See-through Shader

Documentation

Version 1.8
October 2022

https://www.shadercrew.com
support@shadercrew.com

Contents

Introduction

The **See-through Shader** lets you **see your playable characters inside any mesh** without being obstructed by it. It doesn't matter if it is a building, a bridge, a cave, or whatever else you can imagine. You can also use the effect without being dependent on playable characters!
Just apply the shader using one of our provided tools, register a player(Player-based) or not(Independent), and it works. The best thing about it is that it **doesn't need any additional changes to your mesh!**
We included **plenty of super helpful tools to get the effect to work within seconds.**

Please keep in mind that this shader is **meant for use with standard/lit shaders,** and we only support most properties of those.
So if you have a custom shader doing some other fancy stuff, the **See-through Shader** will most likely not help that, as we can not support all possible variations of what a shader could do.

Furthermore, the **See-through Shader** asset is first and foremost a **shader,** and we supply you with methods of applying and using it.

This asset also includes many demos, demonstrating the various ways of applying and controlling the shader, applying Zoning, using the shader for a Third- or First-Person setup, and much more.

What is a shader?

To understand what a shader is, we highly recommend you to read the introduction on https://en.wikipedia.org./wiki/Shader and https://docs.unity3d.com/Manual/shader-introduction.html

We support all 3 RenderPipelines and every stable version of it!

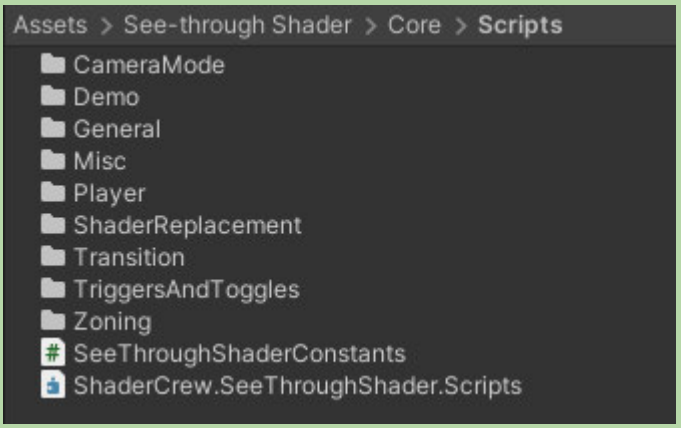
That means we support:

Built-in
URP 2019, URP 2020, URP 2021
HDRP 2019, HDRP 2021, and HDRP 2021.

Note: Please be aware that you do NOT need to understand what a shader is to use our asset. We have plenty of scripts included so that you can entirely focus on the result without worrying about the inner workings.

Structure of Asset Folders

- **core/Scripts**
  These are the main c# scripts folders ordered according usage



- **core/Resources/STSReferenceMaterials**
  Reference Materials with the See-through Shader applied used in the demo scenes

- **core/Resources/STSTreeMaterials**
  Tree Materials used in the demo scenes

- **core/Misc (Deprecated)**
  Helper Scripts

- **core/Editor**
  Scripts for the Unity Editor custom UI

- **core/Icon**
  See-through Shader Logo

- **core/Shader/HDRP**
  HDRP 2019, 2020 and 2021 version of the see-Through Shader

- **core/Shader/Standard-builtin**
  Standard or builtin (Unity default) version of the see-Through Shader

- **core/Shader/URP**
  URP 2019, 2020 and 2021 version of the see-Through Shader

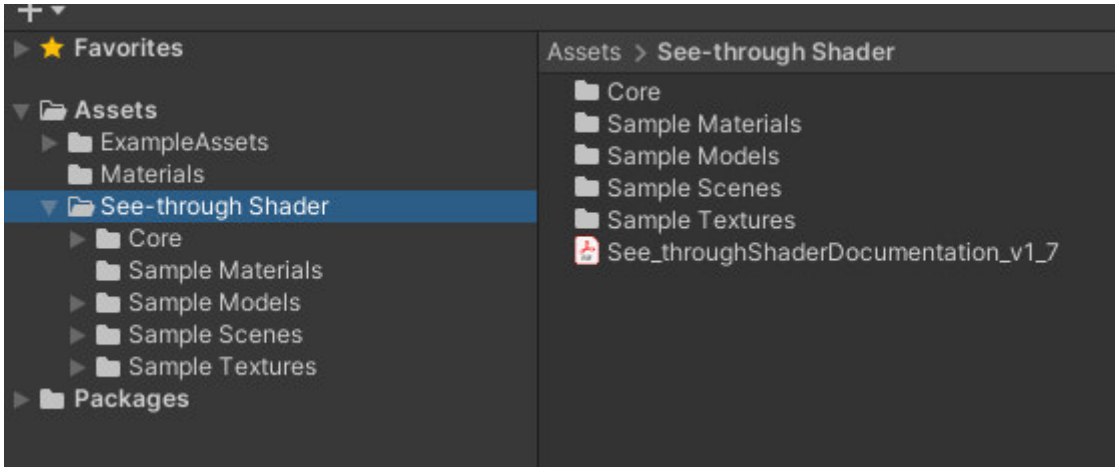
- **Sample Material**
  Optional, here are the material files used in the demo scene

- **Sample Models**
  Optional, here are the building model files for the building used in the demo scene

- **Sample Scenes**
  Optional, here is the demo scene, double click on the scene file to open it and press play

- **Sample Textures**

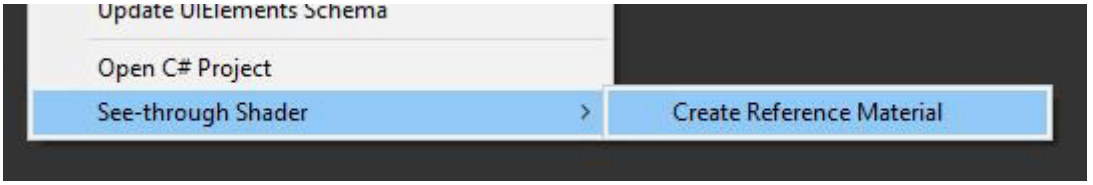
  Optional, here are dissolve textures for the demo and experimenting



Is there a material supplied in the asset package?

Yes, in the ***See-through Shader/Core/Resources/STSReferenceMaterials*** folder, you will find several materials called refMaterial.

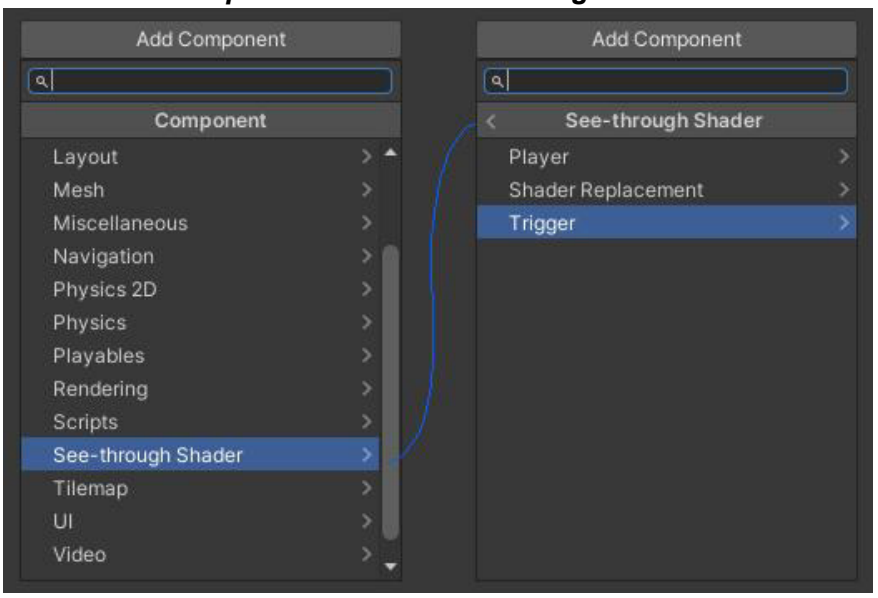
We also added a context menu under ***Assets > See-through Shader > Create Reference Material***. (also located under right-click inside the project window and then ***See-through Shader > Create Reference Material***)



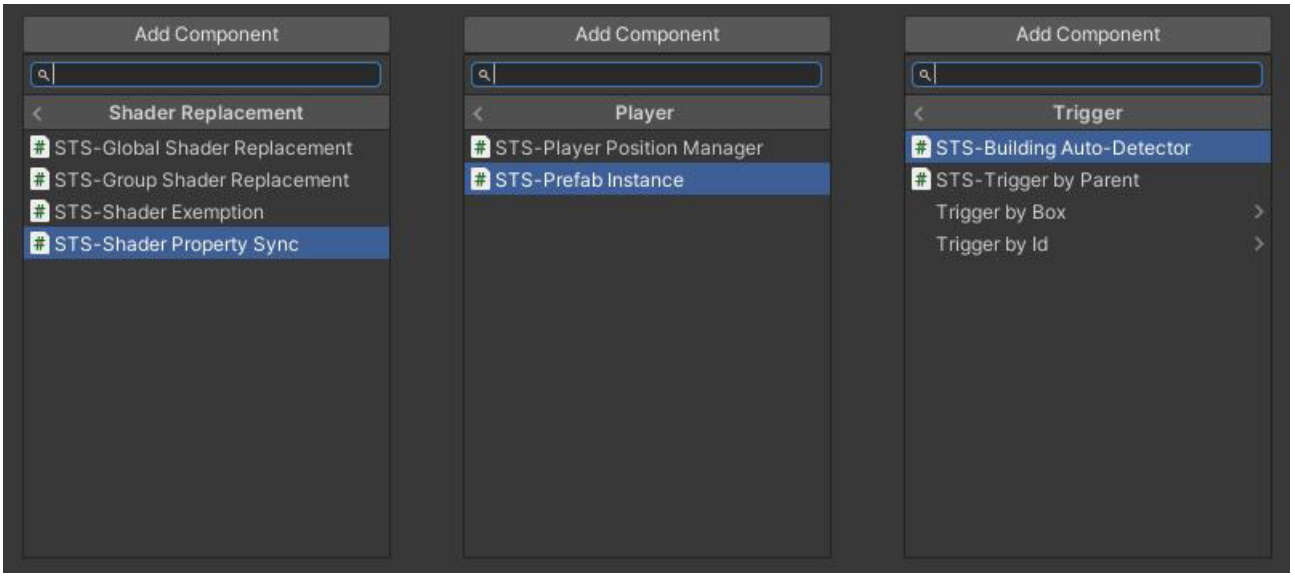
How to Apply the Scripts

Go to your GameObject Inspector and press “Add Component”. After that search for the “See-through Shader”-folder in the component list.
Inside, you will find all scripts you need to use the “See-through Shader”.

Component Menu - See-through Shader



Component Menu - See-through Shader - Sub Folders



How does it work

The **See-through Shader** works on the materials of the gameObject's elements. It does not matter if it's one mesh or thousands of sub-game objects.

There are **two ways of applying the shader** to your buildings:

Either by setting it **manually** as the building's material shader or **automatically** by using either the ***GlobalShaderReplacement.cs***, which applies it **globally** to all GameObjects or the ***GroupShaderReplacement.cs***, which assigns it **locally** to a group, that is, all children of a parent, that this script got attached to.
For more details on this, see *How to apply the shader*

Note: *Both scripts can be limited to specific layers!*

There are **two ways of controlling the effect:**
"Player-based" and "Player Independent"

Player-based

By default, the affected area of the "See-through Shader" is defined by an alterable effect radius, the *Default Effect Radius* property. Still, it can also be explicitly triggered on a per-building/object basis.

There are **two ways of restricting the area of effect to a building/object**:

The recommended way is using Triggers, as they have the best performance and offer the most control! (see chapter on *Triggers*)
When using Triggers, you can choose between letting one collider act as the enter and exit trigger, or having dedicated entrance and exit triggers (e.g., pair of cubes at the entrance) for the buildings, so the effect is applied just when passing those.

We also included a script that automatically detected structures using colliders and the raycast auto-detection algorithm(see chapter on ***BuildingAutoDetector.cs***).

In the most simple setup, *Effect Radius Only*, just apply the **See-through Shader** to your building/object material and add your player to the playable characters position manager, ***PlayerPositionManager.cs***, which can be attached to any GameObject or empty.

The area of the effect of the shader is controlled by setting the size of the *Default Effect Radius*.

If you want the shader to work on a per-building/per-object basis, you can use either *Triggers* or *Auto-Detection*.

Note: Auto-Detection should be used with care and only as a last resort because it requires ongoing calculations to detect buildings!

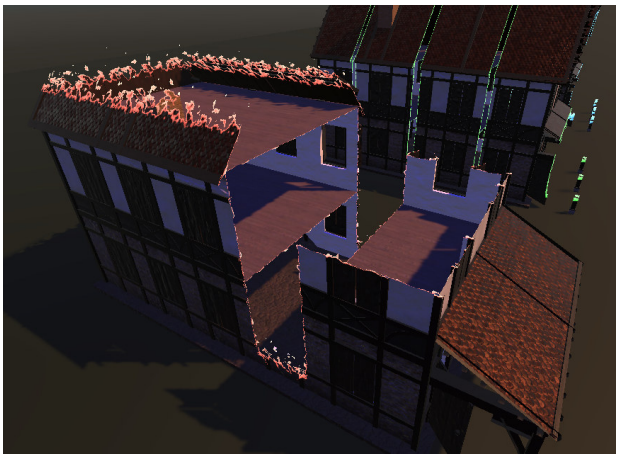
The **See-through Shader** does not need colliders to work. Still, if you want to use it on a per-building basis with enter exit trigger cubes, you need colliders on those objects so players can pass through and activate the enter and exit functions shader or allow the auto-detection algorithm to work.

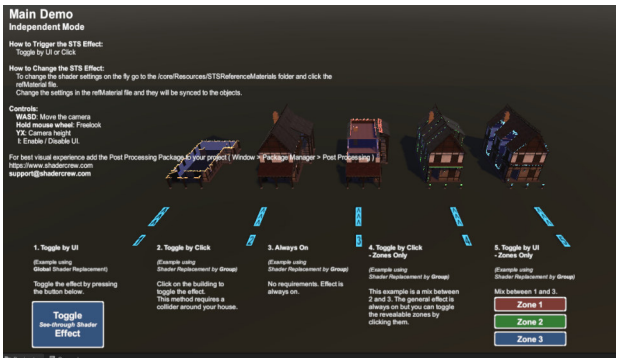
To sum it up, first, you have to attach the ***`PlayerPositionManager.cs`*** to some empty or GameObject and add your playable character to it. Then you have to apply the shader to your buildings. Finally, you can control the effect of the shader by choosing between three different methods of use.

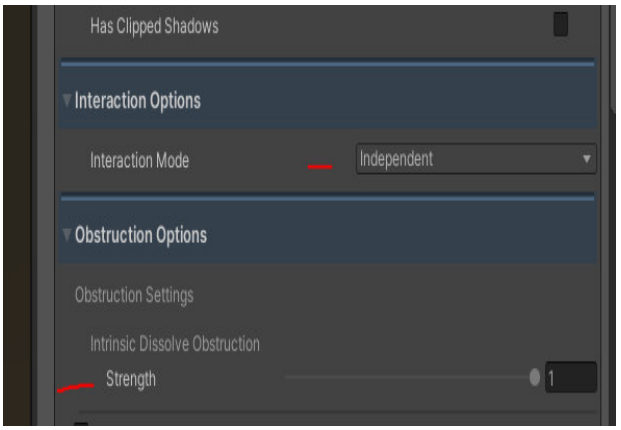
Note: *You can mix every way of assigning and controlling the shader and even use multiple different combinations in the same scene; for example, one building is set up using triggers, some others use the effect radius method, and another GameObject the auto-detection script.*
The See-Through Shader will handle each of them individually.

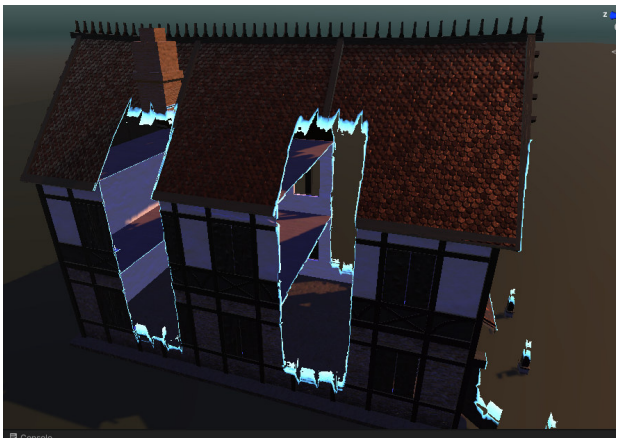
Player Independent

See-through Shader supports a playerless setup called “Independent mode”
This is a non player based setup where you can apply the effect independent of a player and without the Position Manager script.
The Main use case is a click based revealing / dissolving effect and cross sectioning.



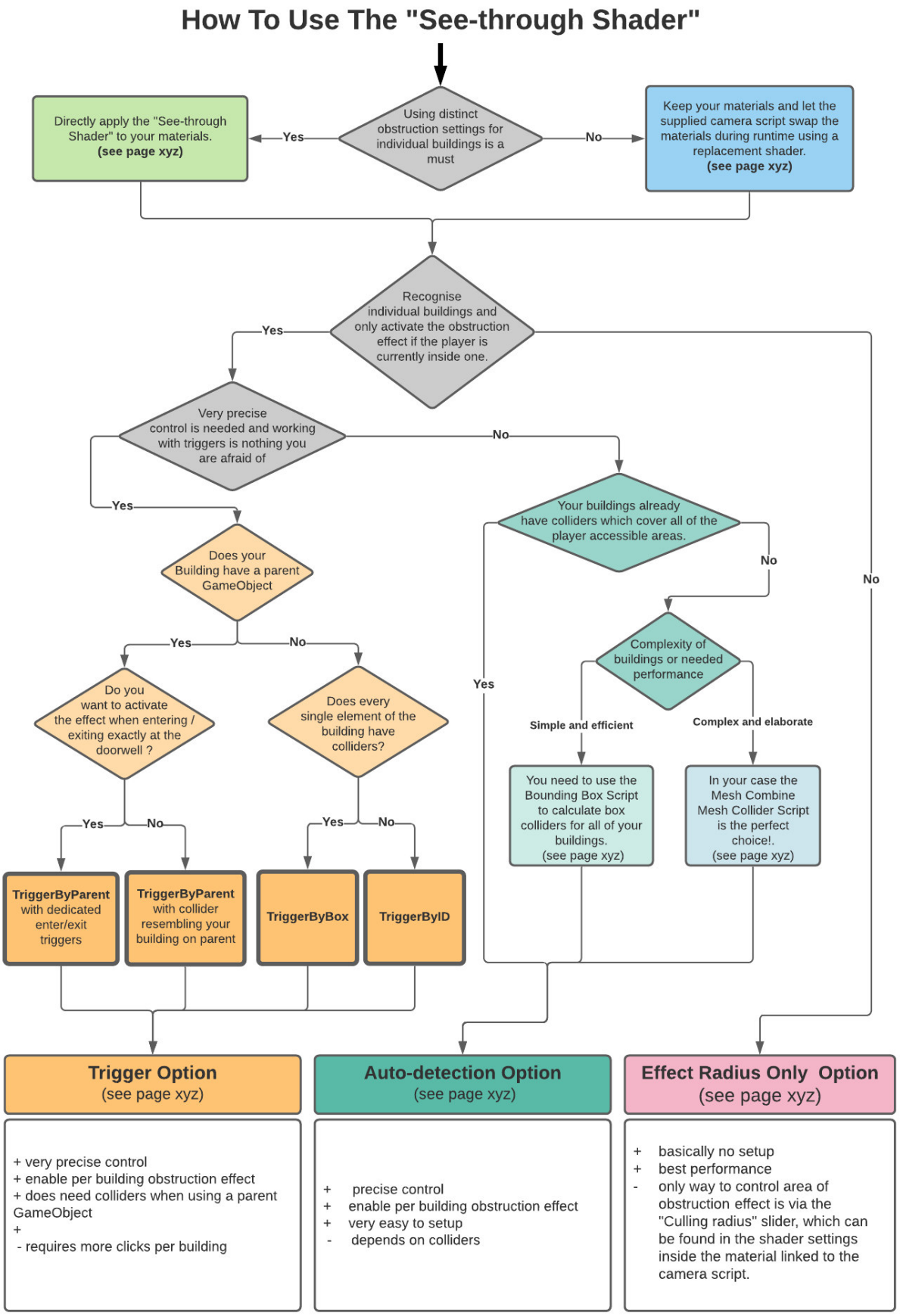






After you applied the shader to your gameobject, you don’t have to do anything else.
To toggle the effect, we added several example scripts like `ToggleByClick.cs`, `ToggleByClickZonesOny.cs`, `ToggleByUI.cs` and `ToggleByClickUIZonesOnly.cs`. Additionally you can see those examples in action in our Independent demo.

Decision schema(Player-based)



How to apply the shader

The first thing you have to do to use this asset is to apply the shader to your buildings/objects. You can either manually do this, see *Manually apply the shader*, or use one of our replacement scripts that automatically applies the “See-through Shader” either to every material of all GameObjects or to a specific group. See *Automatically apply the shader*. You can also limit the reach of those scripts to particular layers, so only GameObject and their materials in distinct layers get affected by the replacement scripts.

Method 1 - Manually apply the shader

To add the shader to your Building, you first need to add a material to your GameObject by clicking **Assets->Create->Materia**l in the project view context menu or from the main menu.
After that, assign the shader to your material using the Shader drop-down menu in the inspector window under Custom.
You could also drag and drop the ***SeeThroughShader.shader*** onto the shader drop-down menu.

For more information regarding seeing the official Unity documentation on this topic:
https://docs.unity3d.com/Manual/materials-introduction.html

Method 2 - Automatically apply the shader

All you have to do is either use the ***GlobalShaderReplacement.cs*** or ***GroupShaderReplacement.cs*** and link a reference material with the “See-through Shader” assigned to it via “Reference Material.”

Global Shader Replacement:

The ***GlobalShaderReplacement.cs*** adds the “See-through Shader” globally to all GameObjects that don’t have the ***SeeThroughShader.shader*** applied already to their materials. You can add this script on an empty or any GameObject.

Group Shader Replacement:

The ***GroupShaderReplacement.cs***, on the other hand, does the same locally.

There are 3 different Group Types to choose from: Parent, Box, Id.

Parent:
Choose a Parent Transform, and all of its children will get the linked reference material with the “See-through Shader” assigned to them.

Box:
Choose a Trigger Box, and all GameObjects inside of this Box will get the linked reference material with the “See-through Shader” assigned to them.

Id:
Choose an Id, and all GameObjects with the chosen TriggerObjectId will get the linked reference material with the “See-through Shader” assigned to them.

Additionally, on both scripts, you have the option to restrict the assignment to GameObjects that reside on selected layers.

How to control the effect of the shader

Player-Based

You have three different options of how to manage the effect of the “See-through Shader.” The most basic and simple one is the *Effect Radius Only* method, the default way of operating the shader.

Suppose you want the effect to be on a per-building/object basis. You have two choices: Either you can go for the *Auto-Detection* method, using the auto-detection script ***BuildingAutoDetector.cs***, which is very easy to use and doesn’t require a lot of setups.
Or you choose the most precise and most performant way, using the *Triggers* solution, which, opposed to the auto-detection script, doesn’t require any ongoing calculations but, on the other hand, requires you to set up some triggers. Because we know that setting up triggers can be a lot of work, we added a couple of helper scripts to make your life easier.

Summary Of Control Methods

Method 1 - *Effect Radius Only*

This method will restrict the shader affected area of effect within a sphere around the player. The effect size is set by a variable radius value found in the shader inspector settings under *Default Effect Radius*.
It’s the easiest and fastest way of using the shader but doesn’t work on a per-building/object basis, which means when the radius is large enough, the shader-affected area is likely to spread among multiple buildings.
This is the **default mode of the shader**, which is **activated if neither the Trigger nor the Auto-Detection method is used** for any individual building.
Additionally, this solution doesn’t support the transition feature, which is limited to the Auto-Detection and Trigger method.

Method 2 - *Auto-Detection*

An easy way to set up but contrary to the “Effect Radius Only” method, supports per-building/object effect restrictions and the transition feature. It has the worst

performance of all control methods, as constant calculations are required for the algorithm to work.
This solution requires you to attach the ***BuildingAutoDetector.cs*** script to your playable character and assign colliders to your buildings. Those colliders should represent the shape of the building as much as possible, as the logic inside of the BuildingAutoDetector will use the information given by those colliders to determine if the player is inside the building or not.
It's not as precise as the TriggerByParent trigger, for example, which allows for exact door positioning and, as a result, for perfectly accurate enter and exit state changes.

A MeshCombine script, ***MeshCombine.cs***, is included with our asset that lets you create a combined mesh collider out of all the meshes, which can help to improve the detection quality on some more complex buildings.

Method 3 - Triggers

The way **triggers** work is to activate and deactivate the effect when the player passes the trigger objects (simple cubes with colliders in isTrigger=true mode in most cases).
Triggers will work very precisely as they can be set up individually for every building, and they work with simple and large complex structures with many sub-elements.

The most used and recommended trigger is the TriggerByParent script (TriggerByParent.cs); this will iterate from a parent GameObject of the building through every element matching a preselected layer.

The other two triggers are for:
There is no parent GameObject, but elements have colliders so that a surrounding box can capture inside colliders (TriggerByBox).

There is no parent GameObject, and the elements have no colliders; you can assign an ID (h; the123, etc.) to every part belonging to a particular building and use the TriggerByID script; the script will then when the player passes the enter/exit colliders find the IDs belonging to the corresponding house.

All the shader modes (Circle, Angle, None, etc.) and the features work with all triggers.

Detailed Description of the Individual Triggers

Trigger by Parent

The TriggerByParent script does not need colliders for the elements; **the effect is triggered by one collider surrounding the whole building or one or more pairs of dedicated entrance and exit cubes** with colliders placed at the entry.

Pro:
- **Most precise and easy to set up**
- **No colliders needed for building elements**

Con:

- **Parent GameObject (container) needed**

Trigger by Box

The idea behind this trigger is to capture the building and or the building elements that have colliders with a surrounding box.
This can be the case when the **scene hierarchy can not be changed**, so the buildings or structures have **no parent GameObject.**
There are three components for this setup: the trigger box surrounds the building and two or more dedicated triggers for the player to pass.
The surrounding box GameObject is dragged onto the trigger objects.
When the **player passes the triggers, the script will capture all the colliders on a specific layer with the help of the surrounding box**.
This trigger can also work well with more dynamic or procedural generated scenes as it captures everything inside a box no matter of the hierarchy in the scene

Pro:

- **Great and effective solution when no parent containers are possible**
- **Works well in dynamic procedural generated scenes**

Con:

- **Little less robust because of the nature of colliders**
- **All building elements need colliders**

Trigger by ID

This was the last use case that came along when developing this asset, so here we thought of **a use case where you do not have any "anchor points" like a parent or colliders**.
All the **building's elements get a manual id that identifies them together as one building** in this use case.
This can be beneficial in dynamic procedurally generated scenes where you add the ids programmatically or for smaller setups manually.
For **Levels with many elements, we do not recommend adding the ids manually**.
This Trigger comes with two or more dedicated trigger cubes that refer to the id assigned to the elements. The trigger will iterate through the scene to find the corresponding building elements according to his ID when the Scene starts.
It is helpful, and there are use cases where you need this trigger, but we recommend the other trigger types as they are faster to set up and have no impact on the load time of the scene.

Pro:

- **No parent GameObject needed**
- **No colliders needed for building elements**

Con:

- **Manually setup of IDs for building elements**
- **Increase load time at start**

As said earlier; the supplied C# scripts are wrappers for the shader to apply the enter and exit action.
Feel free to study the trigger script codes and write your custom trigger; if you have any questions for your specific use case, please email us; we will be happy to support you.

Note: It is best to drag a selection box in unity editor over the elements. With a second hierarchy window open, move them beneath a parent GameObject where you can use the “Trigger by Parent” script.

