
See-through
Shader

See-through Shader Documentation

Version 1.7.3
October 2021

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

Introduction:

What is the “See-through Shader” asset:
The “See-through Shader” asset lets you see your **playable characters** through any obstruction.
The core of the asset is the “See-through Shader” that comes with helper scripts for applying the shader.
This asset is primarily made for creating excellent gameplay within buildings, but of course can also be used in other use cases.

First and foremost the “See-through Shader” asset is a **shader*** and we supply you methods of applying and using it. We are always happy to improve on our product and implement new ways of making use of it. If you have any requests or wishes, please let us know!

*What is a shader and how does this affect the Unity Render Pipelines:

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

With version 1.7 we support ALL 3 RenderPipelines and every version of it. That means we support:
Built-in, URP 2019, URP 2020, HDRP 2019 and HDRP 2020.

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 completely focus on the end result without worrying about the inner workings.

How does it work:

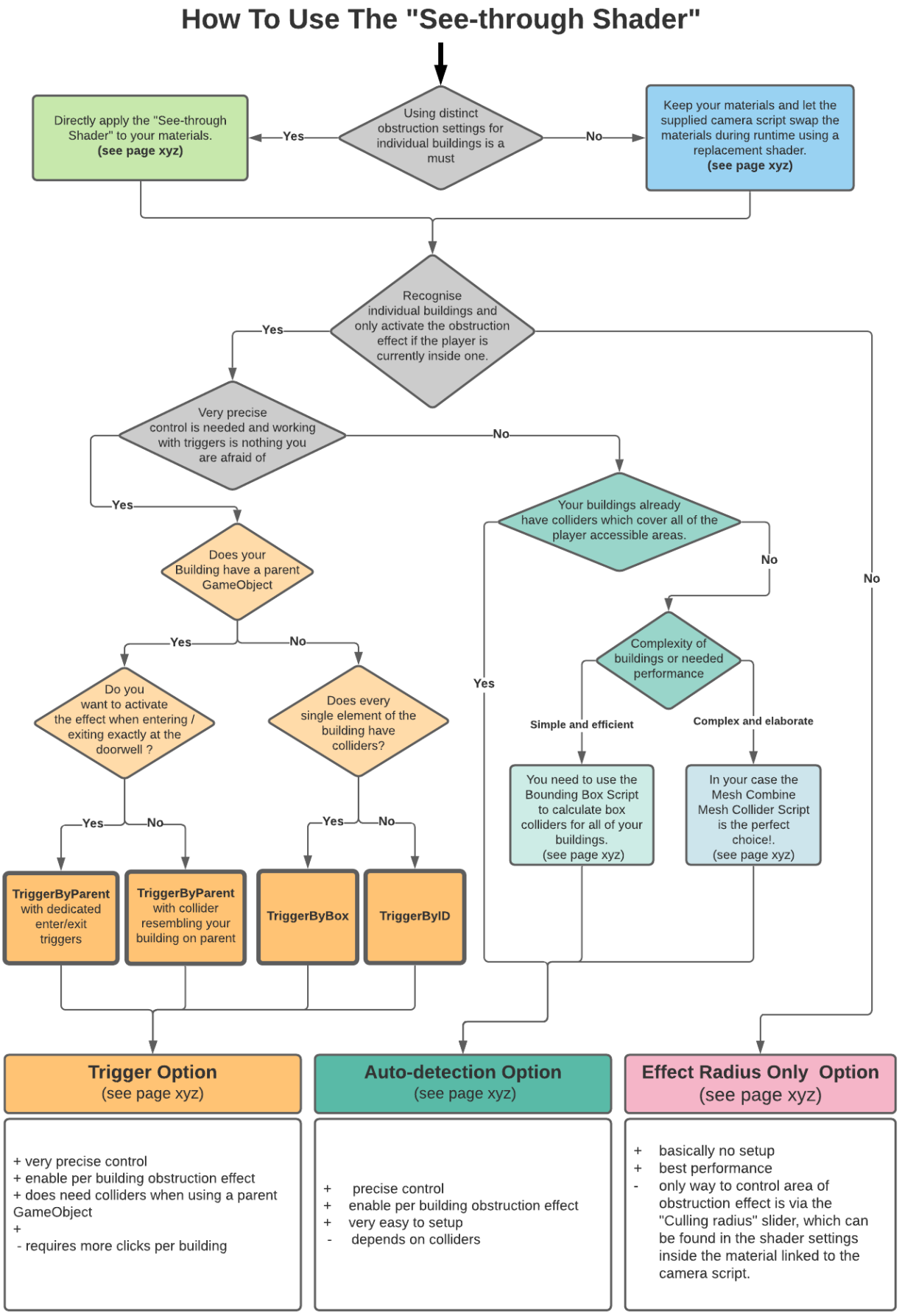
The “See-through Shader” works on the materials of the building's elements, it does not matter if it's one mesh or thousands of sub gameobjects.

There are two ways of applying the shader to your buildings. Either by setting it manually as the building’s material shader, or by using the “*ReplacementShader.cs*” script to automatically apply it to all GameObjects at runtime, which do not have the “See-through Shader” applied yet. (note: this can also be limited to specific layers).

By default the effect area of the “See-through Shader” is defined by an alterable effect radius, but it can also be triggered specifically on a per-building/object basis. There are two ways of restricting the area of effect to a building/object:
Either manual by setting triggers(see chapter on “*Trigger.cs*”) or automatically by using colliders and the raycast auto-detection algorithm(see chapter on “*BuildingAutoDetector.cs*”).

In the most simple setup, just a GameObject with the “See-through Shader” assigned to its material and the playable characters position manager script is added and the area of the effect of the shader ist controlled by setting the effect radius. For more complex buildings you can either supply a collider to your building and let the auto-detection script find out if the player has entered the building or not.
For the most precise control, which is highly recommended, you could also use dedicated enter- and exit-triggers (pair of cubes at the entrance) for the buildings so just when passing the triggers the effect is applied.
“See-through Shader” does not need colliders to work generally but for some dedicated triggers you need colliders on those objects so players can pass through and activate the enter and exit function of the shader or allow the auto detection algorithm to work.

To sum it up, first you have to assign the “*PlayersPositionManager.cs”,* then apply the shader to your building and then you can control the effect of the shader by choosing between three different methods of use.
(In previous releases(before v1.7) there was a script called PlayerPositionToGlobalVector.cs. Please replace that one with the *PlayersPositionManager.cs*, if you are upgrading)
.
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.



Picture:Decision schema on how to apply

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 method 1, or use our replacement script that automatically applies the “See-through Shader” to every material of all GameObjects that don’t have this shader applied to them yet, see method 2. You can also limit the reach of this script to certain layers, so only GameObject and their materials in specific layers get affected by the replacement script.

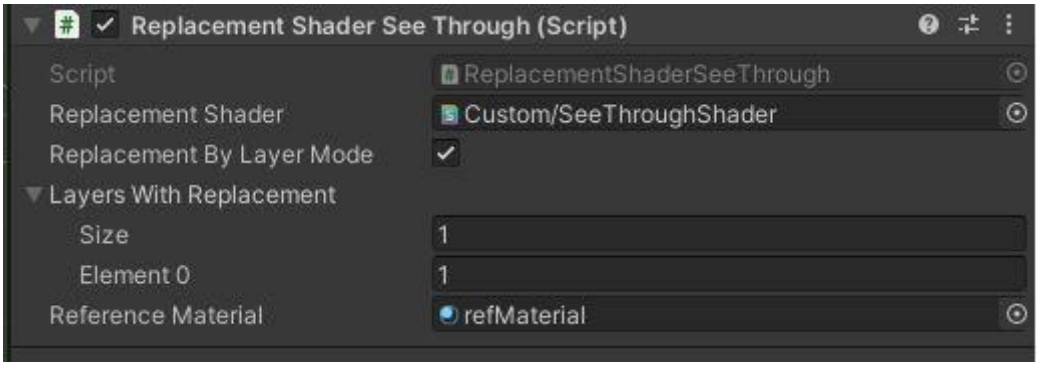
Method 1: **Manually apply the shader:**

To apply a 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 you can assign the shader to your material using the Shader drop-down menu in the inspector window, under Custom.
You could also just drag and drop the “*SeeThroughShader.shader*” onto the Shader drop-down menu.

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

Method 2: **Automatically apply the shader using the replacement script:**

All you have to do is attach the “*ReplacementShader.cs*” to your main camera and link a reference material with the “See-through Shader” assigned to it via “Reference Material”. Additionally by toggling the “Replacement by Layer Mode” checkbox, you can specify that only GameObjects in specific layers are affected by the replacement logic.



How to control the effect of the shader:

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

If you want the effect to be on a per-building/object basis, you have two choices:
Either you can go for the building **“Auto-Detection”** method, using the auto-detection script “*BuildingAutoDetector.cs*”, which is very easy to use and doesn’t require a lot of setup.
Or you choose the most precise way, using the “**Trigger**” 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 Overview of Application Methods:

Method 1: **Effect Radius Only:**

This method will restrict the shader affected area of effect within a sphere around the player. The size of the sphere is controlled by a variable radius 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, that means when the radius is large enough it is highly likely that the shader affected area is 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:**

Also easy to set up but contrary to the “Effect Radius Only” method supports per-building/object effect restrictions and the transition feature.
This solution requires you to attach the “*BuildingAutoDetector.cs*” script to your playable character and have colliders assigned 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: **Trigger :**

So 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 can work with simple and large complex buildings with many sub elements.

