.
- **Max particle count**
  Controls how many liquid particles a liquid instance can have simultaneously. Higher values correspond to a higher possible liquid volume, but results in higher VRAM usage and potentially higher performance cost.
  This parameter cannot be changed on the “live” liquid instance.
- **Grid resolution**
  This parameter controls the size of the liquid simulation grid.
  This is the most important parameter for performance adjustment.
  Higher size of the grid corresponds to a higher quality simulation, and the smaller size of each particle, but results in higher VRAM usage and a higher performance cost.
  Effective grid resolution shows you the size of your liquid grid. VRAM usage and performance cost scales with effective grid resolution.
  Cell size and Particle size shows you calculated size of each cell and

particle respectively.
This parameter cannot be changed on the “live” liquid instance.

- **Run Simulation**
  Freezes the liquid when disabled. Also decreases performance cost when disabled, since liquid physics won’t be simulated.
- **Enable Container Movement Feedback**
  When enabled, liquid will try to stay in place in world space, when moving the container.
  When disabled, moving the liquid container will move liquid with it.
- **Use Fixed Timestep**
  Use Fixed Timestep for physics simulation. Can be used to have similar behavior of liquid between different runs and machines. Use with care, if the physics simulation takes more time than Fixed Timestep, performance will be severely affected.
- **Visualize Scene SDF**
  Allows you to visualize analytical and neural colliders. This option is intended only for debugging.
- **Enable Render Downscale/Downscale Factor**
  Enables the usage of lower resolution for liquid rendering to improve performance. The Resolution scale is controlled with the “Downscale Factor”. Recommended for mobile devices.
  You can not set it to 1.0, if you want full resolution you must disable render downscale.
- **Custom Light (HDRP only)**
  Sets which light will be used for liquid shading. It’s required that you set that parameter. Currently, only 1 light can contribute to liquid lighting on HDRP.
- **Reflection probe**
  Sets reflection probe used for reflection calculation on the liquid. Must be set on HDRP, and it’s strongly recommended to be set on BRP/URP.
- **Current Rendering Mode**
  Selects which renderer is going to be used for this instance of the liquid. The default and recommended one is Mesh Render. Please note that only Unity Render supports VR in the current release.
- **Current Injection Point**
  Specify at which point in the render graph should liquid be rendered. Recommended options are: Before Forward Alpha if you want other transparent objects to be rendered on top of the liquid, or After Forward

Alpha if you want liquid to be rendered on top of the other transparent objects.

Material parameters

Please note that all material parameters are only used in Mesh Render. In the case of Unity Render mode, you are responsible for material and its parameters.

- **Materials**
  Materials used to shade liquid surface, and optionally upscale liquid when the Enable Render Downscale option is enabled. Most users will not change that option. To use it you will need to create your own custom version of FluidMeshMaterial, and potentially UpscaleMaterial as well. If it is set to None in Editor it will revert back to the default value.
- **Use Cubemap Refraction**
  Whether to use the provided cubemap for refraction calculation instead of calculating screen space refraction.
- **Color**
  Liquid color.
- **Reflection Color**
  Tint for reflections on the liquid.
- **Emissive Color**
  How much light does liquid emit. Unless it's pure black, liquid will glow.
- **Roughness**
  How rough is the liquid surface. Normally, for liquids, that parameter is really low.
- **Metalness**
  How metallic (reflective) the liquid surface is.
- **Scattering amount**
  Defines how much light gets scattered inside the liquid. Higher values correspond to more opaque murky liquid.
- **Absorption amount**
  Defines how much light gets absorbed inside the liquid. Higher values correspond to more opaque liquid.
- **Index of Refraction**
  Determines how refracted the light coming through the liquid will be.
- **Fluid Surface Blur**
  Determines how smooth the surface of liquid will appear.
- **Enabled Foam**
  When enabled, it allows Foam rendering.
  Note that Foam simulation is independent from this parameter.

Note that Foam is currently unavailable on Android. This will be addressed in one of the future updates.

- **Foam Intensity**
  The intensity of foam generation in the liquid.
  Higher values correspond to more foam generated.
- **Foam Decay**
  Rate of foam decay
- **Foam Decay Smoothness**
  Controls the time in the particle lifetime when the particle brightness starts to fade out.
  This parameter determines when the foam particles start to fade out.
  A value of 0.5 will make the particles start to fade out halfway through their lifetime.
- **Foaming Threshold**
  Foam spawn threshold.
- **Foam Brightness**
  Foam particles brightness.
- **Foam Size**
  Foam particle size.
  The size of the foam particles is in voxel space.
  Value of 1.0 will be the size of a single simulation voxel.
- **Foam Diffusion**
  Foam particle diffusion.
  Higher values increase the random motion of the foam particles which can make them appear more natural.
- **Foam Spawning**
  Foam spawning probability.
  Higher values increase the probability of foam particles spawning in the simulation.
- **Foam Motion Blur**
  Foam motion blur.
  The higher the value, the longer the motion trails of the foam particles will be.
- **Max Foam Particles**
  The maximum number of foam particles that can exist in the simulation.
  Higher values correspond to having potentially more foam.
  Setting this parameter to 0 disables foam simulation.
  This parameter cannot be changed on the “live” liquid instance.

Note that Foam is currently unavailable on Android. This will be addressed in one of the future updates.

- **Foam Particle Lifetime**
  The maximum number of simulation frames that the foam particles can exist for.
  Higher values increase the lifetime of the foam particles.
- **Foaming Occlusion Distance**
  How deep underwater foam particles will be seen. Higher values correspond to foam being visible further into the liquid.
- **Material 1/2/3**
  Determine parameters of additional liquid materials used by additional particle species

Advanced Render parameters

Please note that all material parameters are only used in Mesh Render. In the case of Unity Render mode, you are responsible for material and its parameters.

- **Ray Marching Resolution Downscale**
  Configures resolution of raymarching. Higher values correspond to higher quality refraction and Scattering/Absorption calculations, but also higher performance cost. Has no effect in Unity Render mode or when raymarching is disabled.
- **Refraction Bounces**
  Configures how many refraction bounces will be calculated. Two Bounces has higher quality, especially when underwater, but also has a higher performance impact. Has no effect in Unity Render mode or when raymarching is disabled
- **Disable Raymarch**
  Disables calculation of refraction and liquid light absorptions/scattering. This improves performance but makes liquid fully opaque. Has no effect in Unity Render mode.
- **Underwater Render**
  Enables underwater render, when the camera is underwater. Enabling this option has a slight performance impact even when the camera is not underwater. Has no effect in Unity Render mode or when raymarching is disabled
- **Vertex/Mesh optimization parameters**
  Configures how much smoothing will be applied to the liquid surface. Higher iterations count corresponds to higher quality but has a slightly higher performance impact. Higher Step values correspond to more smoothing but potentially more artifacts. You can set all those values to 0 to get voxel liquid.
- **Dual Contouring Iso Surface Level**
  Configures how thick the surface of liquid will be before mesh optimization. Lower values generally result in higher triangle count and so higher quality.
- **Iso Surface Level**
  Configures how thick the surface of liquid will be after mesh optimization. If this option differs too much compared to the previous one you may have some visual artifacts, e.g. bridging between liquid drops.

- **Raymarch parameters**
  Configures how liquid is raymarched for refraction/liquid depth calculations. If misconfigured it can result in opaque liquid or bad refraction. Higher values of Iso Surface parameter correspond to thicker liquid surface for the purpose of that calculation. Higher Max Steps count corresponds to higher quality and less artifacts but has higher performance impact. Higher values of Step Size and Step Factor allow you to decrease Max Steps parameters, but corresponds to more artifacts. Has no effect in Unity Render mode or when raymarching is disabled

Solver parameters

- **Gravity**
  Gravity that affects the entire liquid. If you want radial gravity use Force Field instead.
- **Fluid stiffness**
  Configures how stiff the liquid is (how resistant it is to compression). Higher values are not recommended and may lead to unstable simulation.
- **Particle density**
  Configures the resting particle density. Higher values result in smaller particles, which result in higher quality, but smaller liquid volume.
- **Viscosity**
  Configures how viscous the liquid is (how hard it is to change the shape of the liquid)
- **Surface tension**
  Configures the surface tension of the liquid. Positive values result in liquid that tries to merge together. Negative values result in liquid that tries to split into as many small liquid parts as possible, resulting in a spaghettification effect.
- **Maximum/Minimum Velocity**
  Limits the maximum/minimum velocity of the particles. Setting minimum velocity to values higher than 0 results in liquid that will never stop. Minimum Velocity is unavailable in the Free version
- **Force interaction strength**
  Scales the force that the liquid applies to objects with force interaction enabled. This has a logarithmic scale.
- **Heightmap resolution**
  Resolution of terrain heightmap texture used for Terrain SDF. Higher values result in more accurate Terrain SDF, but have higher performance/memory impact. Has no effects when no Terrain SDFs are used.
- **Foam Buoyancy**
  Determines buoyancy of foam compared to normal liquid
  Value of 0 represents same buoyancy between foam and liquid
- **Material indices**
  Determines which material will be used for liquid rendering by setting corresponding weights. Default material's weight is equal to 1 - sum of all other weights. You can select a single material by setting its value to 1, or

you can mix multiple materials by setting different weights for each that sums up to 1.

- **Additional Particle Species**
  Defines the number of additional particle species and their parameters. You can add additional particle species in runtime, and increase the number of particle species up to 16 over what it was when liquid started simulation.

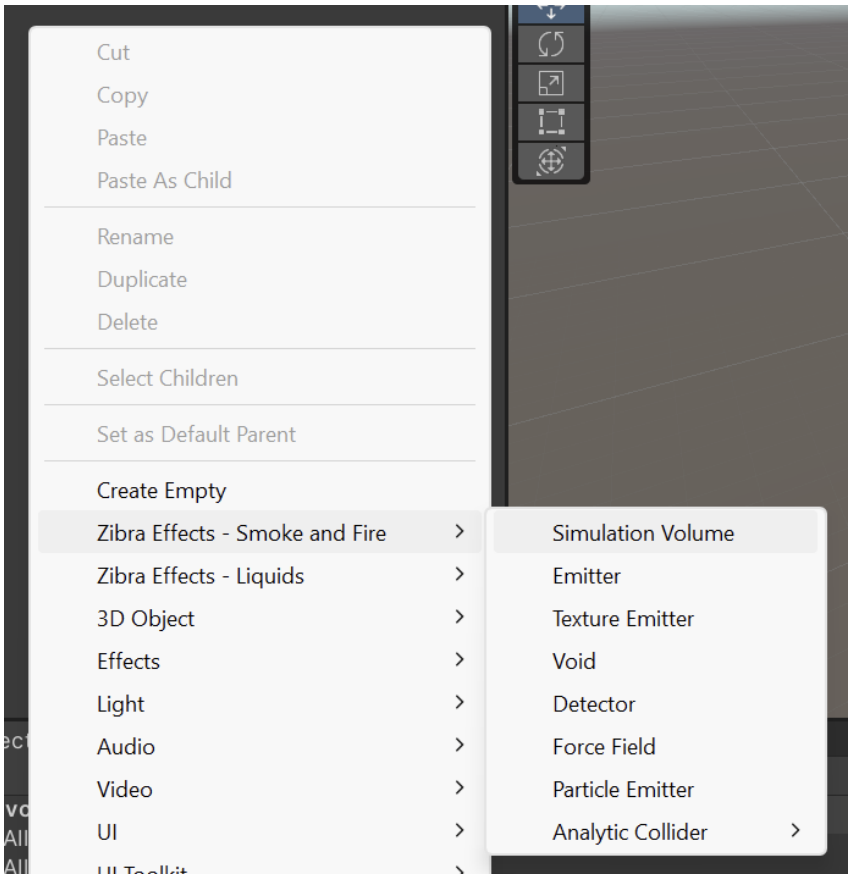
Smoke & Fire Setup

The first thing you need to do is to register your plugin.
After that you can start setting up smoke & fire in your scene.

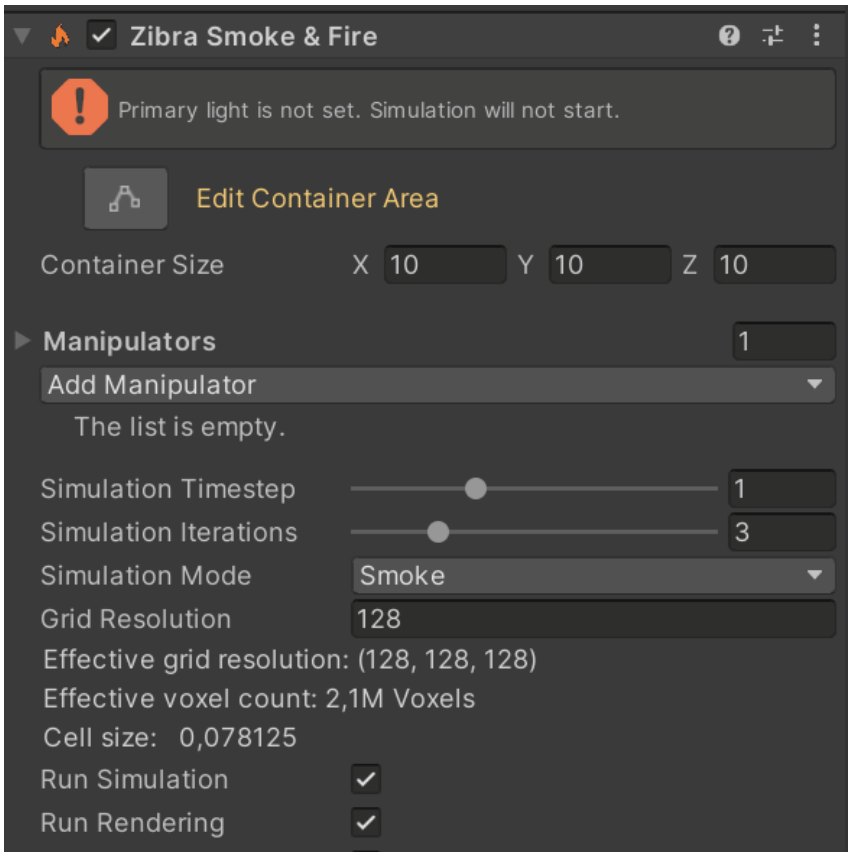
Creating Smoke & Fire Simulation Volume

To create a Zibra Smoke And Fire instance:

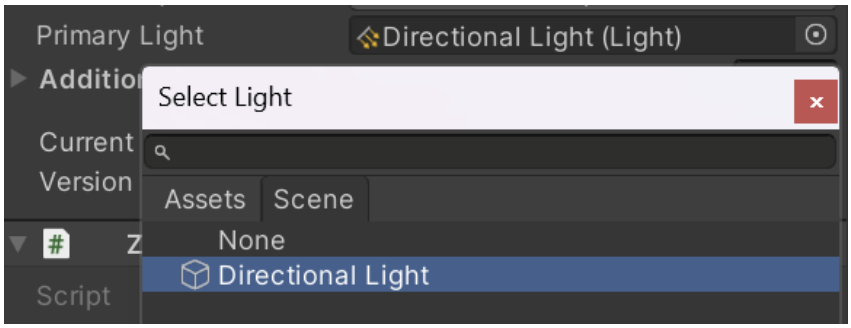
1. Right click in the Hierarchy window and select “Zibra Effects - Smoke and Fire → Simulation Volume”.



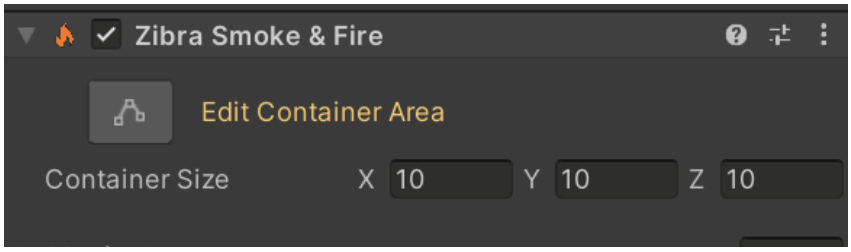
2. In the Inspector window, you'll see the Zibra Smoke And Fire parameters: Note that you'll have an error that it's not properly configured yet.