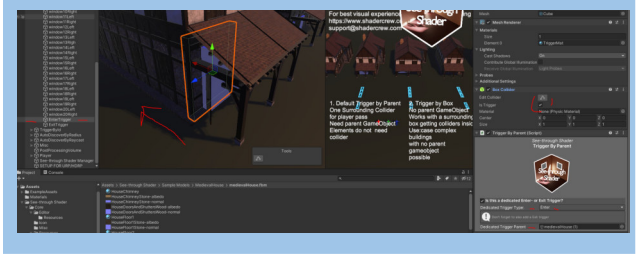
Overview trigger use cases

TriggerByParent (TriggerByParent.cs)

Global Enter / Exit Collider on the parent object

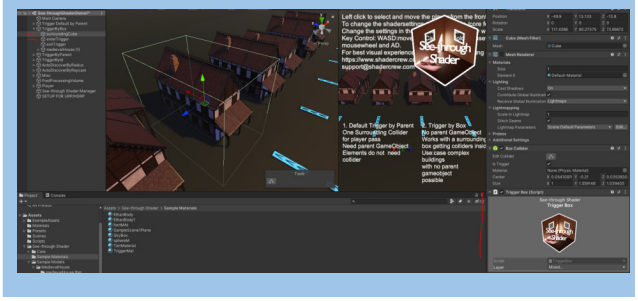
TriggerByParent (TriggerByParent.cs)

Dedicated Enter and Exit trigger objects Most accurate



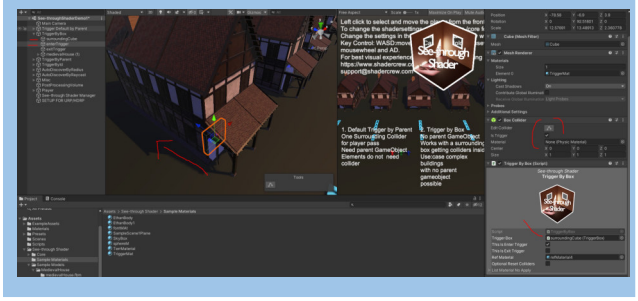
TriggerByBox (surrounding box)

Surrounding TriggerBox capturing elements with colliders inside for the TriggerByBox Use case



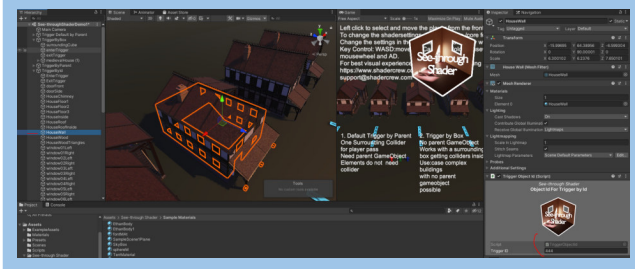
TriggerByBox Enter / Exit Triggers

Dedicated Triggers used to apply the effect to the elements captured within the TriggerBox



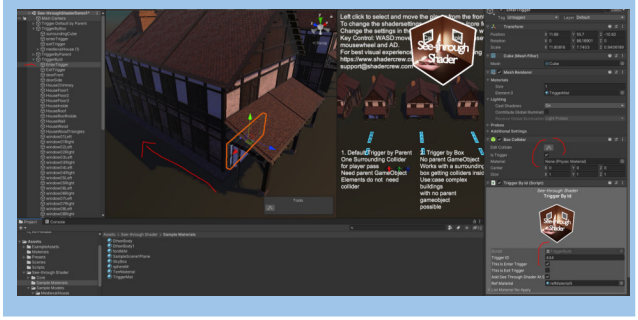
TriggerById

No colliders and no parent object available id set up on the element



TriggerById

No colliders and no parent object available enter/exit trigger

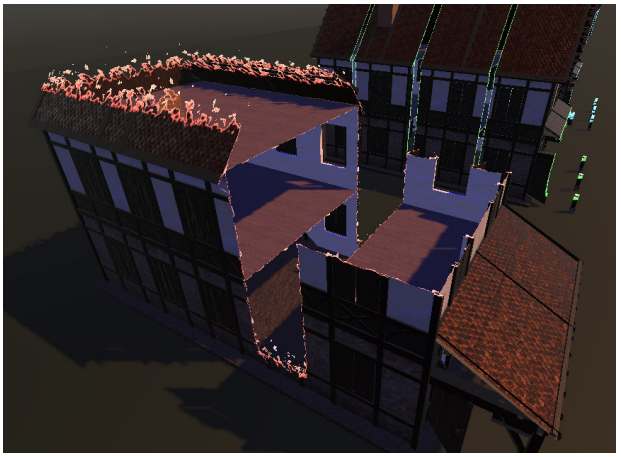


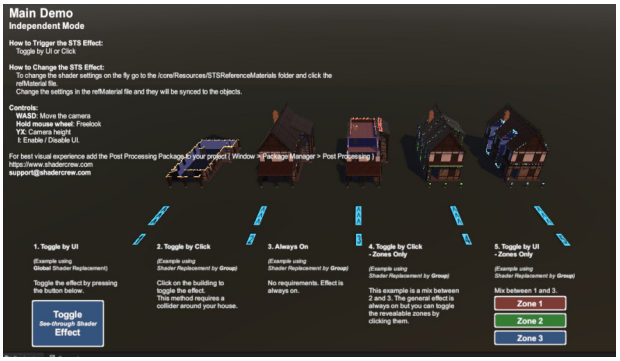
Player Independent

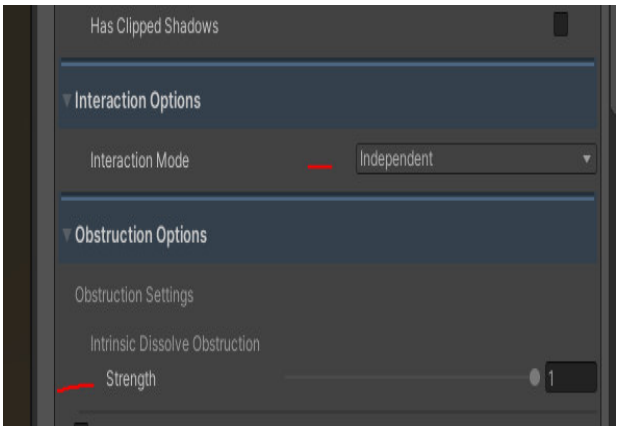
The See-through Shader works with a player and also a non player based setup.
They can also be mixed together where some Objects work on Layers and with the Materials in player mode and some Objects with the independent setup.

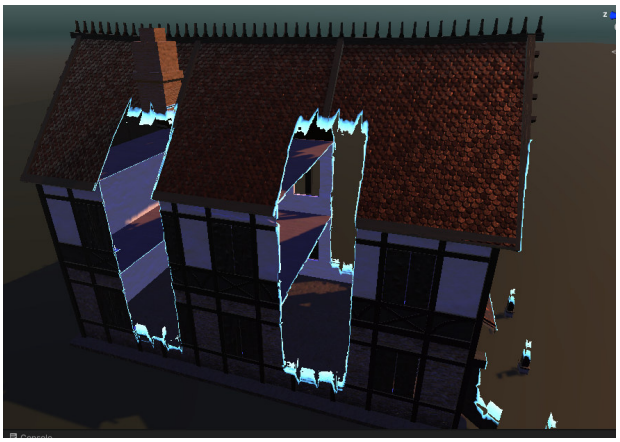
Independent mode works with the “none” obstruction mode and uses the “intrinsic strength” slider to seamlessly regulate the dissolve effect.
The main use cases for the independent mode are Cross Sectioning and when you want to work with the See-through Shader independent of a player or its position.

It comes with programmatic features for activate and deactivate the effect and we deliver scripts for easy Click / UI Button approaches.





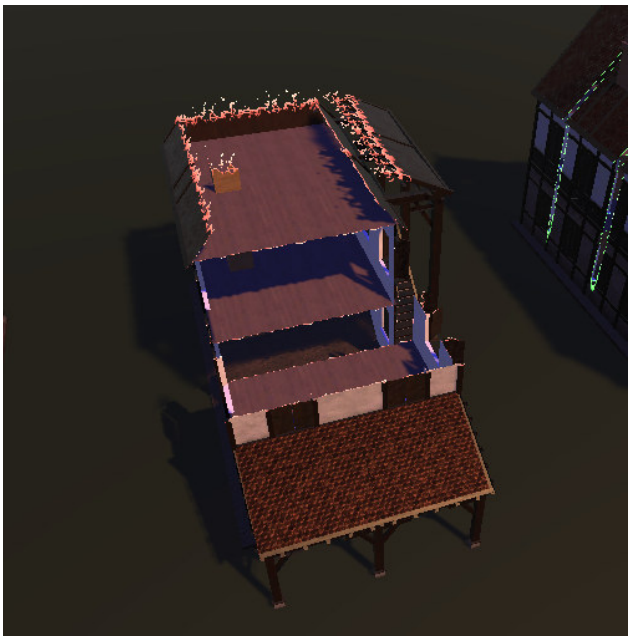




Summary Of Control Methods

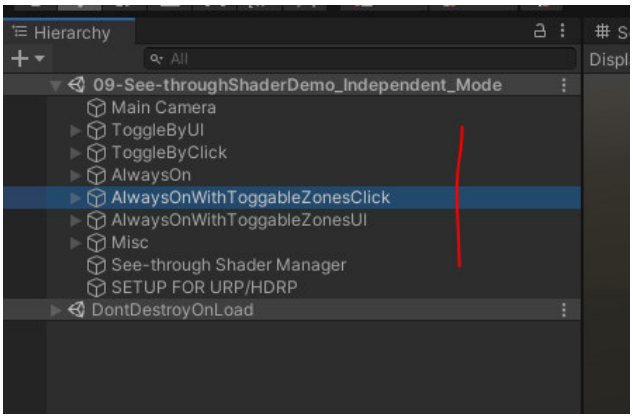
Method 1 - *Always On*

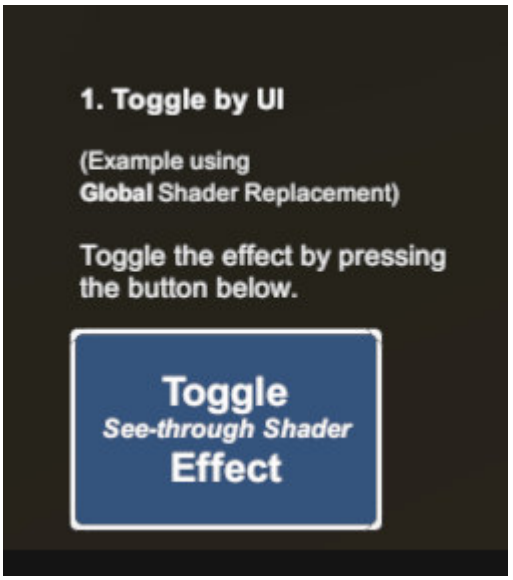
This use case applies the effect at any time; it is the opposite of the Toggle Method where you want to switch the effect on and off.
It is the third setup in our independent demo scene found under Assets/See-through Shader/Demo Scenes/



Method 2 - *Toggles*

Toggles are basically switches we implemented as components for turning the effect on and off by logic.
Mostly it is used with Mouse clicks / Touch but the feature can also be used programmatically in other use cases.
Check out the description below of the Toggles included in the Asset.

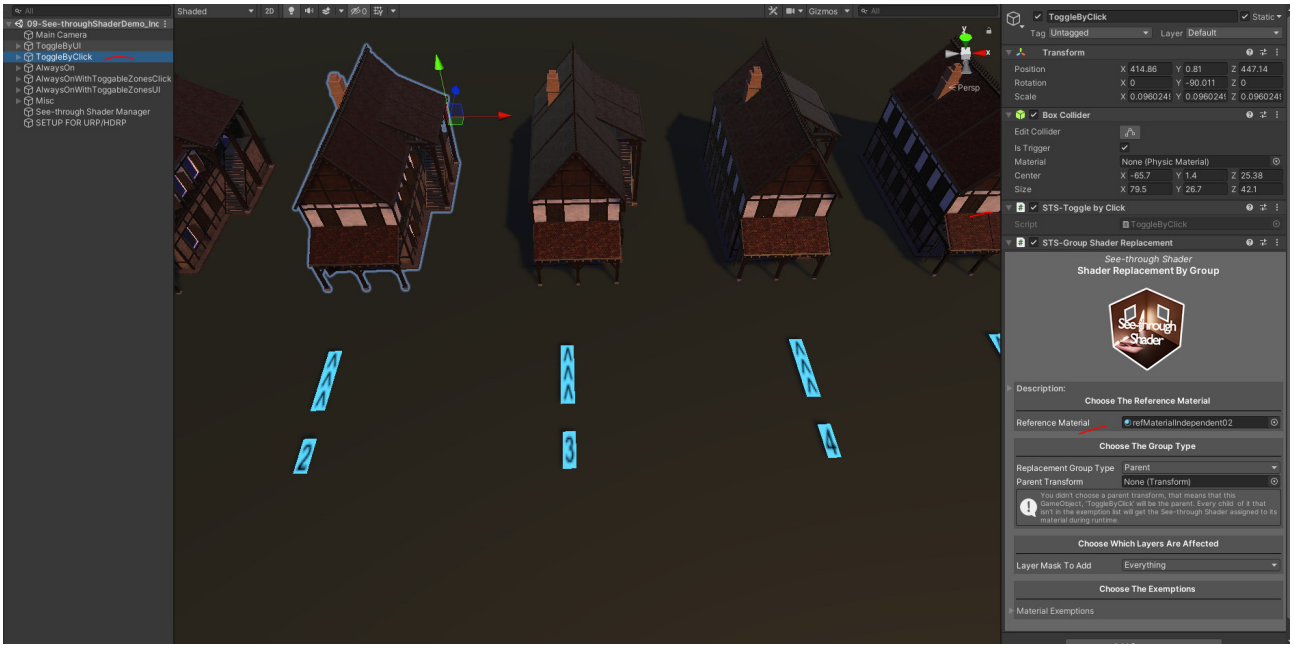


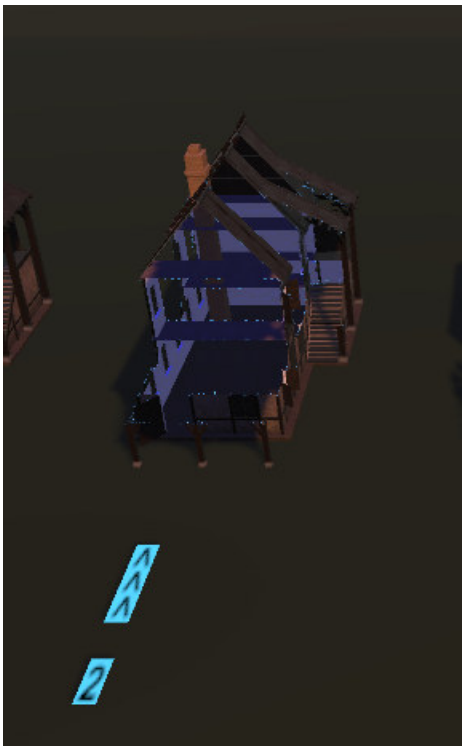


Detailed Description of the Individual Toggles

Toggle By Click

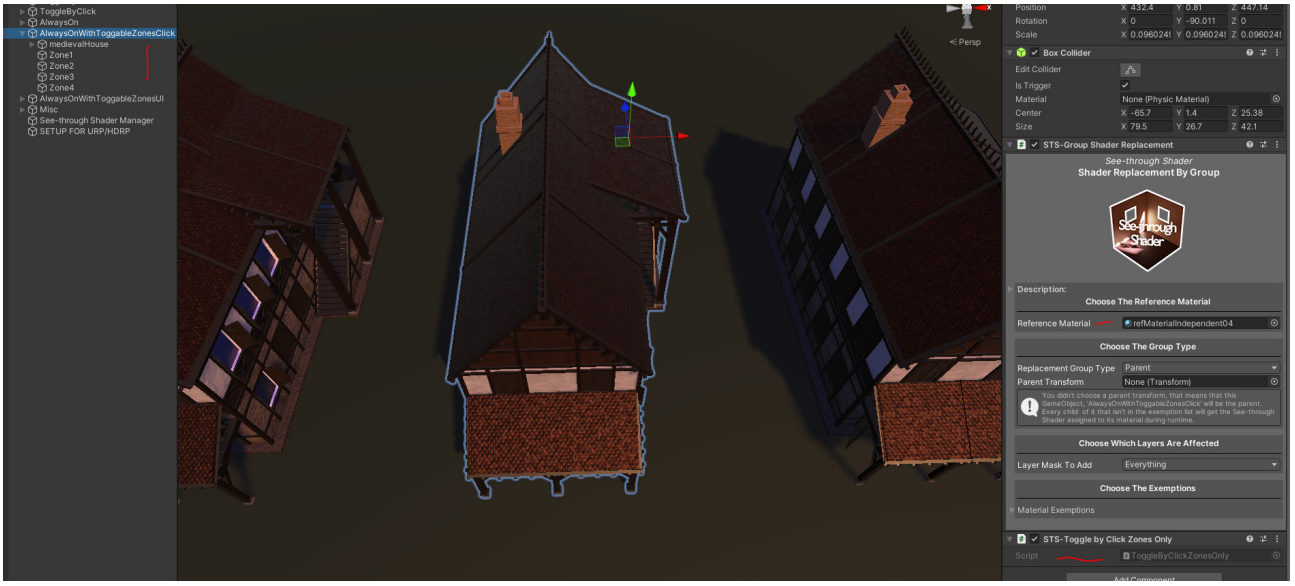
This component is used for clicking a Object and activating / deactivating the effect.
Add a collider to the “root” GameObject of the target and the ToggleByClick script.
Now by Clicking it will trigger the effect on the Object and its ChildObjects and vice versa.

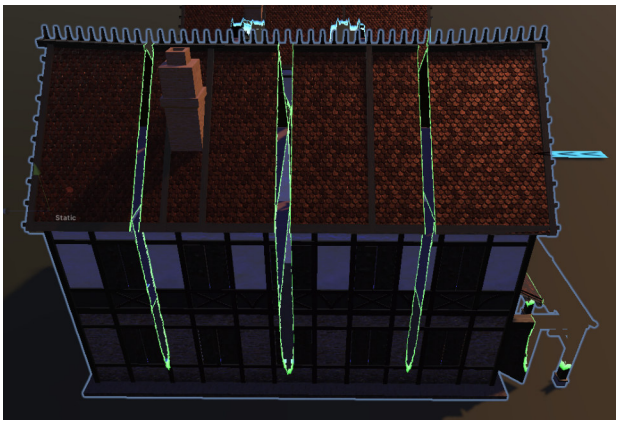


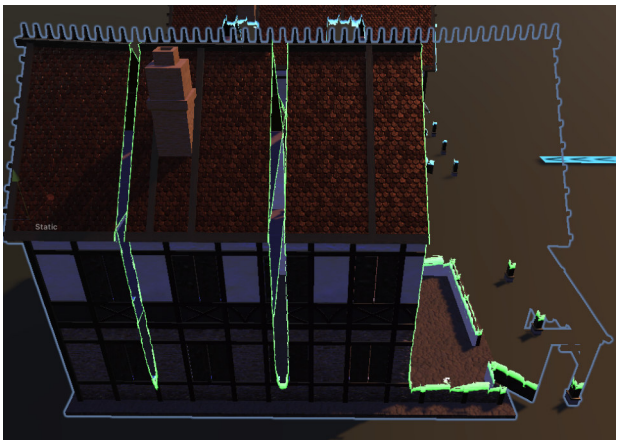


Toggle By Click - *Zones Only*

This components works with ZoneBoxes and will enable or disable the effect just for the Zone.
Add this to a Objects root and the Zoneboxes beneath as childs.
It can be used for a default always on with hidden Areas which can be revealed.

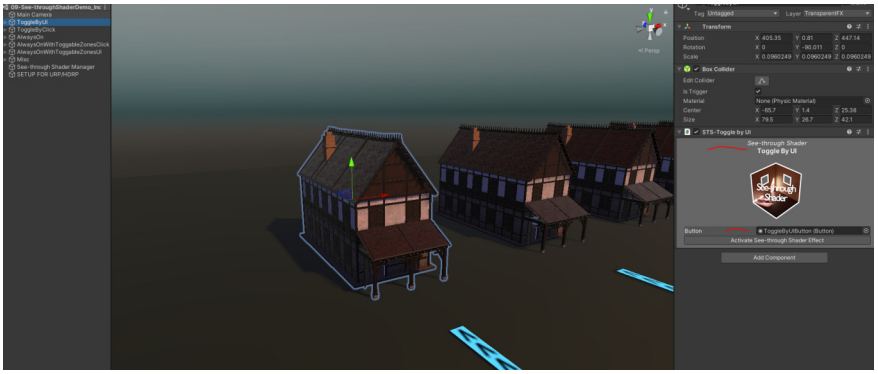






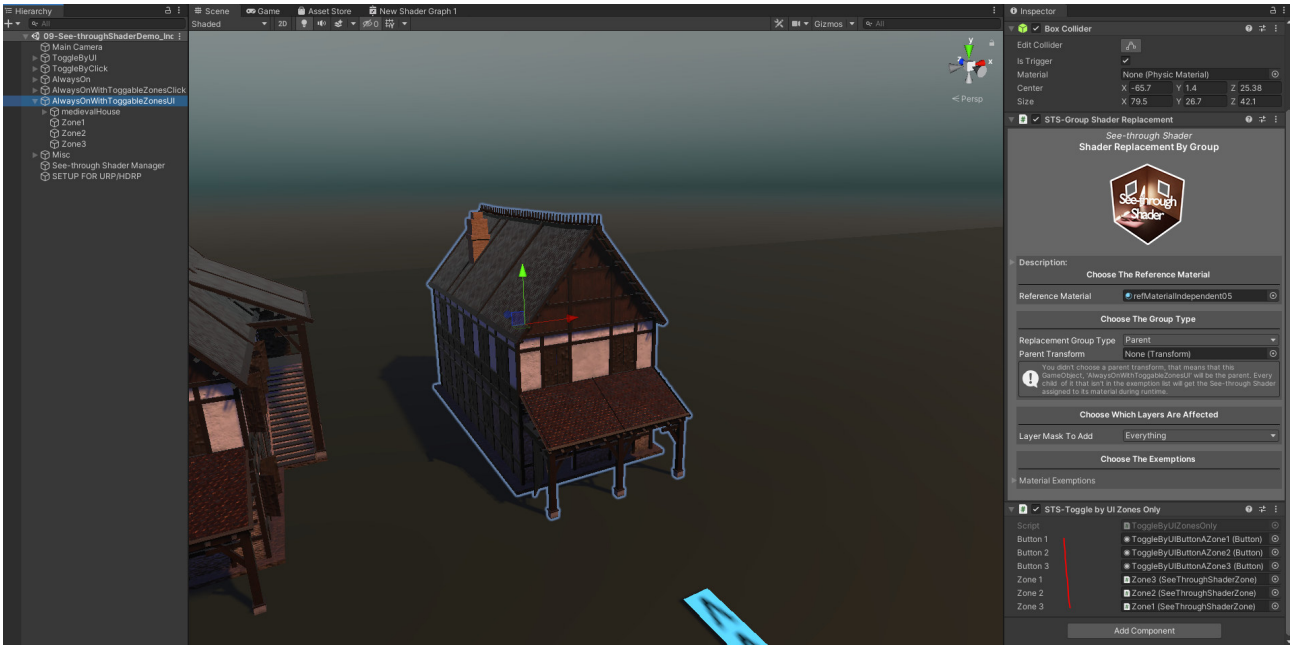
Toggle By UI

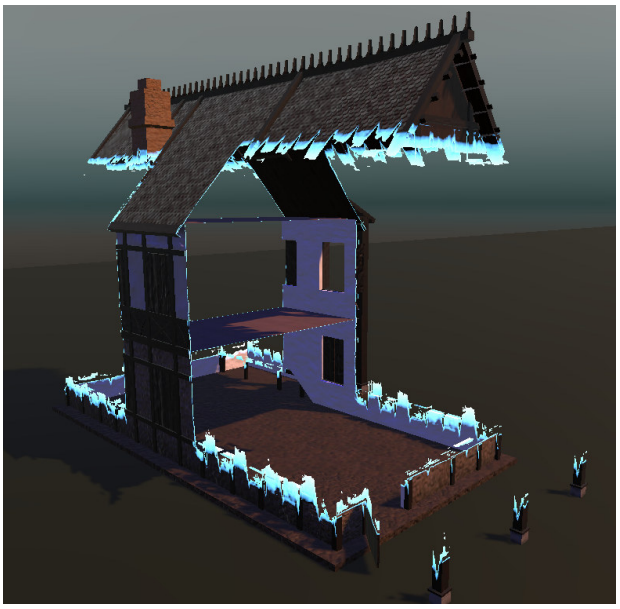
These components work similarly to the “Click” implementation except they are for UI use. Add the STS-Toggle by UI component to a root GameObject and drag a UI Button into the Button field.
Now on click the See-through Shader effect can be activated / deactivated.



Toggle By UI - *Zones Only*

Analog to the Toggle By UI the Zones Version of this script activates just the ZoneBoxes so also with a UI Based setup you can work with Zones.
Add the STS-Toggle by UI Zones only script to the root of a GameObject and drag UI Buttons and ZoneBox GameObject to the corresponding fields.





The See-through Shader Overview

Built-in
SeeThroughShader.shader
URP
SeeThroughShaderURP2019.shader, SeeThroughShaderURP2020.shader, SeeThroughShaderURP2021.shader
HDRP
SeeThroughShaderURP2019.shader, SeeThroughShaderURP2020.shader, SeeThroughShaderURP2021.shader

Introduction

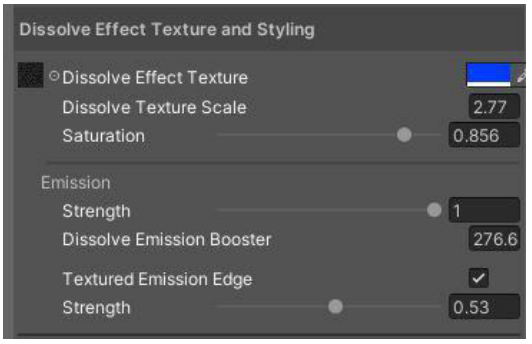
As said earlier, the core of the See-through Shader Asset is the *See-through Shader*, whose dedicated features can be controlled by the materials settings.

The various scripts with this asset help set up the Shader on each building element and sync the settings from a reference material by start and/or runtime.
Suppose you have buildings with just a few elements. Of course, the shader settings can also be done manually by clicking the elements of the building and changing the settings on the material.
If you want different features for a set of buildings, you need to create several material templates and apply them to the specific buildings.

See-through Shader settings are separated into five sections

1 - Dissolve Effect Texture and Styling.

Here, you can **make things glow and colorful**, together with the ability to change the scale of the texture, awesome visual effects can be achieved. Feel free to experiment with your own textures.

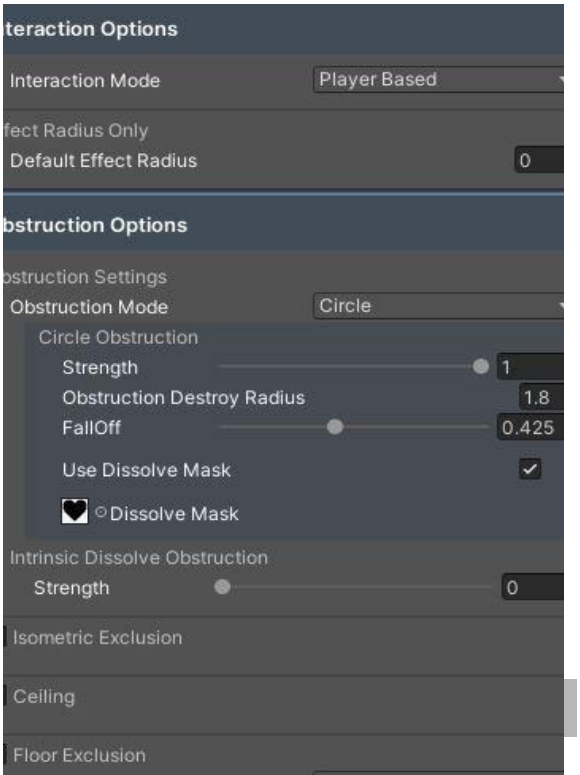


See *Dissolve Effect Texture and Styling*

2 - Player Obstruction Options.

This is the shader's primary function. The settings on how the obstructed area should be cleared can be found here **from shapes to totally clear (none mode).**

One of the key features of this asset is to let you choose from where (manually or relative to the player) you want to draw floors and the walls. So it does draw the floors and part or fully draws the walls up to an adjustable y height, you can choose if just the roof should be dissolved or the walls down to the floor.

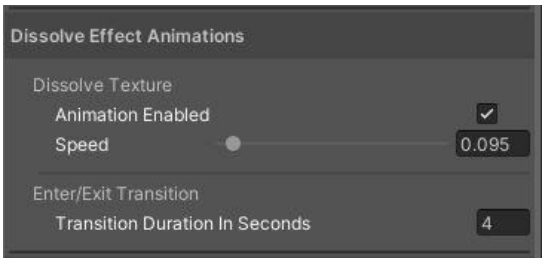


Complementary to the floor feature, we added a ceiling feature to precisely control the shader's primary function to draw from certain Y Levels and with the Ceiling feature it is from the roof downwards. You will also find the Isometric exclusion feature that uses an adjustable plane to clear or leave the mesh vertically.

See *Player Obstruction Options*

3 - Dissolve Effect Animations.

Another vital feature of the asset is the texture feature, where you can add any texture to the dissolved area and fine-grain control its behavior up to animation speed.



See *Dissolve Effect Animations*

4 - Zoning.

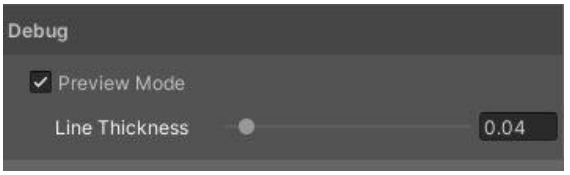
The Zoning feature enables rendering of everything inside or outside the zones.
(Currently, the only available zone type is a box, but more will come in future updates)
This is for use cases where you have non-revealed areas or the opposite, revealing everything inside the zone.

Drag the “See-through Shader” zones (Zone cubes) in the hierarchy under a parent GameObject, which has the “Zone Group” component attached to itself, to make them work together uniformly as one big zone.
For multi-floor buildings, a sync Y option is available to draw the zone boxes according to the Y position of the players or a manual y value.

See *Zoning*

5 - Debug.

The debug view shows the areas that are affected by the shader.
This allows debug and fine adjustment operations on the various shader settings in a full contrast view with black and white colors.
Especially for the zoning feature, the Y sync parameter will show a red line pointing to the Y level concerning the current player position where the shader will draw the zone boxes.



For example, this helps to draw the secret rooms, when the player goes up to the next floor.