The most used and also 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 TriggerByParent script does not need colliders for the elements at all , the effect is triggered by one collider surrounding the whole building or by two dedicated enter exit cubes with colliders placed at the entry.
The other two triggers are for the use cases where there is no parent GameObject but elements have colliders so a surrounding box can capture inside colliders (TriggerByBox).
When there is no parent GameObject and the elements have no colliders you can assign a ID (house123 etc.) to every element 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 corresponding house.

General trigger description can be found prior to this section , here is some more detailed description of the trigger GameObject.

“Trigger by Parent” (building elements do not need colliders)
For simple AND complex buildings with up to many elements , this is the best use case as it does need colliders and you often have a parent gameobject per building.

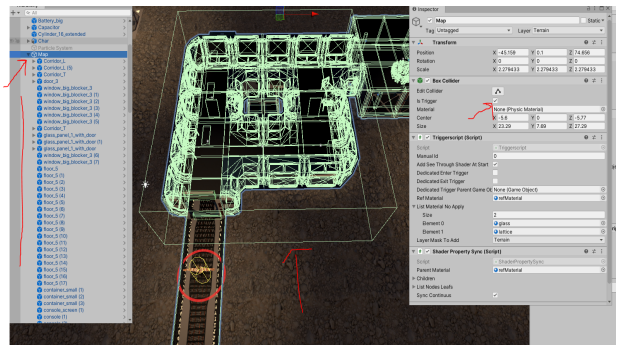
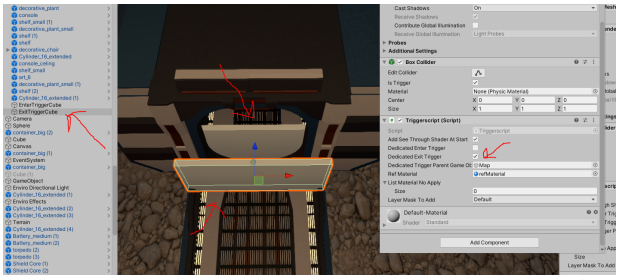
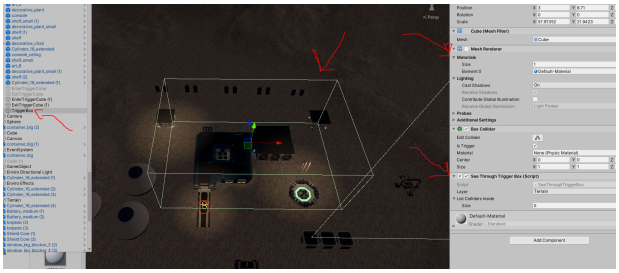
“TriggerByBox”
Works with a collider Box surrounding the building and applying the See-through Shader to collider enabled objects on a specific layer mask you choose. This is useful where you have a complex building with colliders but it has no parent gameobject.

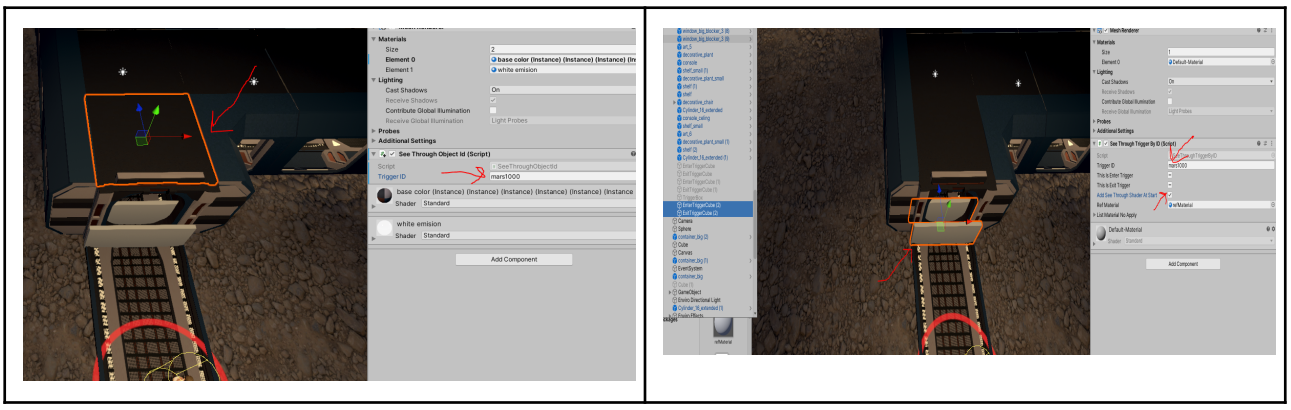
“Trigger by ID”
Is for use cases where you don't have colliders and also no parent , so you set an ID to every element of the building and when passing the trigger these elements will trigger the See-through shader.

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 own custom trigger, if you have any questions for your specific use case please write us an email , we will be happy to support you.

Notice:
Mostly it is best practice to drag a selection box in unity editor over the elements and 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 parent object 	**TriggerByParent (TriggerByParent.cs)** Dedicated Enter and Exit trigger objects Most accurate 
TriggerByBox (surrounding box) Surrounding box 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 setup on element	**TriggerById** No colliders and no parent object available enter / exit trigger



Structure and description of Asset Folders:

- **core**
  Core scripts and Shader file to operate the Asset.

  Shader:
  SeeThroughShader.shader

  Triggers:
  TriggerByParent.cs > Main trigger script , works on the buildings parent gameobject
  SeeThroughTriggerByBox.cs > Trigger that captures all elements within a surrounding box collider
  SeeThroughTriggerById.cs > Trigger that finds all elements of a building by pre set ID

- **core/Misc**
  Sync and helper scripts

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

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

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

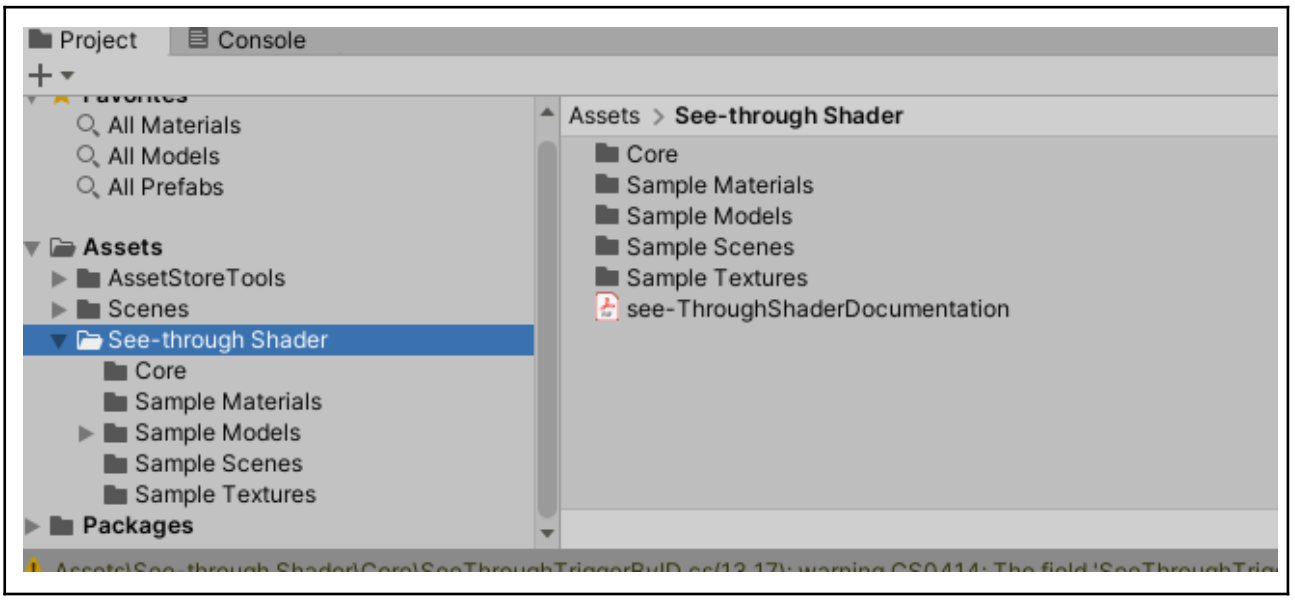
- **Sample Material**

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 /core folder you will find a Material called refMaterial , you can use this when configuring triggers for start or at runtime with the sync script to set the shader settings and they get applied to the building.

Don't forget to also supply a dissolve texture as it will make the overall effect much better.
You can also make your own template Materials by clicking right click with the mouse and just create a unity material , give it a name and under the shader setting select the See-through shader under Custom/See-through Shader.
This Manual will provide a step by step guide.

Scripts description:

Documentation of the c# scripts.

Scriptname: TriggerByParent.cs

Class name: Trigger

Description:
“Trigger by Parent” script where you have a parent gameobject and either a surround box collider as trigger or two dedicated triggers (enter / exit) at a specific place where the player passes.
For the building elements no colliders are needed.
The shader can be pre-applied or with a boolean checkbox at start with a reference material supplied and a layermask to which layers the shader should be applied.

Script Variables in unity Inspector:

Name	Type	Description
DedicatedEnterTrigger	Bool (checkbox)	If this trigger object is a dedicated enter trigger
dedicatedExitTrigger	Bool (checkbox)	If this trigger object is a dedicated exit trigger.
dedicatedTriggerParentGameObject	GameObject	Drag the parent GameObject of the building on this field for the dedicated triggers . If you apply this trigger script on a parent GameObject of a building and add a surrounding Box collider (tisTigger = true) then you don't need to supply this Field. This trigger script object will be taken as parent .