3. Set the “Primary Light” parameter. It’s strictly necessary to set this parameter.



4. If you did everything correctly, that error in Zibra Smoke And Fire will disappear:



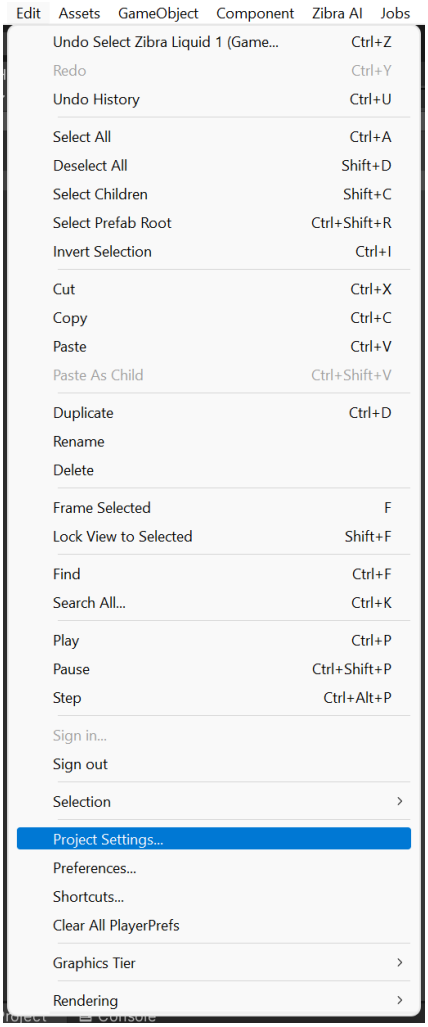
5. If you are using BRP or HDRP, you are now ready to use Smoke & Fire. For URP please proceed to Additional setup on URP.

Additional setup on URP

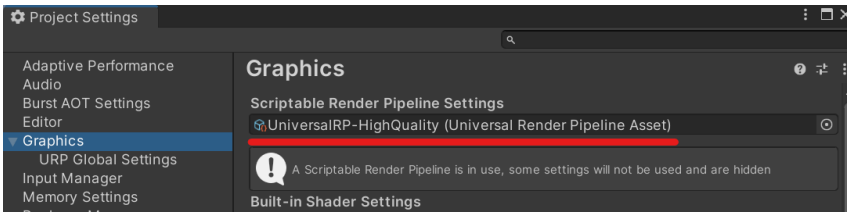
If you are using URP you may see those errors:

To add “URP Smoke And Fire Rendering Component” and fix first error:

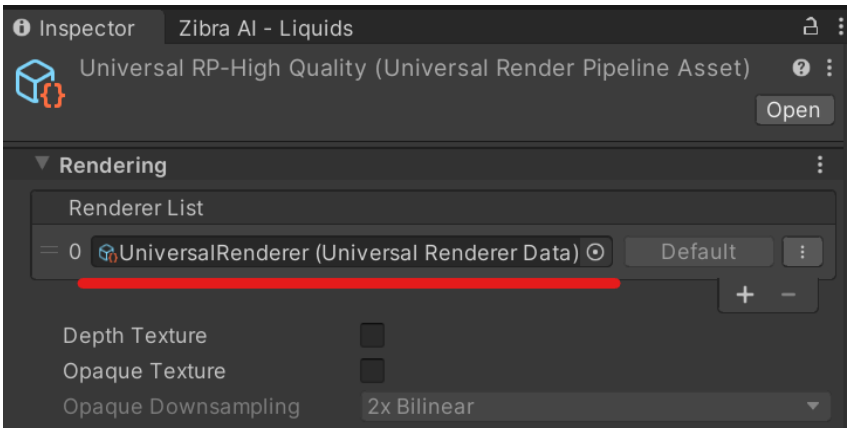
6. Navigate to “Edit → Project Settings...”



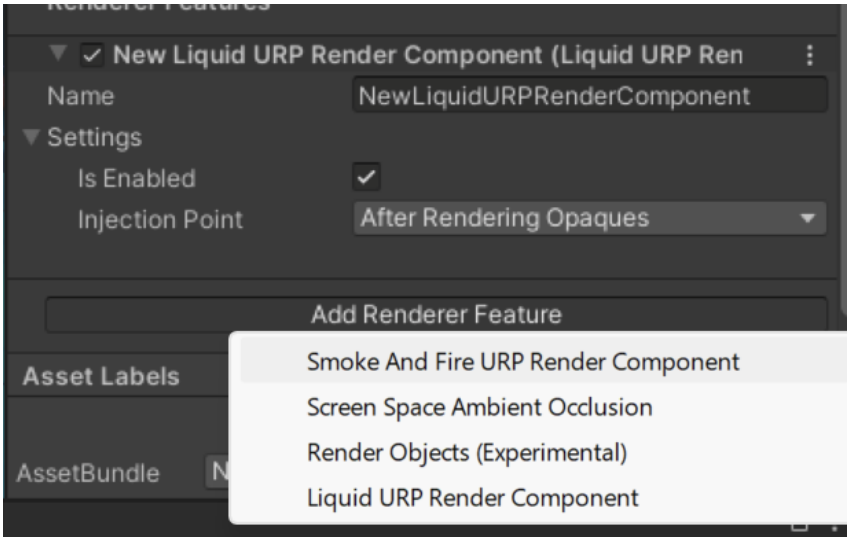
7. From there go to Graphics and open your current Scriptable Render Pipeline Settings asset in the Inspector (you can do it by double clicking it).



8. Now you can see the Renderer List. Open your default Renderer asset in the Inspector (you can do it by double clicking it). You may need to repeat following steps for non default Renderers if you intend on using smoke & fire with them.

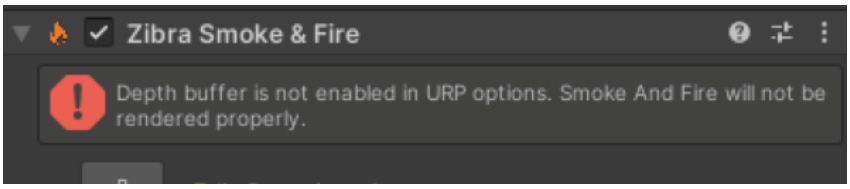


9. Add the “Smoke And Fire URP Render Component”



(Specific UI elements can vary depending on Unity version)

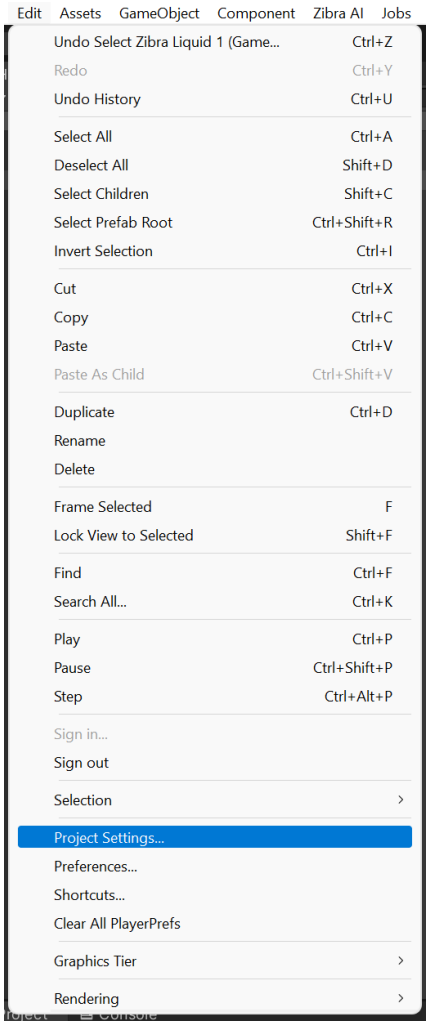
10. If you did everything correctly, that error in Zibra Smoke & Fire will disappear:



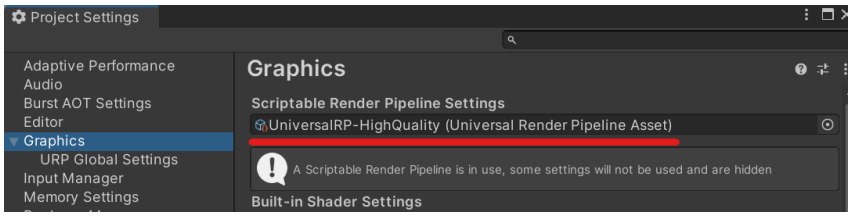
Notes:
Adding “Smoke And Fire URP Render Component” is project wide, and is only needed once.
You may have separate settings for desktop and mobile platforms. In that case, please make sure to add Smoke And Fire URP Render Component to each setting you intend to use with Smoke & Fire.

To enable depth buffer and fix second error:

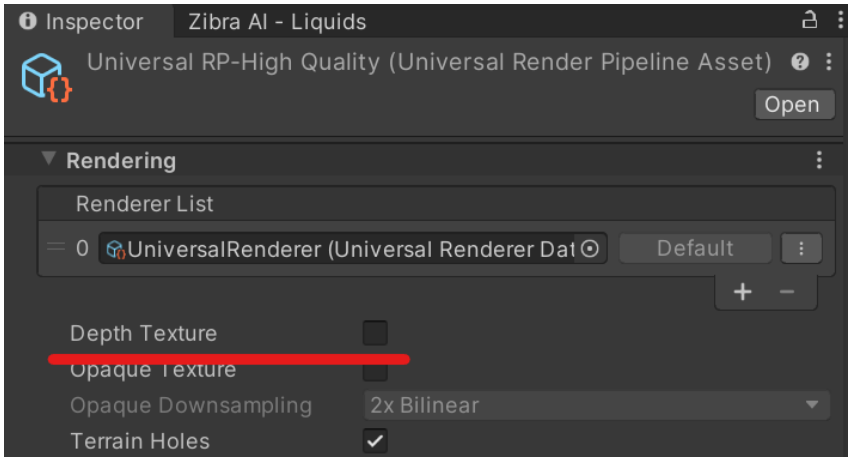
6. Navigate to “Edit → Project Settings...”



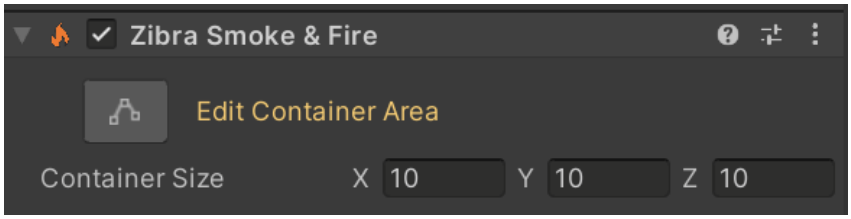
7. From there go to Graphics and open your current Scriptable Render Pipeline Settings asset in the Inspector (you can do it by double clicking it).



8. Enable the Depth Texture option



9. If you have any additional Universal Render Pipeline assets used in your project (e.g. for use on Mobile devices), please enable Depth Texture
10. If you did everything correctly, that error will disappear:



Notes:
Enabling depth texture is project wide, and is only needed once.
You may have separate settings for desktop and mobile platforms. In that case, please make sure to enable depth texture in each setting you intend to use with Smoke & Fire.

Configure Zibra Smoke & Fire

Main Smoke & Fire parameters

- **Container size**
  Size of the simulation volume.
  You can press “Edit Container Area'' to enable resizing gizmos.
  This parameter cannot be changed on “live” simulation instances.
- **Simulation Timestep**
  Controls the time interval between each simulation frame. Smaller values result in more frequent updates, providing a slower but smoother and more accurate simulation. A lower timestep also results in more viscous behavior.
- **Simulation Iterations**
  Number of simulation iterations per simulation frame. The simulation does ⅓ of the smoke simulation per iteration, and extrapolates the smoke movement in time for higher performance while keeping smooth movement. To do a full simulation per frame you can set it to 3 iterations, which may be beneficial in cases where the simulation interacts with quickly moving objects.
- **Simulation Mode**
  Setting that determines the type of simulation being performed, with options including Smoke, Colored Smoke, and Fire. Smoke mode simulates a single colored smoke/fog/etc. Colored Smoke mode allows emitting smoke of a given color. Fire mode simulates smoke, fuel, and temperature components, allowing control of burning fuel to produce fire.
- **Grid Resolution**
  Size of the simulation grid.
  This is the most important parameter for performance adjustment.
  Higher size of the grid corresponds to a higher quality simulation, but results in higher VRAM usage and a higher performance cost.
  Effective grid resolution shows you the size of your simulation grid.
  VRAM usage and performance cost scales with effective voxel count.
  Cell size shows the calculated size of each cell.
  This parameter cannot be changed on the “live” simulation instance.
- **Run Simulation**
  Freezes simulation when disabled. Also decreases performance cost when disabled, since simulation won’t run. Disabling this option does not prevent simulation from rendering.

- **Run Rendering**
  Enables rendering of the smoke/fire. Disabling rendering decreases performance cost. Disabling this option does not prevent simulation from running.
- **Limit Framerate/Maximum Framerate**
  Limits the number of simulation frames per second to avoid excessive performance impact. Maximum Framerate setting specifies the desired limit.
- **Fix Volume World Position**
  When enabled, moving simulation volume will not disturb simulation. When disabled, smoke/fire will try to stay in place in world space. If you want to move the simulation around the scene, you want to disable this option.
- **Enable Render Downscale/Downscale Factor**
  Enables the usage of lower resolution for rendering to improve performance. The Resolution scale is controlled with the “Downscale Factor”. Recommended for mobile devices.
  You can not set it to 1.0, if you want full resolution you must disable render downscale.
- **Current Injection Point**
  Specify at which point in the render graph should Smoke & Fire be rendered. Recommended options are: Before Forward Alpha if you want other transparent objects to be rendered on top of the Smoke & Fire, or After Forward Alpha if you want Smoke & Fire to be rendered on top of the other transparent objects.
- **Downscale options**
  If Downscale enabled, allows you to render Smoke & Fire in lower resolution.
- **Primary Light**
  Directional light that will be used for Smoke & Fire lighting. Must be set, otherwise simulation will not start. Can be freely modified at runtime.
- **Additional Lights**
  List of point lights that contribute to Smoke & Fire lighting. Can be freely modified at runtime. You can add up to 16 lights to that list.

Material parameters

- **Materials**
  Materials used to render Smoke & Fire volume, and optionally upscale rendered image when the Enable Render Downscale option is enabled. You can replace built-in rendering shaders by using those options, but it's only intended for advanced Smoke & Fire users to do so. To use it you will need to create your custom version of Materials. If it is set to None in Editor it will revert back to the default value.
- **Smoke Density**
  Optical density of smoke. Higher values correspond to more opaque smoke. Note that emitters may emit both smoke and fuel, but they have separate density values. Smoke is also created as a result of fuel burning.
- **Fuel Density**
  Optical density of fuel. Higher values correspond to more opaque fuel. Note that emitters may emit both smoke and fuel, but they have separate density values. Only has an effect in the fire simulation mode.
- **Absorption Color**
  Color of the light that can pass through the smoke.
- **Scattering Color**
  Color of the light that can scatter inside the smoke.
- **Shadow Absorption Color**
  The shadow absorption color. Defines color of the light for shadow render purposes. Only has effect when shadow projection is enabled.
- **Scattering Attenuation**
  Controls the distance light can travel through the volume.Higher values correspond to a more localized illumination effect.
- **Scattering Contribution**
  Determines the amount of scattered light that contributes to the final rendered image, affecting the overall brightness of scattered light..
- **Object Primary Shadow**
  When enabled, simulation volume renders shadow from the point of view of Primary Light on objects occluded by smoke.
- **Object Illumination Shadows**
  When enabled, simulation volume renders shadow from the point of view of Additional Lights on objects occluded by smoke.

- **Illumination Brightness**
  Controls the overall brightness of the volume's illumination from point lights. Higher values correspond to more illuminated volume.
- **Illumination Softness**
  Determines the softness of the volume's illumination, with higher values resulting in a more diffuse and softer appearance.
- **Black Body Brightness**
  Brightness of black body radiation. Black body radiation is light emitted based on the temperature of the object emitting light.
- **Fire Brightness**
  Controls the overall brightness of the fire. This is a multiplier for brightness. Another factor that affects brightness is speed of fuel combustion reaction. Only has an effect in the fire simulation mode.
- **Fire Color**
  Determines the color of the fire in Fire simulation mode, allowing for a more customized appearance. This parameter changes the entire color of the fire and does not depend on temperature. . Only has an effect in the fire simulation mode.
- **Temperature Density Dependence**
  Controls the optical density of the volume, decreasing with increasing temperature. Higher values correspond to more transparent fire. Only has an effect in the fire simulation mode.
- **Enable Projected Shadows/Object Shadow Intensity**
  Determines whether the volume casts shadows in the scene and controls the intensity of the shadows cast by the volume on the objects in the scene.
- **Shadow Distance Decay**
  Controls the rate at which the intensity of the shadows cast by the volume decreases with distance, affecting how far the shadows extend.
- **Shadow Intensity**
  Intensity of shadowing effect.
- **Shadow Projection Quality Level**
  Sets the quality level of the projected shadows, the “tricubic” setting results in more detailed and accurate shadows but higher performance cost.
- **Volume Edge Fade Off**
  Controls the amount of fading of the optical density at the edges of the

simulation volume, creating a smoother transition between the volume and its surroundings.