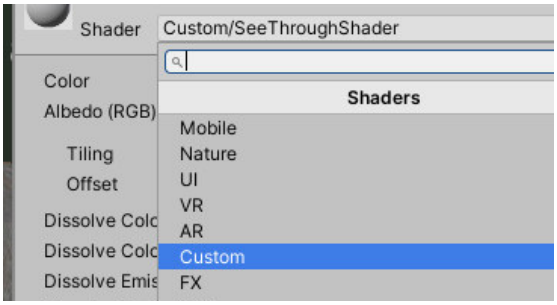
See *Debug*

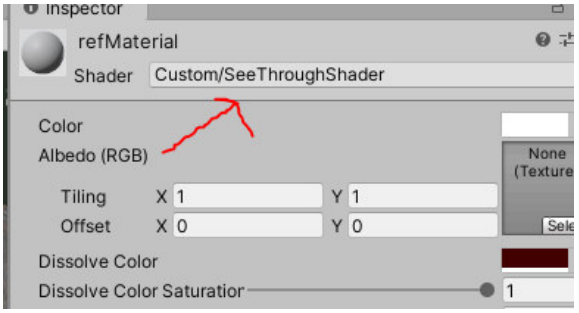
See-through Shader Features and Settings Details

Shader Selection

Select from the Shaders List the See-through Shader to apply it to a Material as shown in the Pictures below.
This is important when creating your template materials; after creating them with the right mouse and creating material, assign the Custom/SeeThroughShader(select the right one according to your RenderPipeline choice) to use them with this asset.

Picture/s 1 Custom/SeeThroughShader add



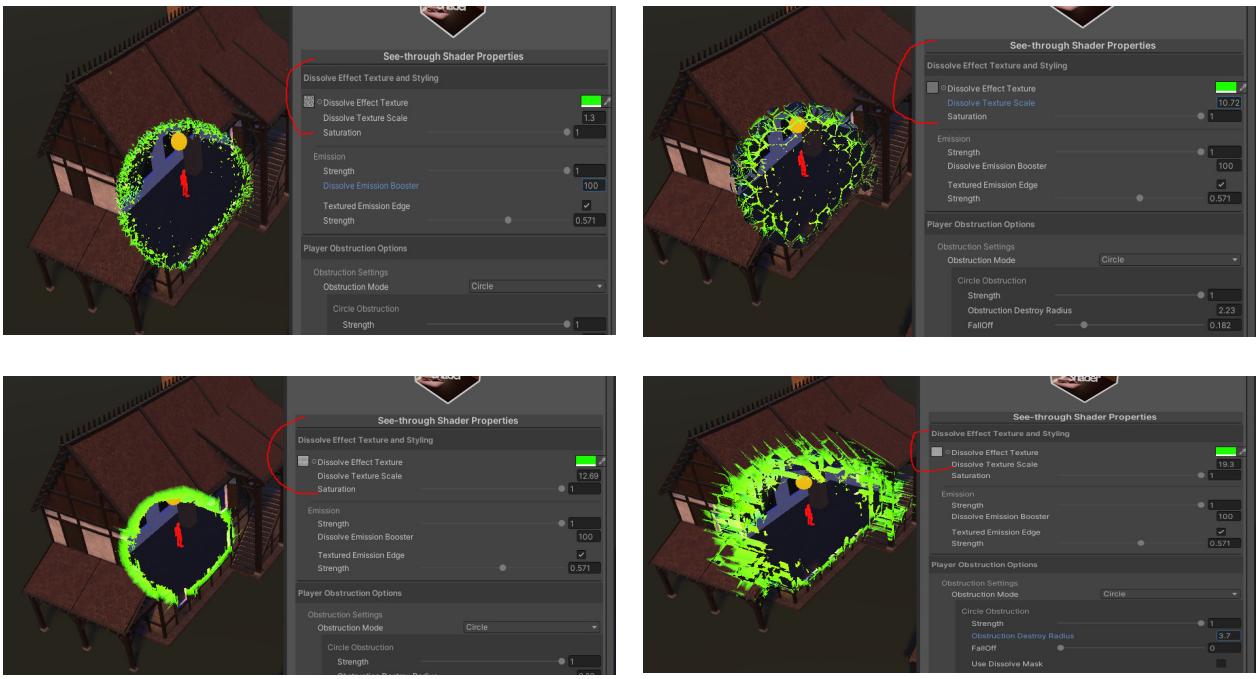


Dissolve Effect Texture and Styling

Property 1 - *Dissolve Effect Texture*

Any 2D dissolve texture used to show visual effects while transitioning and on the dissolve borders.
Feel free to experiment with different textures and scales as they show surprising visual effects.
We ship three textures within this asset for you to use.

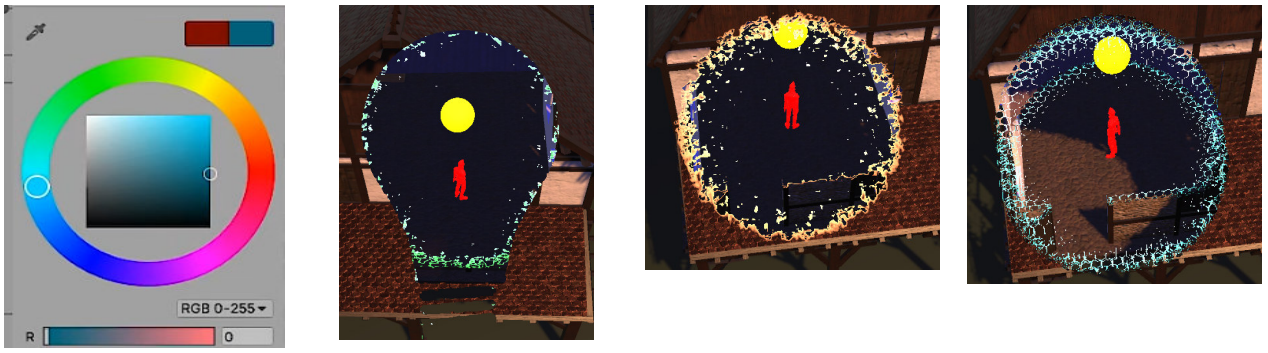
Picture/s Dissolve texture:



Property 2 - *Dissolve Effect Color*

Set the color of the region affected by the dissolve texture algorithm

Picture/s Dissolve Effect Texture and Color

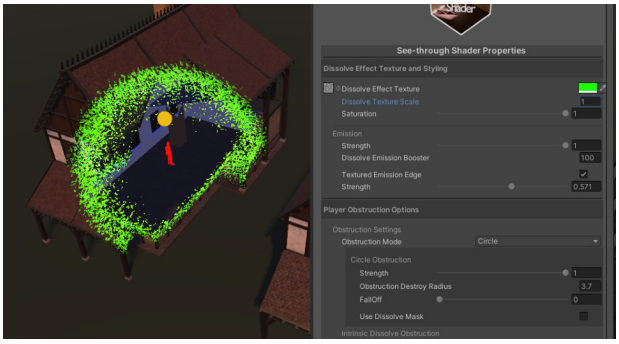


Property 3 - *Dissolve Texture Scale*

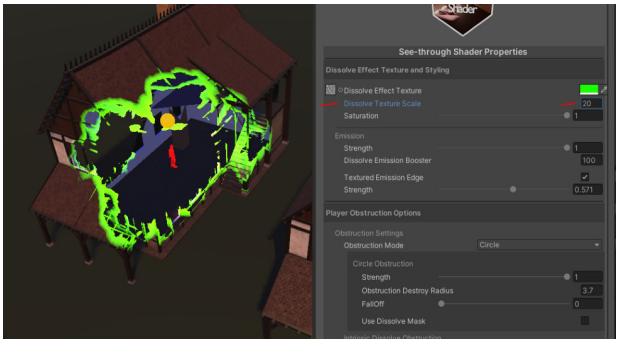
The scale of the dissolve texture, less value means finer resolution.
Please feel free to experiment with this value.

Picture/s Dissolve texture scale

Noise Scale value 1

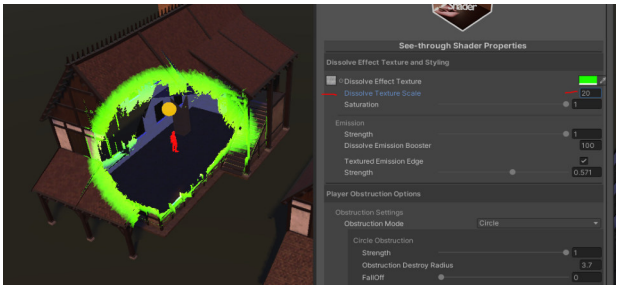


Scale value 20



Tiles Scale value 1

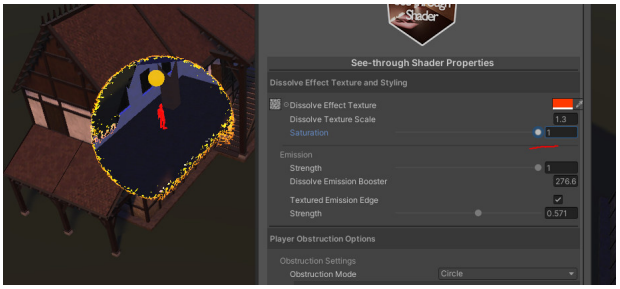
Scale value 3



Property 4 - *Dissolve Texture Color Saturator*

Set the color brightness of the dissolve texture according to your setup or use case.

Picture/s Dissolve texture color

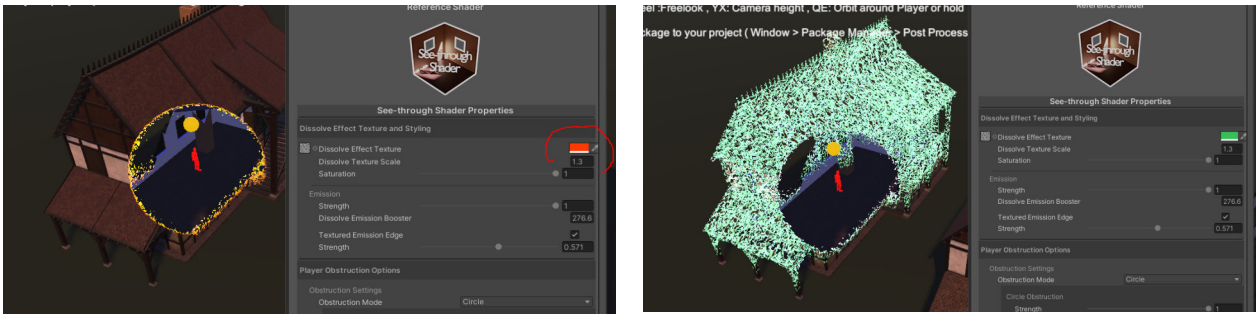


Emission

Property 5 - *Dissolve Emission Strength*

Set the dissolve texture color emission to have glow effects

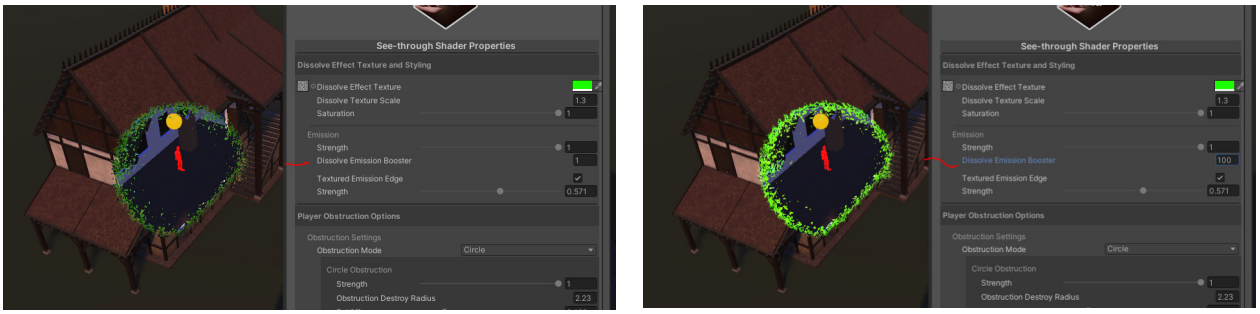
Picture/s Dissolve texture color



Property 6 - Dissolve Texture Emission Booster

Set the dissolve texture booster for some extra glow emission

Picture/s Dissolve texture emission booster



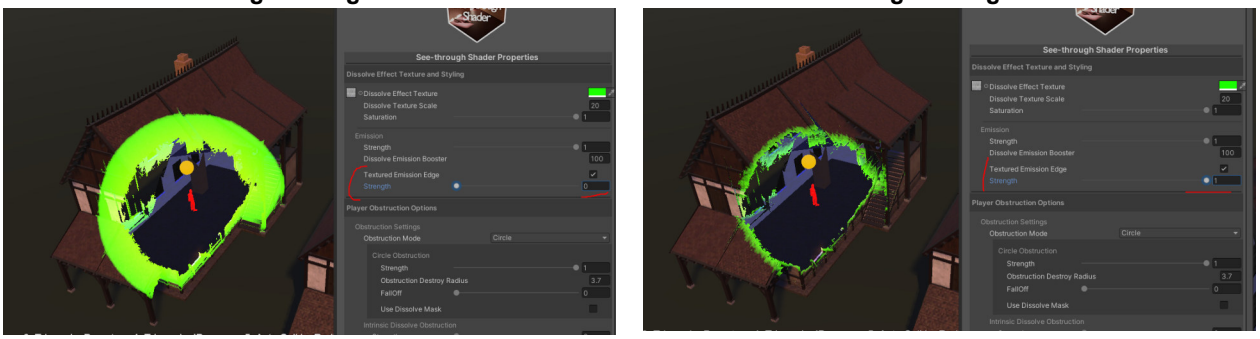
Property 7 - Dissolve Texture Edge Strength

Textured emission edge allows the dissolve texture to influence the emission shape.

Picture/s 17 Textured Emission Edge Strength

Textured Emission Edge Strength 0

Textured Emission Edge Strength 1



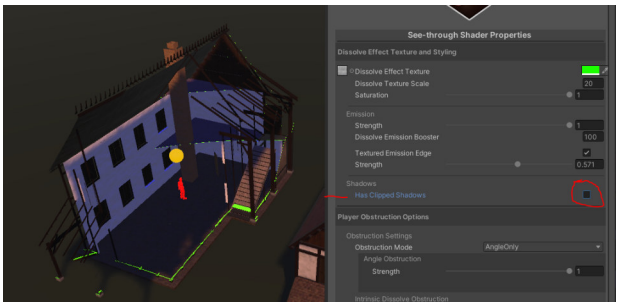
Shadows

Property 8 - *Has Clipped Shadows*

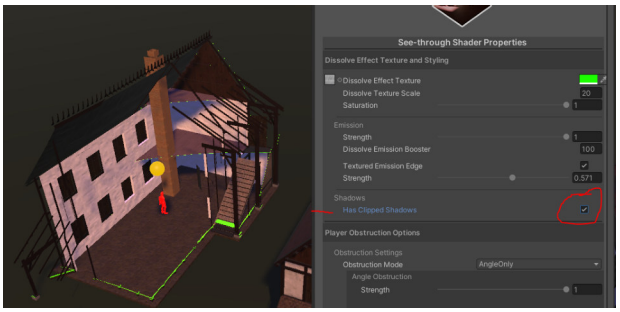
Controls if the shadows should be drawn in relation to the light outside or within the building.
Determines if the shadows are the same before the dissolve effect was applied. Imagine a house with a roof on a sunny day. Let the player be inside the house, and the dissolve effect removes the roof. If you disable "Has Clipped Shadows," the sun is still blocked from entering.

Picture/s 15 Has Clipped

Shadows off



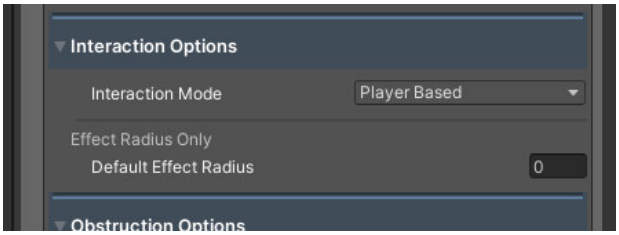
Shadows on

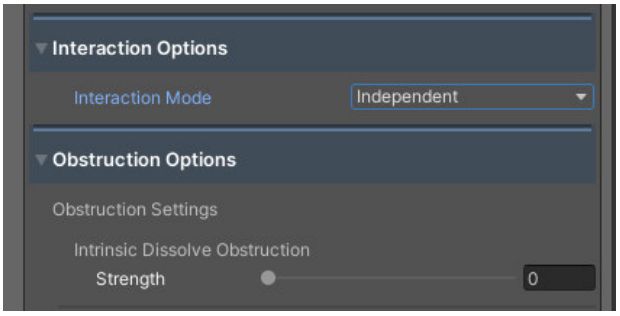


Interaction Options

Property 1 - *Interaction Mode*

Select PlayerBased or Independent mode of the Shader.
As described earlier “PlayerBased” tracks the position of players and “Independent” is a non player approach to handle the effect independent of player(s).





Settings for Player-Based Mode

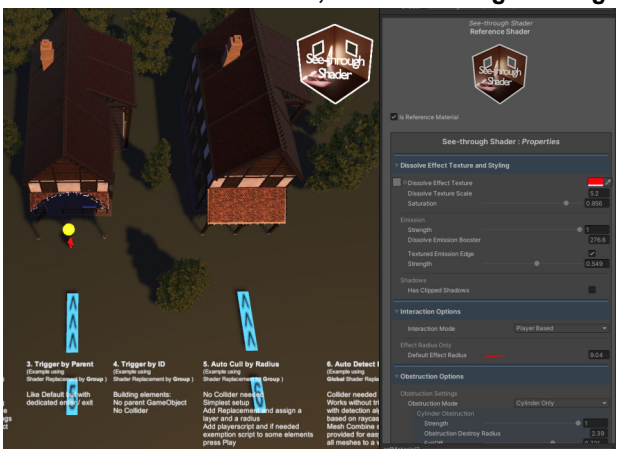
This only appears if you selected the Player-Based Mode as the Interaction Mode.

Property 2 - *Default Effect Radius*

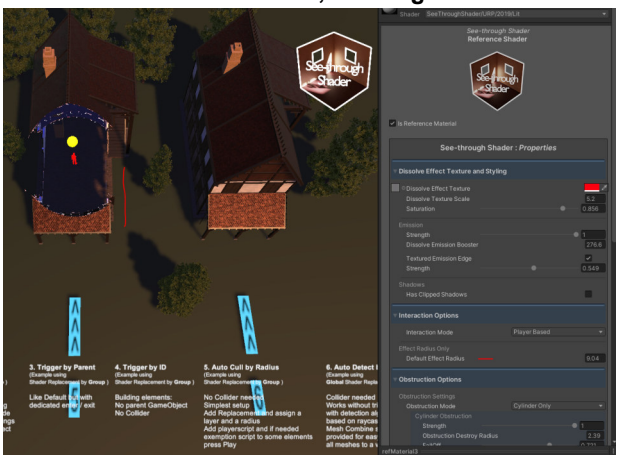
Default Effects Radius defines a always-on radius around the player within the mesh is cleared.
The default effect feature does not need any triggers as it is the default “run mode” of the shader, when no Triggers or AutoDetection is used.
This feature is also handy in procedurally created worlds. You would just add the replacement shader script and use this feature to always clear mesh within the default effect radius.
Beware that when using Triggers, the Default effect will not work as it is meant for different use.

Picture/s Default Effect Radius

Picture Default effect radius, before entering building



Picture Default effect radius, building entered



Player Obstruction Options

Picture/s 8 Dissolve obstruction modes:

Dissolve obstruction modes

This is the **most important and core feature of the See-through Shader Asset. This controls the mode of clearing the obstruction**.
The level of clearance is seamlessly adjustable, so semi-visibility is possible as well as drawing the silhouette of the building with the Dissolve texture for special effects.

Following obstruction modes are available:

Angle

Clear obstruction by the angle between the camera and player
angle based: uses the angles from the mesh to the player, and the distance from the mesh and the player to the camera to calculate which parts of the mesh do not draw.

Cone

Remove parts of the mesh contained inside a cone. The tip of the cone is at the transform position of the player, and the center of the base is at the camera transform position.
The radius is optional and can be set using "Cone Obstruction Destroy Radius

Cylinder

Removes parts of the mesh that are contained inside a Cylinder. The base of the Cylinder is at the transform position of the player, and the center of the base is at the camera transform position.
The radius is optional and can be set using "Cylinder Obstruction Destroy Radius

Circle

Clear obstruction with a focused circle around the player, is the recommended mode as it works well with simple and complex buildings

Curve

Control the shape of the obstruction removal via a curve.
x=0 is where the player is, and x=1 is where the cam is. the y values[0,1] of the graph dedicate the “Radius Width Multiplier”. The radius width is the product of Obstruction Destroy Radius times the radius width multiplier, aka the corresponding y-value from the graph.

Angle and Cone

Combination of Angle and Cone

Angle and Cylinder

Combination of Angle and Cylinder

None

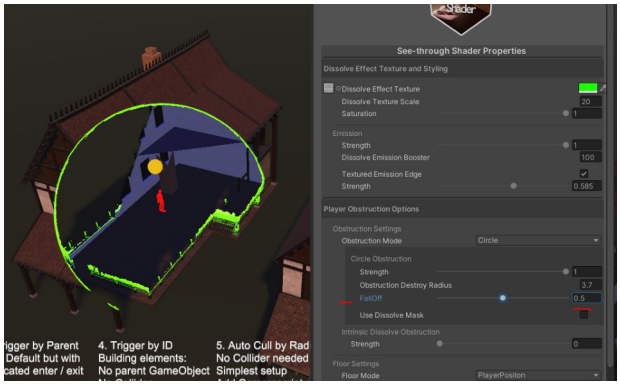
Applies the mesh uniformly across the whole mesh. Use this to clear the whole mesh or give it a textured semi-transparent look.
This is useful for visual effects and removing any obstruction

Settings for Cone, Cylinder, and Circle mode

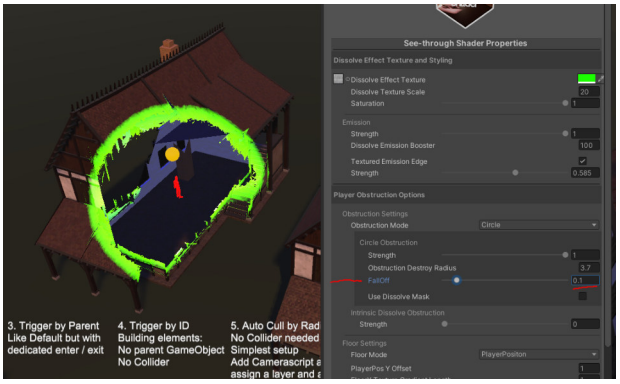
Property 1 - *Dissolve Falloff*

Controls how smooth the blending between the original mesh and the dissolved area is. A value of 0 produces a soft edge, whereas increasing the value closer to 1 produces an increasingly harder edge.

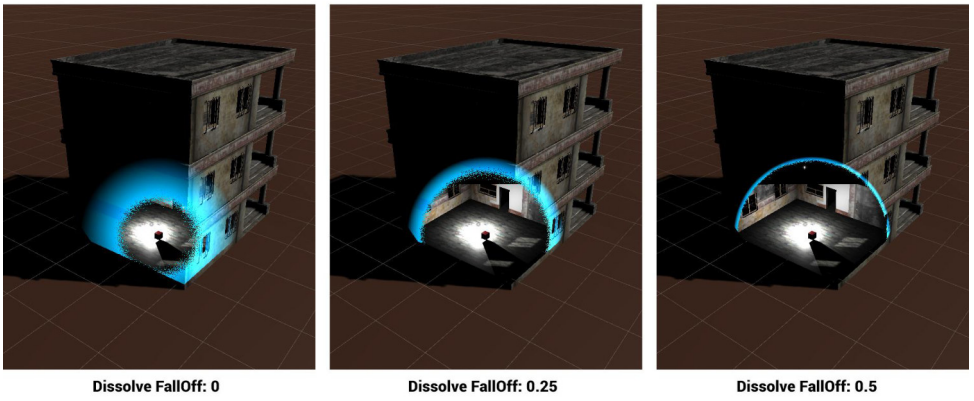
Dissolve falloff value 0.5 (circle mode)



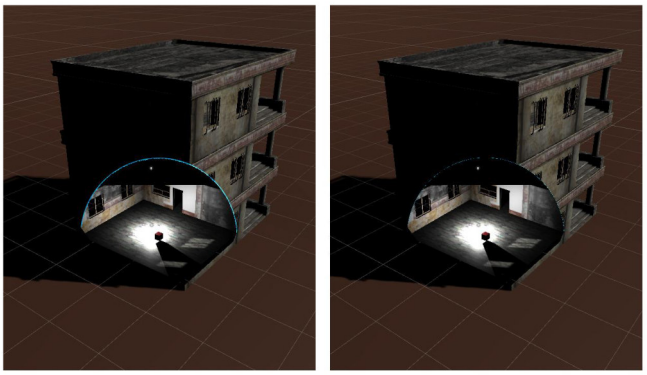
Dissolve falloff value 0.1 (circle mode)



Picture/s Dissolve falloff



Dissolve FallOff: 0 | Dissolve FallOff: 0.25 | Dissolve FallOff: 0.5



Dissolve FallOff: 0.75 | Dissolve FallOff: 1

Property 2 - *Dissolve Mask*

Use 2D Black and White Textures to create your own custom dissolve shapes. Drag them to the Dissolve Mask Field and enjoy!

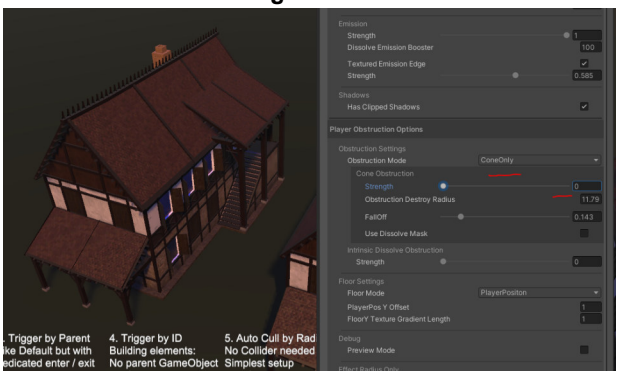
Settings for ConeOnly Mode

Property 1 - *Cone Obstruction Strength*

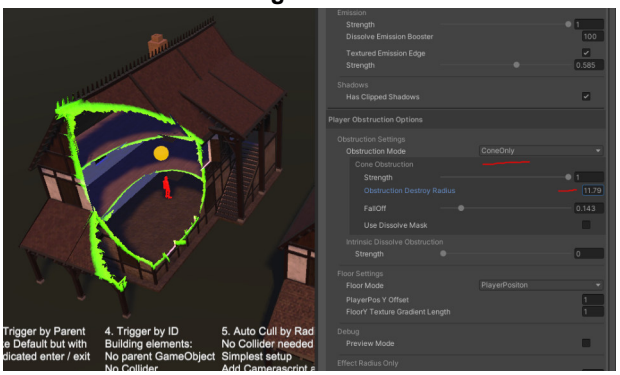
Strength of clearance, the highest value 1 clears the most.

Picture/s Cone Obstruction Strength

Cone Obstruction Strength value 0



Cone Obstruction Strength value 1

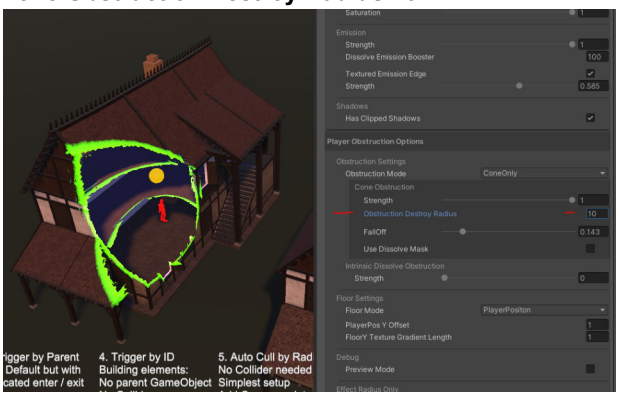


Property 2 - *Cone Obstruction Destroy Radius*

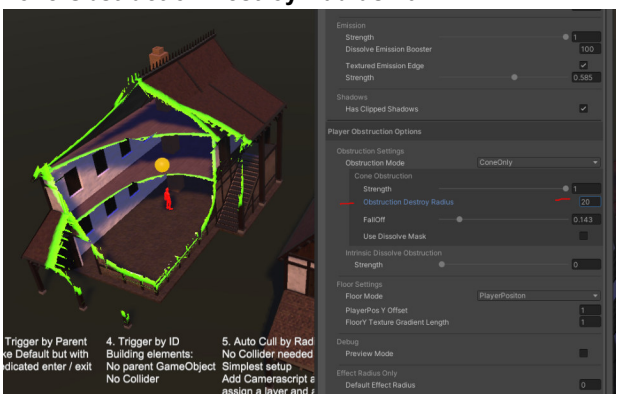
The radius of the cone within this mesh will be dissolved

Picture/s Cone Obstruction Destroy Radius

Cone Obstruction Destroy Radius 10



Cone Obstruction Destroy Radius 20



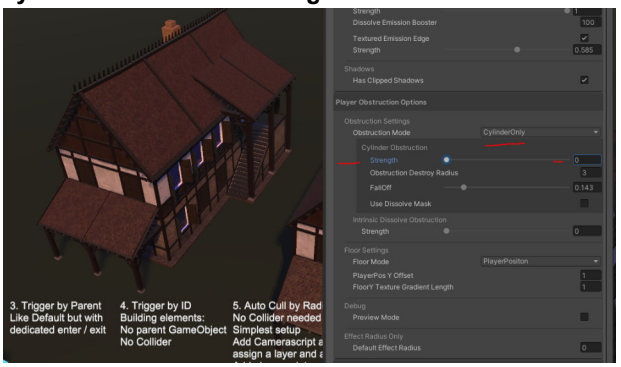
Settings for CylinderOnly Mode

Property 1 - *Cylinder Obstruction Strength*

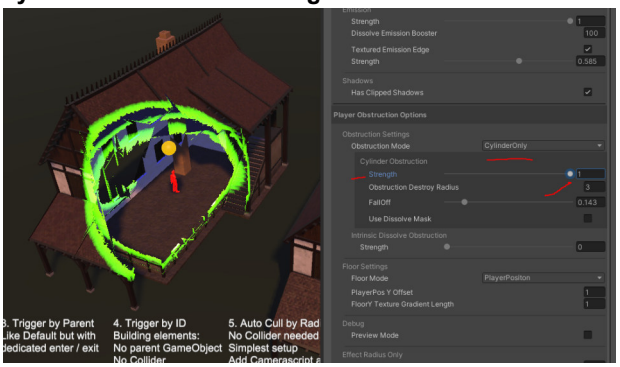
Strength of clearance, the highest value 1, clears the most.