RefMaterial	Material	Drag here the Material called refMaterial from the /core Folder or your own created template Materials to get the shader settings applied from this Material file.
listMaterialNoApply	List<Material> (Material list in Editor)	Add an integer for the amount of Materials you want to drag in and press Enter . Then drag here the Materials you want to exempt from See-through Shader like Glas etc.
LayerMaskToAdd	LayerMask (Layers to select in the unity editor)	Select here the unity layers where See-through Shader should be applied when adding it at start. Multi selection is possible.

Scriptname: PlayersPositionManager.cs

Class name: PlayersPositionManager

Description:
This is the script that MUST be applied to some empty or wherever you want. You have to add some 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 in that list. If you require this asset to support more, please let us know!

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

Scriptname: BuildingAutoDetector.cs

Class name: BuildingAutoDetector

Description:
This script determines automatically if the player is within a building or not .
This will work for most buildings and for some rare building forms the usual triggers can be used where a parent GameObject is supplied.
This script is mostly used in a click and go environment where the buildings are not too 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 see through” script on the camera and press play .
A combination of this script and the „standard“ triggers is possible so use the AutoDetection Scripts and the more exact Trigger scripts as you need it.
This script works with an algorithm and raycasts to determine where the player is relative to the buildings around.
Just 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 is provided.

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

Scriptname: HideIfDisableDrawer.cs
Class name: HideIfDisableDrawer

Description:
When working with the See-through Shader on Materials or on the refMaterial this helper class is used to make only active parameters visible in the unity inspector.
This script works automatically from the /core folder , no interaction is required .
Please do not remove this file.

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

Scriptname: LampTriggerScript.cs
Class name: LampTriggerScript

Description:
Optional trigger script to enable or disable a light when the player passes.

Script Variables in unity Inspector:

Name	Type	Description
LightBulb	Light	Add this script to GameObject with a Collider set to „trigger“ acting as trigger. When the player passes this trigger (collider) then the light bulb will be enabled or disabled.

Scriptname: MeshCombine.cs

Class name: MeshCombine

Description:
Helper Script that makes 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 own 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 just work with simple to middle complex buildings but with thousands of meshes of complex interior the combined mesh will be mostly like the original meshes.
Please use this script for cases described above , we will also provide here videos and documentation with example buildings.

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

Scriptname: ReplacementShader.cs

Class name: ReplacementShader

Description:
Script for the Camera that adds the See-through Shader to all GameObject which match the unity Layer supplied in the „List of Layer Numbers“ Variable.
It has a Radius Variable that will cull the building mesh within this radius if its > 0 and the element has no other trigger.

In the simplest setup you add the Player Position to Global Vector script to the Player and this script to the camera , then set the Layers you want this to apply , effect radius and press play.
Of course a combination is possible you can still have other possible trigger options in the asset for your more complex buildings and use this script together for the simpler buildings where you want no additional modification or overhead of work per / object .

Script Variables in unity Inspector:

Name	Type	Description
ReplaceMentshader	Shader (can be found in / core folder)	Optional: This script will find the See-throughShader automatically but for safety you can also drag it in here manually in the unprobably case that at start the shader is not loaded already. In

		most cases you can leave this field empty .
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
_CullingRadius	Float (5.0 , 10.0 for example)	The Radius in meters around the player where the mesh will be culled when the player is within reach.

Scriptname: SeeThroughShaderController.cs

Class name: SeeThroughShaderController

Description:
Important core C# Script for the See-through Shader to control the entry and exit sequence.
All trigger scripts work with this controller automatically so no need for adding this to a GameObject.
It will take a List<Material> from own made triggers , if you are interested in making your own triggers have a look at the code in this class.

Script Variables in unity Inspector:
This Script has no public variables but public Methods and two constructors.

Scriptname: SeeThroughShaderExemption.cs

Class name: SeeThroughShaderExemption

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

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

Scriptname: PlayerDemoController.cs

Class name: PlayerDemoController

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 using unity Navmesh.
Please use your own player controller for the purpose of your use case , for our Videos we used Assets for Moving , Navigation and Pathfinding.
We would be happy to share the Links and Help how to implement them.

Script Variables in unity Inspector:

Name	Type	Description
Speed	Float	Player speed when moving

Scriptname: ShaderPropertySync.cs

Class name: ShaderPropertySync

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 whether at start or at runtime in a continuous mode.
You can use this script or build your own upon this to apply changes from a reference Material to a building or object with See-through Shader active.
Mostly you will load your initial settings with the other scripts at start and sometimes use in game changes and sync them to the child elements of the parent where this script is applied.

Script Variables in unity Inspector:

Name	Type	Description
parentMaterial	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 start will be synced to. Mostly you will use the sync continues option to sync to all gameobject beneath without specifying them forehand. But in some cases the game designer wants just a handful of materials be pre added in the editor This Option will of course work with the continuous sync option
Sync Continuus	Bool (checkbox)	Sync the See-through Shader settings from the parentMaterial File to all the children beneath this GameObject which have the See-through Shader

Scriptname: TriggerObjectId.cs

Class name: TriggerObjectId

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 own player controller for the purpose of your use case , for our Videos we used Assets for Moving , Navigation and Pathfinding.
We would be happy to share the Links and Help how to implement them.

Script Variables in unity Inspector:

Name	Type	Description
triggerID	string	Assign a unique id to any elements within the same

		building so that the trigger by id script can find them.

Scriptname: TriggerByID.cs

Class name: TriggerByID

Description:

Use this script to set up a 3D Object in unity with a collider (isTrigger = true) acting as trigger.
When the player passes this trigger then 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 parent can not be used.
This trigger comes as a pair of enter and exit trigger, 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 enter 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 so that the trigger by id script can find them.
thisIsEnterTrigger	bool (checkbox)	Set to true for the enter trigger
thisIsExitTrigger	bool (checkbox)	Set to true for the exit trigger
addSeeThroughShaderAtStart	bool (checkbox)	Add the See-through Shader to the building elements material at start
refMaterial	Material	Drag the refMaterial Material from the /core folder or your own template Material wtith the See-through Shader assigned to sync the shader settings at start to the elements of the building

listMaterialNoApply	List<Material> (List of Material in unity editor)	Populate this list with Materials to exempt from the effect like Glas or any Material. This is mostly useful for Elements with multiple Materials where you want to exempt one from See-through Shader.

Scriptname: TriggerBox.cs

Class name: TriggerBox

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 where you can choose which layers should be processed.

Script Variables in unity Inspector:

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

---

Scriptname: TriggerByBox.cs

Class name: TriggerByBox

Description:

This script will use the TriggerBox script to capture the buildings elements within the collider of the surrounding box.
This script is added to two cubes with colliders set to isTrigger = true and set at a place where the player passes when entering and exiting.

When passing the triggers with this script , it will apply the effekt of the buildings elements.
Drag the refMaterial Material file or your own Material templates with the See-through Shader selected to sync these settings at game start.
In the demo scene you will a example for a setup of this script.

Script Variables in unity Inspector:

Name	Type	Description
triggerBox	SeeThroughTriggerBox (the script above)	Drag the surrounding box gameobject onto this field.
thisIsEnterTrigger	bool (checkbox)	Set to true for the gameobject (cube) acting as enter trigger
thisIsExitTrigger	bool (checkbox)	Set to true for the gameobject (cube) acting as exit trigger
refMaterial	Material	Drag the refMaterial file from /core or your own Material with the See-through Shader selected onto this field to sync the shader settings from this file to all elements
listMaterialNoApply	List<Material> (List Material in unity editor)	Optional: Populate this list with materials to exempt from See-through Shader. This is mostly used for elements with multiple materials where some should not have the effect like glas.

Shader: SeeThroughShader.shader

See-through Shader Settings

Introduction:

As said earlier the core of the See-through Shader Asset is the See-through Shader which dedicated features can be controlled by the Materials settings.
The various scripts that can come with this Asset help set up the Shader on each element of the building and sync the settings from a reference material by start and or by runtime.
If you have buildings with just a few elements of course the shader settings can also be done manually by clicking the buildings elements and changing the settings on the material.
If you want different features for a set of buildings than you need to create several material templates and apply them on the specific buildings.