- **Ray Marching Step Size**
  Ray marching step size used for the volume rendering process. Smaller values correspond to more accurate rendering but higher performance cost.
- **Shadow Resolution**
  The resolution of the primary shadow 3D texture relative to the simulation volume. Smaller values correspond to more accurate shadows but higher performance cost.
- **Shadow Step Size**
  Ray marching step size that is used for calculating the primary shadows, affecting the accuracy and detail of the shadows.
- **Shadow Max Steps**
  Maximum number of ray marching steps used for calculating the primary shadows. Higher values correspond to more accurate shadows but higher performance cost.
- **Illumination Resolution**
  The resolution of the Additional Lights's shadow 3D texture relative to the simulation volume. The lower the resolution the faster the shadows are computed.
- **Illumination Step Size**
  Ray marching step size used for illumination calculation. Smaller values correspond to more accurate illumination but higher performance cost.
- **Illumination Max Steps**
  Maximum number of ray marching steps used for illumination calculation. Higher values correspond to more accurate illumination but higher performance cost.
- **Max Effect Particles**
  Controls the maximum number of effect particles in the simulation. If you hit the limit, you won't be able to emit more, until some of the particles get deleted. Higher values correspond to potentially more effect particles but higher performance cost.
- **Particle Lifetime**
  Determines the lifetime of the effect particles in the simulation, affecting how long the particles exist before disappearing.
- **Particle Occlusion Resolution**
  Controls the resolution of the occlusion texture used in the effect particle

rendering. Higher values correspond to more accurate effect particles occlusion but higher performance and memory cost.

Solver parameters

- **Gravity**
  Strength and direction of the gravity force applied to the whole simulation.
- **Smoke Buoyancy**
  Controls the strength of the buoyancy force applied to the smoke in the simulation, affecting how quickly smoke rises or falls due to differences in density.
- **Heat Buoyancy**
  Determines the strength of the upward force applied to heat in the simulation, affecting the speed at which hot gases rise.
- **Temp Threshold**
  Combustion threshold. As soon as fuel exceeds this threshold, fuel combustion reaction starts.
- **Heat Emission**
  Rate of heat emission from the combustion reaction of the fuel in the simulation.
- **Reaction Speed**
  Rate at which combustion reaction occurs in the simulation.
- **Maximum Velocity**
  Maximum velocity of the gas allowed in the simulation.
- **Minimum Velocity**
  Minimum velocity of the gas allowed in the simulation.
- **Sharpen**
  Strength of sharpening effect in the simulation. Higher values correspond to more detailed simulation, but can lead to visible aliasing.
- **Sharpen Threshold**
  Threshold that differentiates between regions that need to be simulated more accurately. This threshold is based on the average velocity of the gas. Lower values correspond to larger volume inside the simulation getting simulated more precisely but higher performance cost.
- **Color Decay**
  This parameter is responsible for reducing the optical density of smoke with time. In case of the fire simulation mode this parameter is also responsible for controlling temperature decay.
- **Velocity Decay**
  Rate at which velocity is dissipated or decayed over time.

- **Pressure Reuse**
  Determines how much of the pressure data from previous simulation iteration gets reused to simulate current one. Higher values correspond to more stable simulation, but slower response to fast moving objects.
- **Pressure Projection**
  To enforce physically correct behavior of the simulation this parameter needs to be set to 1.0. Values lower than 1.0 will result in the volume behaving more viscous. Values higher than 1.0 will enhance the turbulent behavior of the volume, but it might become unstable, for values usually in the range from 1.7.
- **Pressure Solve Iterations**
  The number of iterations used to compute the pressure. Higher values correspond to more accurate simulation, but higher performance cost. Single iteration is sufficient for most cases.

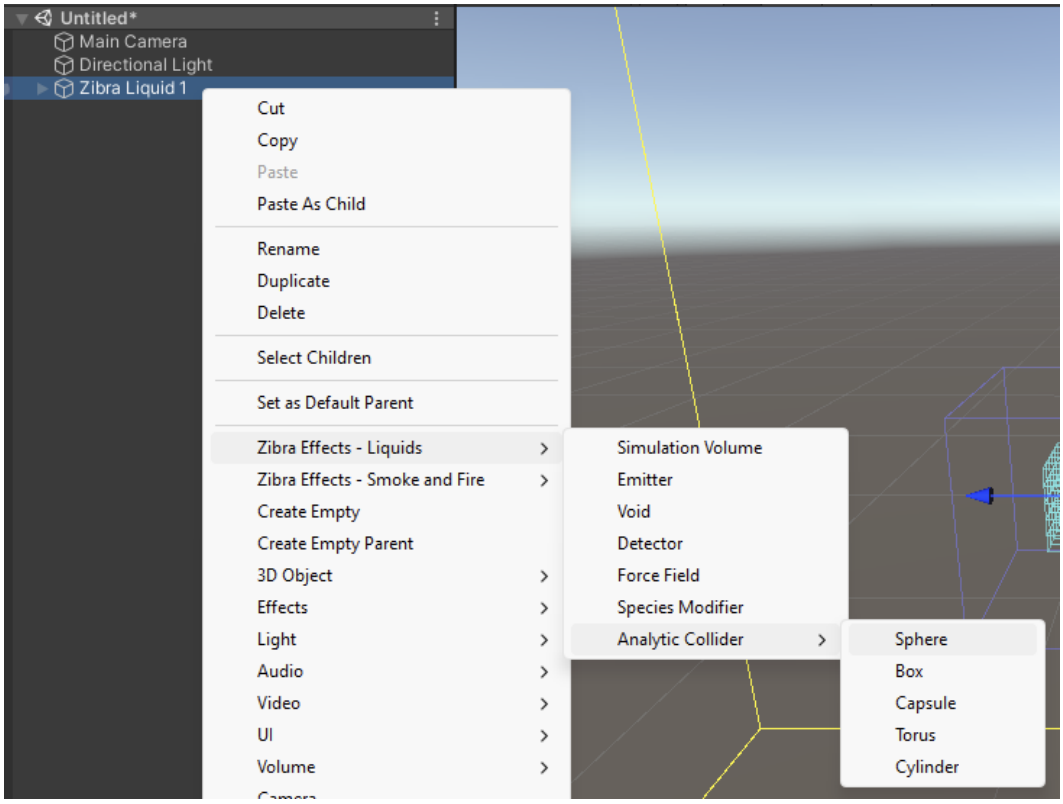
Liquid Manipulators

Collider

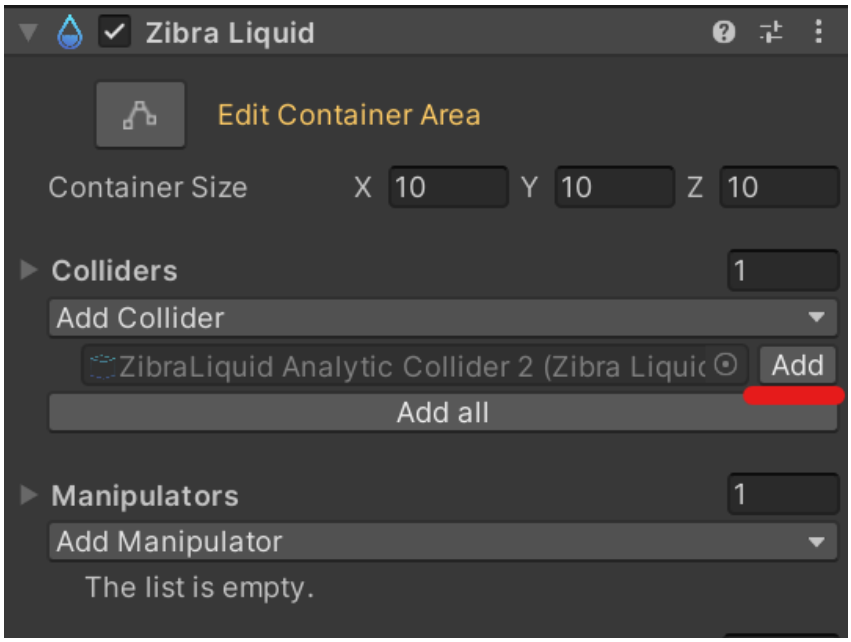
Colliders are manipulators that collide with liquids. They apply force to the liquid and can optionally receive force back from the liquid.

Creating Analytic Colliders GameObjects

1. Right click in the Hierarchy window, and select “Zibra Effects - Liquids → Analytic Collider → Your preferred shape”
   That creates a new GameObject, with Liquids Collider and Analytic SDF components and chosen shape pre-configured.
   Please note, if you right clicked specifically on your Zibra Liquid object, you can skip step 2



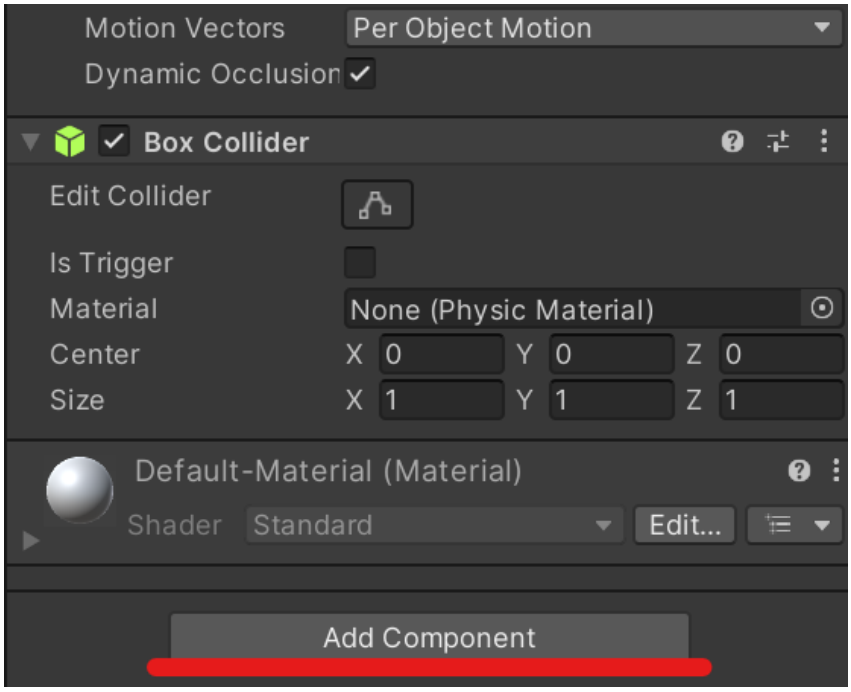
2. Open Inspector for your Liquid object, and press the Add button in the list of colliders to add. You can also use the Add all button to add all colliders to your liquid instance.
   You may also need to expand the list of colliders to add by pressing the “Add Collider” button first.

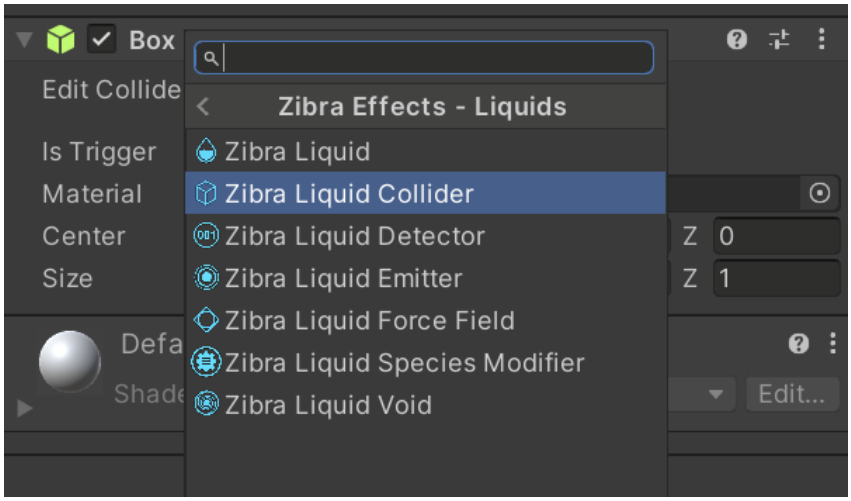


After that your collider is ready to be used. You can set its transform in the editor, or move it at runtime.

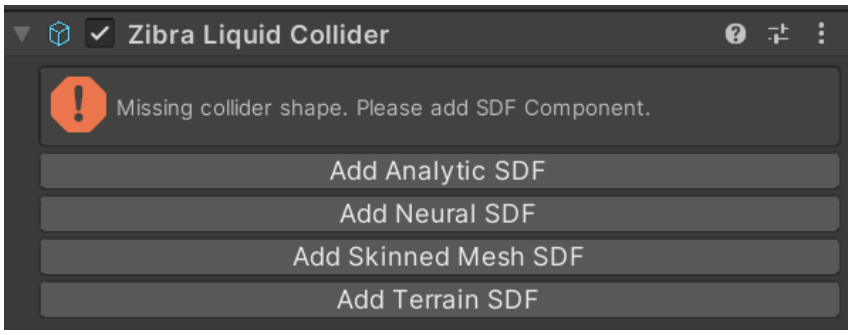
Adding Colliders to existing GameObjects

1. Open inspector for your selected GameObject. Press Add Component and select “Zibra Effects - Liquids → Zibra Liquid Collider”.

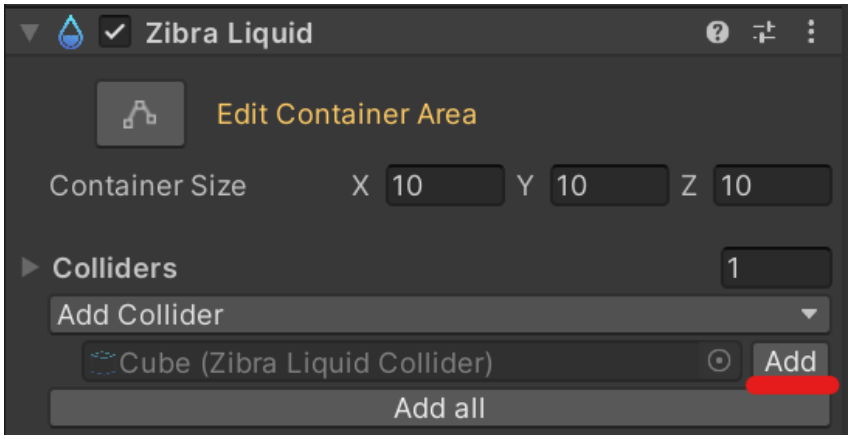




2. In Zibra Liquid Collider add one of the SDF components.
   For more information about SDF components, please see SDFs.



3. Configure newly created SDF component
4. Open Inspector for your liquid object, and press the Add button in the list of colliders to add. You can also use the Add all button to add all colliders to your liquid instance.
   You may also need to expand the list of colliders to add by pressing the “Add Collider” button first.



After that your collider is ready to be used. You can set its transform in the editor, or move it at runtime.

Collider parameters

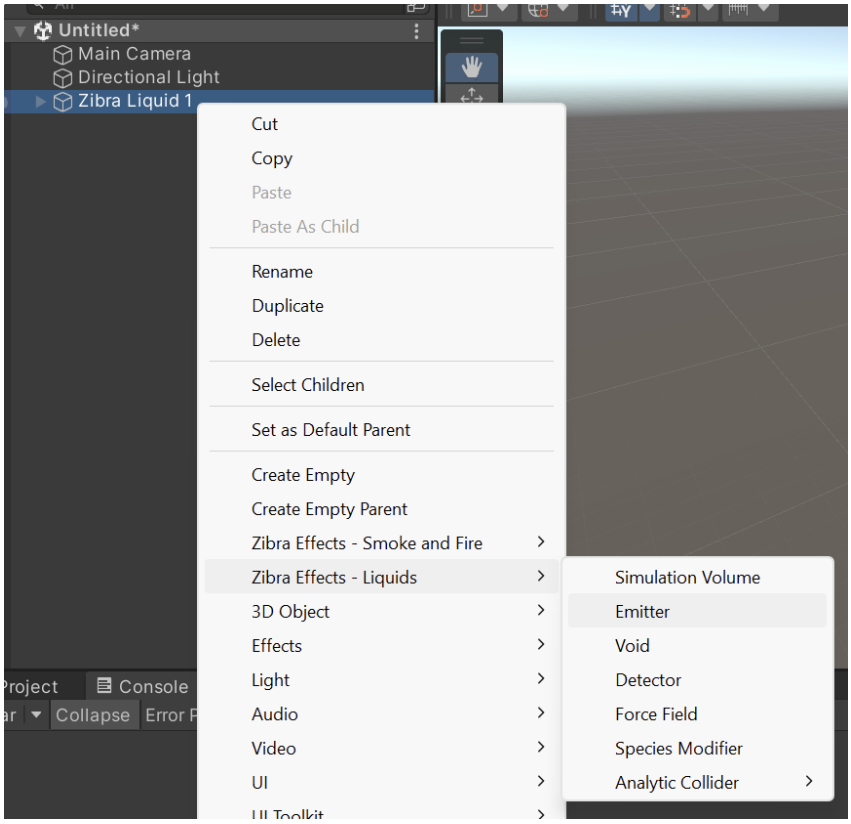
- **Friction**
  Controls the surface friction of the collider, used for liquid interaction.
- **Force Interaction**
  Controls whether liquid can apply force back to the object. You need to add a rigidbody component to the object that has a collider for liquid to apply force to.
- **Force Interaction Callback**
  Specifies a callback that gets Force Interactions force and can modify it. Will be called even when Force Interaction is disabled, with correct calculated force, but that force won’t be applied in this case.