Picture/s Cylinder Obstruction Strength

Cylinder Obstruction Strength value 0



Cylinder Obstruction Strength value 1

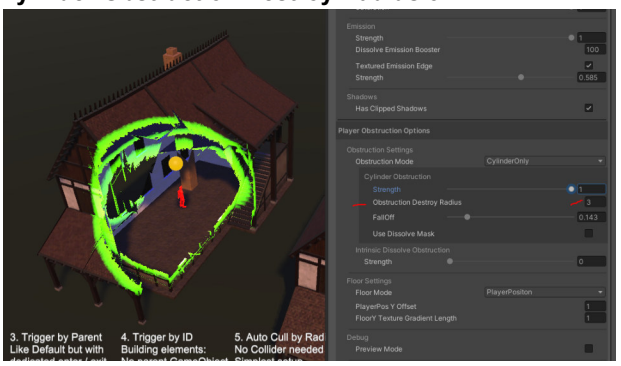


Property 2 - *Cylinder Obstruction Destroy Radius*

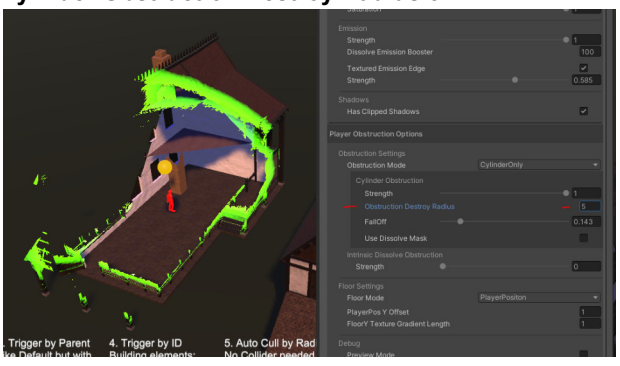
Radius of the cone within this mesh will be dissolved

Picture/s Cylinder Obstruction Destroy Radius

Cylinder Obstruction Destroy Radius 3



Cylinder Obstruction Destroy Radius 5



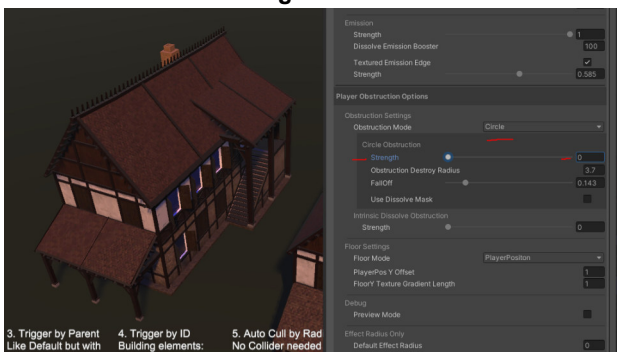
Settings for Circle Mode

Property 1 - *Circle Obstruction Strength*

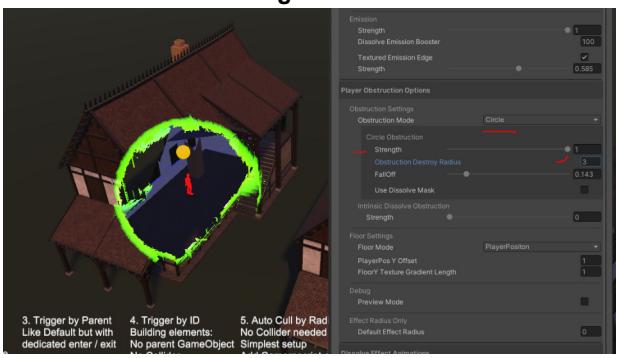
Strength of clearance, the highest value 1 clears the most.

Picture/s Circle Obstruction Strength

Circle Obstruction Strength value 0



Circle Obstruction Strength value 0.5

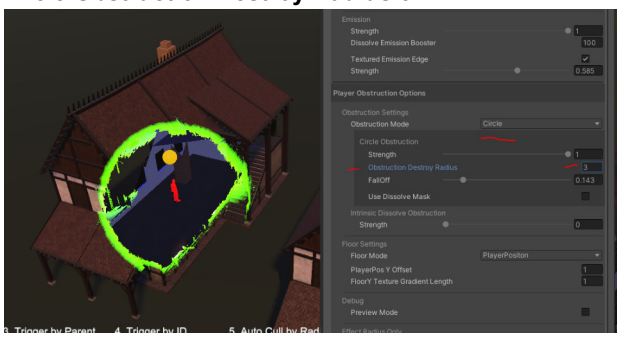


Property 2 - *Circle Obstruction Destroy Radius*

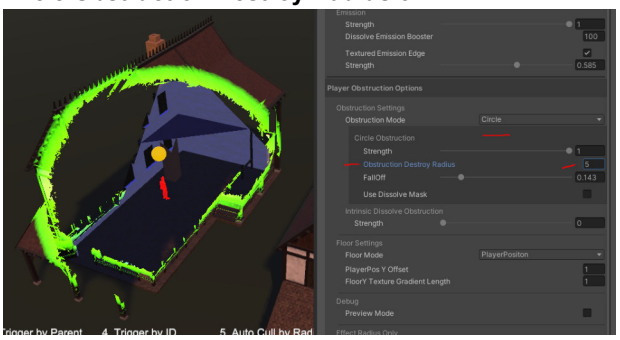
The radius of the cone within this mesh will be dissolved

Picture/s Circle Obstruction Destroy Radius

Circle Obstruction Destroy Radius 3



Circle Obstruction Destroy Radius 5



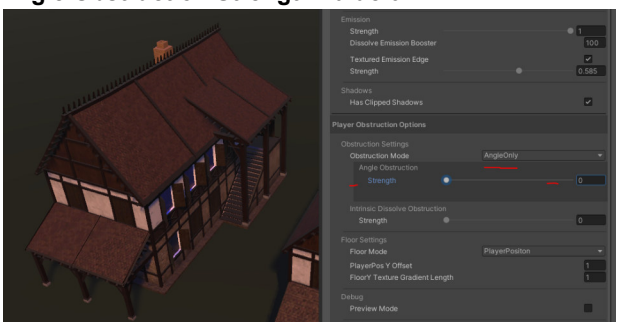
Settings for Angle Mode

Property 1 - *Angle Obstruction Strength*

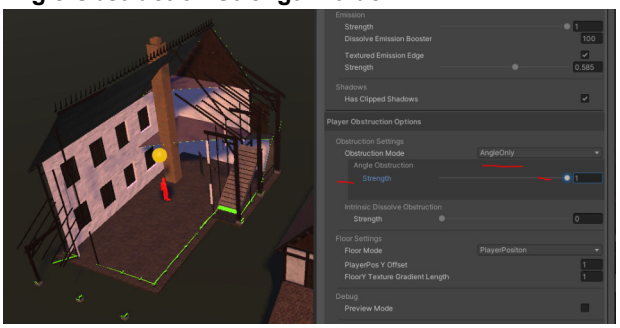
Strength of clearance when seen the player from the angle of the camera

Picture/s Angle Obstruction Strength

Angle Obstruction Strength value 0



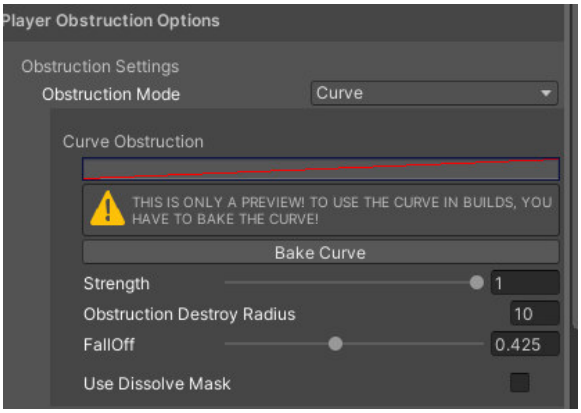
Angle Obstruction Strength value 1



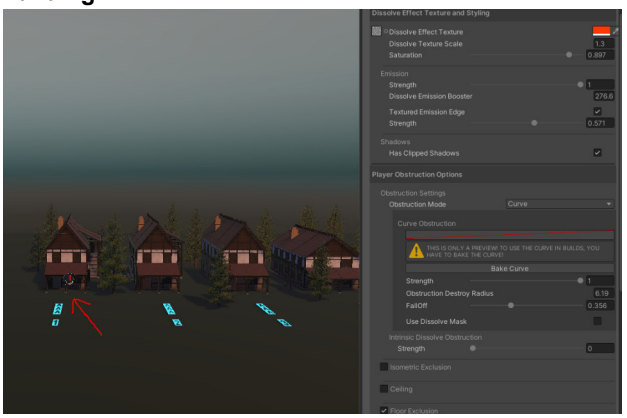
Settings for Curve Mode

Dissolves Mesh with a variable cutout size with the help of a visual Graph set in the refMaterials.
In Unity 2019 there is a known bug when using the curve editor window and then maximizing the window. Everything works fine in 2020 and 2021.

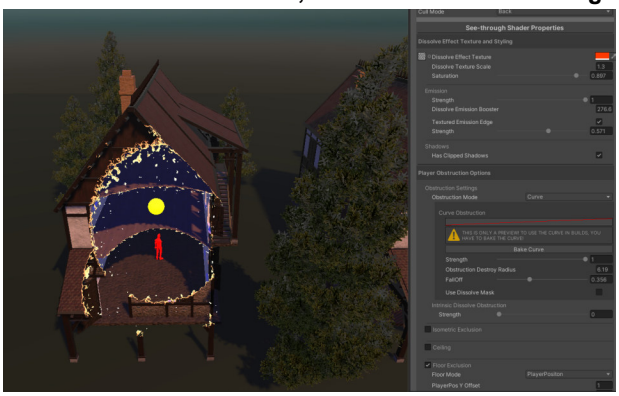
Picture Curve mode Overview



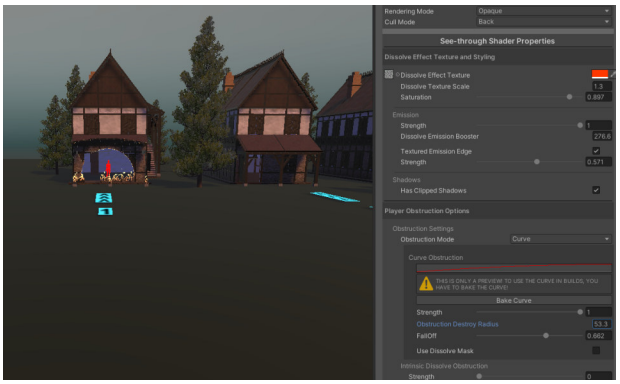
Picture Curve mode result, Camera far to the Building

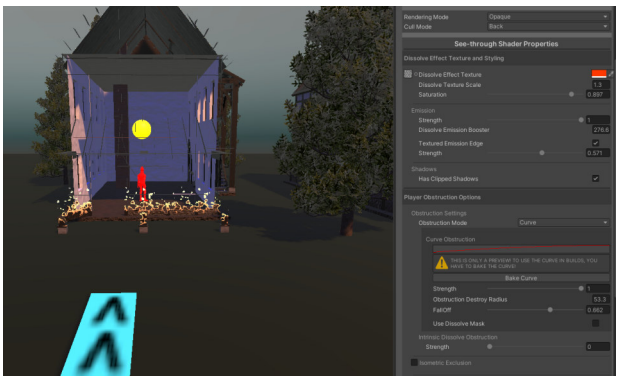


Picture Curve mode result, Camera near the Building



Picture Curve mode Result Example 2



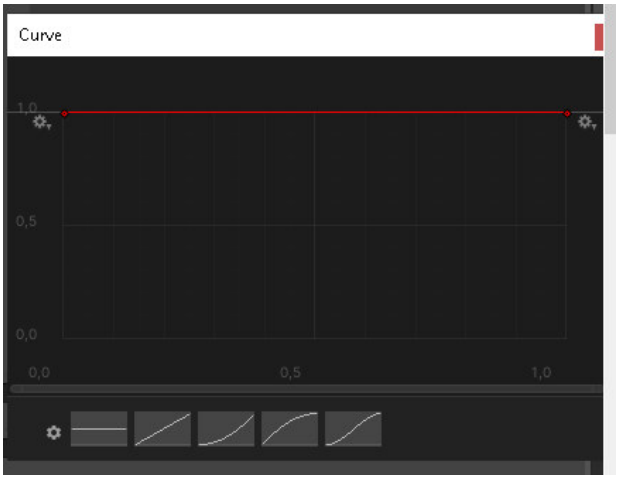


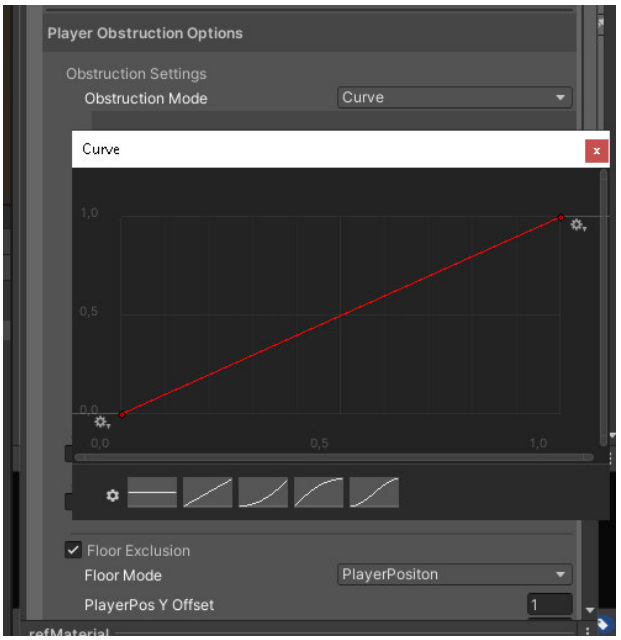
Property 1 - *Curve Obstruction*

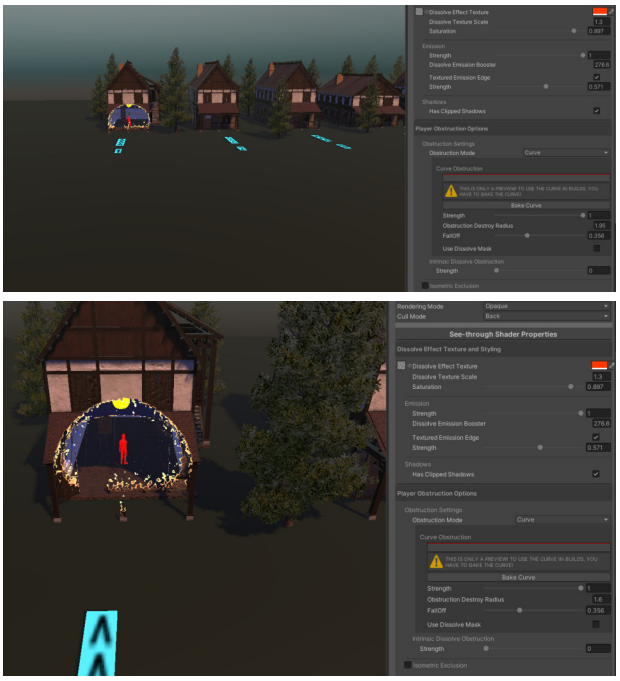
Picture(s) Curve Obstruction Graph

Picture Curve Presets

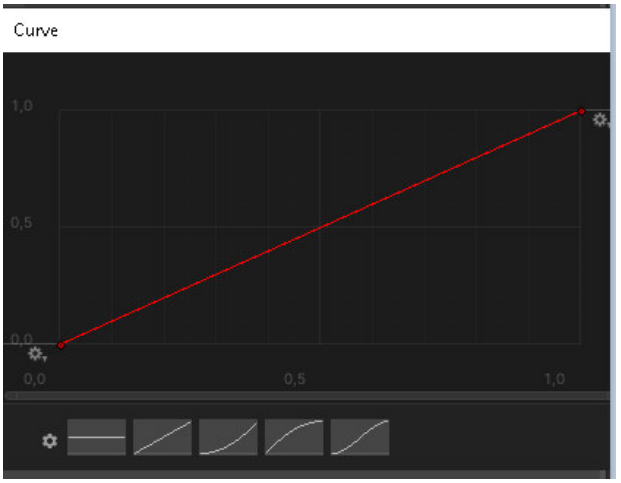
Picture Curve Preset 1
No change of cutout size when approaching



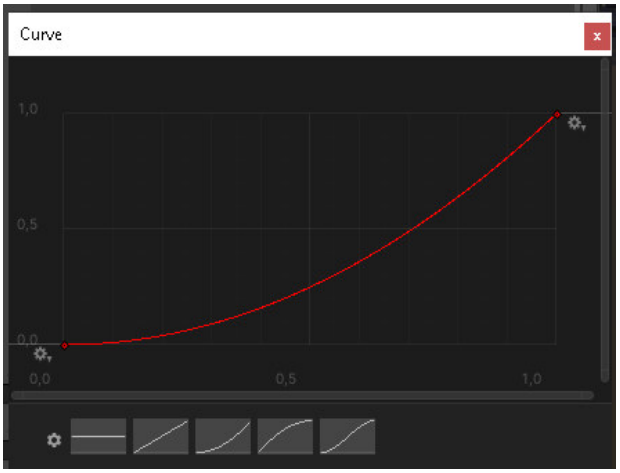




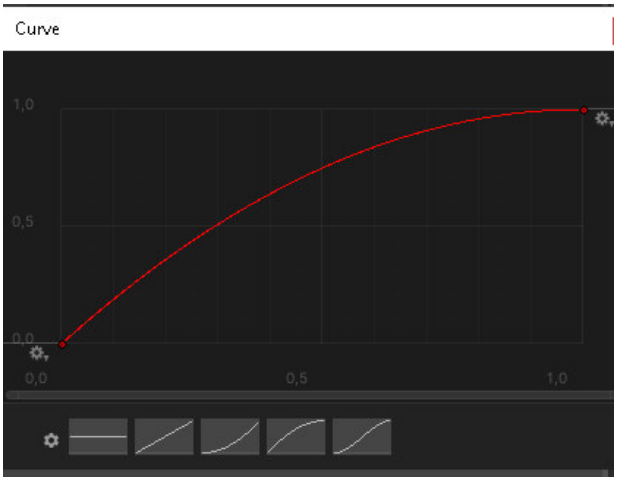
Picture Curve Preset 2



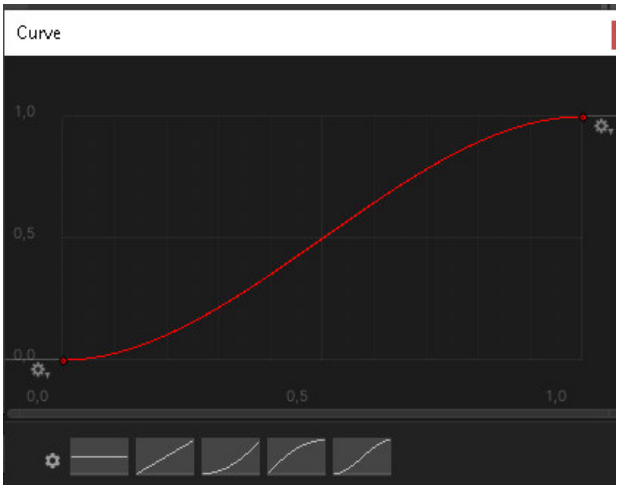
Picture Curve Preset 3



Picture Curve Preset 4



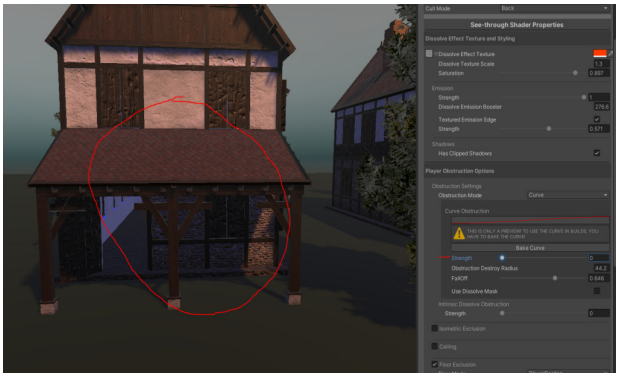
Picture Curve Preset 5



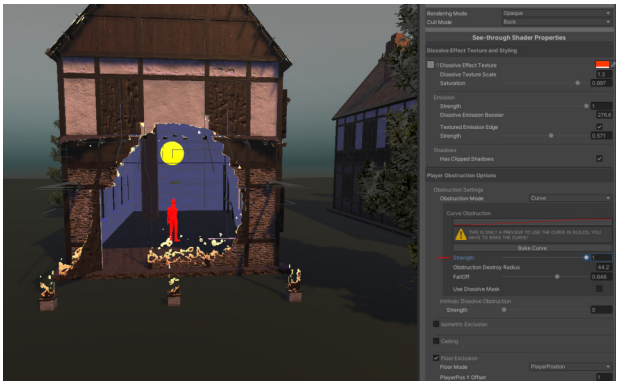
Property 2 - Strength

Property that affects the dissolve strength in the cutout area.

Picture Curve strength 0



Picture Curve strength 1



Property 3 - Obstruction Destroy Radius

Radius of the dissolve area

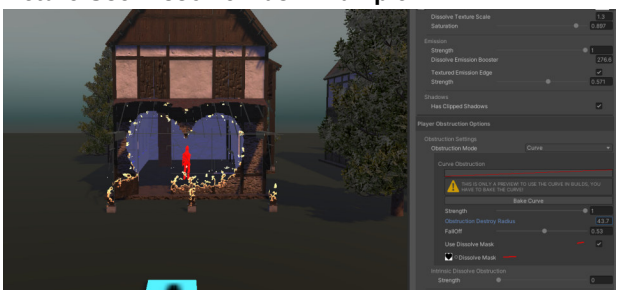
Property 4 - FallOff

Gradient textured Area at the borders varies between no textured border to full textured.

Property 5 - Use Dissolve Mask

Activates the Shape feature to use custom shape masks.

Picture Use Dissolve Mask Example



Intrinsic Dissolve Obstruction

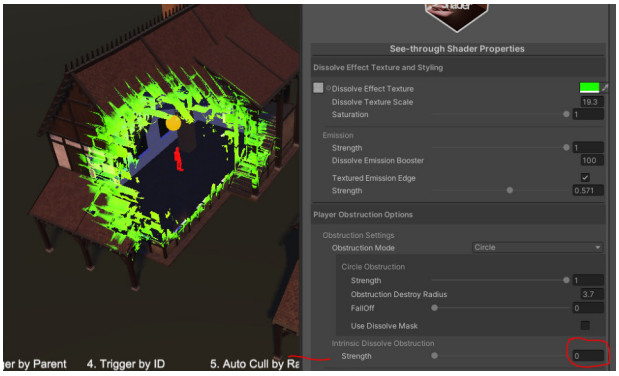
Property 1 - Intrinsic Obstruction Dissolve Strength

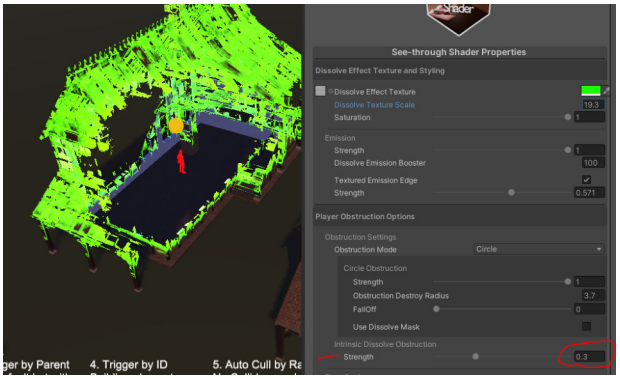
Dissolves seamless and adjustable whole mesh outside of the dissolved area (player area) upwards beginning with the PlayerPos y offset value.
This value set to 1 is all clear; this is useful with the obstruction mode “none” to draw the silhouette of a building or make it semi-transparent.

Picture/s 9 Intrinsic dissolve strength

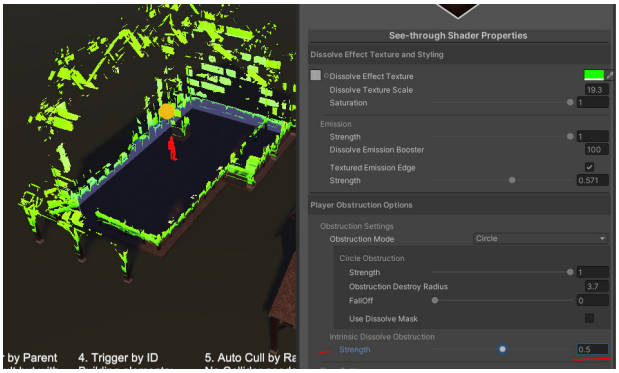
Intrinsic off

Intrinsic 0.3

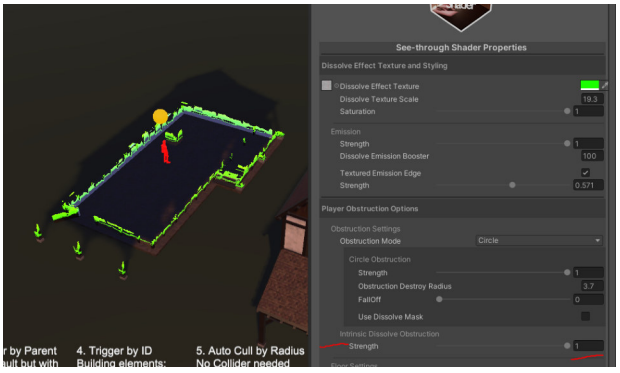




Intrinsic 0.5



Intrinsic 1



Isometric Exclusion

Uses a virtual plane to clear or leave the mesh horizontally in relation to the player. This feature can enhance or fine-grain the result of the effect combined with the obstruction modes.

Property 1 - *Isometric Exclusion Toggle*

Enable/Disable the Isometric Exclusion feature

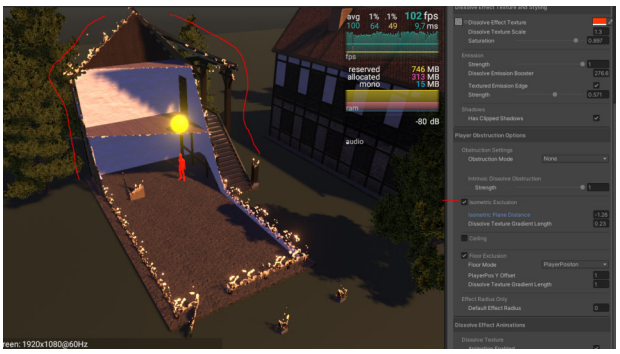
Property 2 - *Isometric Plane Distance*

Distance to the player of the virtual plane

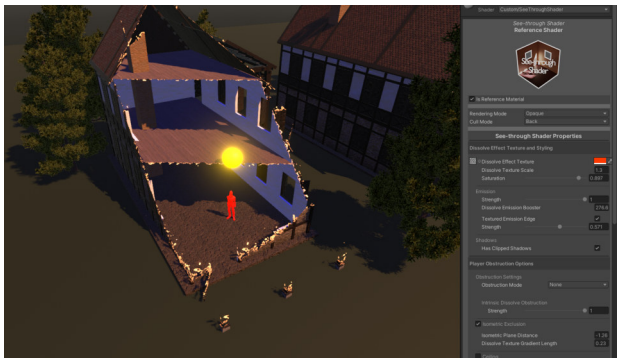
Picture/s Isometric Plane Distance

Picture Isometric Exclusion off

Picture Isometric Exclusion enabled



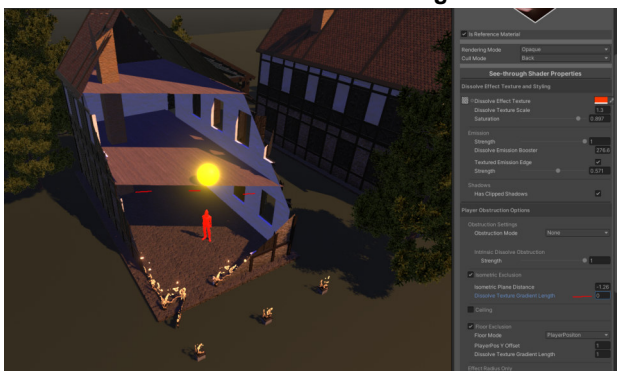
Picture Isometric Exclusion, the player moved to the front



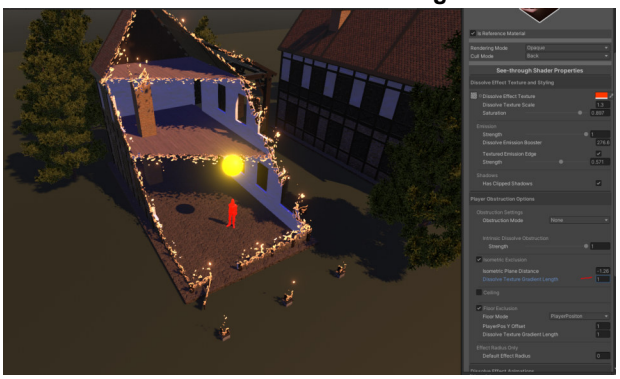
Property 3 - Dissolve Texture Gradient Length

Picture/s Dissolve Texture Gradient Length

Picture Dissolve Texture Gradient Length 0



Picture Dissolve Texture Gradient Length 1



Ceiling

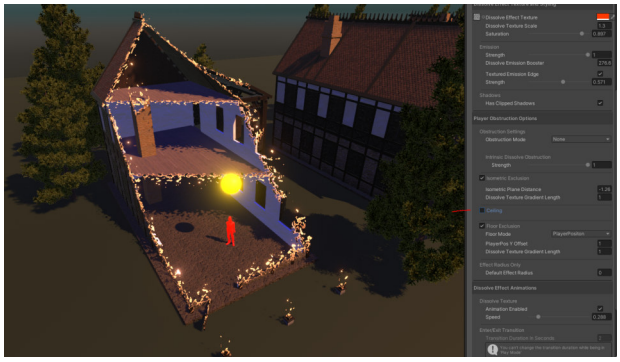
Control of dissolving effect from top-down, principally this is useful for ceilings.
This feature has two modes called Additive and Subtractive.
In Additive mode, it will add ceiling mesh even though it would be cleared usually, and in Subtractive mode, it will clear the mesh from the top-down hence the “Ceiling” Feature.
It can be seen as the opposite feature to the Y Floor feature.