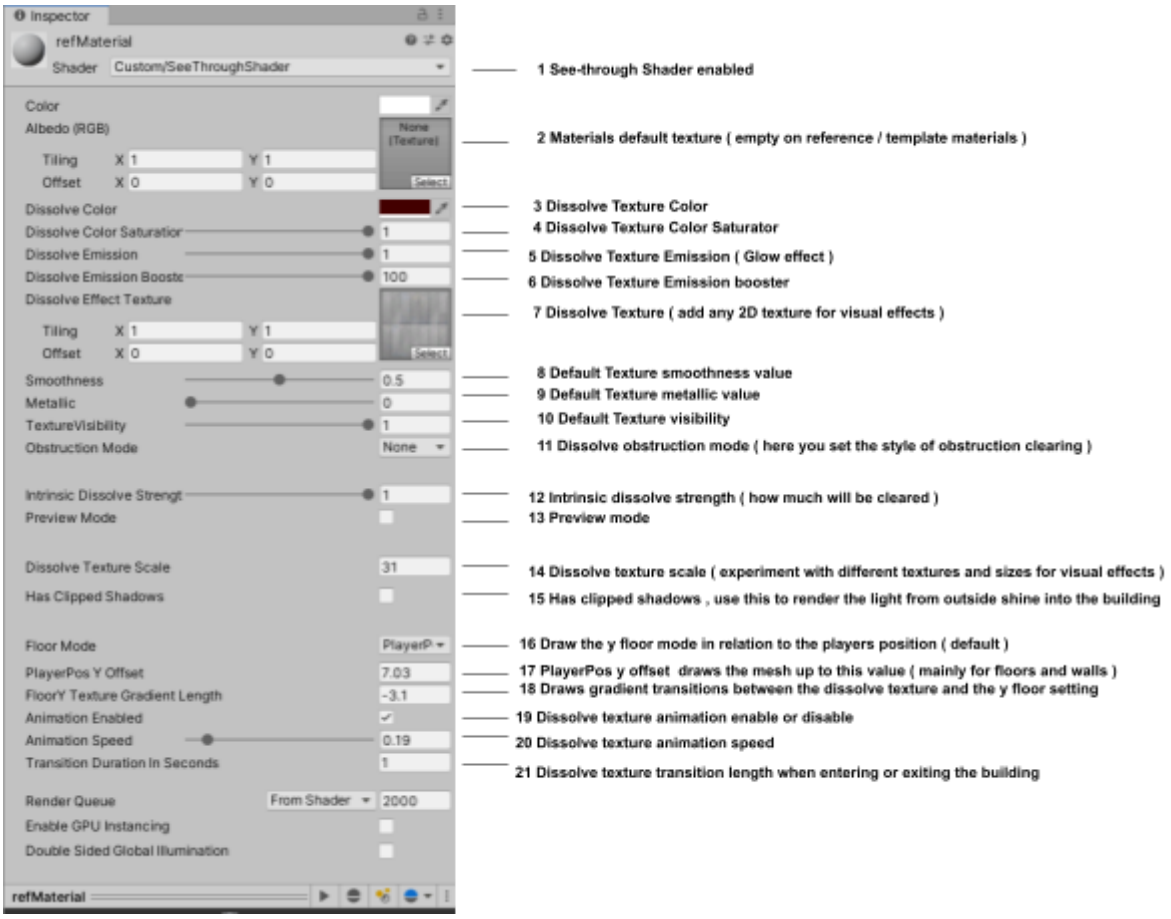
See-through Shader settings come in three main Areas:

- obstruction mode
  This is the main function of the shader , the settings how the obstructed area should be cleared can be found here , from shapes to totally clear (none mode)

- texture settings and animation
  Also very important feature of the asset is the texture feature where you can add any texture to the dissolved area and fine grain control the behaviour up to animation

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

Overview Picture:

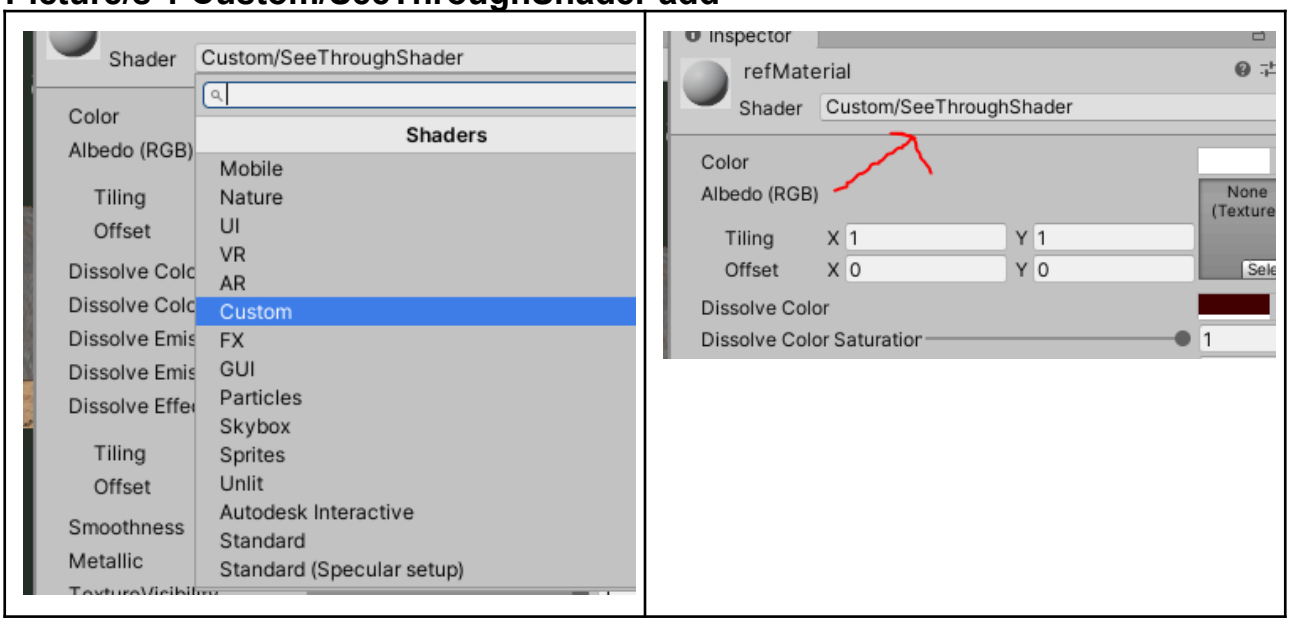


See-through Shader Description of Features and Settings:

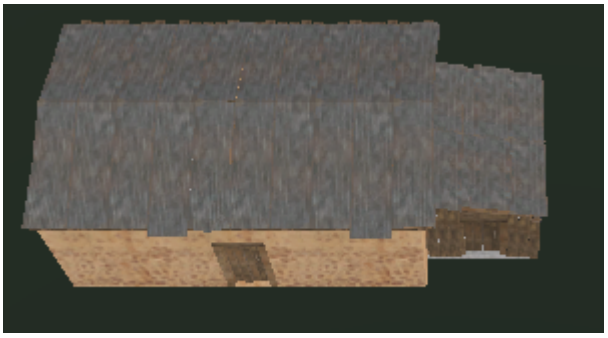
1 Custom/SeeThroughShader:

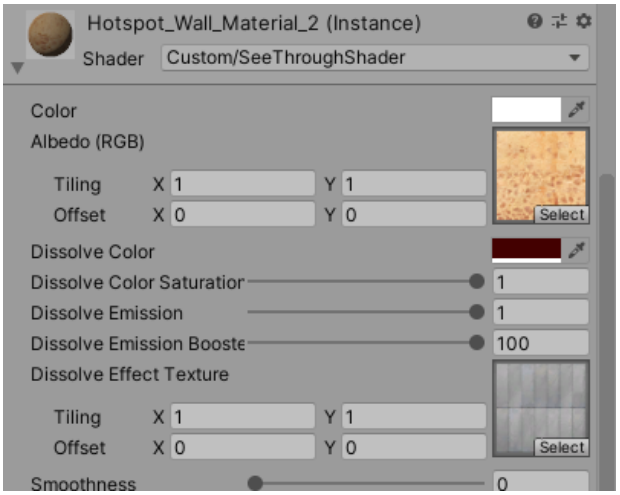
If this is selected then the See-through Shader is active for this material .
This is important when creating your own template materials , after creating them with right mouse and creating material, assign the Custom/SeeThroughShader to use them with this asset.

Picture/s 1 Custom/SeeThroughShader add



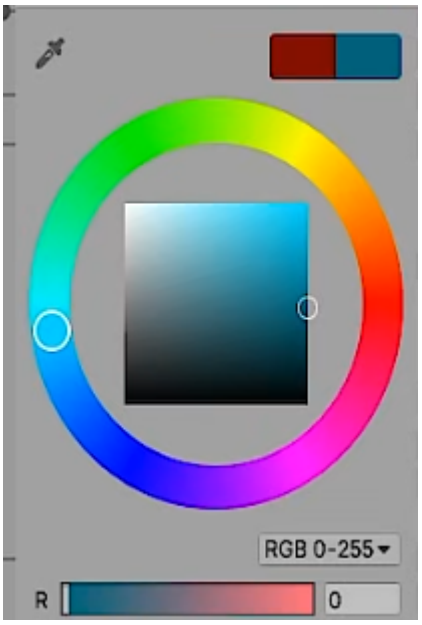
2 Albedo (RGB):
This is the default or main texture of the building element , in reference or template material this will be empty as it will get filled with the texture of the object when applied.
Picture/s 2 Default albedo texture of material