Emitter

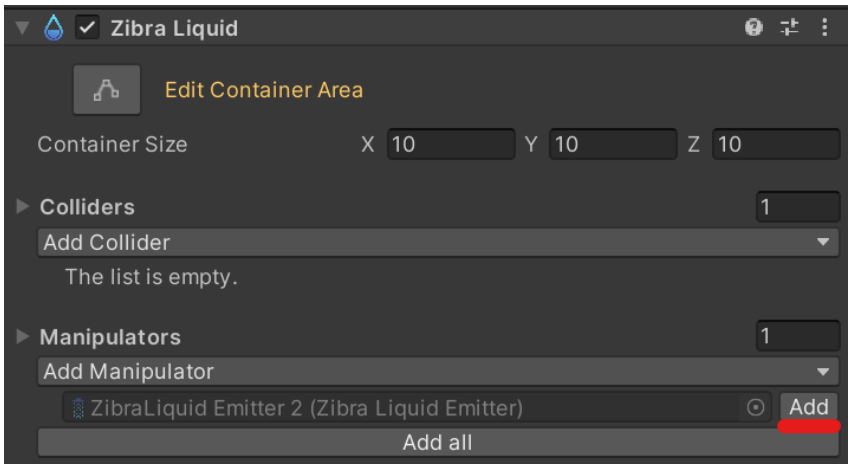
Liquid emitters are used for emitting liquid, if you want to spawn liquid you need these.
When you first create Zibra Liquid, the Emitter is automatically created and added to it. If you don't need additional ones you can skip adding additional emitters and go right to Emitter parameters

Adding Emitter

1. Right click in Hierarchy, and select "Zibra Effects - Liquids → Emitter"
   Note that if you right click specifically on your liquid object you can skip step 3



2. Open Inspector for your liquid object, and press the Add button in the list of manipulators to add. You can also use the Add all button to add all manipulators to your liquid instance.
   You may also need to expand the list of colliders to add by pressing the “Add Manipulators” button first.



Emitter parameters

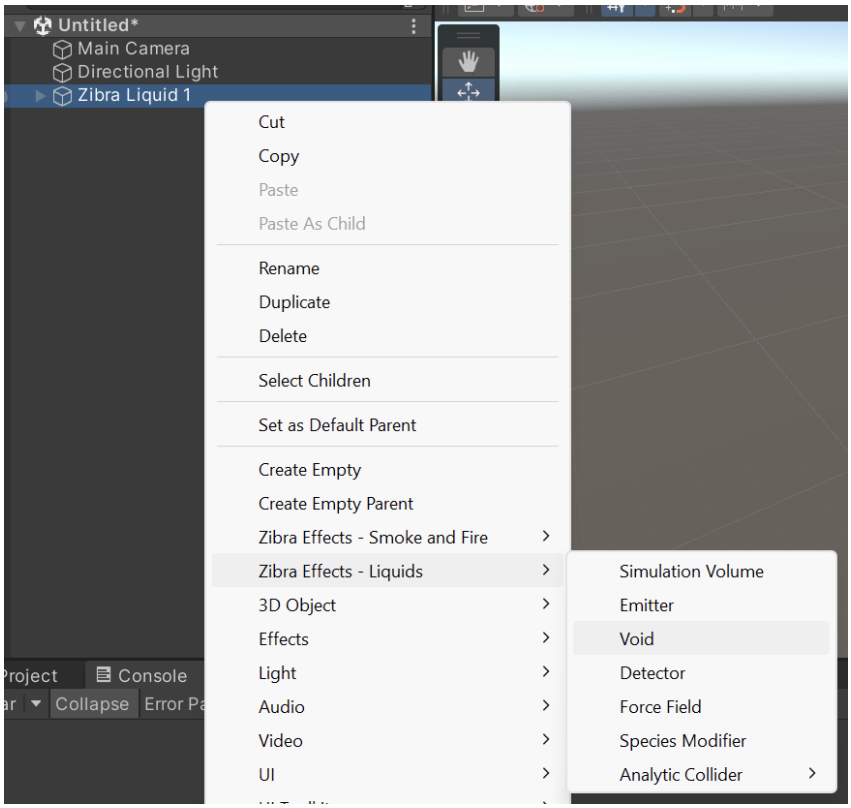
- **Emitter statistics** (read only)
  When you start the play mode you'll be able to keep track of how many particles were emitted. You can also access those values via scripts, via "createdParticlesTotal" and "createdParticlesPerFrame" attributes.
- **Volume Per Sim Time**
  Amount of liquid that is emitted per unit of Simulation Time.
- **Initial Velocity**
  Initial speed of the emitted particles.

Void

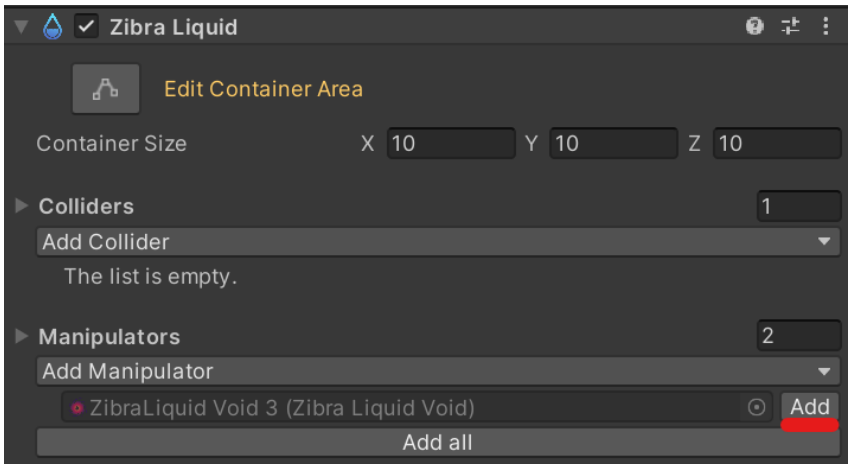
Voids are used to destroy liquid. Since there's a limit to the maximum number of particles in Zibra Liquid instance, you need to destroy particles for your emitters to keep emitting liquid continuously. But if it's fine for you that the emitters will stop at some point, the usage of voids is optional.

Adding Void

1. Right click in Hierarchy, and select "Zibra Effects - Liquids → Void"
   Note that if you right click specifically on your liquid object you can skip step 2



2. Open Inspector for your liquid object, and press the Add button in the list of manipulators to add. You can also use the Add all button to add all manipulators to your liquid instance.
   You may also need to expand the list of colliders to add by pressing the “Add Manipulators” button first.



Void parameters

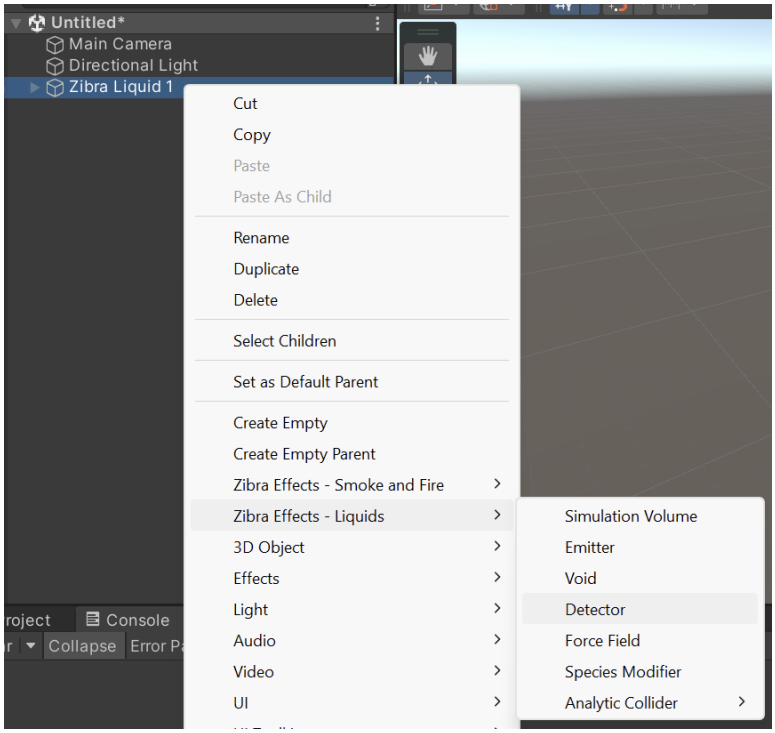
- **Void statistics** (readonly)
  When you start play mode you'll be able to keep track of how many particles were deleted. You can also access those values via scripts, via "deletedParticleCountTotal" and "deletedParticleCountPerFrame" attributes.
- **Delete Percentage**
  What percentage of liquid will be deleted each second if simulation speed is set to default value. Changing simulation speed in liquid will scale deletion speed accordingly.

Detector

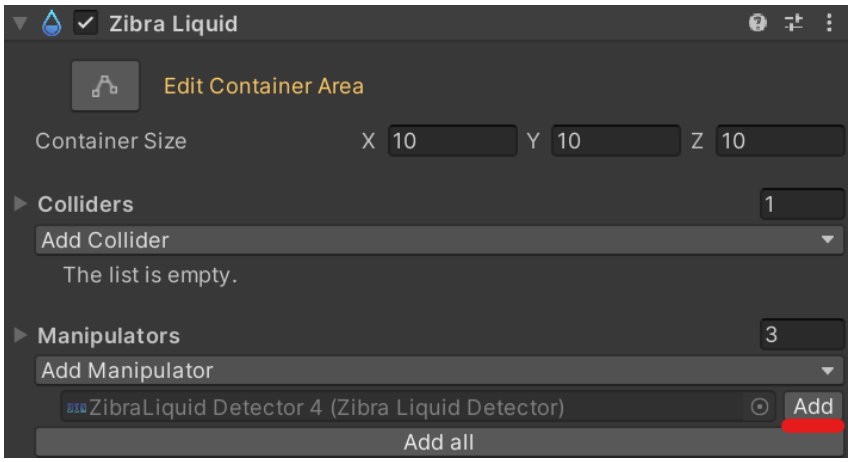
Detector is used to detect how many liquid particles are in the specified volume.

Adding Detector

1. Right click in Hierarchy, and select “Zibra Effects - Liquids → Detector”
   Note that if you right click specifically on your liquid object you can skip step 2



2. Open Inspector for your liquid object, and press the Add button in the list of manipulators to add. You can also use the Add all button to add all manipulators to your liquid instance.
   You may also need to expand the list of colliders to add by pressing the “Add Manipulators” button first.



Detector parameters

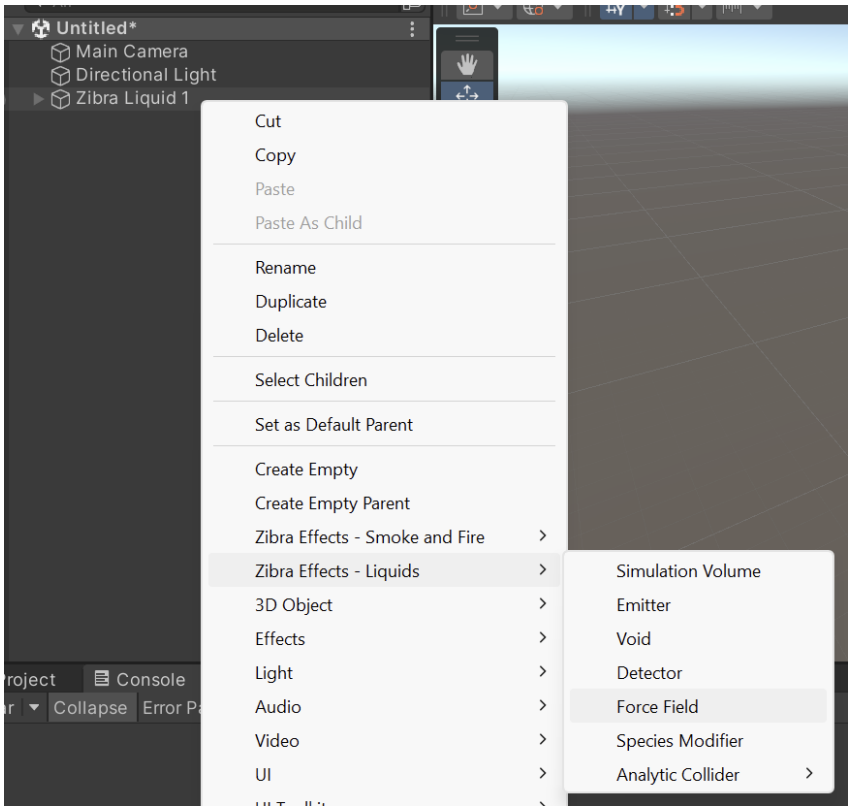
- **Detected particle count** (read only)
  When you start play mode you’ll be able to keep track of how many particles are inside the detector. You can also access those values via scripts, via the “particlesInside” attribute.

Force Field

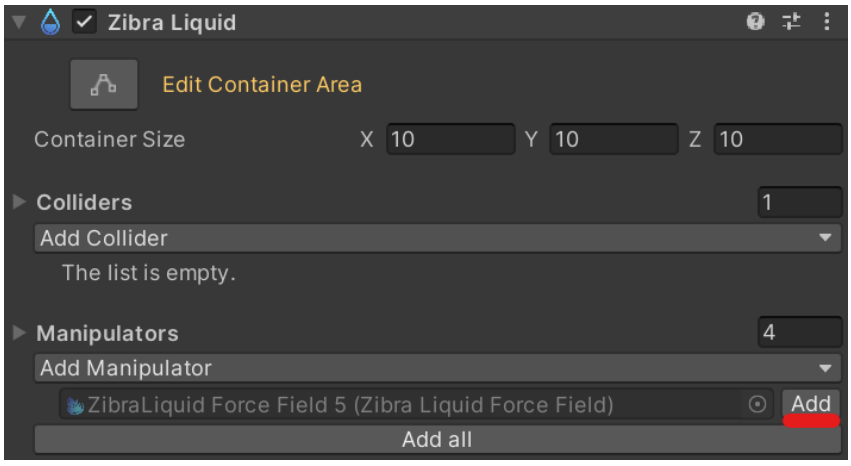
Force fields are used to apply force to liquid in a specific way, and in a specific region. You can for example use force fields to create radial gravity, to use instead of the default vector one. Or you can create a fountain by putting a force field that pushes liquid up in the body of water.

Adding Force Field

1. Right click in Hierarchy, and select “Zibra Effects - Liquids → Force Field” Note that if you right click specifically on your liquid object you can skip step 2



2. Open Inspector for your liquid object, and press the Add button in the list of manipulators to add. You can also use the Add all button to add all manipulators to your liquid instance.
   You may also need to expand the list of colliders to add by pressing the “Add Manipulators” button first.