Property 1 - *Ceiling Toggle*

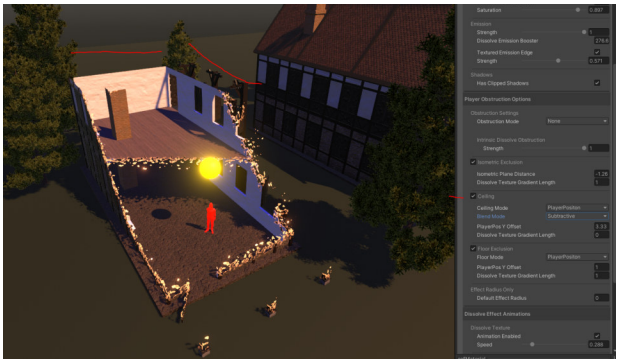
Activates Ceiling.

Picture/s Ceiling Toggle

Picture Ceiling Toggle off



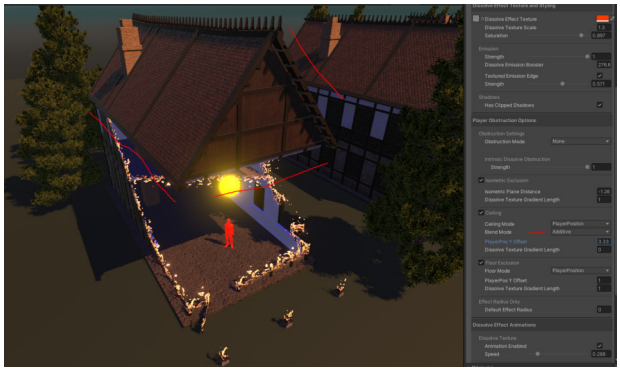
Picture Ceiling Toggle on



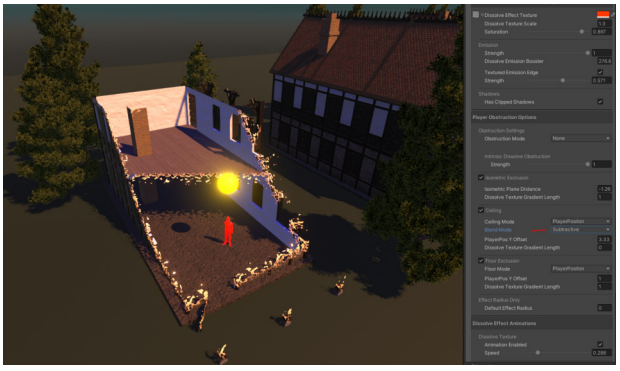
Property 2 - *Blend Mode*

Picture/s Ceiling Blend Mode

Picture Additive blend mode



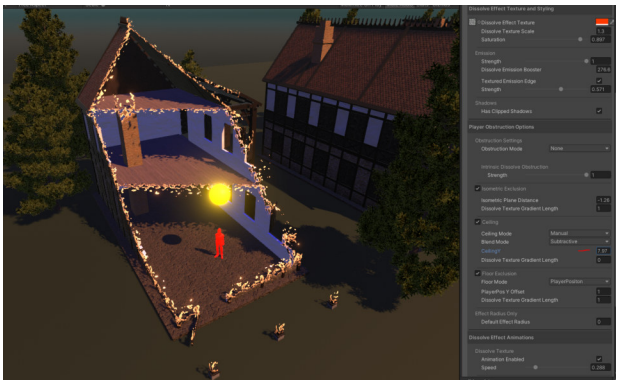
Picture Subtractive blend mode

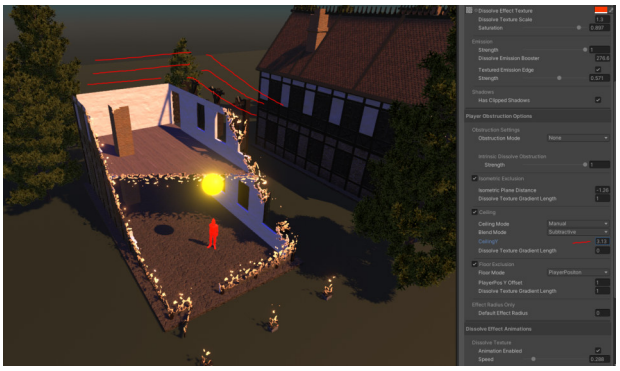


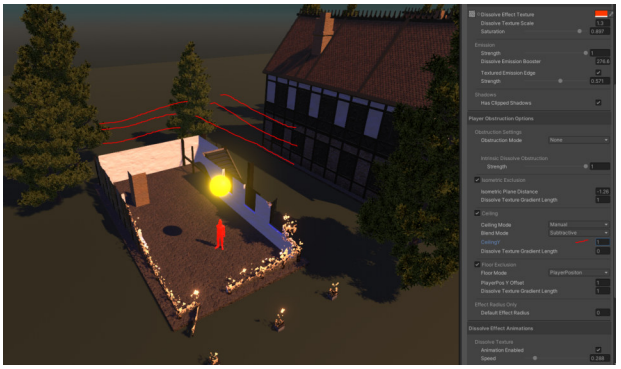
Property 3 - *CeilingY*

Manually Y value draw control

Picture/s Different CeilingY Settings





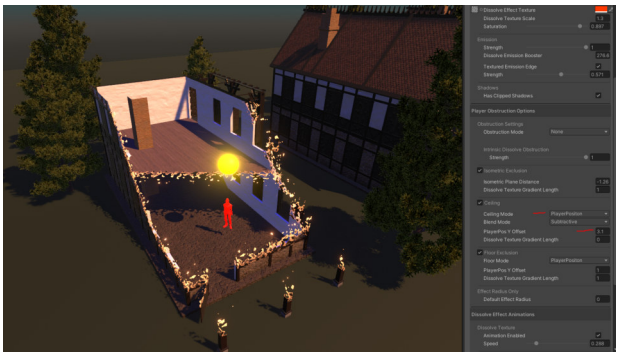


Property 4 - *PlayerPosition Y Offset*

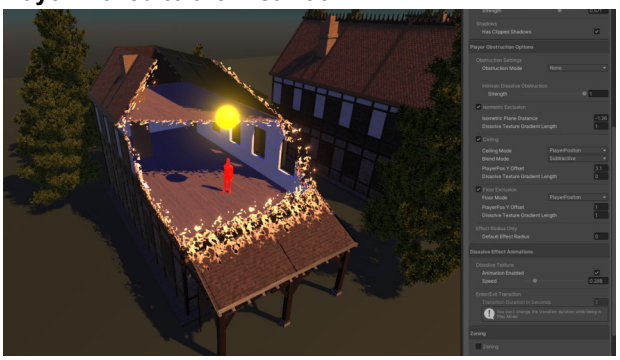
Automatic Ceiling Y offset draw control based on the player position

Picture/s PlayerPositionY Offset

PlayerPositionY Offset



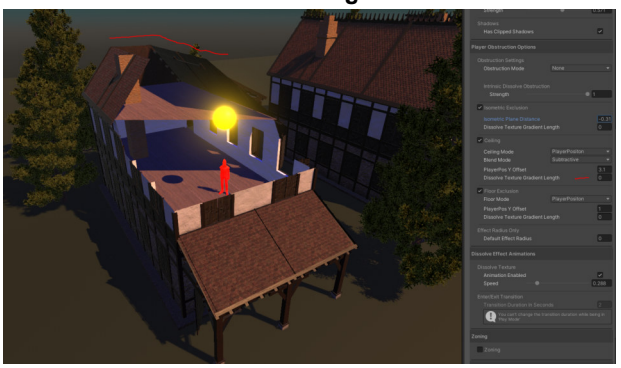
Ceiling PlayerPosition Y Offset,
Player moved to the first-floor



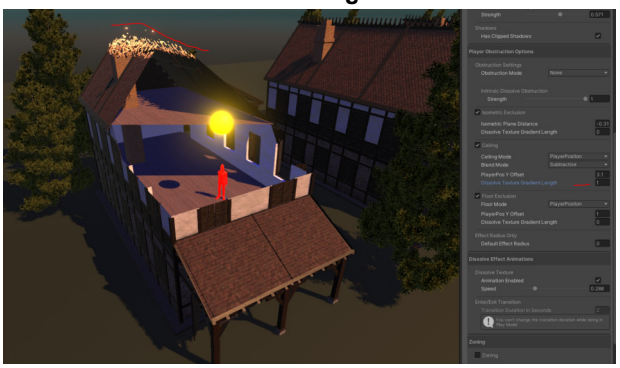
Property 5 - *Dissolve Texture Gradient Length*

Picture/s Dissolve Texture Gradient Length

Dissolve Texture Gradient Length 0



Dissolve Texture Gradient Length 1



Floor Exclusion

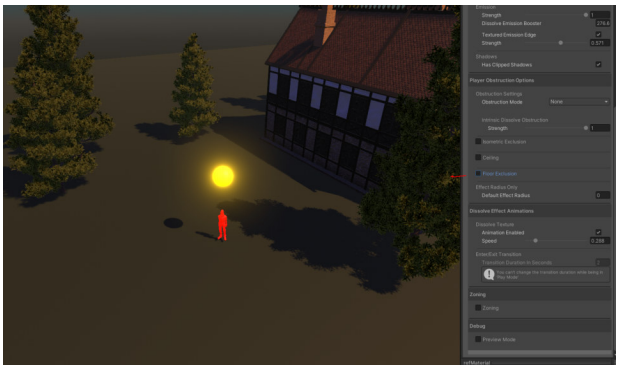
Floor Exclusion is an important feature and manges the behavior of drawing floors and or walls up to a variable height.
Everything below that y value will be fully drawn, so this feature is mainly used to have a base floor and or walls that are not affected by the dissolve effect.

Property 1 - *Floor Toggle*

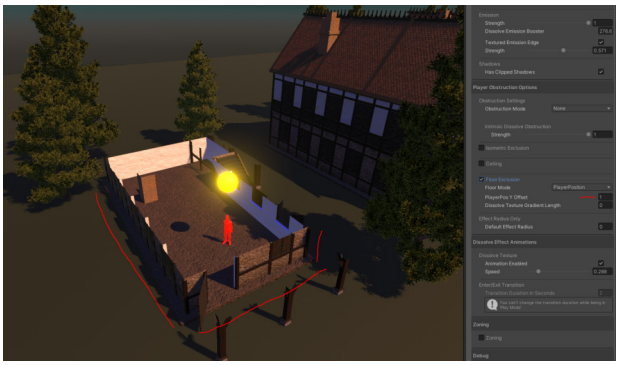
Activates Floor Exclusion.

Picture/s Floor Toggle

Floor mode off



Floor mode on

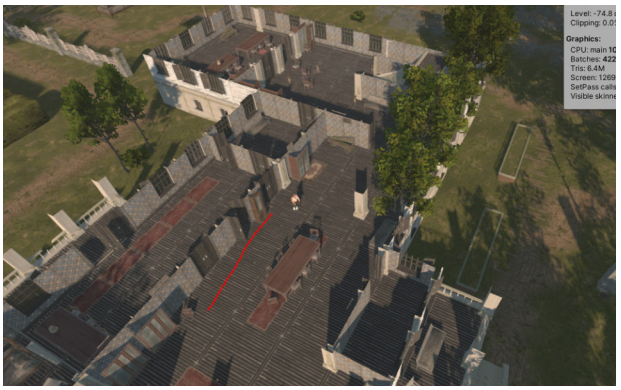


Property 2 - *Floor Mode*

Set the floor mode to the player if you want to draw the floor and other mesh to the player's y position or set it to manually.
This is especially important where you have several floors, so the value of the shader drawing is always accurate to the player's position.

Picture/s floor mode

Floor mode player automatic drawing mesh at player position + player position y offset value, so the floor and the walls and interior are drawn

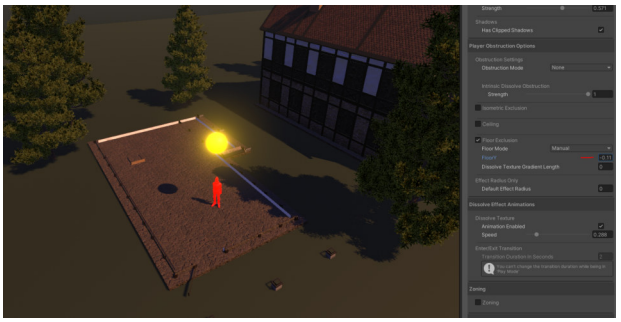


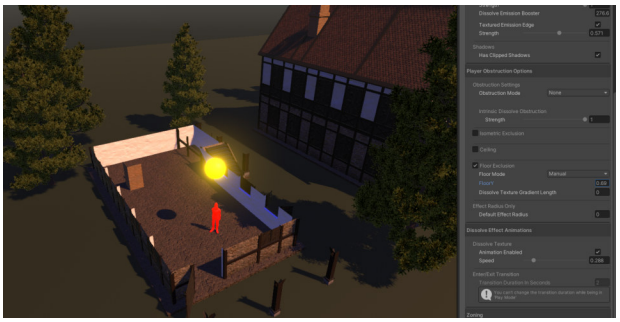
Floor mode player automatic drawing mesh to the second floor + y offset for next walls and floors

Property 3 *- Floor Mode Manual Y*

Use this option where you don't want to adjust the effect according to the Player Position but instead have a manual Y which defines the “floor base”

Picture/s Various Manual Y Values





Property 3 *- PlayerPosition Y Offset (Floor Mode PlayerPositon)*

Active when Floor mode = player.

As you want to draw the floors interiors and or the walls (any mesh belonging to the building), this value determines the units floors and walls are drawn in according to the player position.
Mainly this is important with the floor mode player in multistory buildings as when you move the player to the second floor; the whole first floor is automatically drawn in addition to the floor of the second floor and so on.
Also, use this value to regulate if you want to remove the roof or want walls and interior to be seen to a height of x units.

Picture/s 17 PlayerPosition y offset

Just show the floor

Also, show the walls

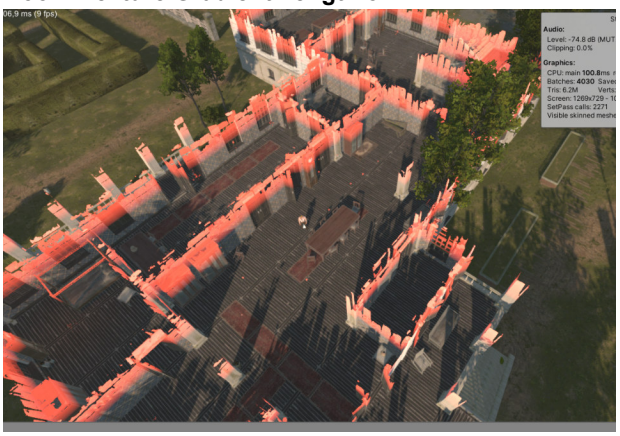
Property 4 - *Dissolve Texture Gradient Length*

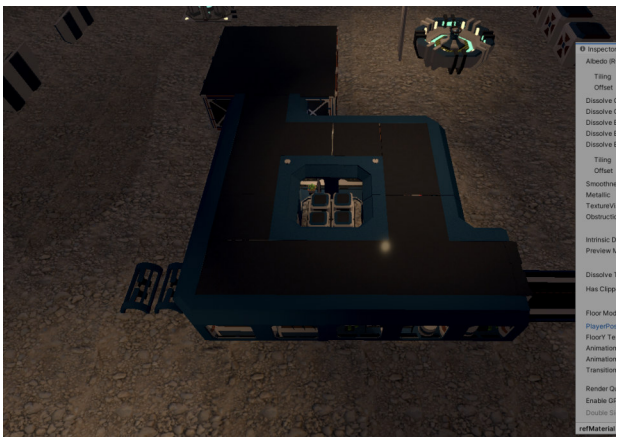
Controls gradient area between dissolve texture and mesh.

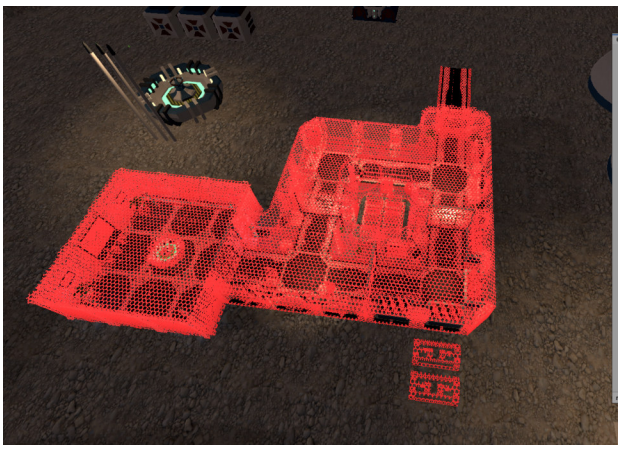
Picture/s Gradient transition length

FloorY Texture Gradient Length off

FloorY Texture Gradient Length on







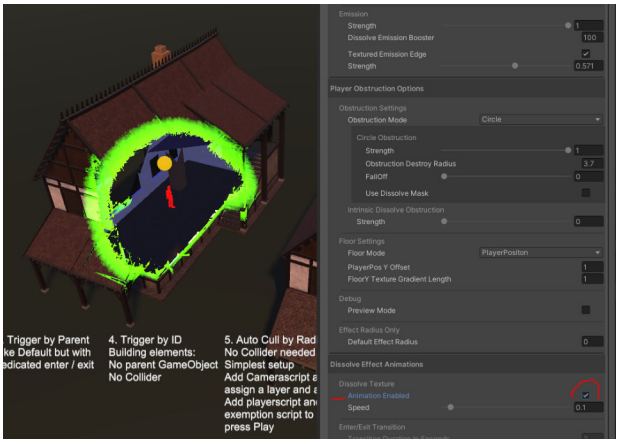
Dissolve Effect Animations

Property 1 - *Animation Enabled*

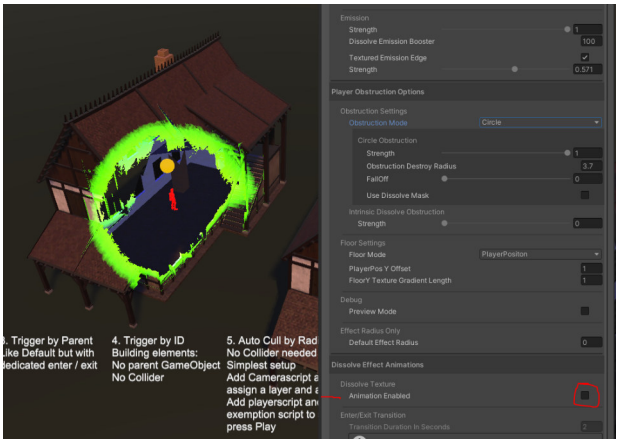
Dissolve texture animation enabled or disabled

Picture/s 15 Animation Enabled

On



Off

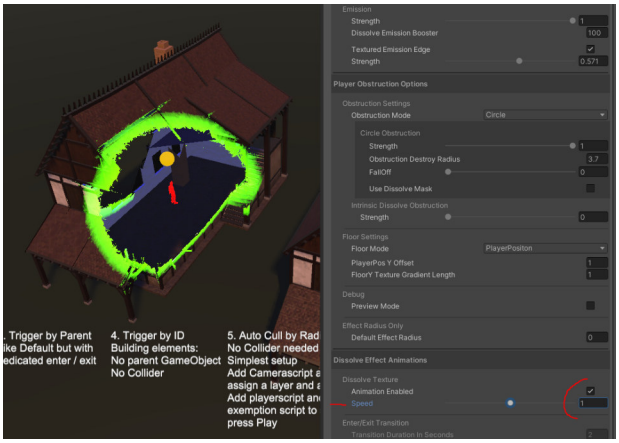


Property 2 - *Animation Speed*

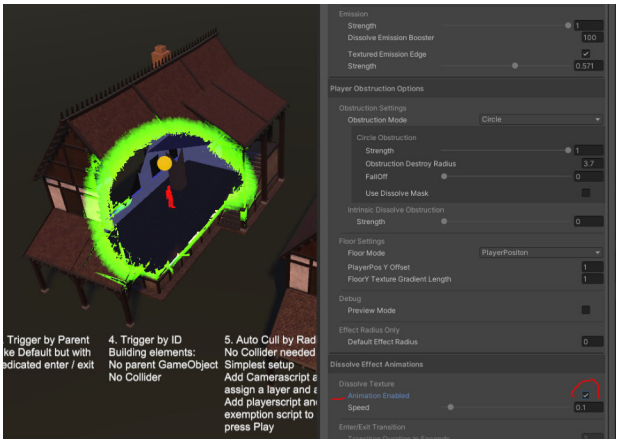
Dissolve texture animation enabled or disabled

Picture/s 16 Animation Speed

Animation speed 1



Animation speed 0.1

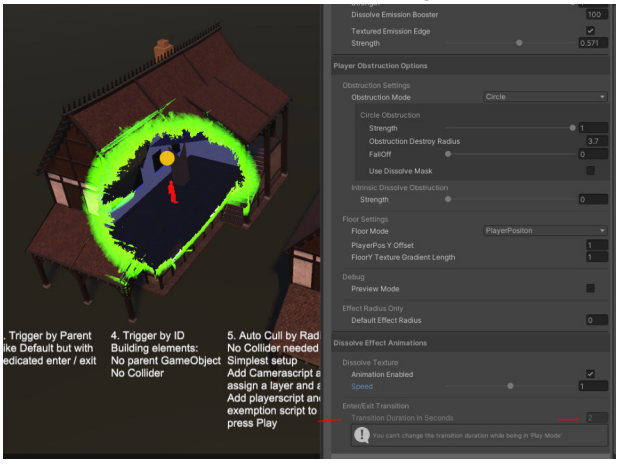


Property 3 - *Transition Duration in Seconds*

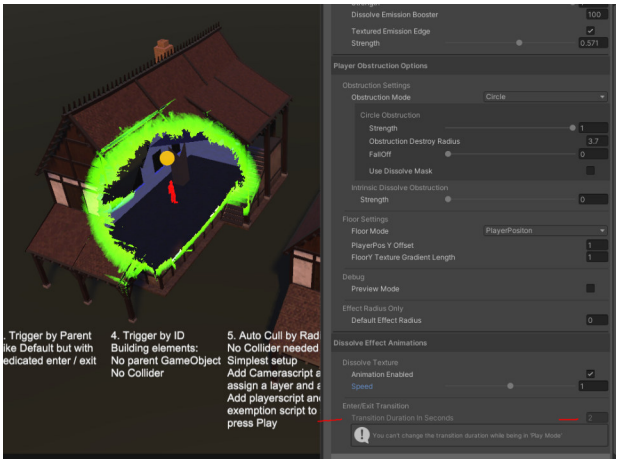
Length of the transition animation effect when entering and exiting a building.

Picture/s 21 Transition Duration in seconds

Transition duration in seconds setting

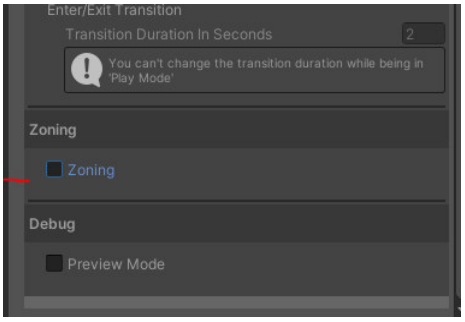


Set to 2



With the **Zoning feature**, you can add Cubes and configure the shader settings to **dissolve everything outside the cube or inside**.
The primary use cases are unrevealed rooms (Additive mode) or revealing individual rooms.
Cross-section or cut use cases are also possible where part of the building is cleared with the help of a Zone box.
Zoning supports grouping (connecting) Zone Boxes together across buildings.
Zones Boxes have colliders that will trigger the effect when entered by the player(s) entered.

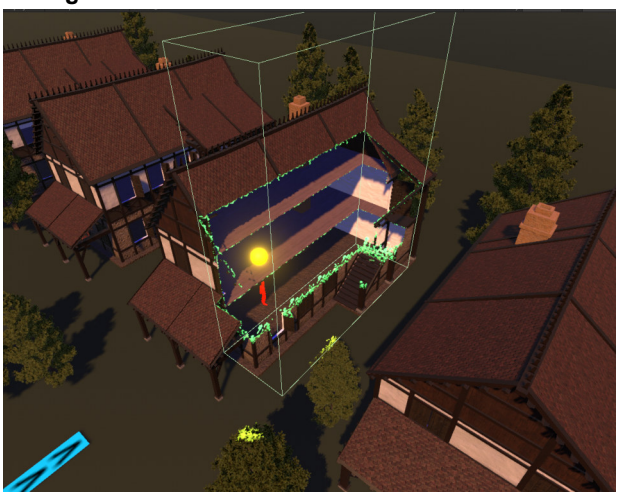
Property 1 - *Enable / Disable Zoning*



Picture/s Zone effect enabled / disabled

Zoning Disabled

Zoning Enabled



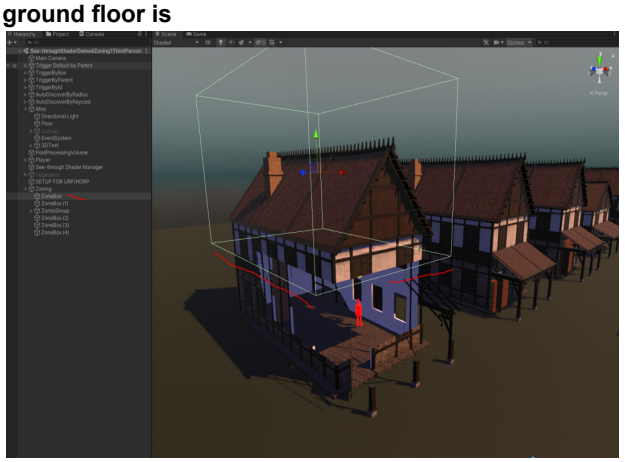
Property 2 - *Zoning Mode*

Set the Zoning mode feature to an additive for dissolving everything outside or subtractive to dissolve the inside of a box.