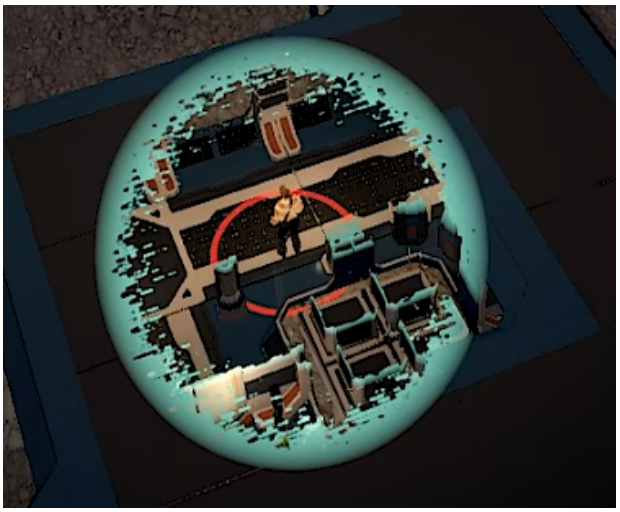


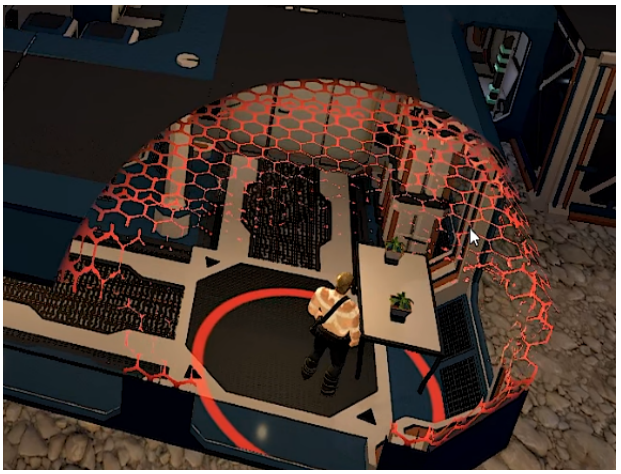


3 Dissolve texture color:
Set the color of the region that is affected by dissolve texture algorithm

Picture/s 3 Dissolve texture color



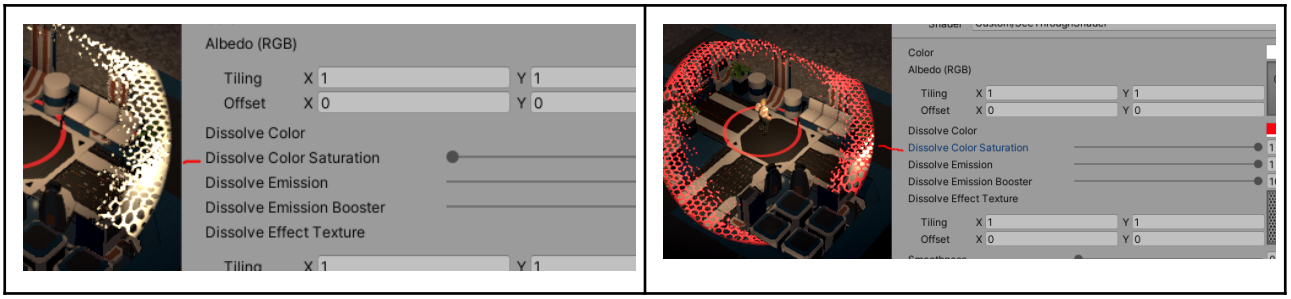




---

4 Dissolve texture color saturator:
Set the color brightness of the dissolve texture according to your setup or use case

Picture/s 4 Dissolve texture color

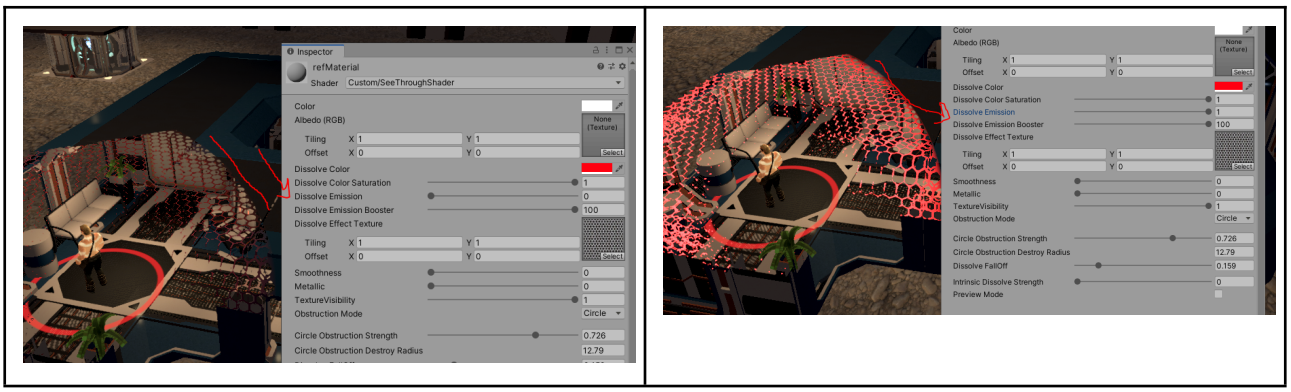


---

5 Dissolve texture emission:

Set the dissolve texture color emission to have glow effects

Picture/s 5 Dissolve texture color

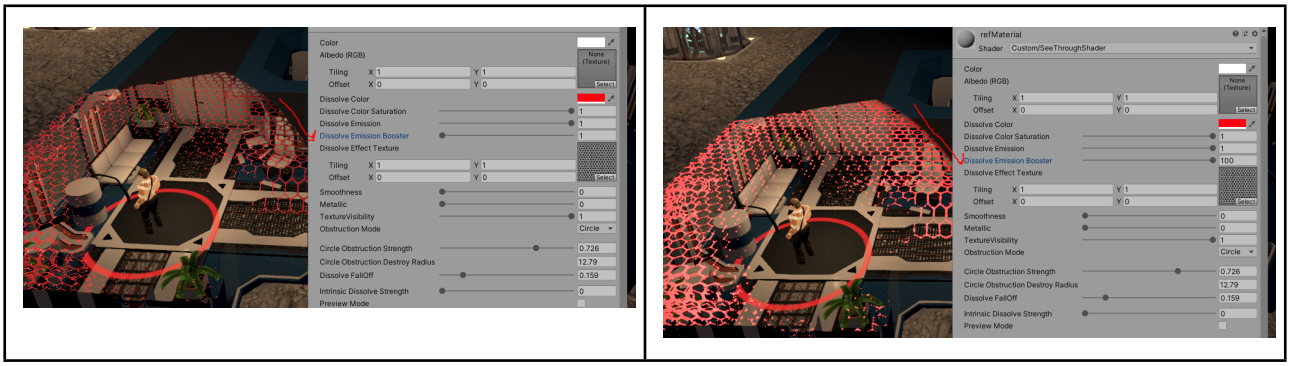


---

6 Dissolve texture emission booster:

Set the dissolve texture booster for some extra glow emission

Picture/s 6 Dissolve texture emission booster

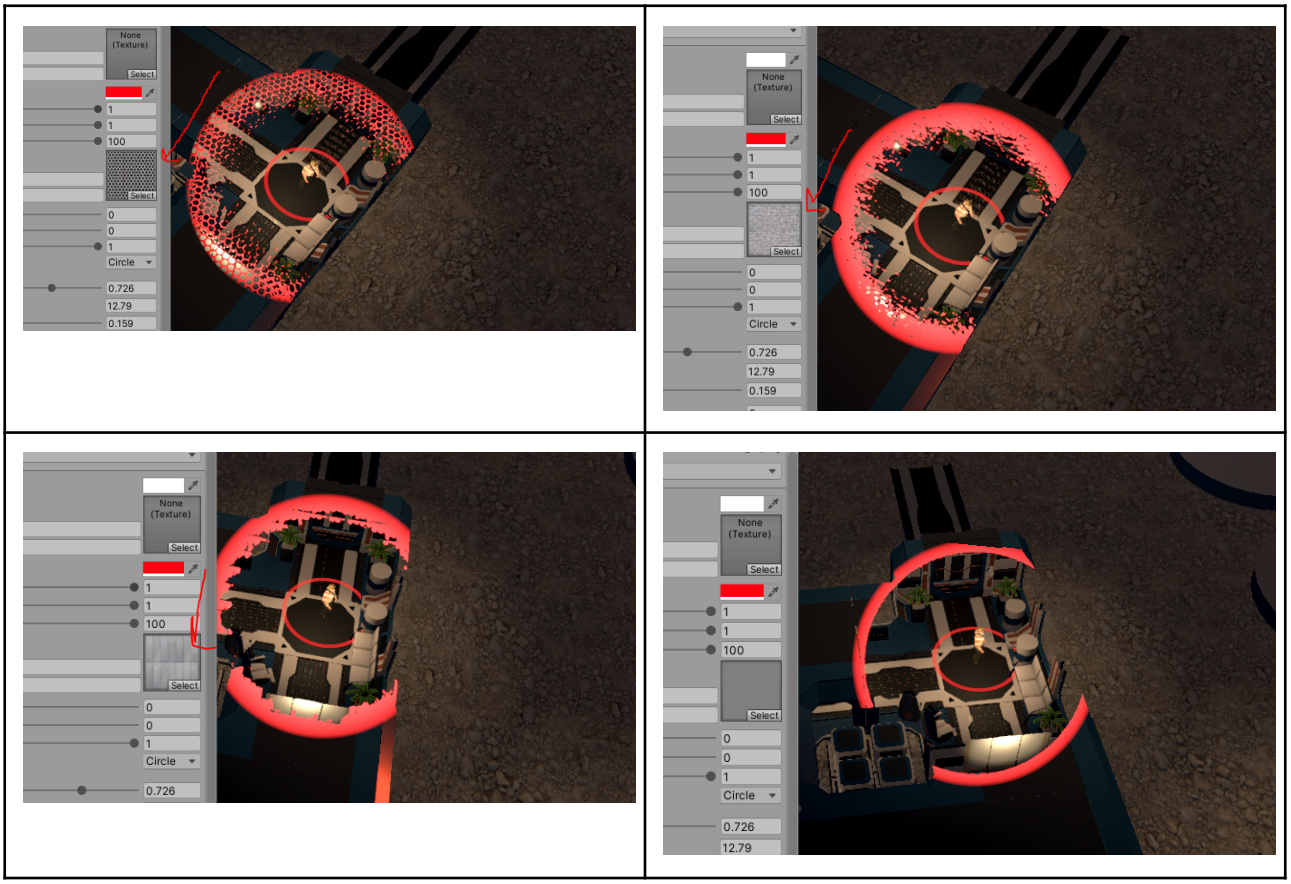


7 Dissolve texture:

Any 2D dissolve texture that is used to show visual effects while transitioning and on the dissolve borders.