Force Field parameters

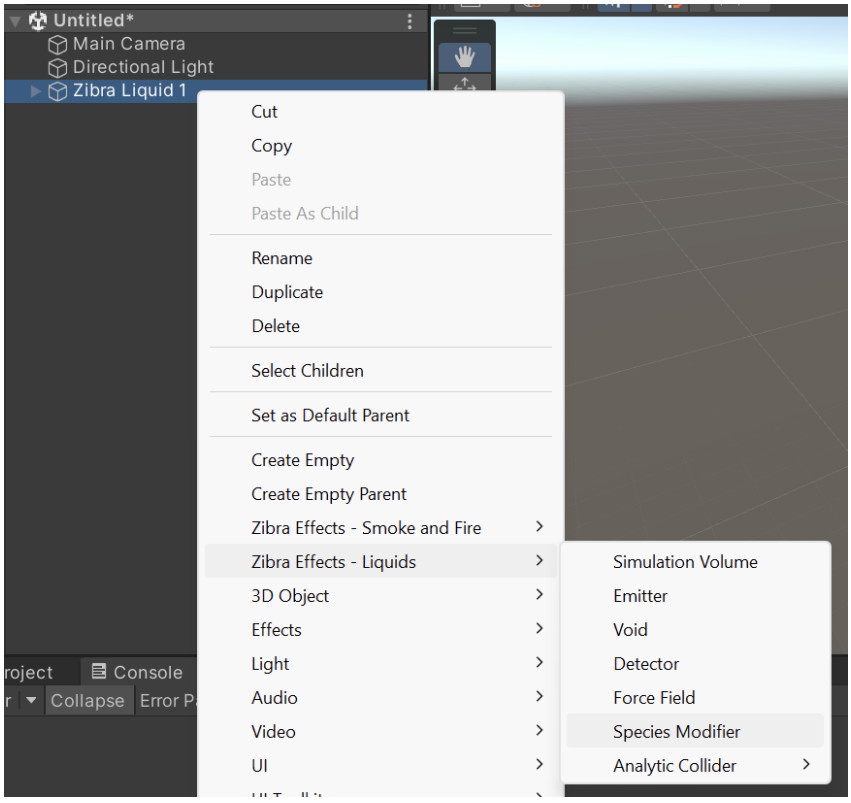
- **Type**
  Configures how Force Field applies force to liquid.
  Radial - pushes or pulls liquid towards the Force Field.
  Directional - pushes liquid in a specified direction.
  Swirl - creates a whirlpool force around the Force Field.
  In the case of the Directional and Radial settings, the direction can be adjusted by rotating the Force Field.
- **Strength**
  Configures how strong the strength of the force applied by Force Field will be. Can be negative to reverse direction. (For radial type, positive force is pulling, and negative is pushing)
- **Distance decay**
  Configures how fast the force strength decays with the distance from the Force Field.
- **Distance Offset**
  Offset for distance to surface. That offset will be applied for purposes of calculating Distance decay.
- **Disable Force Inside**
  Enable to disable applying force to the liquid inside force field.
- **Force Direction**
  Direction of the force applied by force field. Only used if Force Field's Type is Directional.

Species Modifier

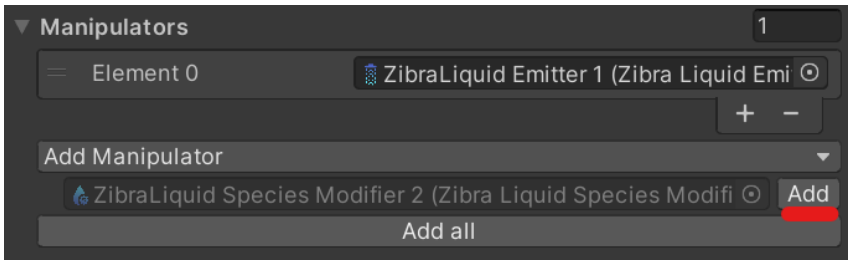
Species modifiers are used to change species of liquid particles. You can for example use it to change color of the liquid, invert gravity, or change any other parameter that can be configured per species.

Adding Species Modifier

1. Right click in Hierarchy, and select “Zibra Effects - Liquids → Force Field” Note that if you right click specifically on your liquid object you can skip step 2



2. Open Inspector for your liquid object, and press the Add button in the list of manipulators to add. You can also use the Add all button to add all manipulators to your liquid instance.
   You may also need to expand the list of colliders to add by pressing the “Add Manipulators” button first.



Species Modifier parameters

- **Target Species**
  Defines which species the particle will get after modification.
- **Probability**
  Defines probability that a particle inside a species modifier will change its species each frame.

Common manipulator parameters

Particle species parameters are shared between all manipulator types, except colliders. They are used to filter which particle species can interact with each manipulator.

- **Current Interaction Mode**
  Defines which particles can be affected by manipulator
  All Particle Species - Manipulator affects everything
  Only Selected Particle Species - Only affects particles of species specified in Particle Species parameter
  Except Selected Particle Species - Affects all particles except species specified in Particle Species parameter
- **Particle Species**
  Selects particle species for use with Current Interaction Mode parameter

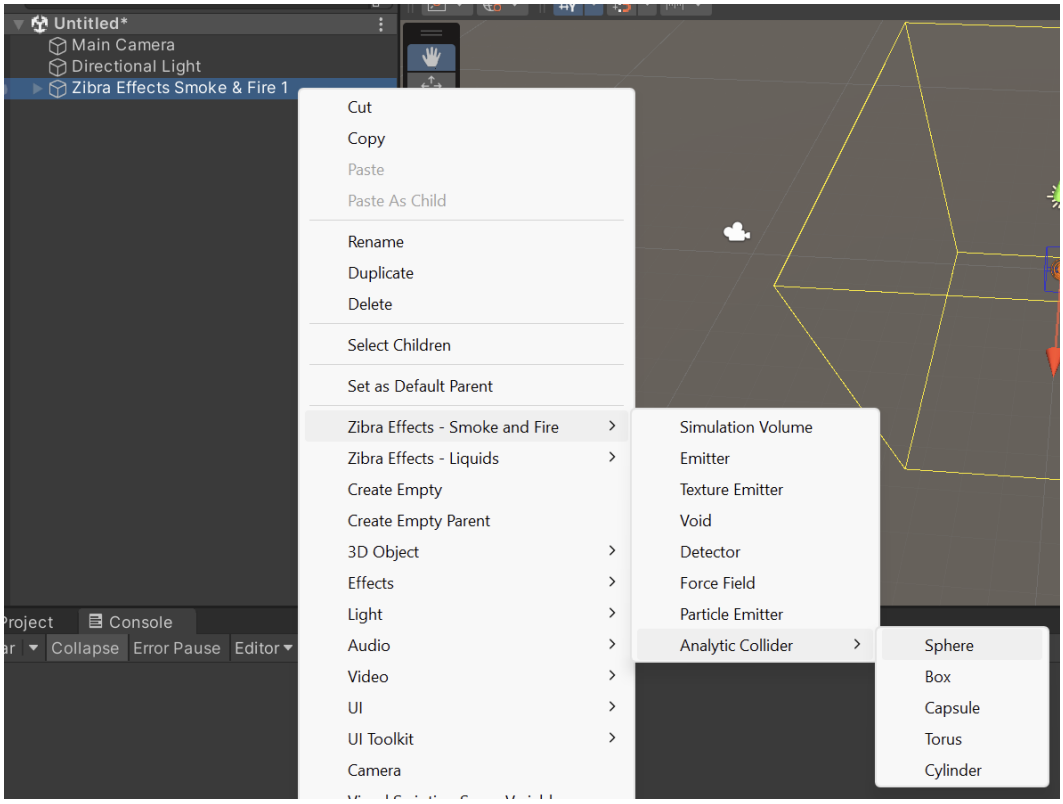
Smoke Manipulators

Collider

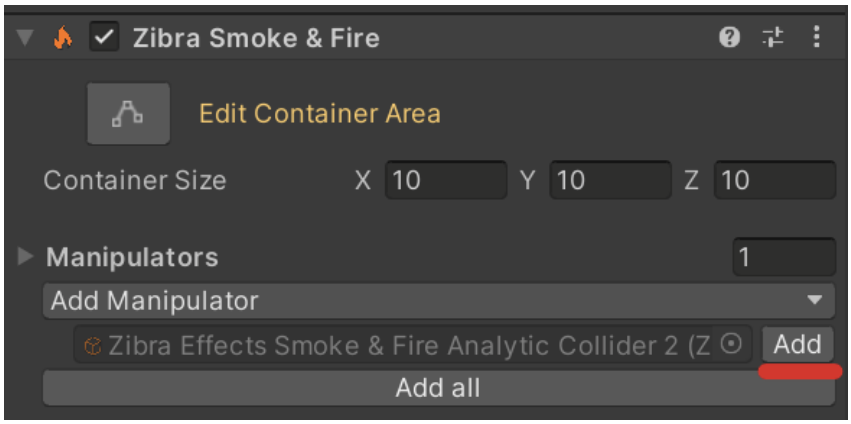
Colliders are manipulators that collide with simulation. They can apply force to Smoke & Fire.

Creating Analytic Colliders GameObjects

1. Right click in the Hierarchy window, and select “Zibra Effects - Smoke and Fire → Analytic Collider → Your preferred shape”
   That creates a new GameObject, with Smoke & Fire Collider and Analytic SDF components and chosen shape pre-configured.
   Please note, if you right clicked specifically on your Zibra Smoke & Fire object, you can skip step 2



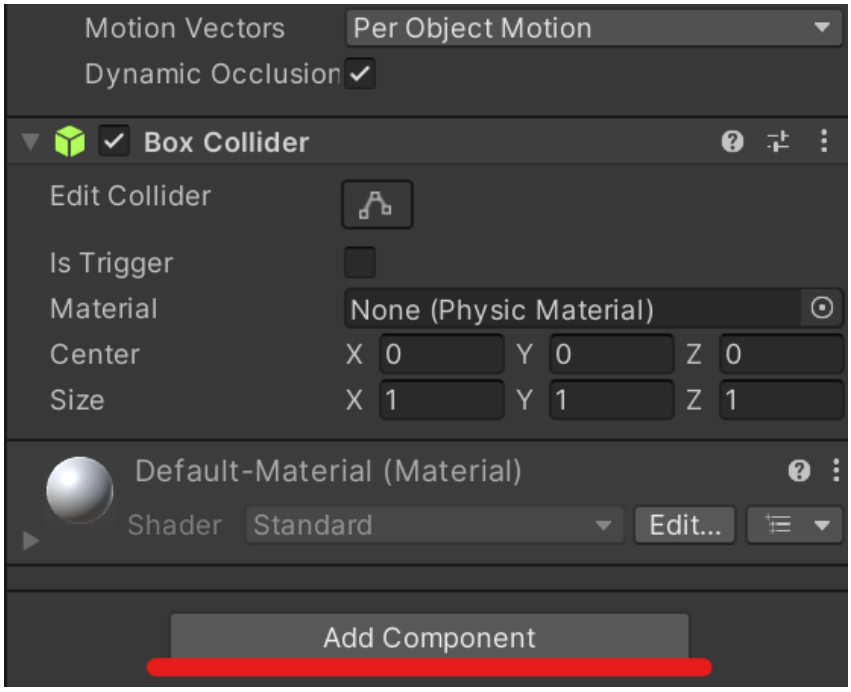
2. Open Inspector for your Smoke & Fire object, and press the Add button in the list of colliders to add. You can also use the Add all button to add all colliders to your Smoke & Fire instance.
   You may also need to expand the list of colliders to add by pressing the “Add Collider” button first.

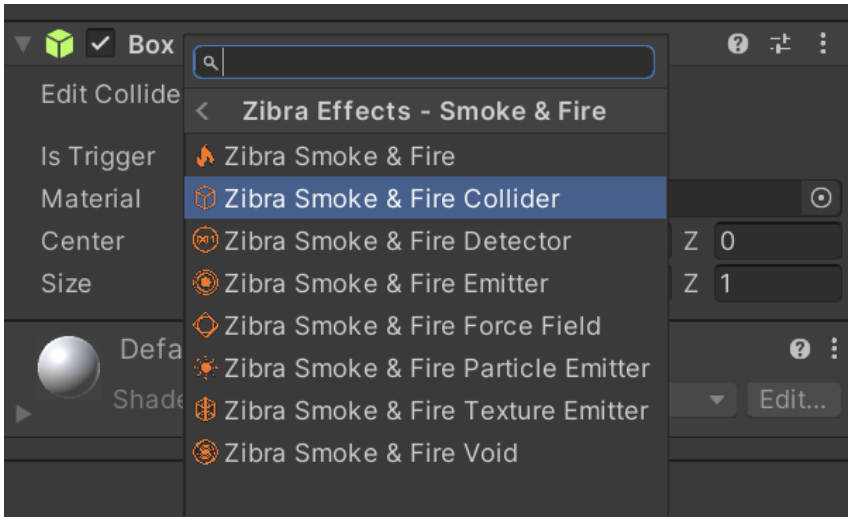


After that your collider is ready to be used. You can set its transform in the editor, or move it at runtime.

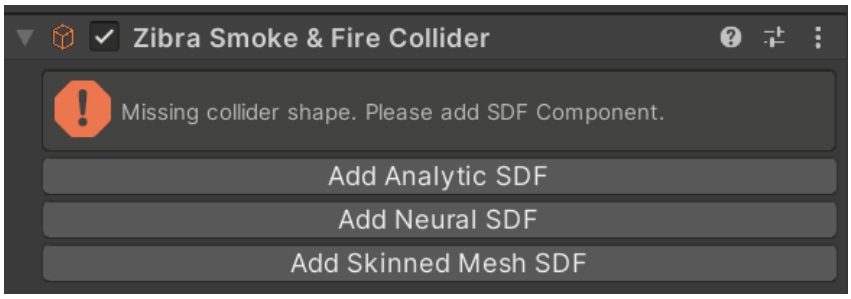
Adding Colliders to existing GameObjects

1. Open inspector for your selected GameObject. Press Add Component and select “Zibra Effects - Smoke & Fire → Smoke & Fire Collider”.

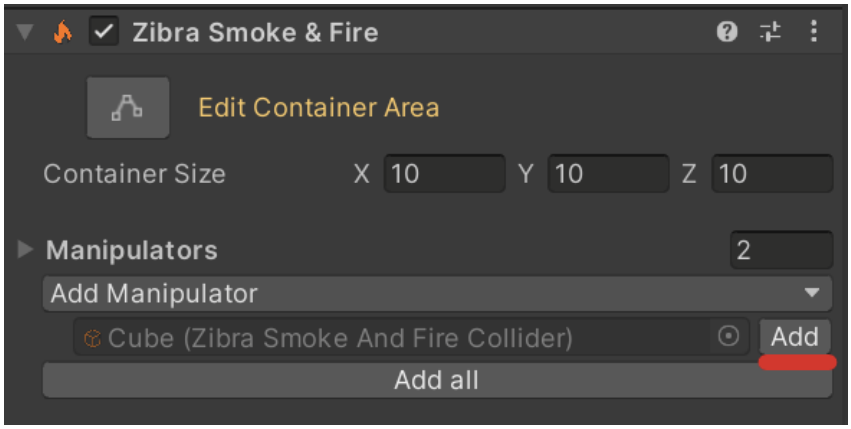




2. In Smoke & Fire Collider add one of the SDF components.
   For more information about SDF components, please see SDFs.



3. Configure newly created SDF component
4. Open Inspector for your Smoke & Fire object, and press the Add button in the list of manipulators to add. You can also use the Add all button to add all colliders to your Smoke & Fire instance.
   You may also need to expand the list of colliders to add by pressing the “Add Manipulator” button first.



After that your collider is ready to be used. You can set its transform in the editor, or move it at runtime.

Collider parameters

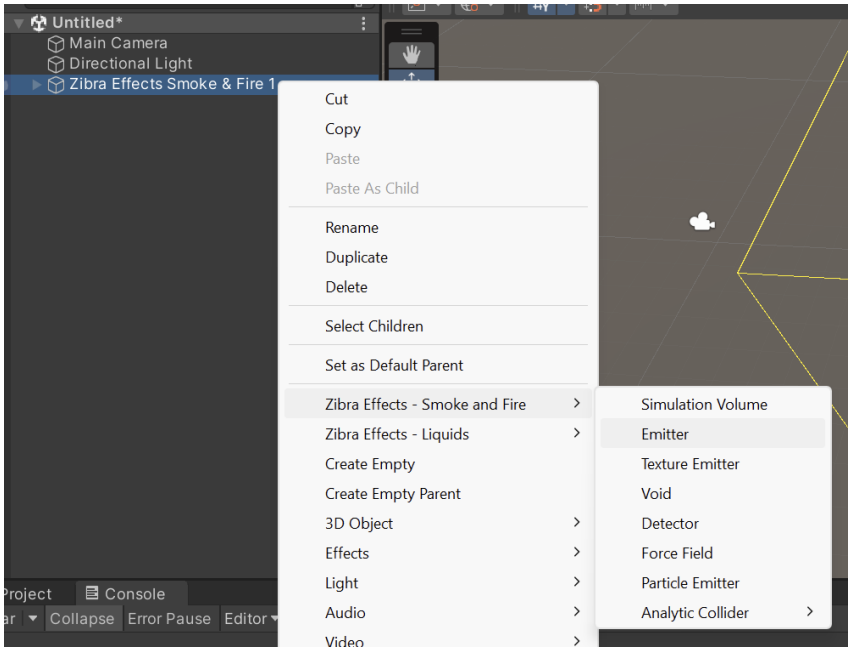
- Fluid Friction
  Controls the surface friction of the collider, used for Smoke And Fire interaction.