Picture/s Zoning mode additive examples

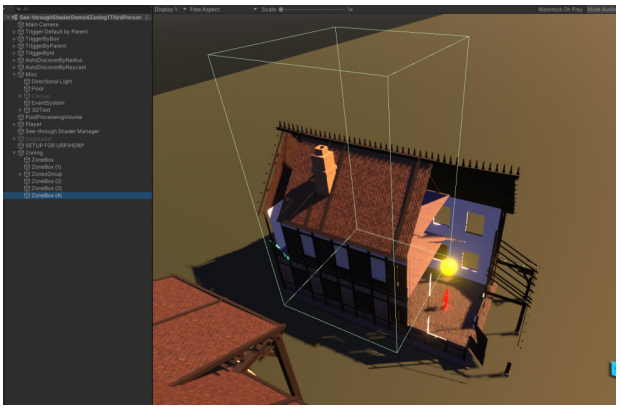
Picture Zoning mode additive example, Rooms with Zoneboxes remain unrevealed until revealing

Picture Zoning mode additive example. The roof with a Zone box is not revealed. At the same time, the ground floor is

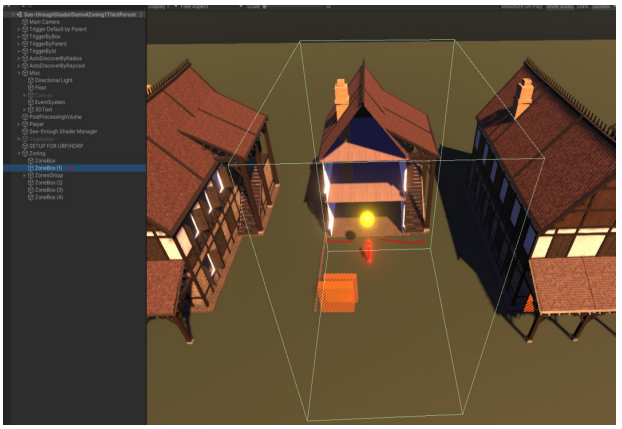


Picture/s Zoning mode subtractive examples

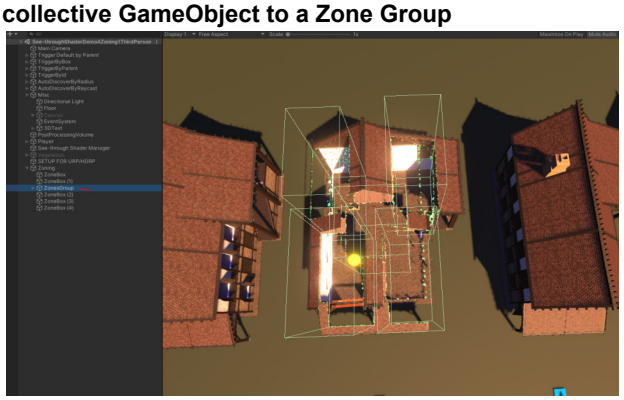
Picture unrevealed Area; the player has to enter the Zone box to reveal



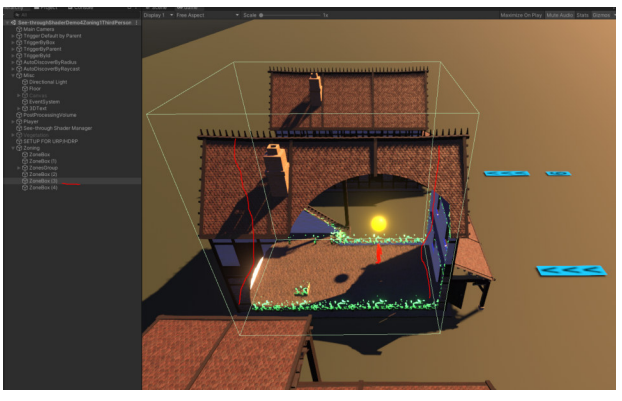
Picture Achieve cross-section “cuts” by using a Zone box and disable the floor mode feature



Picture Multiple Zone Boxes grouped under a collective GameObject to a Zone Group



Picture Zoning mode subtractive with the default effect radius feature

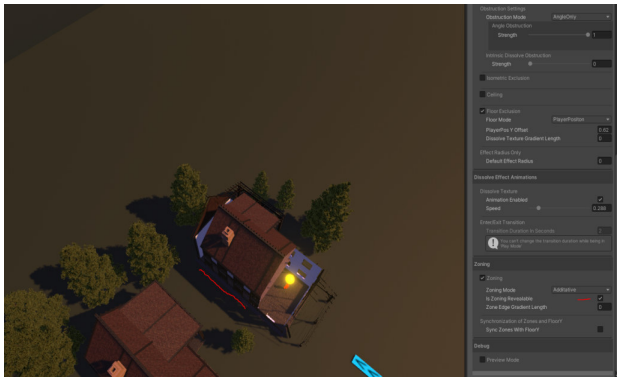


Property 3 - *Is Zone revealable*

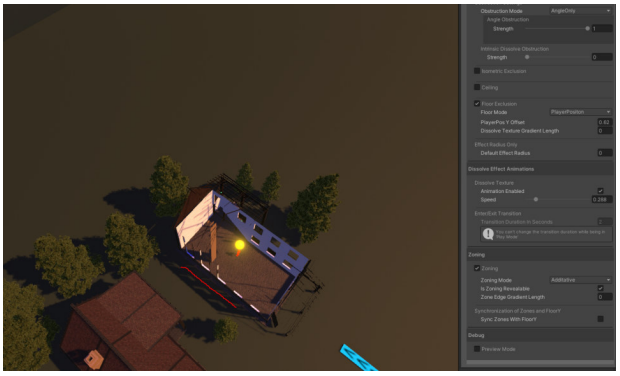
Checkbox to handle if the Zone box can be revealed.
If this is true and the player enters the Zone box, it will be revealed with a transition animation.
Use this checkbox to make the zone revealable according to your game use case.

Picture/s Reveal option of the Zoning feature

Picture Zone Box not revealed



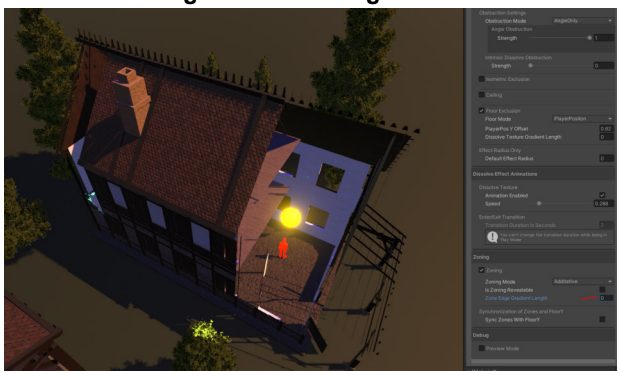
Picture Zone Box revealed



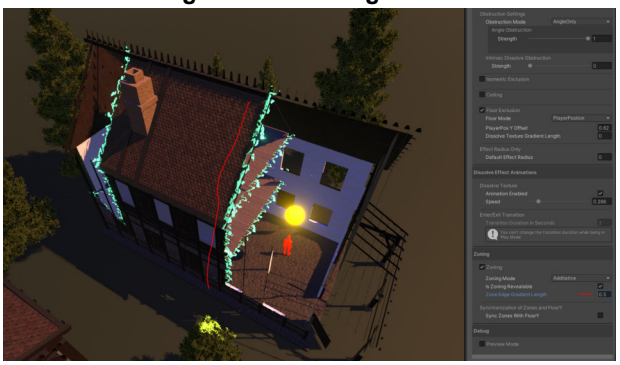
Property 4 - *Zone Edge Gradient Length*

Controls gradient area of the textured edges, zero means no animated edge texture.
Use this property according to your needs, whether you want an extra texture effect on the edges or a clean, sharp edge.

Picture Zone Edge Gradient Length 0



Picture Zone Edge Gradient Length 0.5

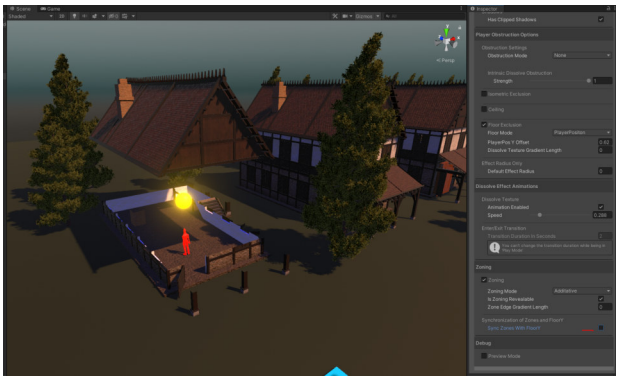


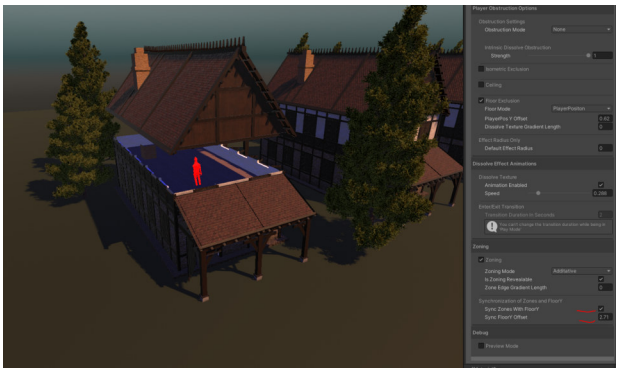
Property 5 - *Sync Zones with Floor Y*

Enable / DIsable the Y sync function for Zones
Synchronizes the drawing of Zoneboxes in relation to the player position.
This is mainly used in Multi-floor buildings to draw the Zone Area (Secret Rooms, etc.) when the player increases.

Picture Additive Zonebox with Disabled Sync
The Secret room at the roof is visible on the ground floor

Picture Additive Zonebox with Enabled Sync
The Secret room at the roof is visible ONLY from the first floor upwards

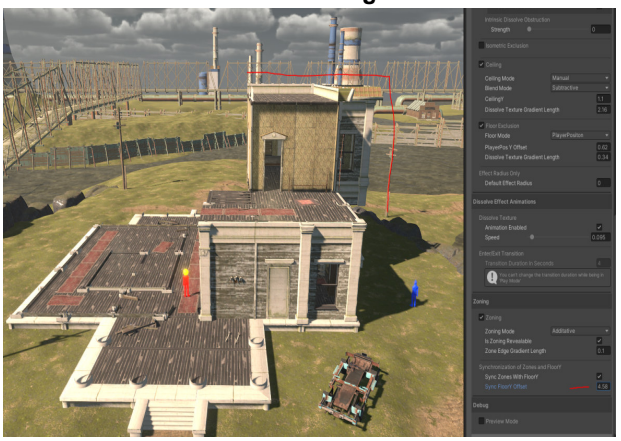




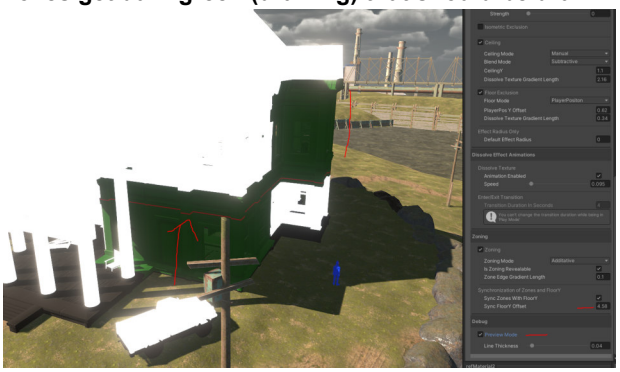
Property 6 - *Sync FloorY Offset*

This value is handling the Zoning Y Floor feature. Hence, its adjustment affects the Y Level at which the shader should draw the Zone Boxes.
Use the preview function and the red indicator line to set this value to your building's floor height.

Picture Zoning Sync FloorY offset too high, the hidden room is visible from the ground floor

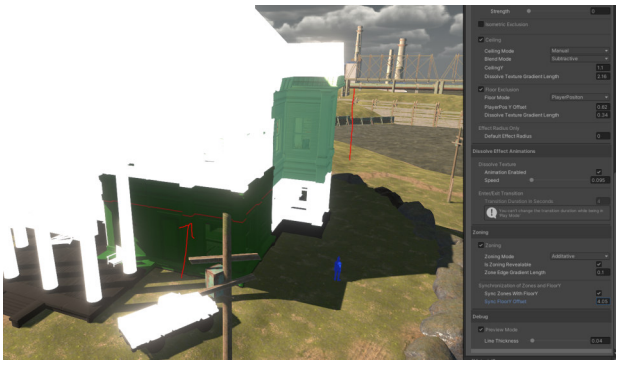


Picture Zoning Sync FloorY offset too high, in preview mode
Adjust the red line with the offset value, so just Zone Boxes get dark green (drawing) that should be drawn



Picture Zoning Sync FloorY offset value correctly adjusted, just the Zone Boxes on the ground floor are now visible. They become visible when the player goes up the feet

Picture Preview mode with the correct Y offset adjustment, now the Zone Box in the first floor turned light green (not drawing)



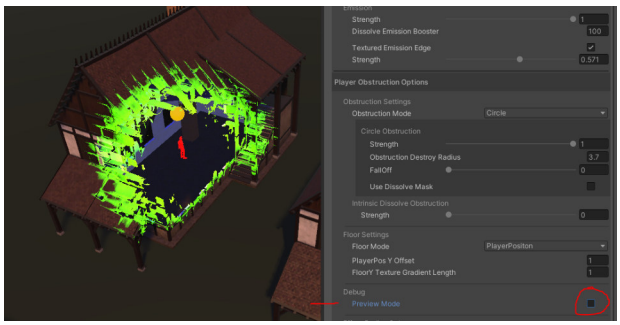
Debug

Property 1 - *Preview Mode*

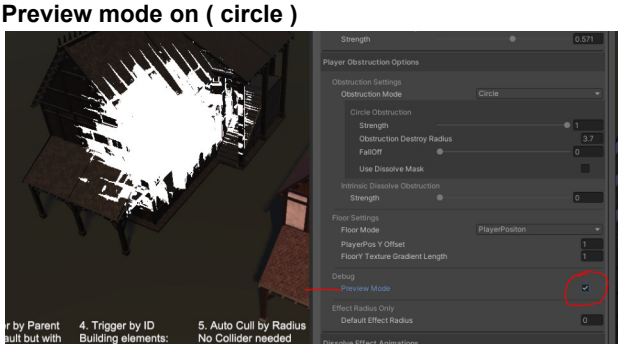
Preview mode for the runtime that shows the dissolve areas within a range of white (100 % dissolve and black (no dissolve).
This is useful for having many objects with different textures and having a more straightforward debug mode when adjusting settings.
Especially for the Zoning feature, we built a helper logic in the preview mode to help set up the offset correctly for showing the following Zone Boxes on floors above the player. A red line indicates the Y offset, and light and dark green color areas show if the Zone will be visible at the current player position height level.

Picture/s 10 Preview mode

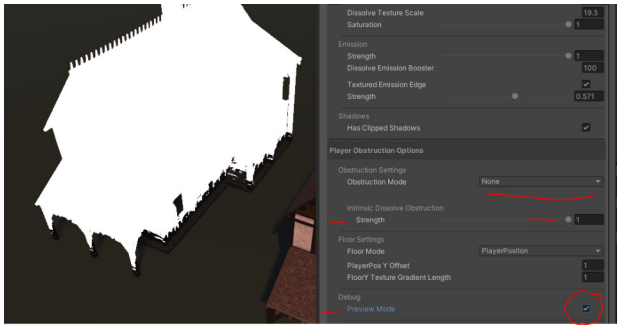
Preview mode off



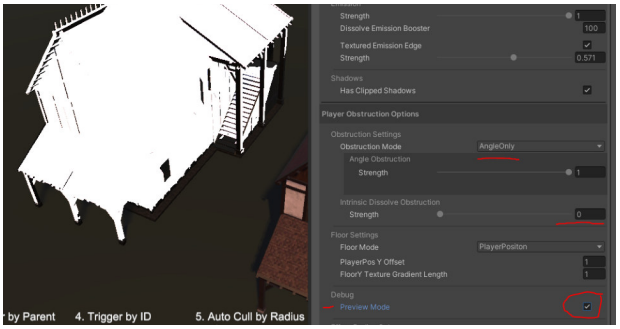
Preview mode on (circle)



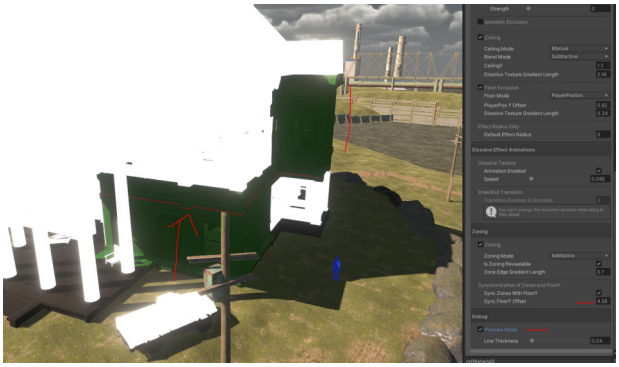
Preview mode (none)



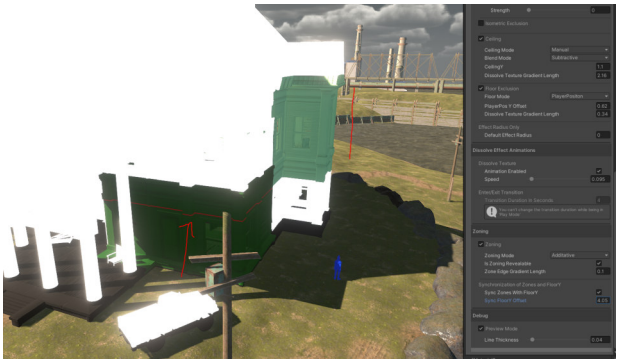
Preview mode (angle)



Picture Preview mode zoning helper
Dark green color indicates that the Zone Box on the floor above is visible at current player position height level



Picture Preview mode zoning helper
Light Green color indicates that the Zone Box in the floor above is not visible at existing player position height level



Extend Your Existing Custom Shaders

Note: Currently you can't use your custom STS shader in conjunction with the Global- and Group Replacement Scripts. We are working on this and will add it as soon as possible in a future update!

ShaderGraph

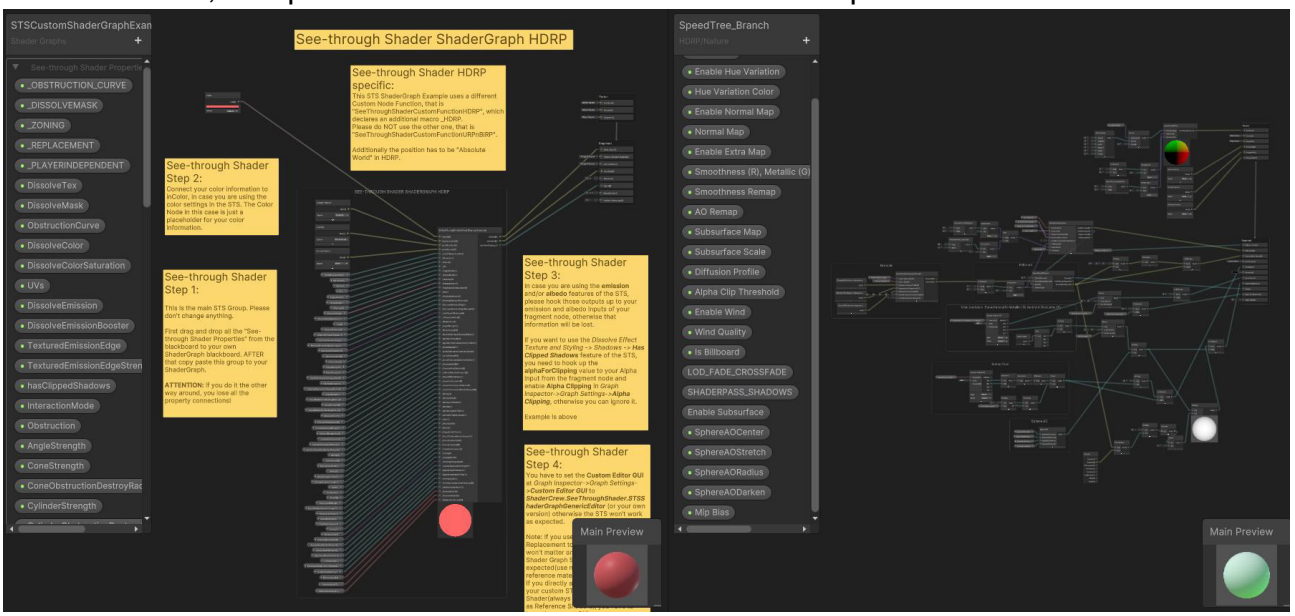
Extending your own ShaderGraph with the "See-through Shader" functionality is very easy and will take you mere seconds.

Step 1 - *Open up the ShaderGraphs:*

First go to **Assets > See-through Shader > Core > STS Shader Customization > STS ShaderGraph** and depending on your your RenderPipeline choose the **URP** or **HDRP** folder.

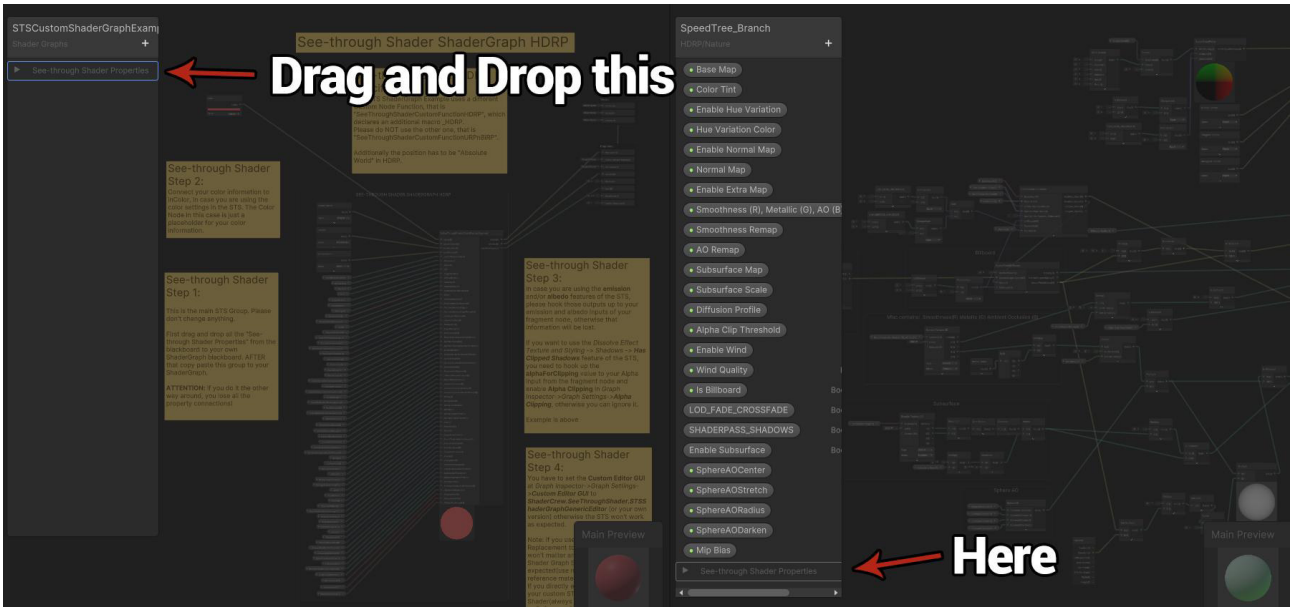
Inside those folders are example ShaderGraphs, like in this case for HDRP:

After you opened up the example, do the same for your existing ShaderGraph, that you want to extend, and place the window next to the STS example one:



Step 2 - Add the See-through Shader Properties:

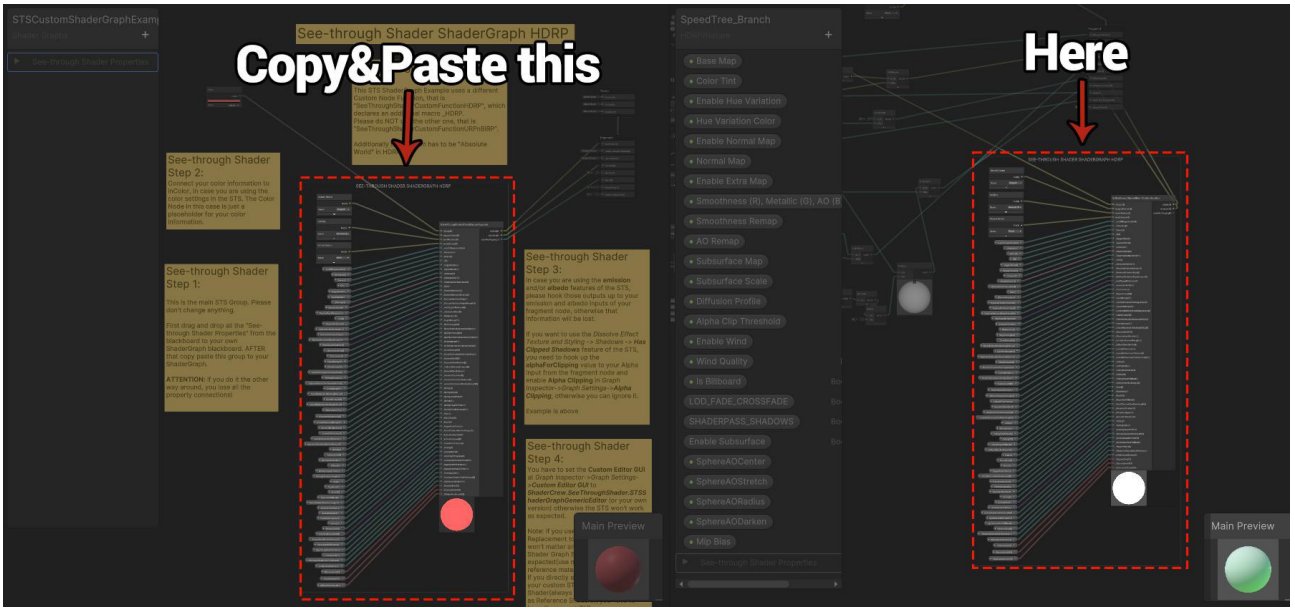
After this select the “See-through Shader Properties” category in the STSCustomShaderGraphExample Blackboard and drag it into your own ShaderGraph Blackboard:



Note: Do the next step only after you added the STS properties to your Blackboard. If you already did Step 3, undo all your changes to your own ShaderGraph and start the tutorial from the beginning, as otherwise you will have to assign all the properties manually to the STS Node Group, which takes a lot of time, trust me, and you don't want to do that! ;)

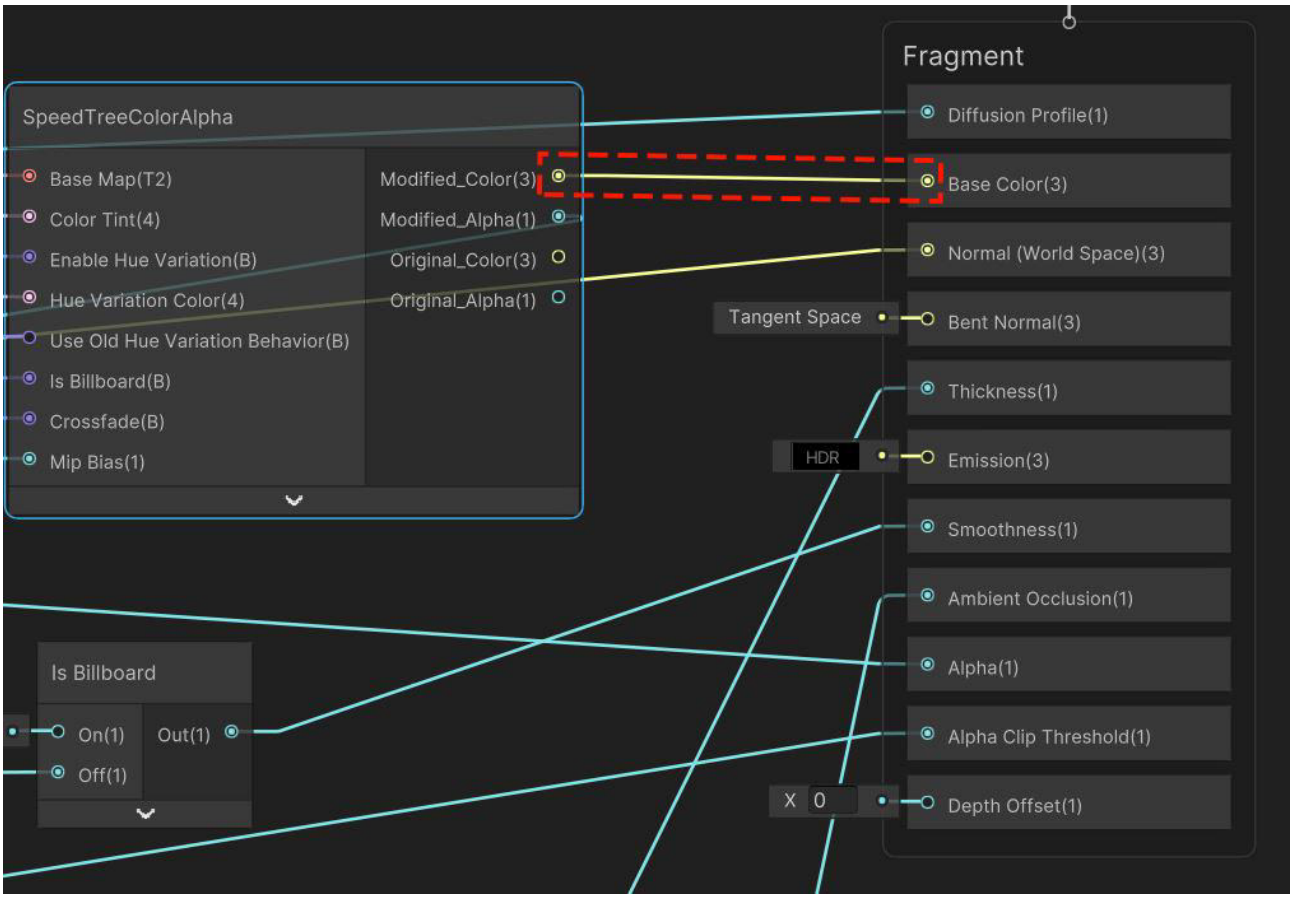
Step 3 - Add the STS Group Node to your ShaderGraph:

Now select the **SEE-THROUGH SHADER SHADERGRAPH** Node Group inside the STS example ShaderGraph and copy&paste it into your own Shadergraph:

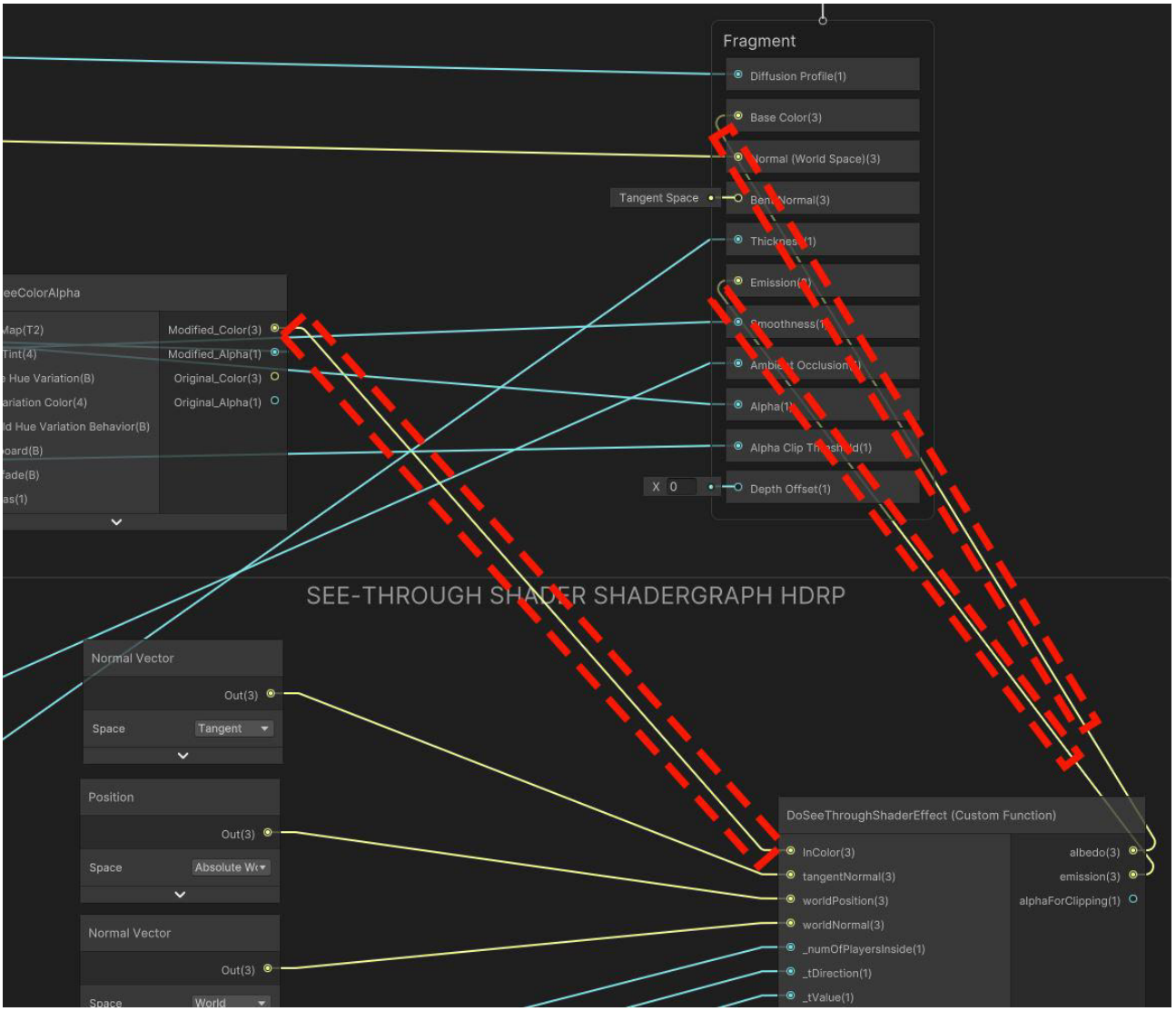


Step 4 - Hook up the STS Group Node to your Fragment Node:

This step requires you to figure out which Vector3 is going to the Base Color of your Fragment Node.