Feel free to experiment with different textures and texture scales as they show surprising visual effects.
We ship three textures within this asset for you to use.

Picture/s 7 Dissolve texture:



8 Smoothness:
Default material smoothness value

---

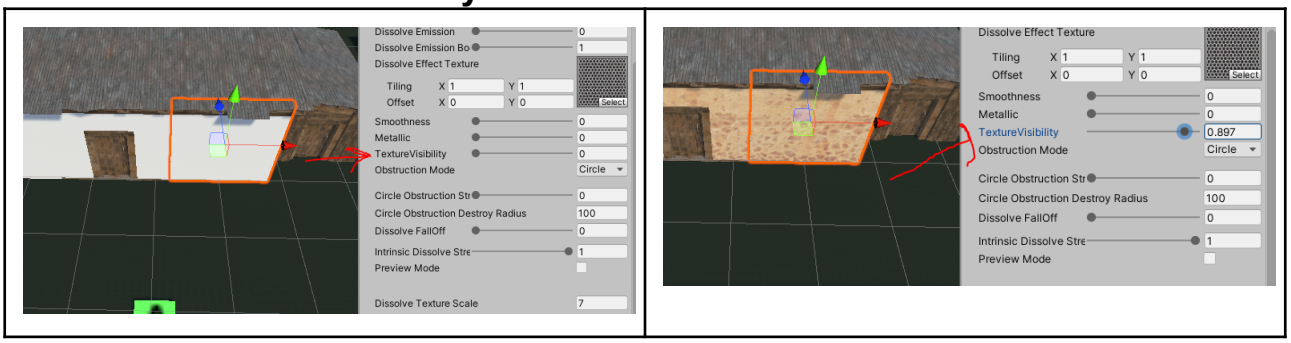
9 Metallic:
Default (this) material metallic value

---

10 Texture visibility:

Texture visibility value controls how much of the default material texture is visible

Picture/s 10 Texture visibility:



---

11 Dissolve obstruction mode:
This is the most important and core feature of the See-through Shader Asset as this controls the mode of clearing obstruction .
The level of clearance is seamlessly adjustable so semi visibility is possible as well as showing the silhouette of the building with the texture for special effects or buildings which must be unlocked by the player.

Following obstruction modes are available:

- **Angle:**
  Clear obstruction by angle between 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 to not draw.
- **ConeOnly:**
  Removes parts of the mesh that are 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
- **CylinderOnly:**
  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

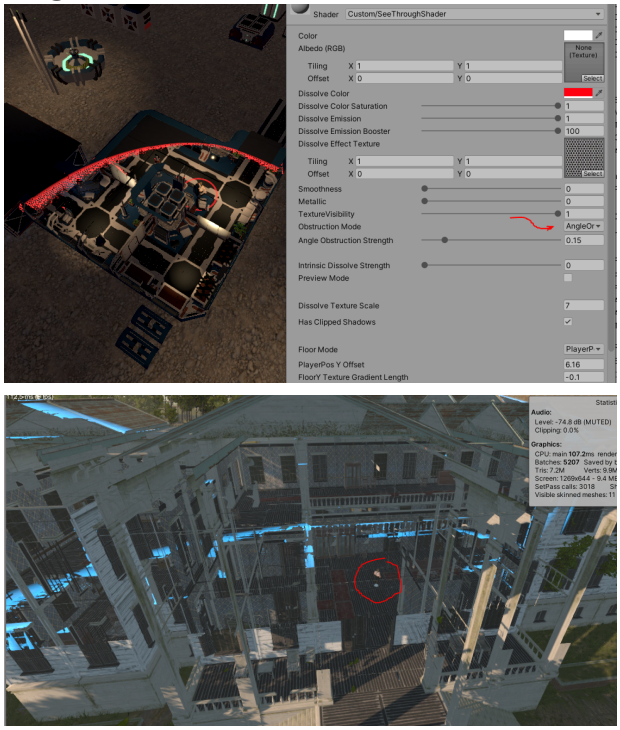
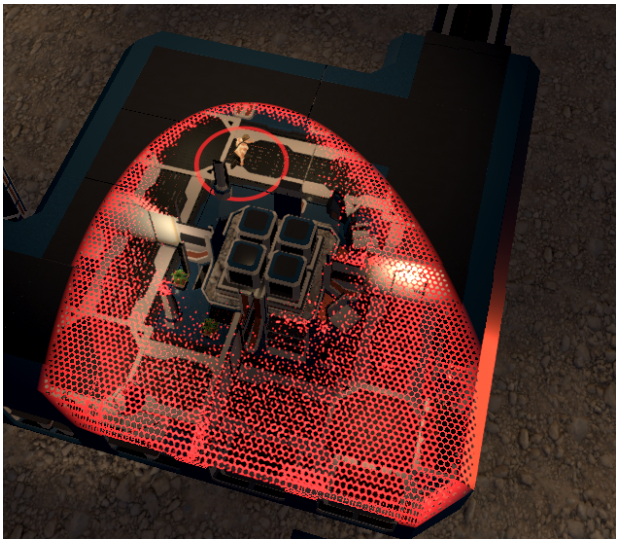
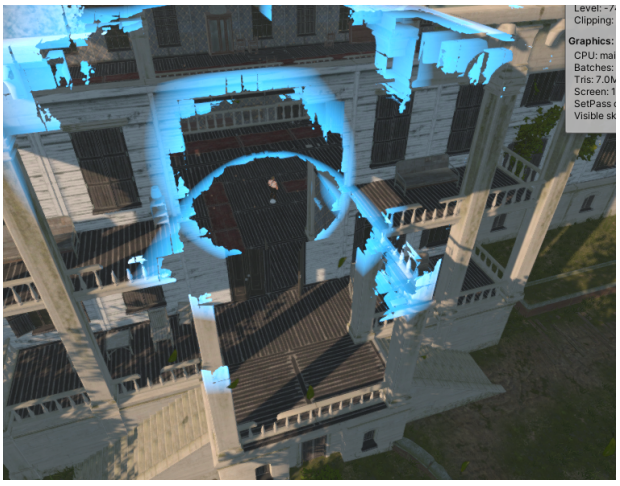
- **Circle:**
  Clear obstruction with a focused circle around the player , this is the recommended mode as it works well with simple and complex buildings

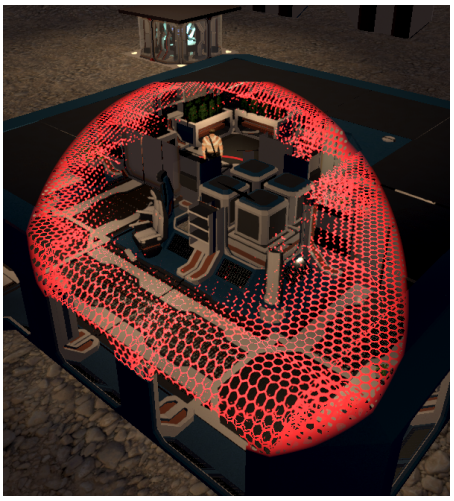
- **AngleAndCone:**
  Combine Angle and Cone
- **AndIgeAndCylinder:**

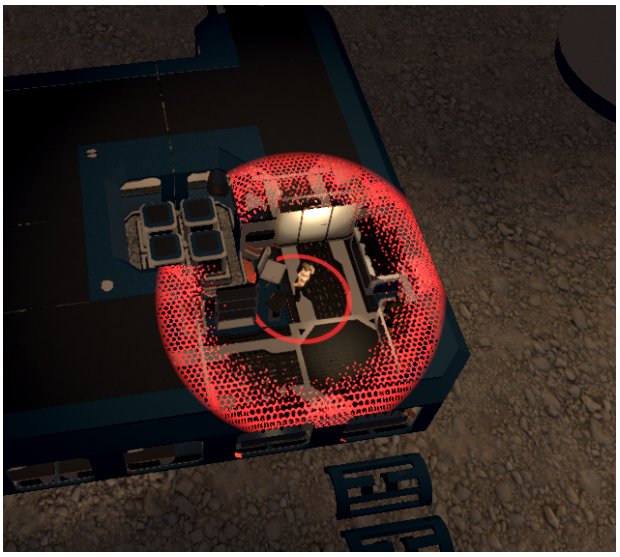
Combine Angle and Cylinder

- **none mode**
  Do not use a shape and clear everything from “PlayerPos y offset” parameter up, so also the whole building can be affected and the silhouette is drawn in with the dissolve shape.
  This is useful for visual effects and to completely remove any obstruction