Emitter

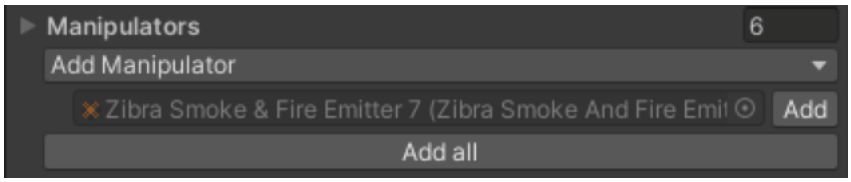
Emitters are used to emit Smoke and/or Fuel at specified temperature.
You need at least one Emitter or Texture Emitter for simulation to have any smoke or fire.
When you first create Zibra Smoke And Fire, one Emitter is automatically created and added to it.

Adding Emitter

1. Right click in Hierarchy, and select “Zibra Smoke And Fire → Emitter”.
   Note that if you right click specifically on your Smoke And Fire object, as shown on the screenshot, you can skip step 2.



2. Open Inspector for your Smoke And Fire object, and press the Add button in the list of manipulators to add. You can also use the Add all button to add all manipulators to your Smoke And Fire instance.
   You may also need to expand the list of colliders to add by pressing the “Add Manipulators” button first.



Emitter parameters

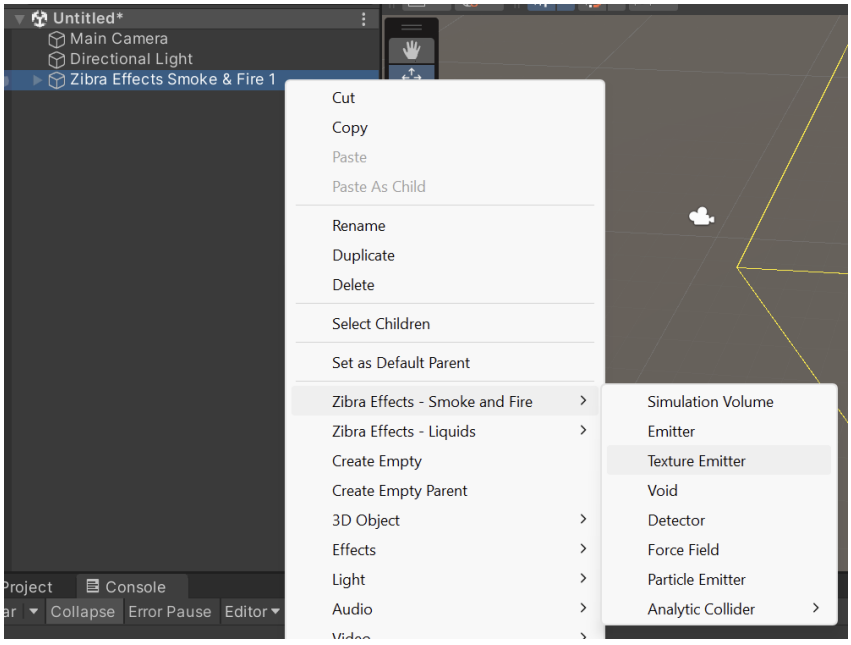
- **Initial Velocity**
  Initial speed of the emitted smoke/fuel.
- **Smoke Color**
  Color of emitted smoke. Only has effect in Colored Smoke simulation mode.
- **Smoke Density**
  Amount of smoke emitted each simulation iteration.
- **Emitter Temperature**
  Temperature of emitted fuel. Only has effect in Fire simulation mode.
- **Emitter Fuel**
  Amount of fuel emitted each simulation iteration
- **Use Object Velocity**
  Whether to apply velocity of emitter GameObject to emitted smoke/fuel. If enabled, sum of GameObject's velocity and Initial Velocity is used for smoke/fuel emission.

Texture Emitter

Texture Emitters are used to emit Smoke and/or Fuel at specified temperature while having more fine grained control over emission. Texture Emitters have 3D texture that you must write to, to specify how much smoke/fuel will be emitted, and at what temperature.
You can also combine data in 3D texture with SDF, to further fine tune the emission volume.
You need at least one Emitter or Texture Emitter for simulation to have any smoke or fire.

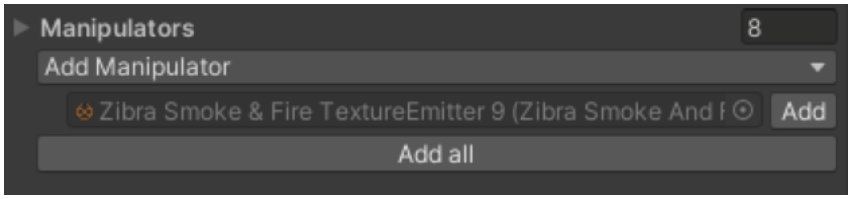
Adding Texture Emitter

1. Right click in Hierarchy, and select “Zibra Smoke And Fire → Emitter”. Note that if you right click specifically on your Smoke And Fire object, as shown on the screenshot, you can skip step 2.



2. Open Inspector for your Smoke And Fire object, and press the Add button in the list of manipulators to add. You can also use the Add all button to add all manipulators to your Smoke And Fire instance.
   You may also need to expand the list of colliders to add by pressing the

“Add Manipulators” button first.



Texture Emitter parameters

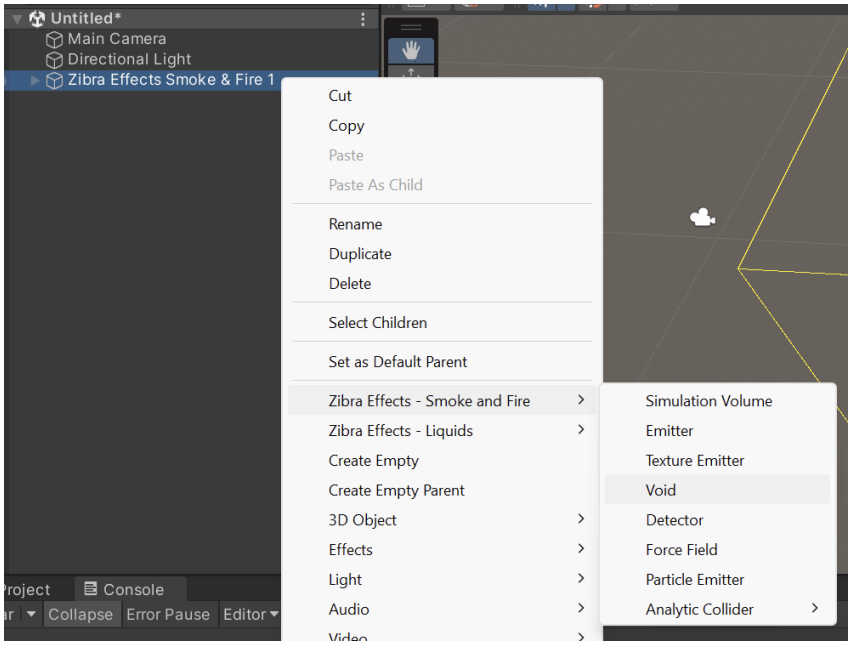
- **Emitter Texture**
  Texture used to control emission. Each texel in the texture corresponds to a point inside the Texture Emitter. Texel value defines emission parameters. You can set it via script, but you must do that before the simulation gets initialized. If not filled with correct data, emitter will not emit anything
  Texture data:
    - In case of Fire simulation mode
        - R - Smoke Density
        - G - Emitter Fuel
        - B - Emitter Temperature
    - In case of Colored smoke simulation mode
        - RGB - Smoke color * density
            - Note that brighter colors will have more density
            - Note that you can’t emit black smoke
    - In case of Smoke simulation mode
        - RGB - Smoke density (average value of R G and B is used)
- **Initial Velocity**
  Initial velocity of the emitted smoke/fuel.
- **Use Object Velocity**
  Whether to apply velocity of emitter GameObject to emitted smoke/fuel. If enabled, sum of GameObject’s velocity and Initial Velocity is used for smoke/fuel emission.

Void

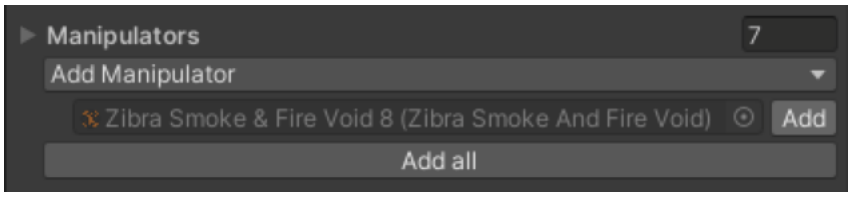
Voids are used to remove Smoke and/or Fuel. You don't necessarily need to remove them manually, and just set up smoke/fuel to dissipate. Also, Voids can create negative pressure, pulling smoke/fuel inside.

Adding Void

1. Right click in Hierarchy, and select "Zibra Smoke And Fire → Void".
   Note that if you right click specifically on your Smoke And Fire object, as shown on the screenshot, you can skip step 2.



2. Open Inspector for your Smoke And Fire object, and press the Add button in the list of manipulators to add. You can also use the Add all button to add all manipulators to your Smoke And Fire instance.
   You may also need to expand the list of colliders to add by pressing the "Add Manipulators" button first.



Void parameters

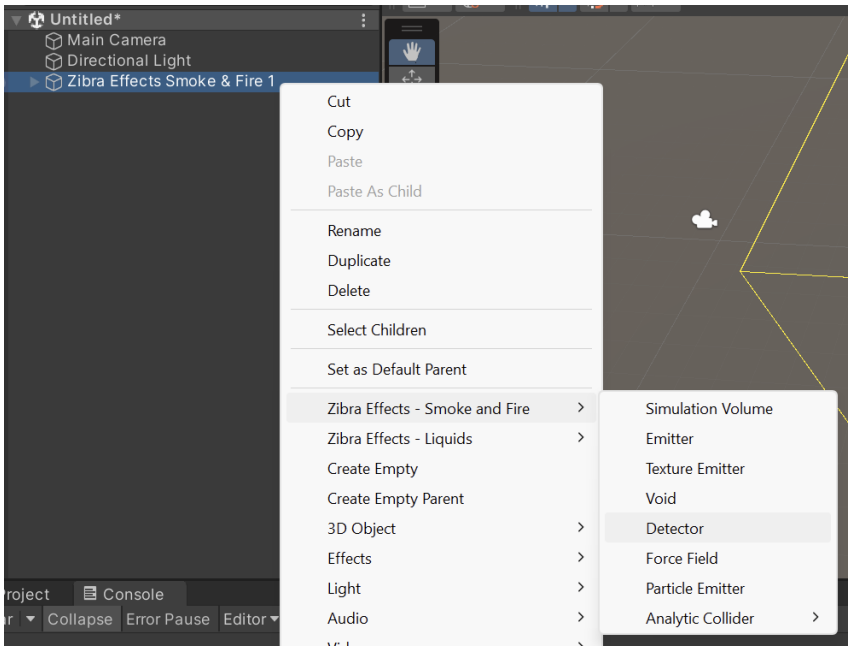
- **Color Decay**
  Controls how quickly the amount of smoke or fuel is decreasing. 1.0 will have no decay, 0.5 will remove half of the total density each simulation iteration.
- **Velocity Decay**
  Controls how quickly the velocity of the smoke or fire is decreasing each frame. 1.0 will have no decay, 0.5 will remove half of the velocity each simulation iteration.
- **Pressure**
  This parameter controls how much pressure to add to the given region. Negative values will suck in the smoke, positive values will move the smoke away.

Detector

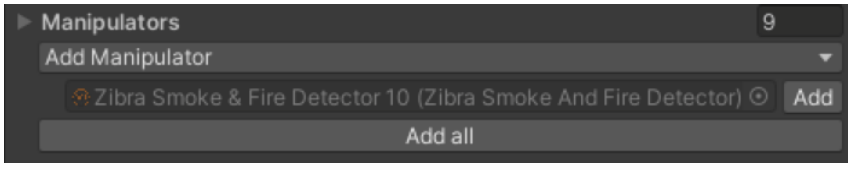
Detector is used to read various data from the simulation. It can also be used to light up the scene based on the simulated fire.

Adding Detector

1. Right click in Hierarchy, and select “Zibra Smoke And Fire → Detector”. Note that if you right click specifically on your Smoke And Fire object, as shown on the screenshot, you can skip step 2.



2. Open Inspector for your Smoke And Fire object, and press the Add button in the list of manipulators to add. You can also use the Add all button to add all manipulators to your Smoke And Fire instance.
You may also need to expand the list of colliders to add by pressing the “Add Manipulators” button first.



Detector parameters

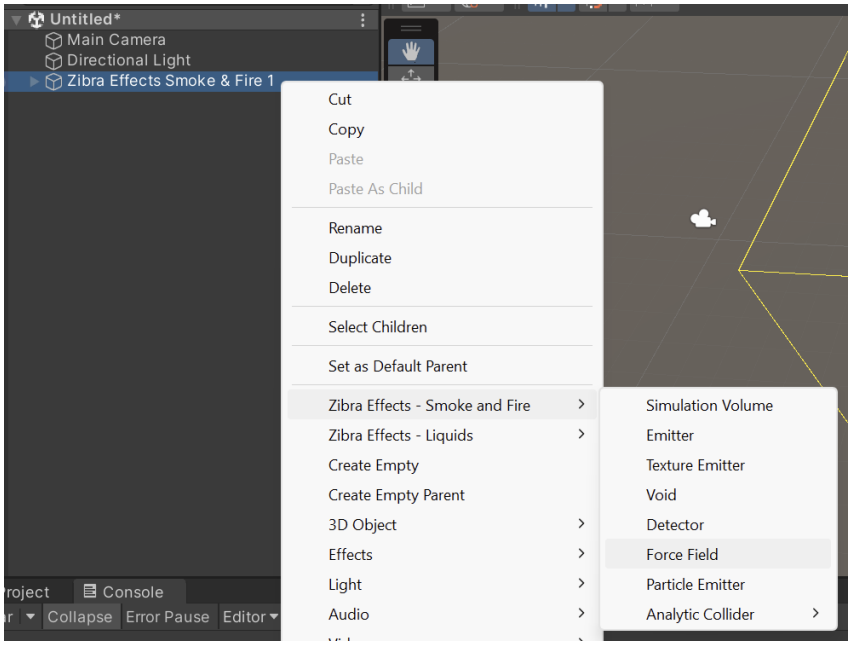
- **Light To Control**
  Selects light that will be used by this detector to pass light from simulation to the scene. Optional.
- **Relative Brightness**
  Scale of brightness set to the light. Does nothing if Light To Control is set to None
- **Average brightness** (read-only)
  Average brightness inside the detector.
- **Relative center of illumination** (read-only)
  Center of illumination relative to detector volume.
- **Avg. smoke amount** (read-only)
  Average amount of smoke inside the detector.
- **Avg. fuel amount** (read-only)
  Average amount of fuel inside the detector.
- **Avg. heat energy** (read-only)
  Average amount of heat energy inside the detector.
- **Volume** (read-only)
  Calculated volume of the detector.

Force Field

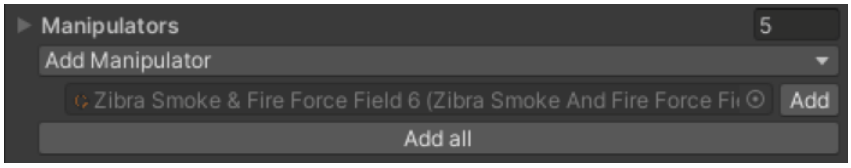
Force fields are used to apply force to the simulation.

Adding Force Field

1. Right click in Hierarchy, and select “Zibra Smoke And Fire → Force Field”. Note that if you right click specifically on your Smoke And Fire object, as shown on the screenshot, you can skip step 2.



2. Open Inspector for your Smoke And Fire object, and press the Add button in the list of manipulators to add. You can also use the Add all button to add all manipulators to your Smoke And Fire instance.
   You may also need to expand the list of colliders to add by pressing the “Add Manipulators” button first.