You need to hook it up instead to the **InColor** property of the STS Group Node and then hook up the **albedo out** node from the STS Group Node to the **Base Color** Property from the Fragment Node instead.
After this hook up the **emission out** node from the STS Group Node to the **Emission** Property of the Fragment Node.



If you want to use the ***Dissolve Effect Texture and Styling -> Shadows -> Has Clipped Shadows*** feature of the STS, you need to hook up the ***alphaForClipping*** value to your Alpha input from the fragment node and enable ***Alpha Clipping*** in ***Graph Inspector->Graph Settings-> Alpha Clipping***, otherwise you can ignore it.

Step 5 - Adding a STS based Custom Editor GUI

Please add **ShaderCrew.SeeThroughShader.STSShaderGraphGenericEditor** as the Custom Editor GUI of your ShaderGraph. This is needed so the functionality of the STS gets preserved:

You can also extend the SeeThroughShaderEditorAbstract and add your own custom GUI for everything not related to the See-through Shader.

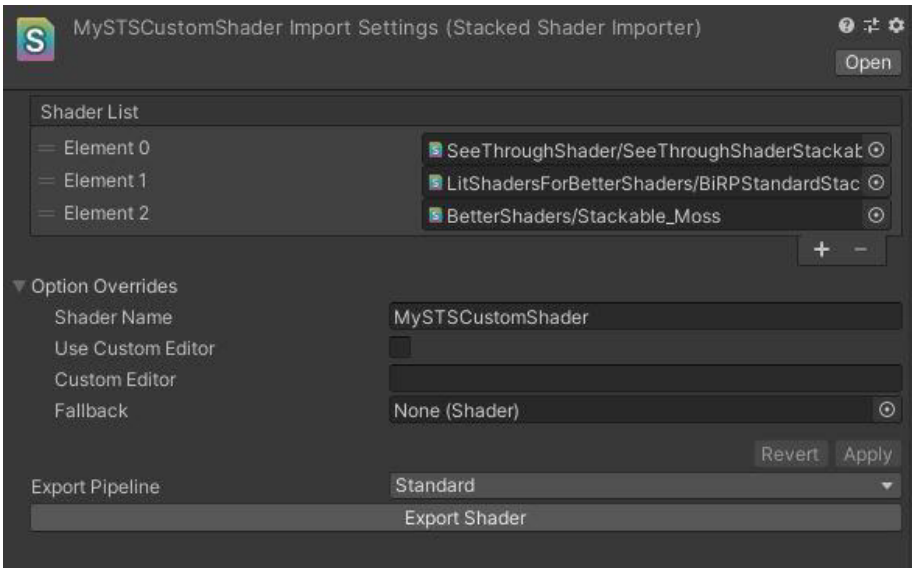
```
public class STSShaderGraphGenericEditor : SeeThroughShaderEditorAbstract
{
```

FINISHED - Congratulations! You did it

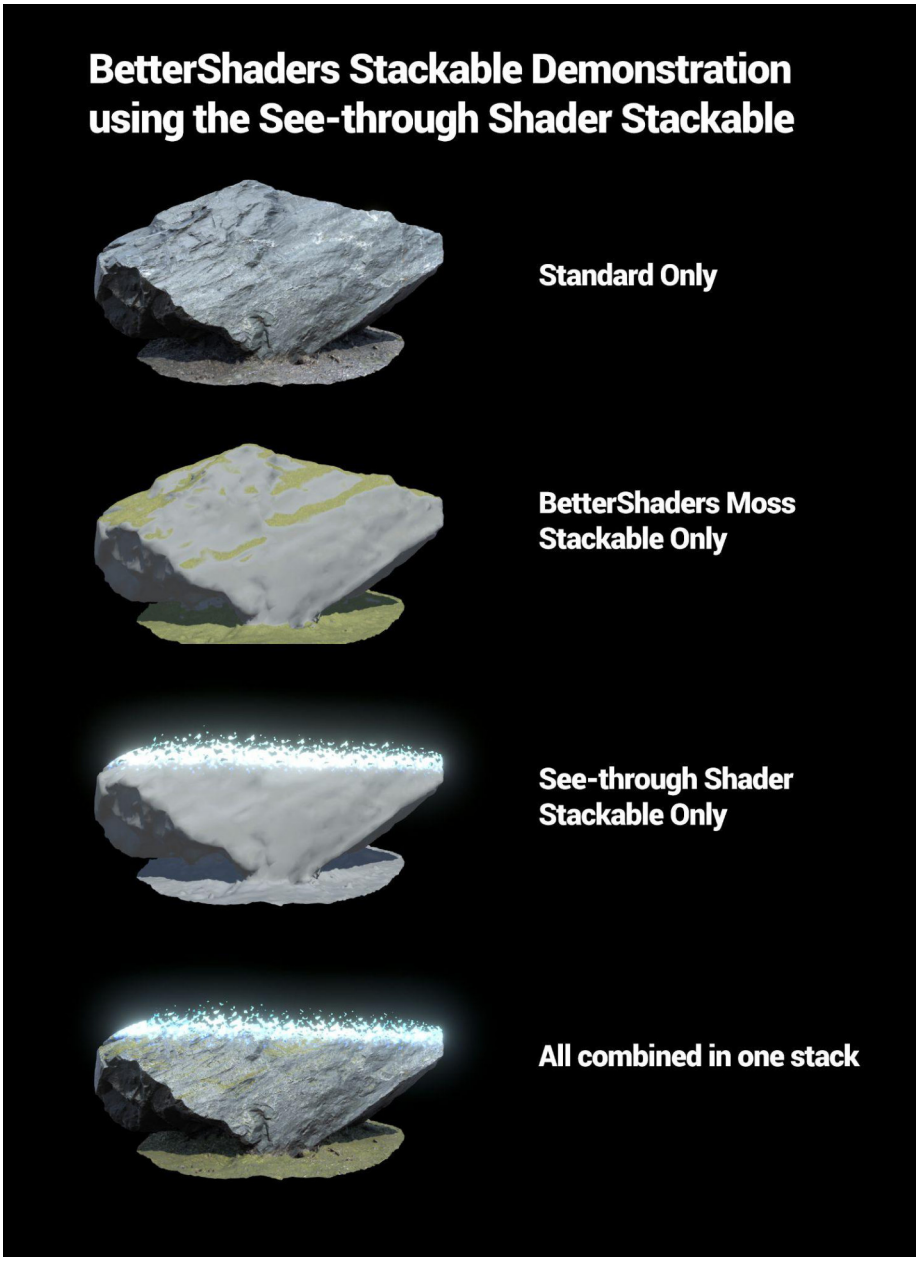
If you got any further questions, please don't hesitate to contact our support. We are always happy to help! :)

BetterShaders Stackable

Using the See-through Shader Stackable is very straight-forward. Just go to ***Assets > See-through Shader > Core > STS Shader Customization > STS Stackable for BetterShaders*** and drag the ***SeeThroughShaderStackable.surfshader*** inside your BetterShaders Stacked Shader.
Or search for "SeeThroughShader/SeeThroughShaderStackable" in the shader selection window in your BetterShaders Stacked Shader.



With a visual demonstration below:



Scripts Overview

Player

Script - *PlayersPositionManager.cs*

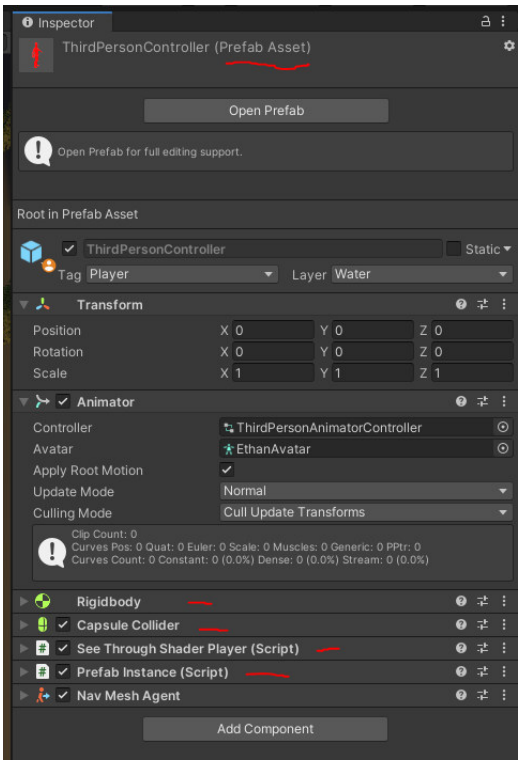
Description:
This is the script that **MUST be applied** to an empty or helper GameObject. You have to add playable characters to the playable character list for the asset to work.
Drag this script on some Gameobject in the unity editor.
This script makes the positions of the playable characters in the list globally available to the “See-through Shader.”
You can have up to 100 playable characters* on that list. If you require this asset to support more, please let us know!

*theoretically even more if you adjust the array length inside the shader and in the Constants.cs script

Script Variables in unity Inspector:
This Script has no public variables.

Script - *PrefabInstance.cs*

Script for adding Prefabs to the See-through Shader System after Instantiation.
You need this script if your player is not in the scene at the Game start and will come later into the Game.
Don't forget to add also a RigidBody, a Collider, and the SeeThroughShaderPlayer script to the prefab as they are also needed.



Script Variables in unity Inspector:
This Script has no public variables.

Shader Replacement

Script - *GlobalShaderReplacement.cs*

Description:
Script for the Camera or a helper / empty GameObject that adds the See-through Shader to all GameObject which match the LayerMask supplied.
It is mainly used to add the shader to all building materials in contrast to the "SeeThroughShaderGroupReplacement" script, which works from a parent GameObject downwards.

Script Variables in unity Inspector:

Name	Type	Description
LayerMask	LayerMask	The Layer of the elements where the shader should be applied to
ReferenceMaterial	Material	Reference Material from where the Shader settings should be taken. Use own template Materials with See-through Shader selected under „Shader" or The refMaterial supplied with the Asset under /core

Script - *GroupShaderReplacement.cs*

Description:
This script will assign the Shader to building elements belonging to a group.
In contrast to the replacement shader, which will add the shader to a whole layer, this script will work with just a group of GameObject and so is more specific.
It also synchronizes the settings from its assigned reference Material to the parent transform.

Script Variables in unity Inspector:

Name	Type	Description
Reference Material	Material	The reference Material where the shader is taken from and also synchronized
Replacement Group Type	Enum	Parent,Box or Id
Parent Transform	Transform	The parent transform of the building or structure where the

		shader should be applied to
Trigger Box	TriggerBox	The box that surrounds all GameObjects the shader should be applied to
Trigger ID	string	Assign a unique id to any elements within the same building to find the trigger by id script.
LayerMaskToAdd	LayerMask	The Layer of the elements beneath the parent transform where the shader should be used
MaterialExemptions	List<Material>	List of Materials to exempt from the shader

Script - *ShaderPropertySync.cs*

Description:
Optional helper class.
Add this script to the parent GameObject of a building to sync See-through Shader settings from the refMaterial reference Material file supplied under /core or your own made template Materials.
The Sync can be done at the start or runtime in a continuous mode.
You can use this script or build your own to apply a reference material's changes to a building or object with See-through Shader active.
Mainly, you will load your initial settings with the other scripts at the start and sometimes use in-game changes and sync them to the parent's child elements where this script is applied.

Script Variables in unity Inspector:

Name	Type	Description
parent material<	Material	Drag the refMaterial supplied in the /core folder or your own made template materials with the See-through Shader enabled on this field to sync the shader settings from
Children	List<Material> (List of Material in unity editor)	Optional List of manually added Materials that at the start will be synced to. Mostly you will use the sync continues option to sync to all game objects beneath without specifying their forehand. But in some cases, the game designer wants just a handful of materials to be pre-added in the editor

		This Option. Will, of course, work with the continuous sync options
Sync Continuous	Bool (checkbox)	Sync the See-through Shader settings from the parent material File to all the children beneath this GameObject, which has the See-through Shader

Script - *SeeThroughShaderExemption.cs*

Description:
Optional helper class.
Add this manually to building elements that you want to exempt from See-through Shader.
In most cases, you will control this behavior by setting the layers in the scripts.

Script Variables in unity Inspector:
This Script has no public variables.

Trigger

Trigger by Parent

Script - *TriggerByParent.cs*

Description:
This is the main “Trigger script” as it works with a parent GameObject of the building or object. It is used with a whole collider surrounding the building or a dedicated enter/exit trigger pair positioned where the player passes, mostly at entrances.
For buildings with several entries, this script can be added to several “helper” GameObjects to support that.l

For the building elements, no colliders are needed. Just the trigger works with colliders in the “isTrigger=true” mode.
The Shader itself can be applied with the “***GlobalShaderReplacement***” script on the Camera globally or with the “***GroupShaderReplacement***” script on a parent GameObjectt.

Script Variables in unity Inspector:

Name	Type	Description
DedicatedTriggerType (Enter/ Exit)	Dropdown	In dedicated mode, if this is the enter or exit trigger object
Dedicated Trigger Parent	GameObject	In dedicated mode, the parent, drag the parent GameObject to this field. In non-dedicated mode, the attached GameObject will be taken as parent GameObject. A collider with isTrigger=true is necessary for both modes

RefMaterial	Material	Drag here the Material called refMaterial from the /core/Editor/Resources/STSReference Mmaterials Folder or your own created template Materials to get the shader settings applied from this Material file.
Is this a dedicated enter or exit trigger	bool	Dedicated mode enabled / disabled

Trigger by Id

Script - *TriggerByID.cs*

Description:
Use this script to set up a 3D Object in unity with a collider (isTrigger = actual) acting as a trigger.
When the player passes this trigger, the script will find all the elements with the corresponding id.
This trigger is for use cases where you have elements of a building that do not have a collider nor a parent object, so the trigger by box and triggering by the parent can not be used.
This trigger comes as a pair of entering and exit triggers, so in most cases, two cubes are created, and the box colliders are set to trigger=true.
Add this script to both cubes and place them where the player will pass it when entering and exiting; set one to be the entered script and the other one to be the exit with the variables described in the table below.

Script Variables in unity Inspector:

Name	Type	Description
triggerID	string	Assign a unique id to any elements within the same building to find the trigger by id script.
thisIsEnterTrigger	bool (checkbox)	Set to true for the enter trigger
thisIsExitTrigger	bool (checkbox)	Set to true for the exit trigger

Script - *TriggerObjectId.cs*

Description:
Optional helper class.
Simple mouse click Player Controller used for the Demo Scene.
This is very basic and was just made for the Demo Scene.
Please use your player controller for your use case; we used Assets for Moving, Navigation, and Pathfinding.
We would be happy to share the Links and help implement them.

Script Variables in unity Inspector:

Name	Type	Description
triggerID	string	Assign a unique id to any elements within the same building to find the trigger by id script.

Trigger by Box

Script - *TriggerByBox.cs*

Description:
This script will use the TriggerBox script to capture the building's elements within the collider of the surrounding box.
This script is added to two cubes with colliders set to isTrigger = true and set the position where the player passes when entering and exiting.
When passing the triggers with this script, it will apply the effect of the building's elements.
In the demo scene, there is a setup example of this script.

Script Variables in unity Inspector:

Name	Type	Description
triggerBox	SeeThroughTriggerBox (the script above)	Drag the surrounding box game object onto this field.
thisIsEnterTrigger	bool (checkbox)	Set to true for the game object (cube) acting as entering the trigger
thisIsExitTrigger	bool (checkbox)	Set to true for the game object (cube) acting as exit trigger

Script - *TriggerBox.cs*

Description:
Helper Script for the Trigger by Box script.
Add this to a cube with a collider and set “isTrigger = true” surrounding the whole building to “capture” the elements inside.
This is for use cases where you have a building with many elements and no parent GameObject, but these elements have colliders.
So when the player passes the Trigger by Box script, this script is used to get the overlapping colliders within the surrounding box.
The script has a Layer setting to choose which layers should be processed.

Script Variables in unity Inspector:

Name	Type	Description
Layer	Layer mask (Multi-select List in unity editor)	Select the layers with the building elements the box should trigger for

Building Auto-Detector

Script - *BuildingAutoDetector.cs*

Description:
This script determines automatically if the player is within a building or not.
This will work for most buildings and some rare building forms; the usual triggers can be used where a parent GameObject is supplied.
This script is mainly used in a click-and-go environment where the buildings are not unusually shaped.
The fastest way for a setup is to add the player global vector script and this script to the player and the camera script to the camera.
After that select the layer you want it to apply on the “replacement shader sees through” script on the camera and press play.
Combining this script and the „standard“ triggers is possible, so use the AutoDetection Scripts and the more exact Trigger scripts as you need them.
This script works with an algorithm and raycasts to determine where the player is relative to the buildings around.
Some of the building elements need a collider for better algorithm detection, and a dedicated video on our youtube channel and screenshots within this manual of building examples are provided.

Script Variables in unity Inspector:
This Script has no public variables.

Toggles

Script - *ToggleByClick.cs*

Description:
Add this to the root of a GameObject together with a collider (Raycast based).
On click it will activate / deactivate the effect to all childelements.

Script Variables in unity Inspector:
This script has no public variables.

Script - ToggleByClickZonesOnly.cs

Description:
Add this to the root of a GameObject together with a collider (Raycast based).
On click it will activate / deactivate the effect within a clicked ZoneBox.

Script Variables in unity Inspector:
This script has no public variables.

Script - ToggleByUI.cs

Description:
Add this to the root of a GameObject you want to work with.
Create a UI Button and drag the Button onto the Button field.
The Button will activate / deactivate the effect for the Object and all childelements on Click

Script Variables in unity Inspector:

Name	Type	Description
Button…	Button	UI Button to trigger the effect

Script - ToggleByUIZonesOnly.cs

Description:
Add this to the root of a GameObject you want to work with.
Create multiple UI Buttons and ZoneBoxes and drag them onto the Button and Zonebox fields.
The Buttons will OnClick activate the effect in the corresponding ZoneBox.
Button1 will Trigger ZoneBox1 in the List.
This script has fields for three Buttons and ZoneBoxes.
You can easily copy this script and add more Buttons and ZoneBoxes

Script Variables in unity Inspector:

Name	Type	Description
Button1	Button	UI Button to trigger the effect on Zone
Button2	Button	UI Button to trigger the effect

		on Zone
Button3	Button	UI Button to trigger the effect on Zone
Zone1	SeeThroughShaderZone	ZoneBox for Button1
Zone2	SeeThroughShaderZone	ZoneBox for Button2
Zone3	SeeThroughShaderZone	ZoneBox for Button3

Zoning

Script - *SeeThroughShaderZone.cs*

Use the hierarchy right-click *See-through Shader->Zoning* context menu for this feature; otherwise, the zone isn't set up correctly.

A script that handles Zone Boxes.

Script Variables in unity Inspector:
This Script has no public variables.

MISC

Script - *MeshCombine.cs*

Description:

Optional helper Script for use with the BuildingAutodetector.cs to make a combined mesh collider from all meshes beneath the parent it is added. It can make a combined collider from the meshes and then deactivate these colliders, so you have one combined mesh for colliding raycasts, etc.
Use it for your use and in your scripts as you like.
After adding this Script to a parent GameObject, it will process the meshes and add a mesh collider with the combined meshes of the beneath mesh.
This will work with **simple to middle complex buildings,** but the combined mesh will be mostly like the original meshes with thousands of meshes of the complex interior.
Please use this optional script for the cases described above; we will also provide videos and documentation with example buildings here.

Script Variables in unity Inspector:
This Script has no public variables.

Script - *PlayerDemoController.cs*

Description:
Optional helper class.
Simple mouse-based click Player Controller used for the Demo Scene.
This is very basic and was just made for the Demo Scene using unity Navmesh.

Script Variables in unity Inspector:

Name	Type	Description
MovementMask	LayerMask	LayerMask for the Ground

Demos Overview

This chapter is about the Demo scenes included in the Asset and their respective use cases.
Path of the Demo scene folder:
Assets/See-through Shader/Sample Scenes/

Demo 01 - ThirdPersonPlayer

This is the **default demo scene** in a **classic third person player** based setup.
Click on the buildings to move them in and trigger the Shader effect.
Each of the 6 Buildings represents a possible way or use case for the Shader to operate.

Building #1:

TriggerByParent Setup with a parent GameObject and surrounding collider in isTrigger=true mode..

Building #2:

TriggerByBox Setup using building elements colliders to capture inside a Box and dedicated trigger / exit Trigger Cubes at the entry

Building #3:

TriggerByParent Setup with a parent GameObject and dedicated enter / exit collider cubes in isTrigger=true mode.

Building #4:

TriggerById Setup with assumed no parent GameObject and dedicated enter / exit collider cubes in isTrigger=true mode.
It is assumed the building gameobjects can not have colliders and grouping with a common parent GameObject is not possible
All the building elements have a common Id that mapps them to a distinct building.

Building #5:

Default effect radius setup.
This is the most basic setup of the See-through Shader with the Replacementshader and the PositionManager script attached to an empty GameObject.
As there are no Triggers used and no BuildingAutodetector script can be used as this building has no colliders the Shader will operate in the Default effect radius mode.
This is basically an adjustable radius around the player set in the RefMaterial that clears Mesh within this range.
It is great for Procedural or large Worlds where you do not want a per building setup (time).
Also Enter / Exit Animation transition is available in this mode.

Building #6:

BuildingAutoDetector script
This is the second Automatic detection scenario after the Default effect radius setup.
As you may have noticed reading through this Document mostly the setup is about Identifying building elements to a distinct building.
This can be done before the Game starts with the setup of triggers and so on or automatic

Demo 02 - ThirdPersonPlayer PrefabPlayer

Like See-throughShaderDemo1ThirdPersonPlayer but with Prefab (Players) Spawning after Start.
This would be the use case if your Player(s) are instantiated into the scene after start for example by pressing a UI Button.

Demo 03 - FirstPersonView NoPlayer

Camera driven (First Person View) use cases where the effect is triggered by a Cube (Crosshair) attached in forward direction to the camera.
When the Cube attached to the Camera is moved within a Building then the effect is applied, vice versa when exiting.
A mix of First Person View and Player(s) in the scene is possible.

Demo 04 - FirstPersonView NoPlayer Raycast

Second First Person View approach using Raycasts to place an invisible Cube within the Buildings and trigger the effect.
As working with Raycasts requires more performance we recommend using the First Person View Setup in the previous Demo Scene.

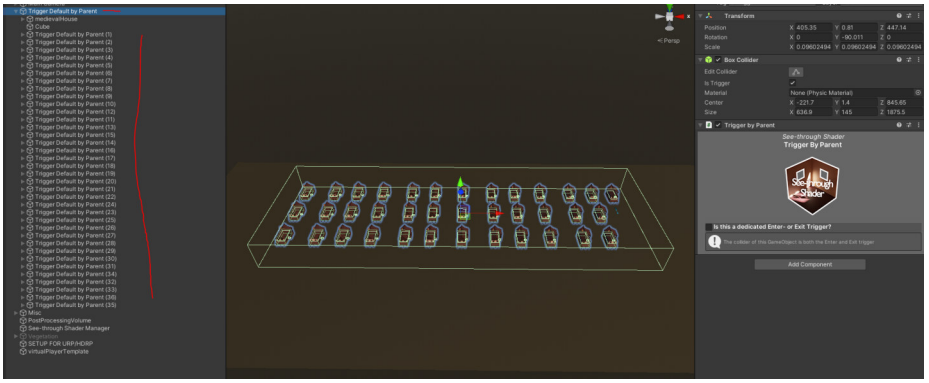
Demo 05 - FirstPersonView **MassPlacement DefaultEffectRadius**

This is a variation of a First Person View demo using the default effect radius with a large radius.
It aims to clear the mesh when the camera comes near a POI.
This setup can also be useful for First Person View in procedural Worlds as the clearing is generic without further setup for individual buildings.

Demo 06 - *FirstPersonView* MassPlacement TriggerAll

Another **Variation of a First Person View** Implementation putting all building GameObjects under a single Parent and the TriggerByParent script on top and a large surrounding collider.
When moving within distance with the camera all the POI will reveal at once with a nice transition effect.

Picture Demo 6



Demo 07 - *Zoning* *ThirdPersonPlayer*

Demo for the **Zoning** feature using the Default Demo 1 scene but with the addition of cube boxes (additive and subtractive setup).
The Demo demonstrates non revealed rooms scenarios as well as different “cuts” through the mesh including full cross section.

Demo 08 - *Zoning* *ThirdPersonPlayer*

Demo for the **Zoning** feature aiming for maximum performance by using **combined meshes** but with the **flexibility to have distinct building control.**
It achieves this by using Zone boxes placed at each building that will trigger the effect when player(s) enter.
This means you have the best performance with combining the meshes with the ability to handle each building individually.

Picture Demo 7

Demo 09 - *InteractionMode Independent*

Demo for the **Independent mode with Click and UI Based scripts.**
Also the use of **ZoneBoxes** with the respective **Toggles** is shown**.**

Picture Demo 7

For more information about each Demo and or your specific use case, please feel free to contact us.

Step-by-Step Guides

Prerequisites:
After downloading the package from the unity store, make sure you have the See-through Folders in your unity projects window.
Browse to the /core Folder and click the refMaterial file; now, in the Inspector, you should see the settings of the See-through shader.
If you can see the See-through Shader settings on the Assets/See-through Shader/Core/Resources/STSReferenceMaterials/refMaterial.mat file, you are good to go.

Automatic or trigger mode?
You have to decide to use the automatic mode, the trigger mode, or both combined.
Details are described in this document but here in short:
It's all about entering and leaving the building, you want it to be exact, so this Asset comes with two combinable modes of operation.

The automatic radius mode is for the simplest use case where you have buildings primarily, and it's enough to clear an area with a radius r around the player.

The automatic building detection script goes a step further and has a raycast algorithm to determine the player's position relative to the building.
This will work with simple to middle complex buildings and most of them.

As these two modes are the fastest ways to set up the asset, we recommend testing if one of them is suitable for your use case, and in addition to them, you can have some buildings with more complex structures where you have a trigger by parent script for example.

The second mode of operation is the triggers that will trigger when the player passes dedicated enter/exit colliders.
This will be more precise. For example “trigger by parent” script iterates over all items beneath a parent game object like unity. The most common trigger use case is the trigger by parent. Also, no colliders are needed for the building's elements.
The other two trigger scripts are the trigger by box and trigger by id, “trigger by box” will use an overlapping cube and capture all the colliders inside; this is useful where you have colliders on the building elements but no root parent.

TriggerById trigger script works where no collider and no parent game object available, the game designers assign a unique id or name to every building element so the trigger can find them.