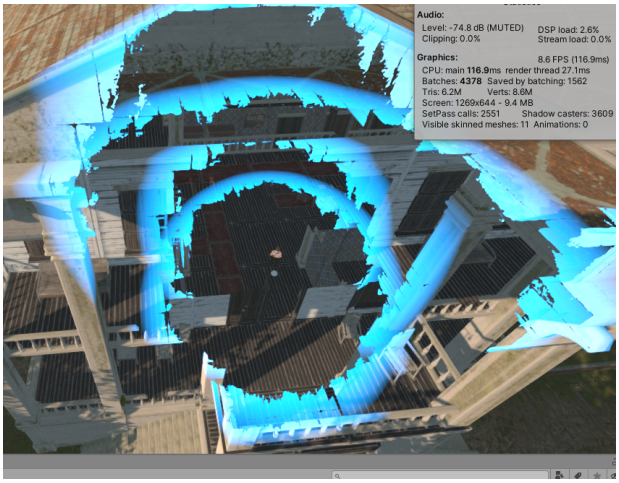
Picture/s 11 Dissolve obstruction mode:

Angle	**Cone**
	 
Cylinder	**Circle**

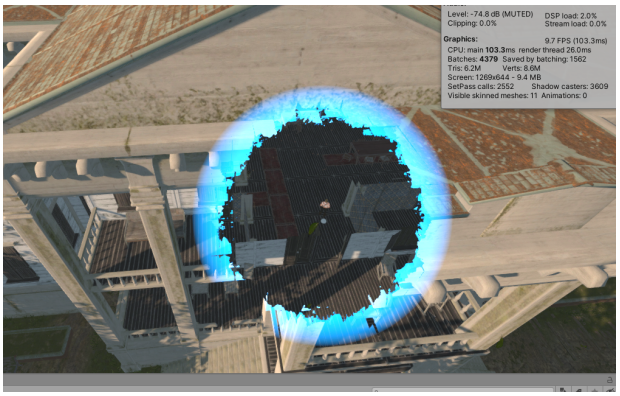




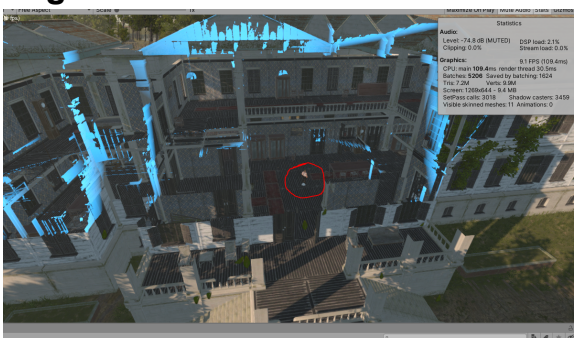
Cylinder



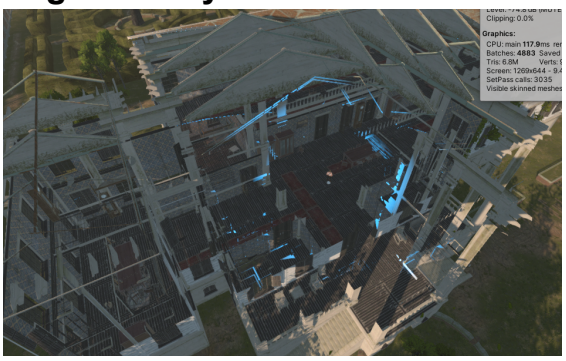
Circle



Angle and Cone

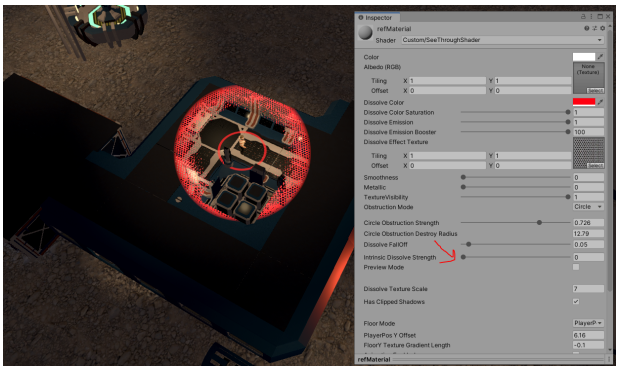
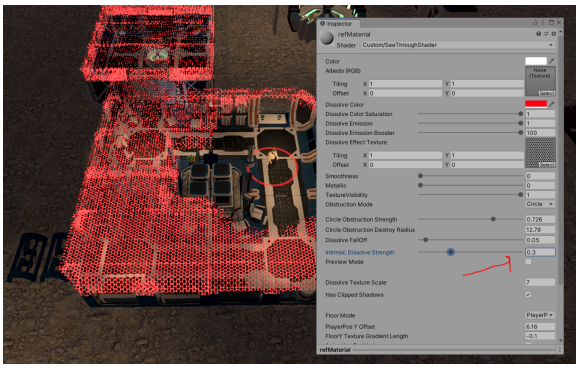


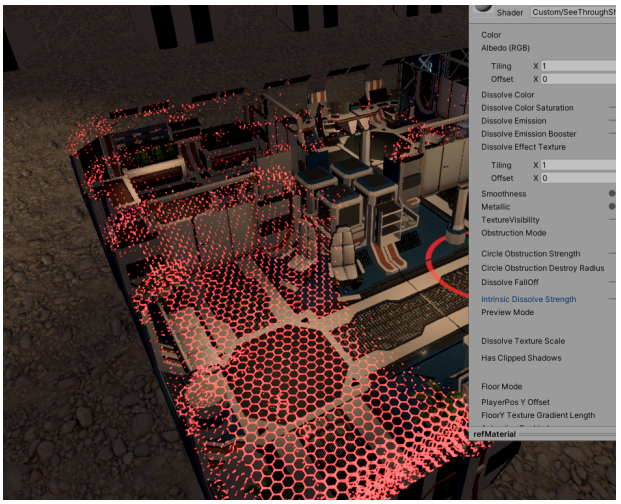
Angle and Cylinder

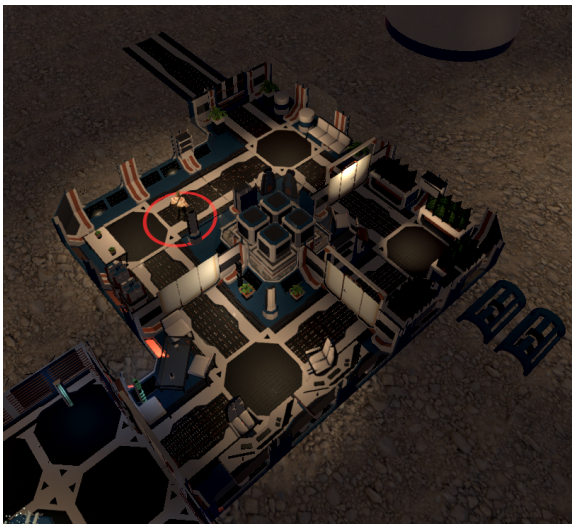


12 Intrinsic dissolve strength:
Dissolves seamless and adjustable whole mesh outside of the dissolve area (player area) upwards beginning with the PlayerPos y offset value.
This value set to 1 is all clear, this is useful with the obstruction mode “none” to draw the silhouette of a building or make it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

Picture/s 12 Intrinsic dissolve strength

Intrinsic off 	**Intrinsic 0.3** 
Intrinsic 0.5	**Intrinsic 1**





13 Preview mode:

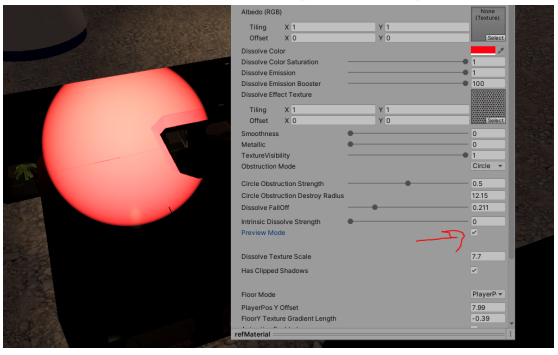
Preview mode for the runtime that shows the dissolve areas within a range of white (100 % dissolve and black (no dissolve).

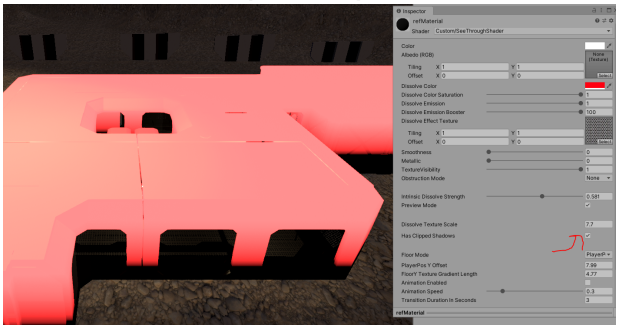
This is useful where you have many objects with different textures and want to have simpler debug mode when adjusting settings.