Force Field parameters

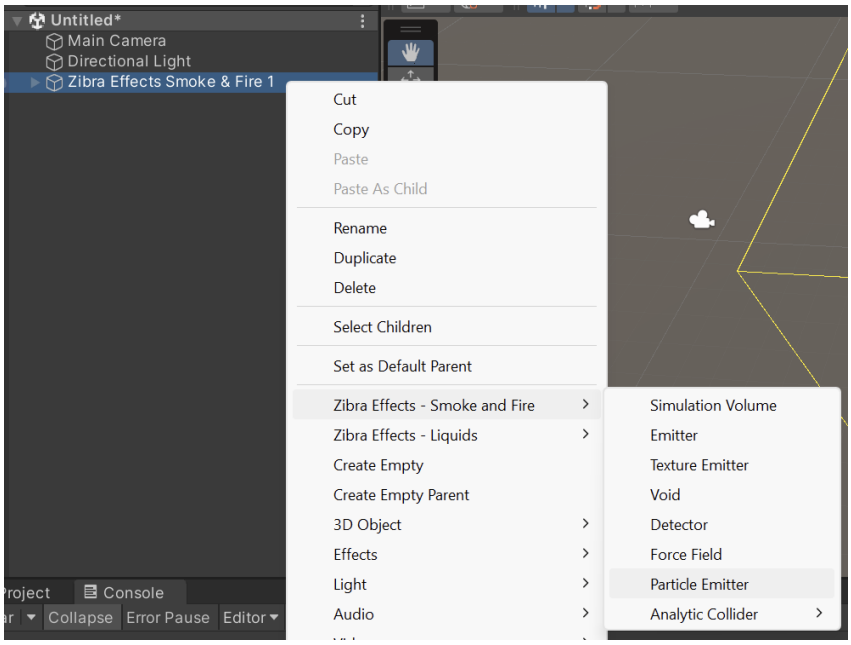
- **Type**
  Configures how Force Field applies force.
  Available types are:
  - Random - Applies random force.
  - Directional - Applies force in the specified direction.
  - Swirl - Creates a whirlpool force around the Force Field.
- **Strength**
  Strength of the force applied by Force Field. Can be negative to reverse direction.
- **Speed**
  Speed of changing randomness. Only has effect when Type set to Random.
- **Random Scale**
  Size of the random swirls. Only has effect when Type set to Random.
- **Force Direction**
  Direction of the force applied by force field. Only used for Directional or Swirl Force Field.

Particle Emitter

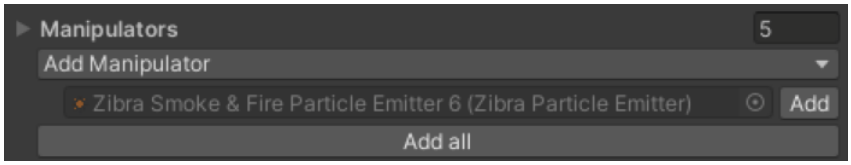
Emitter for Effect Particles. Effect Particles do not affect main Smoke & Fire simulation, and are simulated separately, but they are moved based on the main simulation.

Adding Particle Emitter

1. Right click in Hierarchy, and select “Zibra Smoke And Fire → Particle Emitter”.
   Note that if you right click specifically on your Smoke And Fire object, as shown on the screenshot, you can skip step 2.



2. Open Inspector for your Smoke And Fire object, and press the Add button in the list of manipulators to add. You can also use the Add all button to add all manipulators to your Smoke And Fire instance.
   You may also need to expand the list of colliders to add by pressing the “Add Manipulators” button first.



Particle Emitter parameters

- **Emitter particles per frame**
  Number of Effect Particles emitted per simulation iteration.
- **Render Mode**
  How particles are going to be rendered
  Options:
    - Default - particle visuals are fully controlled via gradients depending on their lifetime.
    - Sprite - sprite will be used to render each particle.
- **Particle Color** (Default Render Mode only)
  Curve that defines color depending on the particle's lifetime.
- **Particle Motion Blur** (Default Render Mode only)
  Scale for motion blur of a particle.
- **Color Oscillation** (Default Render Mode only)
  Define oscillation of particle color with time.
- **Particle Sprite** (Sprite Render Mode only)
  Sprite that will be used to render particles.
- **Particle Brightness**
  Particle’s relative brightness.
- **Size Curve**
  Curve that defines size depending on the particle's lifetime.
- **Size Oscillation**
  Define oscillation of particle size with time.

Note: Curves define the relationship between % of particle lifetime passed and specified values. If a particle goes out of simulation volume, it is immediately destroyed in the middle of lifetime, and so parameters do not reach values specified at the end of gradients.

SDFs

Each manipulator has an SDF component that defines its shape.
You can switch an SDF component for another one, e.g. you can create a force field object, and then switch from analytic SDF to neural SDF.
Those SDF components also have some parameters you can change.

SDF Components

- **Analytic SDF**
  Define one of simple shapes
    - Sphere
    - Box
    - Capsule
    - Torus
    - Cylinder
- **Neural SDF**
  Define complex shape based on static mesh
    - Requires offline generation in Editor
        - Please note that generation sends your mesh to ZibraAI servers for processing
- **Skinned Mesh SDF**
  Define complex shape based on skinned mesh
    - Requires offline generation in Editor
        - Please note that generation sends your mesh to ZibraAI servers for processing
- **Terrain SDF**
  Define terrain shape
    - Can be edited in runtime by modifying terrain
    - Can only be used by Liquid collider

SDF Parameters

- **Chosen SDF type** (Analytic SDF only)
  Defines which shape SDF has.
- **Invert SDF**
  Inverts your shape. For example inverted box will include all points in space, except those that are inside box
- **Surface Distance**
  Offset for distance to surface. With this parameter you can make the SDF a little bit “smaller” or “bigger”.

To adjust size, position and rotation of an SDF, adjust the transform of the GameObject containing corresponding SDF.

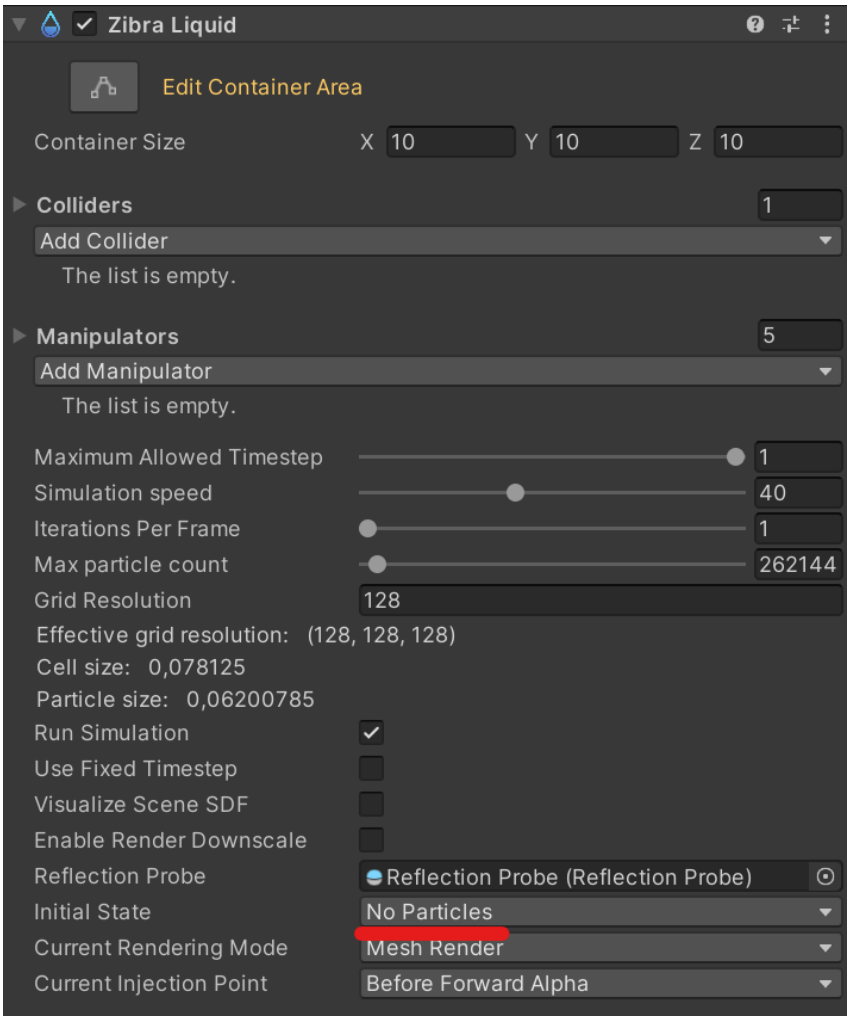
Liquid Initial State Baking

If you want to have a Zibra Liquid instance pre-filled on startup you will need to use Initial State baking.
After baking, the Zibra Liquid instance will remember its initial state and will start simulation from it. Note that we do not support baking animation, simulation over time, etc. Simulation itself is always performed on the device in real time, only the initial state is baked.
Initial State Baking will simulate the first N seconds of your liquid instance, save it, and simulation in play mode will start from that point.
Note that after changing some parameters you'll have to rebake your liquid, specifically: Grid Resolution, Container Size (in case you changed the size non-uniformly, and as a result changed the Effective grid resolution), Max particle count (in case your baked state has more particles that the current max number).

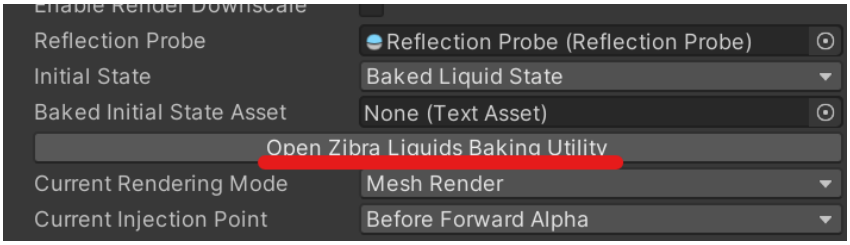
To bake initial state:

1. If you haven't already, add all emitters, voids, and colliders that you are going to have to your liquid.
2. Open the baking utility and select your Zibra Liquid instance to bake.

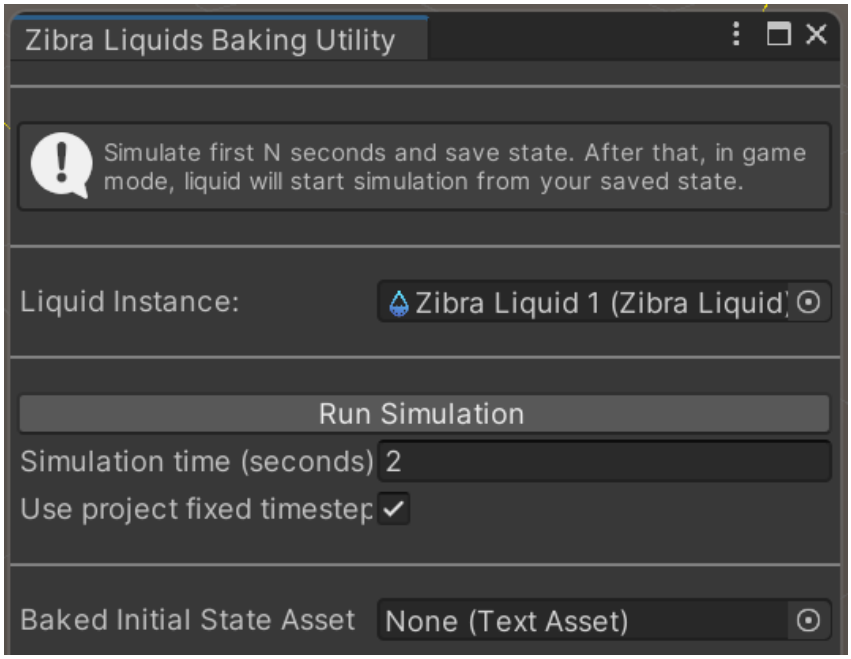
2.1. Open the Zibra Liquid Inspector and set Initial State to “Baked Liquid State”.



2.2. Press “Open Zibra Liquids Baking Utility”

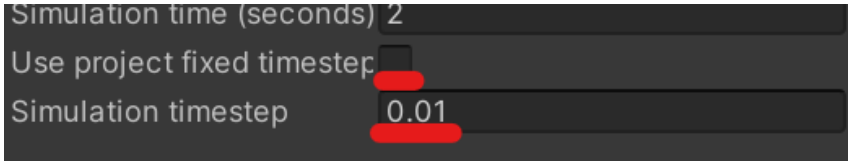


3. Now you see the Baking Utility window

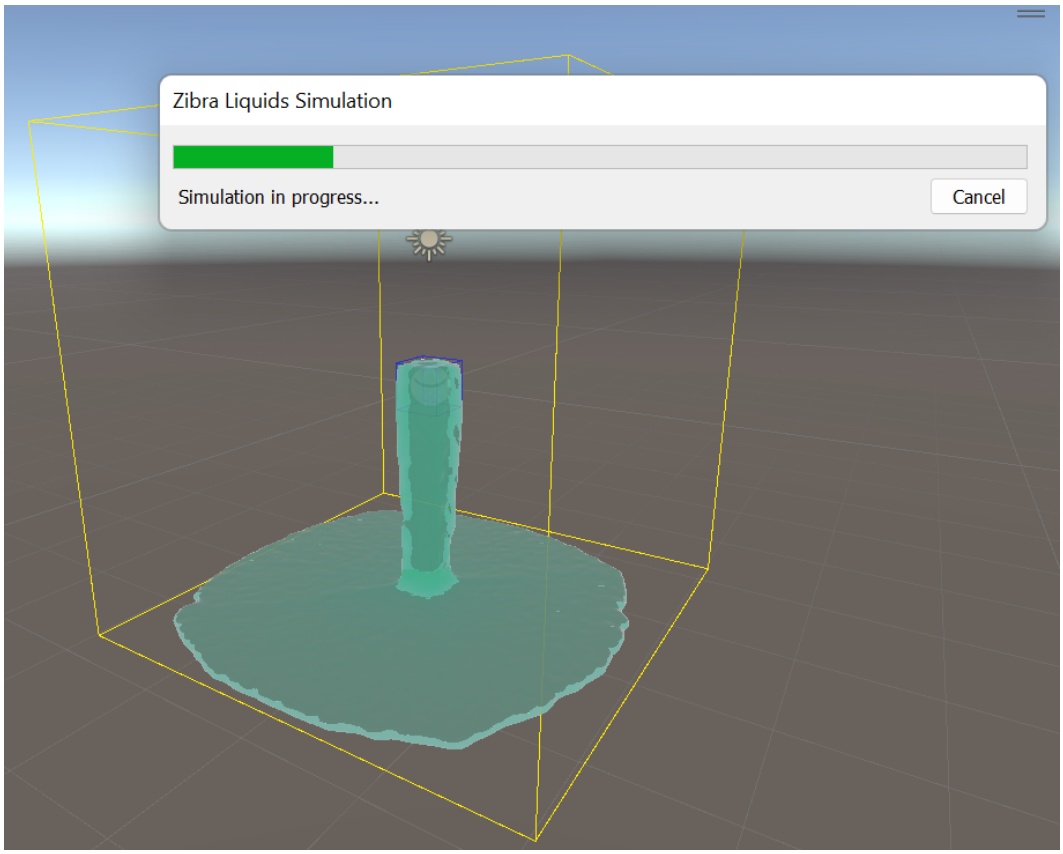


4. Set the Simulation time to the number of seconds you want to simulate.

5. (optional) If you want to run a baking simulation with specific timestamp, you can disable “Use project fixed timestep” and set your desired “Simulation timestep”

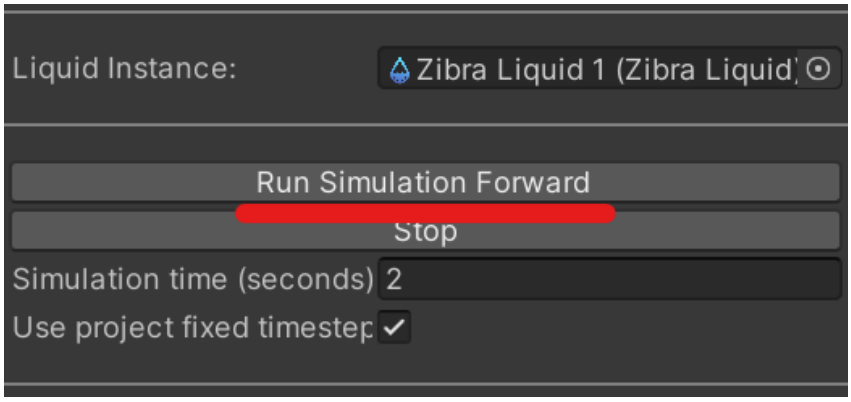


6. Press “Run Simulation”.
You’ll see the simulation running in Scene/Game view (only Game view on URP)

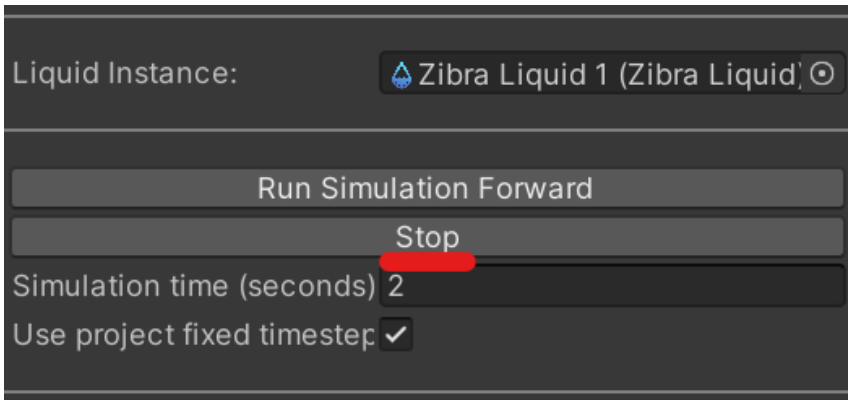


7. (optional) If you want to run the simulation further, set the Simulation time to the amount of time you want to run simulation forward, and press “Run

Simulation Forward”.