We recommend for best practice if you have no parent game object to drag a selection box to select the elements of the building and move them beneath a parent game object as it makes handling more easier and also the sync script for in-game syncing can be applied when there is a parent game object.

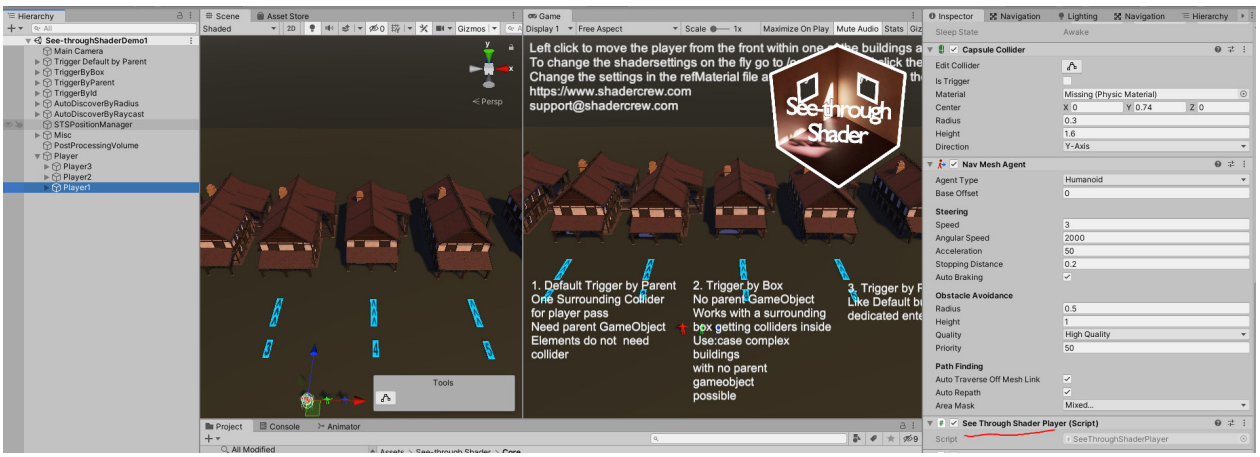
You are free to combine the automatic mode and the trigger mode to fit your game project.

The trigger modes work with colliders to trigger the effect, so the player must have a RigidBody be set to isKinematic=true) to collide.

Step 1 Always needed (Trigger and AutoMode based Setup):
1 Add the “PlayerPositionManager” script to an empty GameObject in the Hierarchy (Menu > GameObject > Create Empty)
1.1 Add as many slots as you have playable Characters and drag them onto the slots.

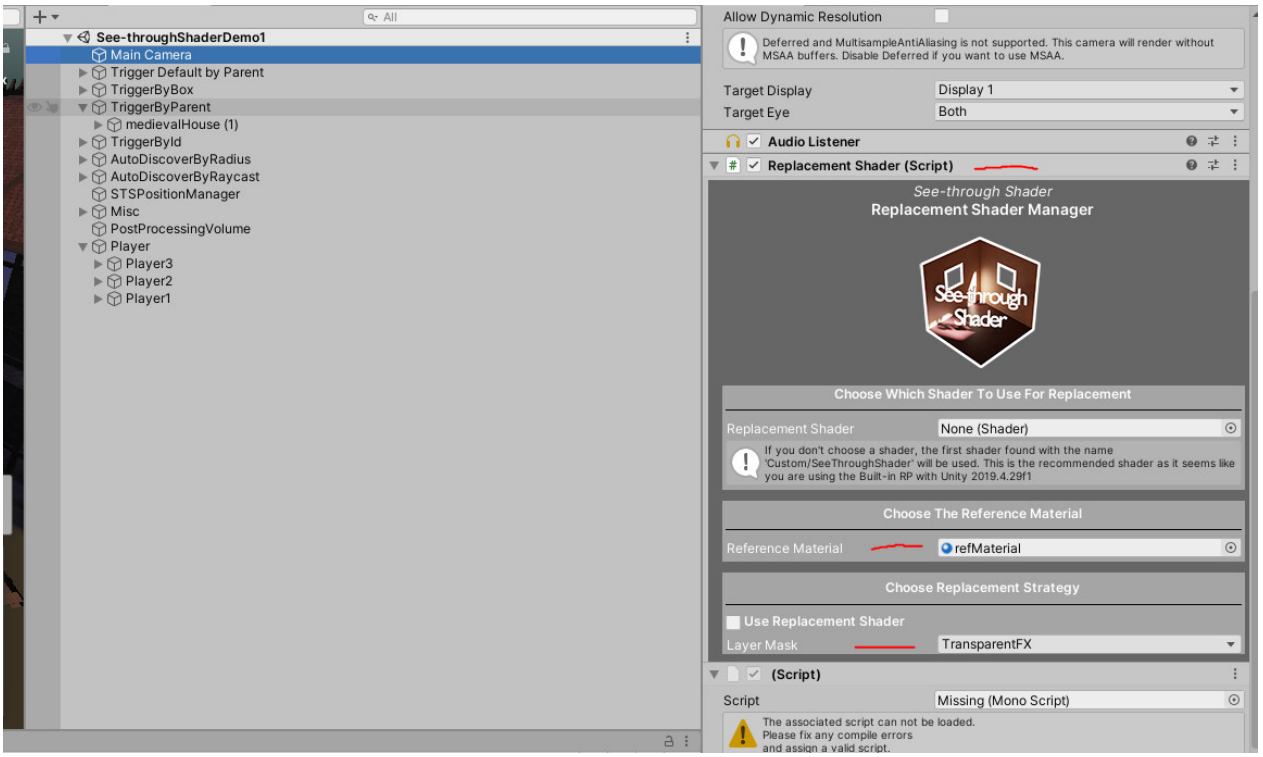
1.1

1.2

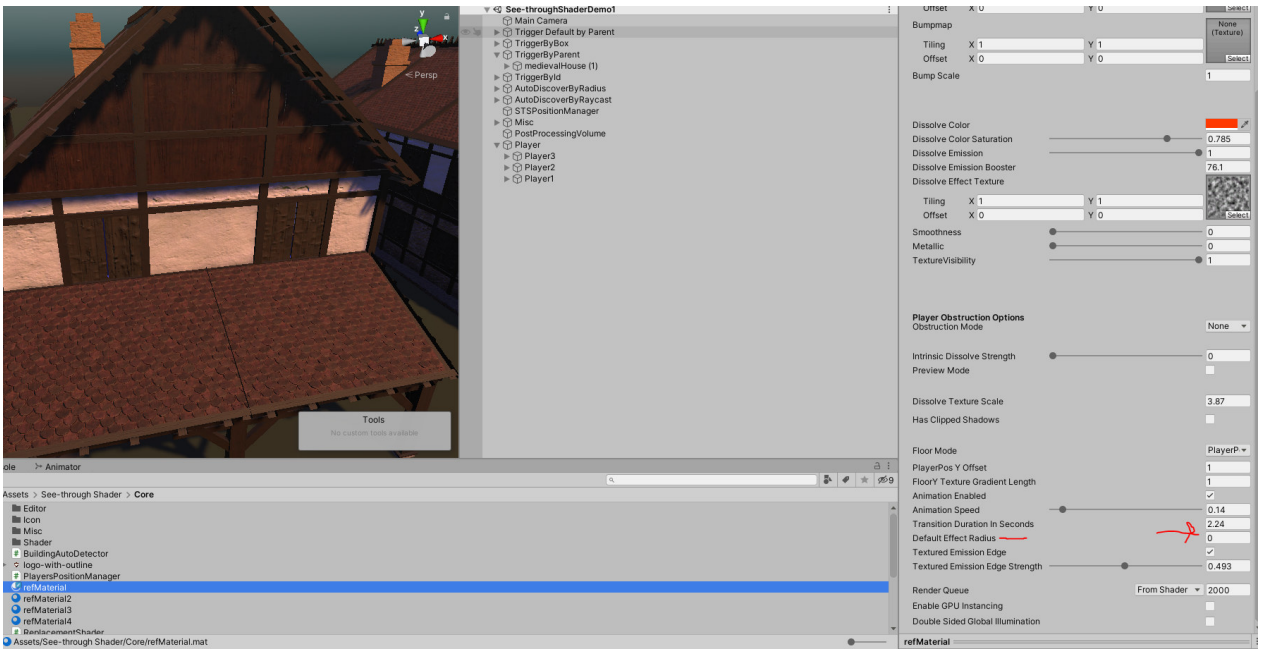


Auto mode Setup:
1.1 Add the ReplacementShader script to the primary Camera
1.2 Set the layer(s) you want the shader to operate on.
Multiple layers are possible.
1.3 If the radius is used, set the radius in the refMaterial or your template Material
1.4 If raycast mode is used, add the BuildingAutodetector to the player game object combinations of 2.3, and 2.4 is possible. Press Play

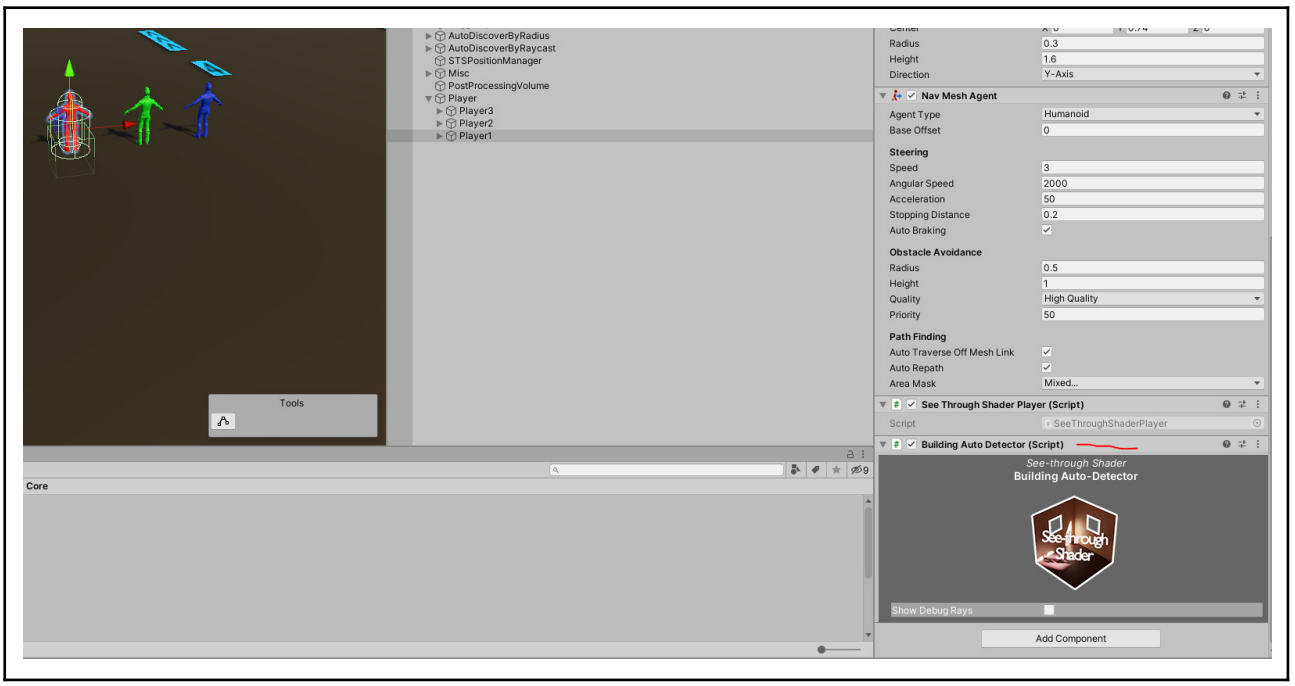
2.1 / 2.2


See-throughShaderDemo1
Main Camera
Trigger Default by Parent
TriggerByBox
TriggerByParent
medievalHouse (1)
TriggerById
AutoDiscoverByRadius
AutoDiscoverByRaycast
STSPositionManager
Misc
PostProcessingVolume
Player
Player3
Player2
Player1
Allow Dynamic Resolution
Deferred and MultisampleAntiAliasing is not supported. This camera will render without MSAA buffers. Disable Deferred if you want to use MSAA.
Target Display
Display 1
Target Eye
Both
Audio Listener
Replacement Shader (Script)
See-through Shader
Replacement Shader Manager
Choose Which Shader To Use For Replacement
Replacement Shader
None (Shader)
If you don't choose a shader, the first shader found with the name 'Custom/SeeThroughShader' will be used. This is the recommended shader as it seems like you are using the Built-in RP with Unity 2019.4.29f1
Choose The Reference Material
Reference Material
refMaterial
Choose Replacement Strategy
Use Replacement Shader
Layer Mask
TransparentFX
(Script)
Script
Missing (Mono Script)
The associated script can not be loaded.
Please fix any compile errors
and assign a valid script.

2.3


Persp
Tools
Bumpmap
Bump Scale
Dissolve Color
Dissolve Color Saturation
Dissolve Emission
Dissolve Emission Booster
Dissolve Effect Texture
Smoothness
Metallic
TextureVisibility
Player Obstruction Options
Obstruction Mode
Intrinsic Dissolve Strength
Preview Mode
Dissolve Texture Scale
Has Clipped Shadows
Floor Mode
PlayerPos Y Offset
FloorY Texture Gradient Length
Animation Enabled
Animation Speed
Transition Duration In Seconds
Default Effect Radius
Textured Emission Edge
Textured Emission Edge Strength
Render Queue
Enable GPU Instancing
Double Sided Global Illumination
Assets > See-through Shader > Core
Editor
Icon
Misc
Shader
BuildingAutoDetector
logo-with-outline
PlayersPositionManager
refMaterial
refMaterial2
refMaterial3
refMaterial4
Assets/See-through Shader/Core/refMaterial.mat
refMaterial

2.4



Trigger mode:

TriggerByParent script:

Trigger by Parent script has two trigger options:

- **one whole collider surround the building or**
- **dedicated enter exit trigger at a certain place the player passes**

Option 1 complete collider setup:

1.1 Select the parent Gameobject of the building and add a collider

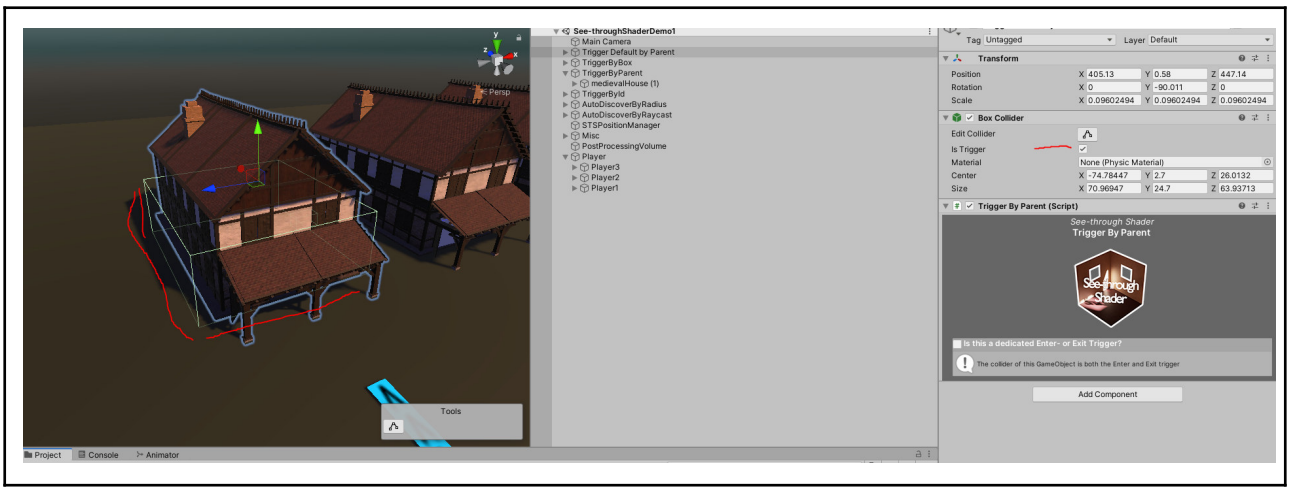
Add the TriggerByParent.cs script to the parent GameObject of the building

Drag the refMaterial file to the refMaterial field

Press play

TriggerByParent Option 1 whole enter/exit collider

1.1



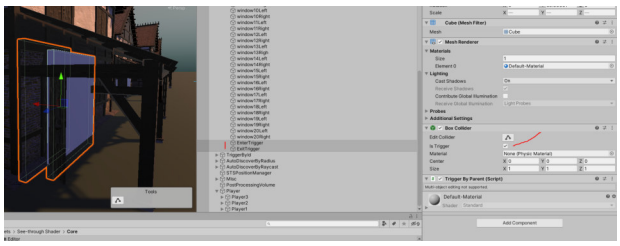
TriggerByParent Option 2 dedicated enter/exit triggers:

When the use case of your game is a building with a door or something like that where you want the player exactly to pass, you can create two cubes acting as the enter exit trigger.
The settings of both will be the same; just the IsEnterTrigger or IsExitTrigger will be different.

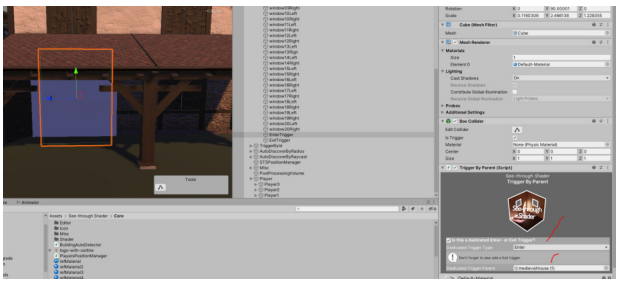
2.1 Create two cubes and place / scale them to fit your entrance, st the isTrigger=true option on the box collider
2.2 Select both in the hierarchy and add the TriggerByParent.cs script to both of them
2.3 If not using ReplacementShader script to globally add the Shader to the Elements the script SeeThroughShaderGroupReplacement can be used to add the Shader to just a Building by adding it to the parent.
After adding, drag the refMaterial or your template Material to the Material field and parent GameObject of the building and set the Layer that reflects the building elements.
2.4 Now, for each trigger, set the isEnterTrigger or IsExitTrigger flag
2.5 (OPTIONAL). Add the ShaderPropertySync script, drag the refMaterial or your template Material file onto the Material field, and check the “SyncContinuus” Box. This can be used in-game in the Editor to change the Shader settings in the refMaterial or your template Material and sync them to the Building.
This is mainly for test purposes and to get known to the Shader settings as for large buildings with many hundred elements; this will cost performance to sync the settings all the time.

TriggerByParent dedicated triggers setup

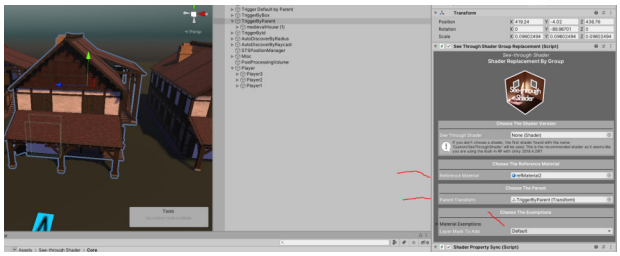
2.1



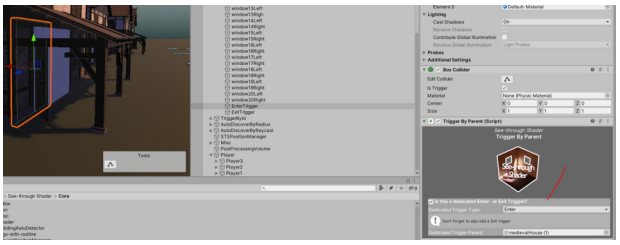
2.2



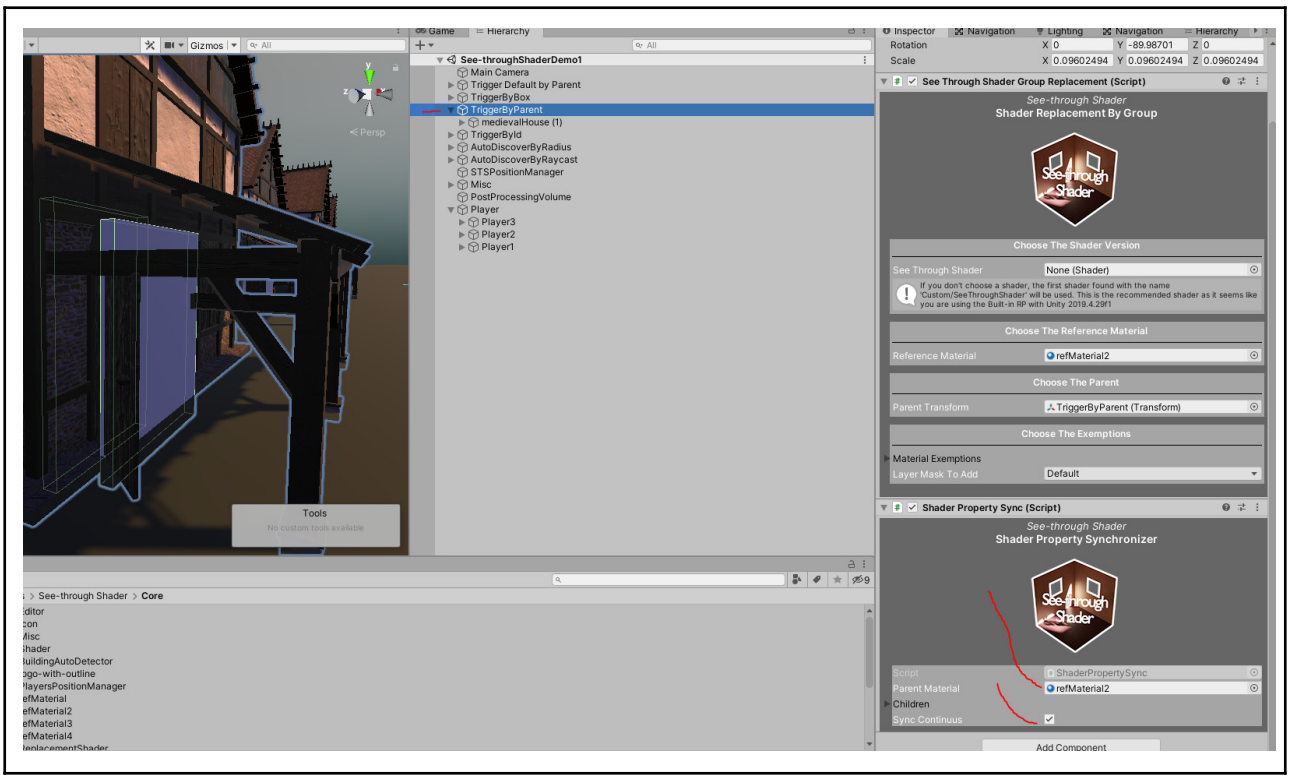
2.3



2.4



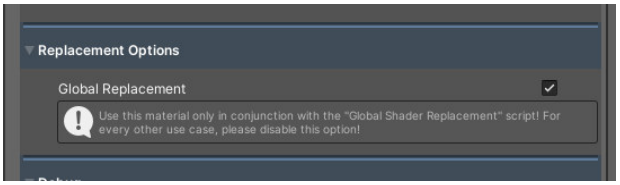
2.5


Gizmos
Persp
Tools
No custom tools available
Game
Hierarchy
See-throughShaderDemo1
Main Camera
Trigger Default by Parent
TriggerByBox
TriggerByParent
medievalHouse (1)
TriggerById
AutoDiscoverByRadius
AutoDiscoverByRaycast
STSPositionManager
Misc
PostProcessingVolume
Player
Player3
Player2
Player1
See-through Shader
Core
Inspector
Navigation
Lighting
Navigation
Hierarchy
Rotation
Scale
See Through Shader Group Replacement (Script)
See-through Shader
Shader Replacement By Group
Choose The Shader Version
See Through Shader
None (Shader)
If you don't choose a shader, the first shader found with the name 'Custom/SeeThroughShader' will be used. This is the recommended shader as it seems like you are using the Built-in RP with Unity 2019.4.29f1
Choose The Reference Material
Reference Material
refMaterial2
Choose The Parent
Parent Transform
TriggerByParent (Transform)
Choose The Exemptions
Material Exemptions
Layer Mask To Add
Default
Shader Property Sync (Script)
See-through Shader
Shader Property Synchronizer
Script
ShaderPropertySync
Parent Material
refMaterial2
Children
Sync Continuus
Add Component

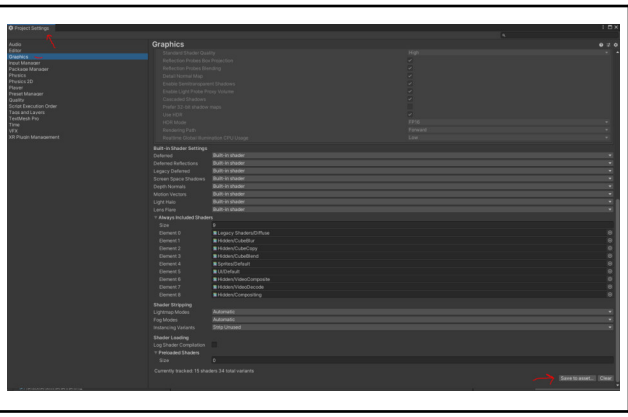
FAQ

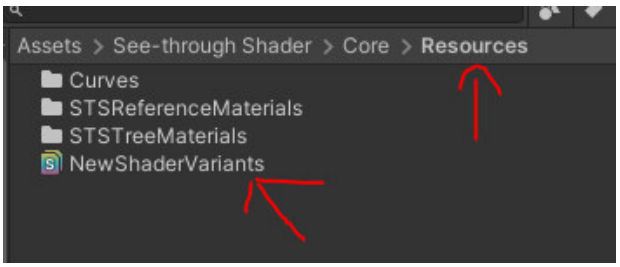
Q 00 - What steps are required for a build ?

1 Check the GlobalReplacement option in your refMaterials when using them together with the STS-Global Shader Replacement Script



2 Create a shader variants file and save it to the See-through Shader/Core/Resources folder



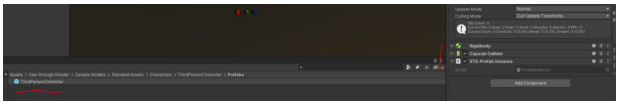


Q 01 - Can I use this asset with Prefabs that aren't in the scene at start ?

Yes this can be done by adding the PrefabInstance script to your Prefab.
Don't forget to add also a RigidBody and a Collider.
A **Demo scene for this setup** can be found here:
Assets/See-through Shader/Sample Scenes/**02_See-throughShaderDemo_ThirdPersonPlayer_PrefabPlayer.unity**

Picture FAQ Question 1

Picture Prefab Setup

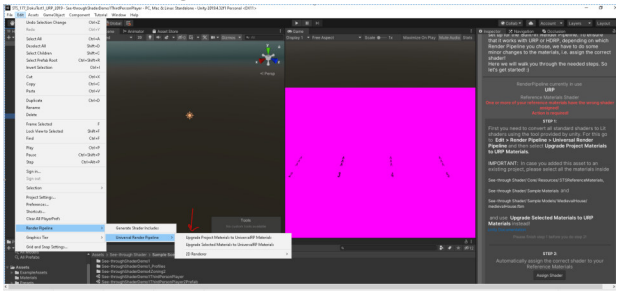


Q 02 - Why is everything pink in URP and HDRP when loading the scene?

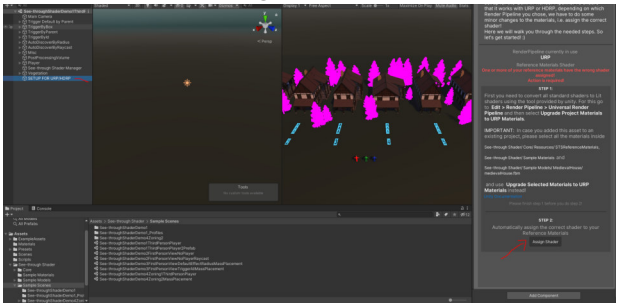
Please upgrade your Materials to the current Rendering Pipeline and execute the See-through Shader Setup as shown in the Pictures below.

Picture FAQ Question 2

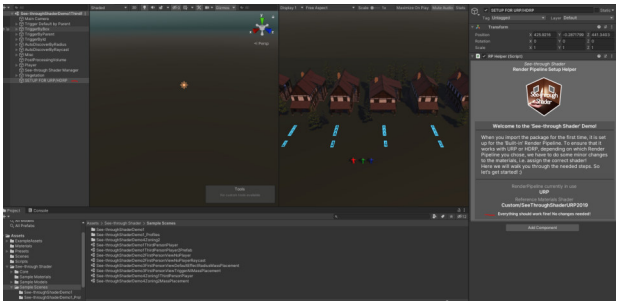
Picture URP Upgrade Materials



Picture URP See-through Shader Setup



Picture URP/HDRP See-through Shader Setup finished



Q 03 - I have a trigger setup and the DefaultEffect radius doesn't work.

When using triggers the default effect radius value has no effect because triggers work with the obstruction modes only.

Q 04 - When my player passes the trigger nothing happens.

Please make sure your Player has **a RigidBody, a Collider, the SeeThroughShaderPlayer** script and the player is added to the PositionManager Component.

Q 05 - Can I use this Asset with custom Shaders like Toon Shaders ?

At the moment we currently just support the Unity Standard / Lit Shaders and BiRP Unlit.

But you can always extend your existing Toon Shader with the See-through Shader functionality, as long as its a BetterShaders Stack or a ShaderGraph.
For more infos about extending your existing custom shaders see “Extend Your Existing Custom Shaders”.

Thank you for using See-through Shader. Any feedback is welcome.
Shadercrew
October 2022
support@shadercrew.com
https://www.shadercrew.com