8. (optional) If you simulated too far, and want to start over, press “Stop” and go back to step 4

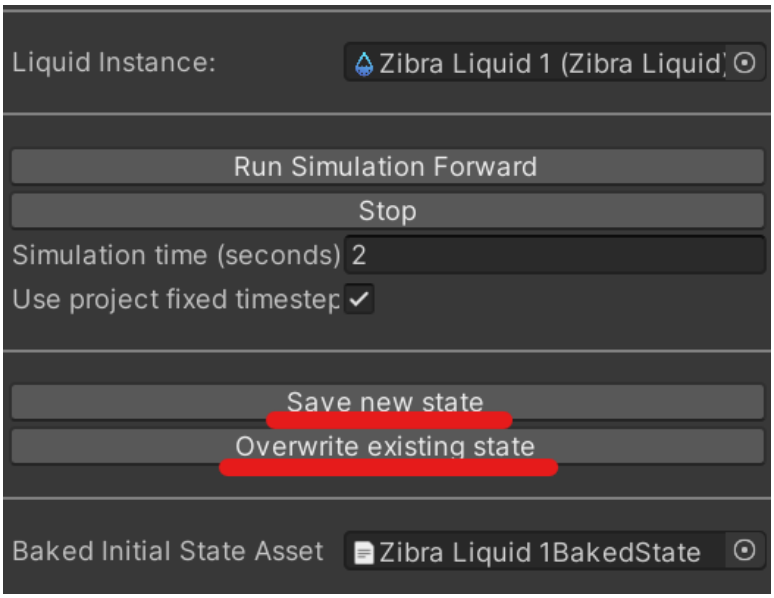


9. Press “Save new state” to save baked state.

That will save the new state to the file in the subdirectory next to your scene file. e.g. for Liquid Instance “ZibraLiquid1” in

"/Assets/NewScene.unity" scene, liquid baked state will be saved into "/Assets/NewScene/" folder as "ZibraLiquid1BakedState.bytes".

9.1. If you already had a previous saved state you'll have 2 buttons: "Save new state" and "Overwrite existing state".



9.2. "Save new state" creates a new file, while "Overwrite existing state" overwrites the old file with a new state.

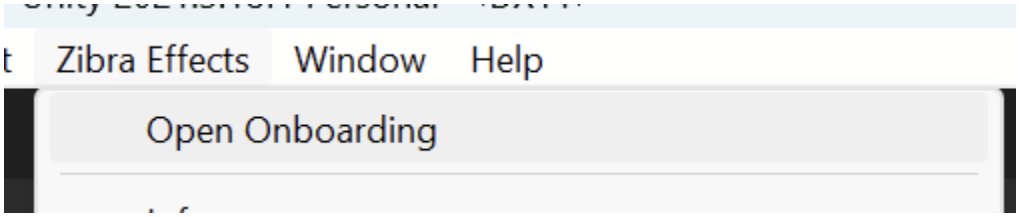
10. (optional) After you are done with baking, press "Stop" to stop the liquid simulation and unlock the parameters that are disabled when the liquid is "live". Next time you'll change a scene or enter playmode this state will reset, so you usually don't need to manually stop it.

Registering Zibra Effects

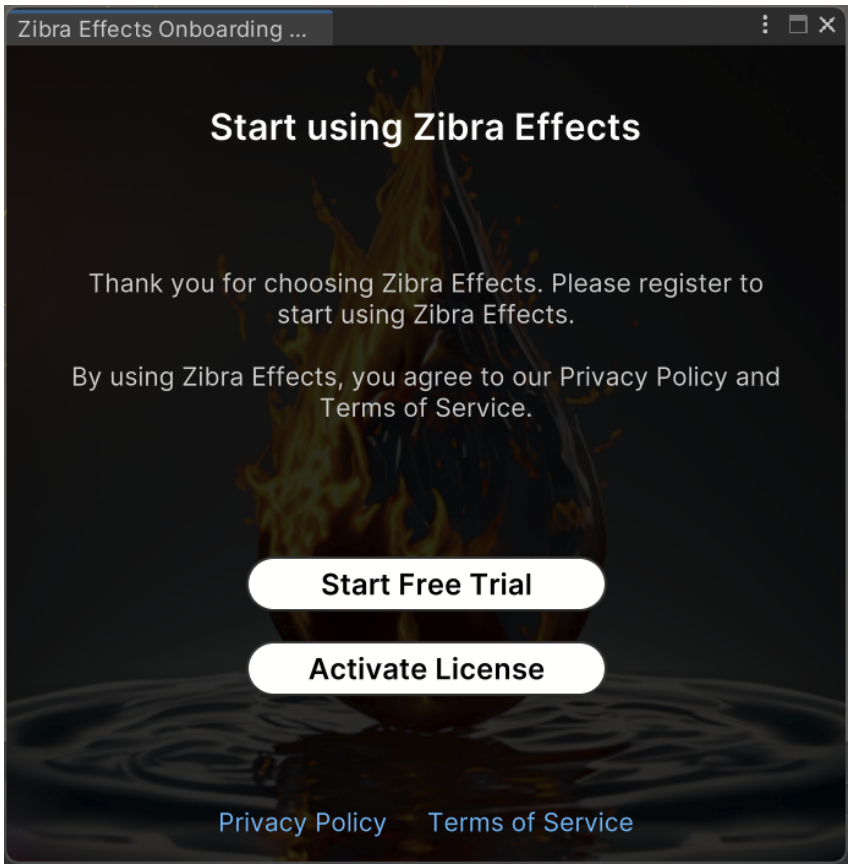
To use certain features you need to validate your license.
Zibra Effects requires license validation before you can

To register your copy of Zibra Effects:

1. If Onboarding window didn't open automatically, In the menu bar, select "Zibra Effects → Open Onboarding"

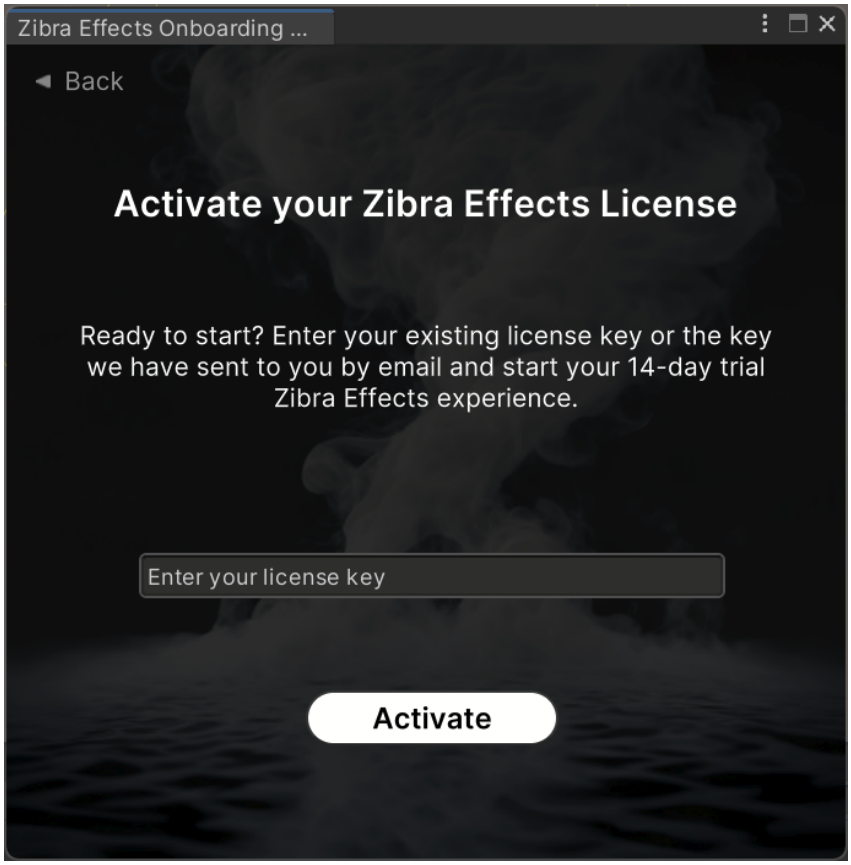


2. In the onboarding window select Stat Free Trial if you don't have license already, or Activate License otherwise

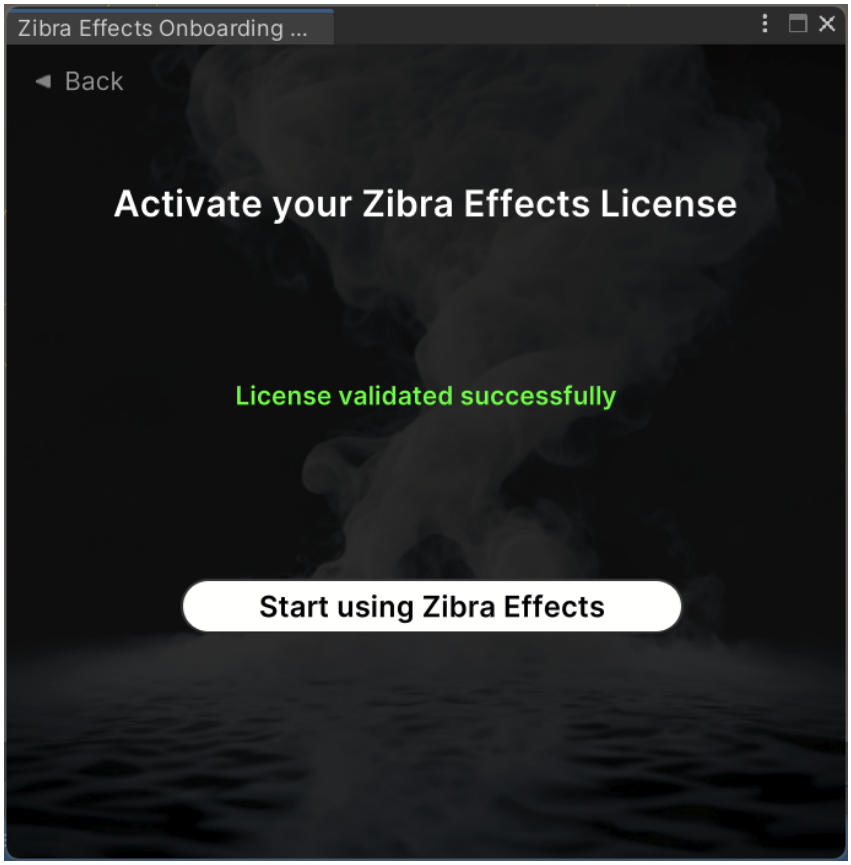


3. If you selected Start Free Trial, sign up and receive your license key

4. Enter your license key

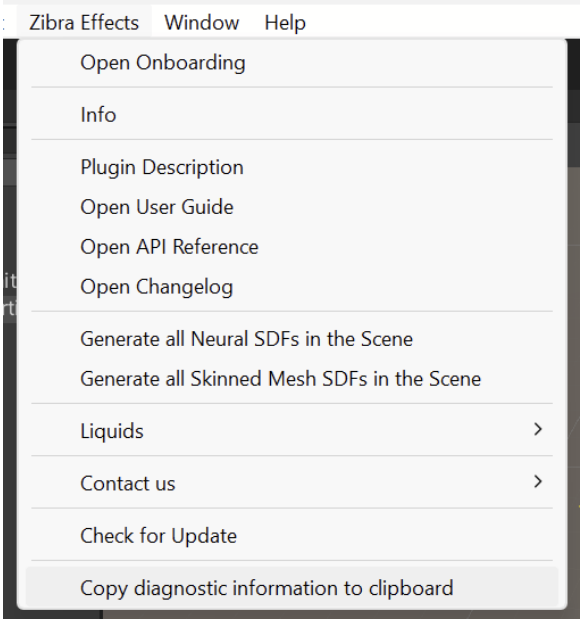


5. After successful validation you can now start using Zibra Effects

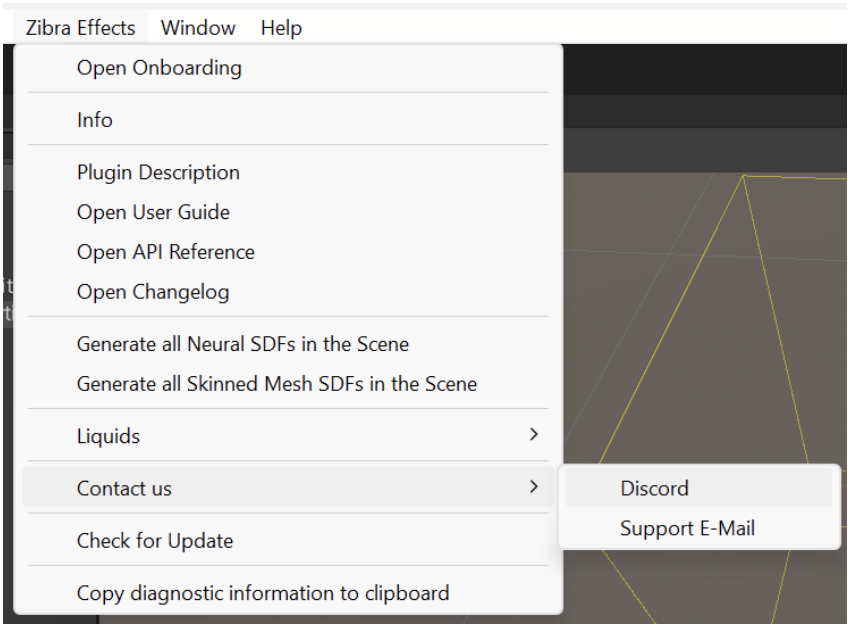


Troubleshooting

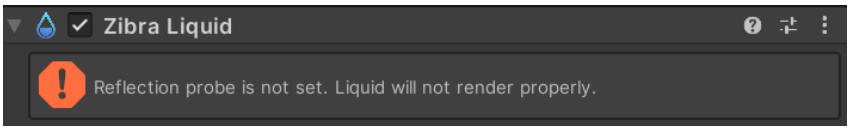
If you have any issues or feedback feel free to contact us on Discord
If you are going to reach out to us for troubleshooting, please export the diagnostics info via this button and send it to us

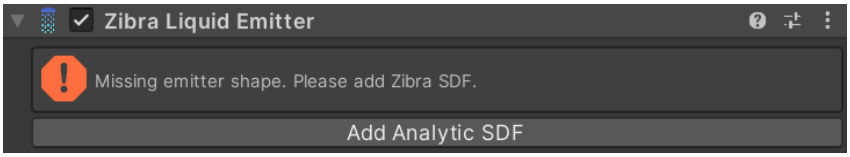


After copying that you can go directly to our discord or email with another buttons



Some common issues you may have and ways to troubleshoot them:

- First thing you want to do in case of any error is to check your Console for any errors.
- Issue: Simulation doesn't work
    - Solution 1: Check whether you have any errors in the Simulation Volume inspector
      e.g.
      

    - Solution 2: (In case of URP) Add "URP Liquid Rendering Component" and enable "Depth buffer"
      See Liquids Additional setup on URP and Smoke & Fire's Additional setup on URP
      Pay extra attention to the fact that mobile and desktop platforms may have separate URP options.
    - Solution 3: Double check System Requirements. We have unsupported platforms (e.g. webGL), Unity version (anything before 2020.3) and some restrictions (For VR to work you need to select Unity Render)
- Issue: manipulators/colliders don't work
    - Solution: check their inspector and ensure all manipulators have an SDF component.
      e.g.
      

- Issue: On HDRP, the Liquid is too bright
    - Solution: Decrease "Reflection color" in Liquid Instance's Material Parameters