Picture/s 13 Preview mode

Preview mode off

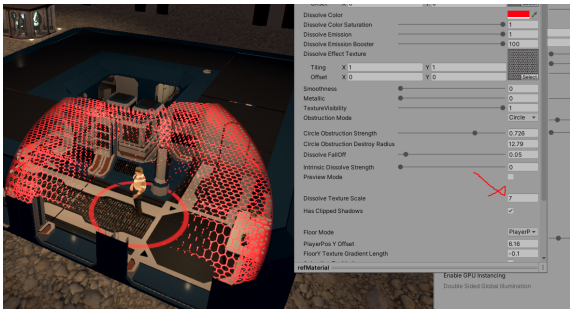
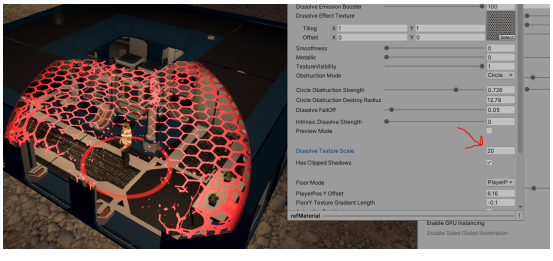
Preview mode on (circle)

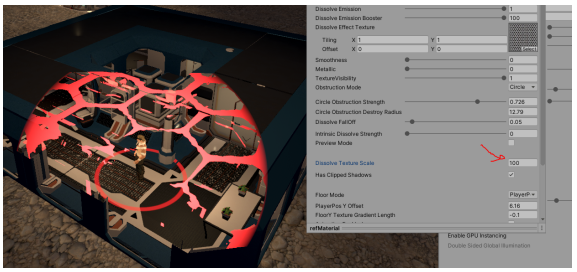
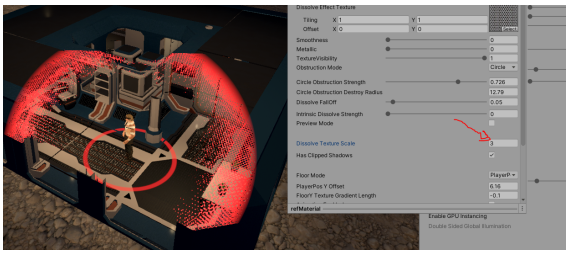


Preview mode (none)	Preview mode (angle)
	

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

Picture/s 14 Dissolve texture scale

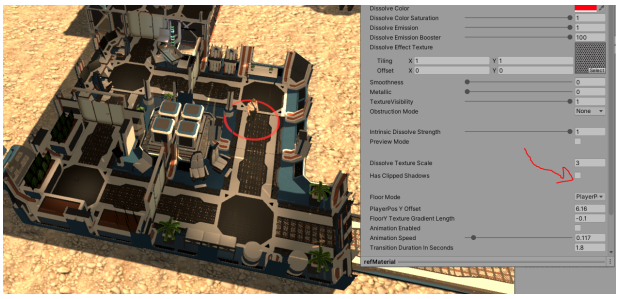
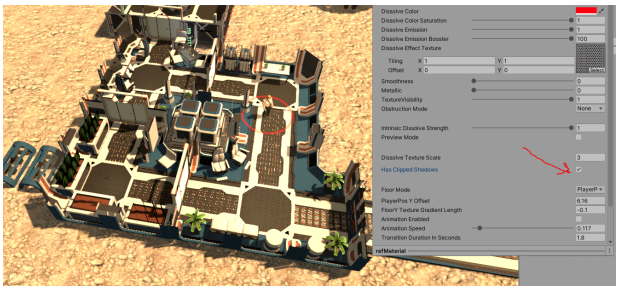
Scale value 7	Scale value 20
	
Scale value 100	**Scale value 3**

---

15 Has clipped shadows:
Controls if the shadows should be drawn in related to the light outside or within the building.
Determines if the shadows are the same as they were 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 remove the roof. If you disable "Has Clipped Shadows" the sun is still blocked from entering.

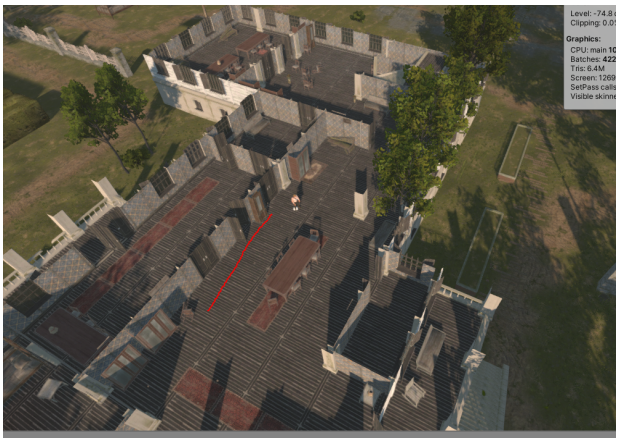
Picture/s 15 Has clipped shadows

Shadows off	**Shadows on**
	

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

Picture/s 16 floor moe

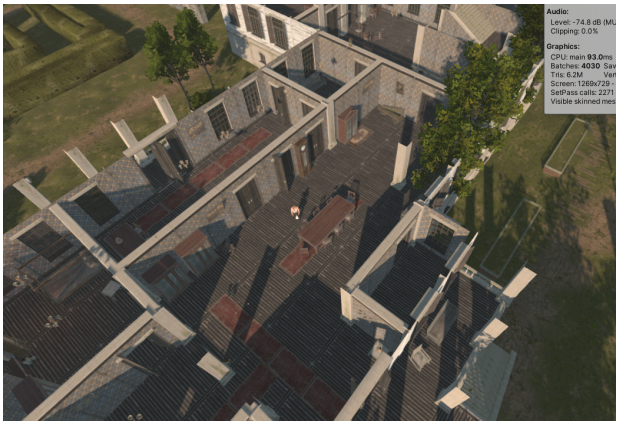
Floor mode player automatic drawing mesh at player position + player position y offset value , so the floor and the walls and interior are drawn	**Floor mode player automatic drawing mesh to second floor + y offset for next walls and floors**



17 PlayerPosition y offset:
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 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 just to remove the roof or do you want walls and interior to be seen to a height of x units.

Picture/s 17 PlayerPosition y offset

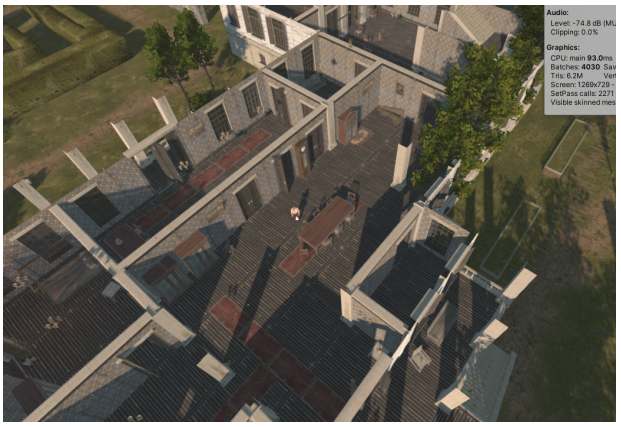
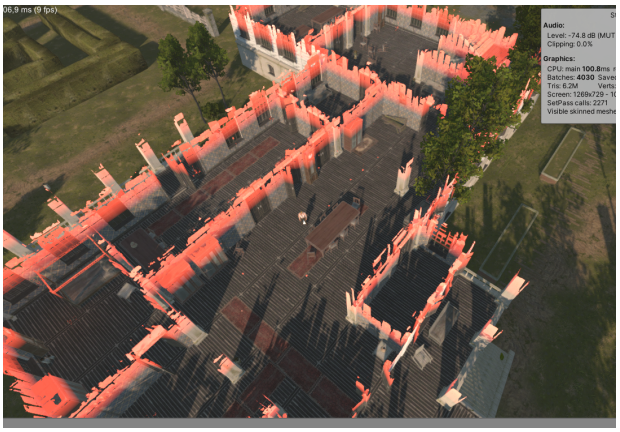
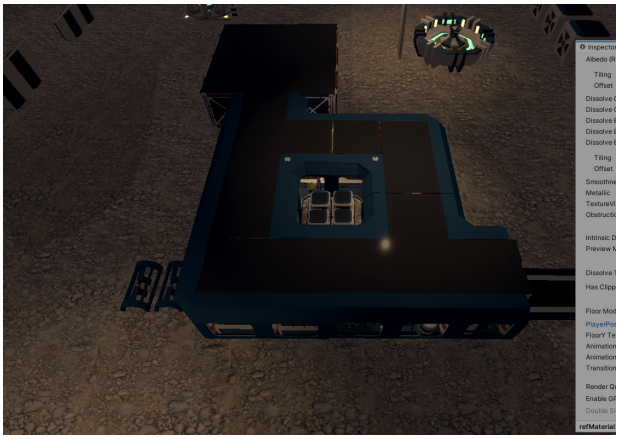
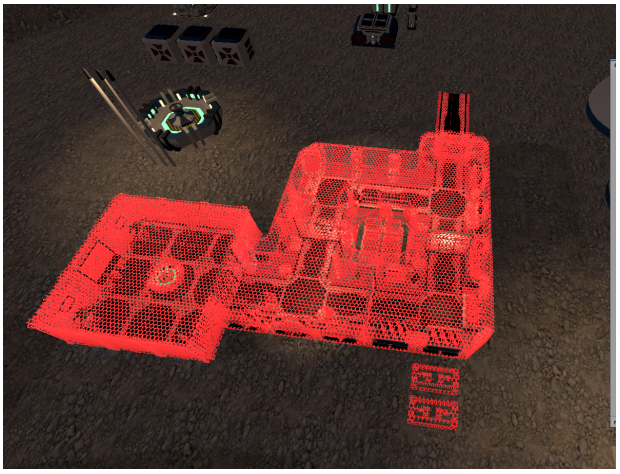
Just show the floor	**Also show the walls**
	

---

18 FloorY Texture Gradient Length:
Controls gradient area between dissolve texture and mesh.

Picture/s 18 Gradient transition length

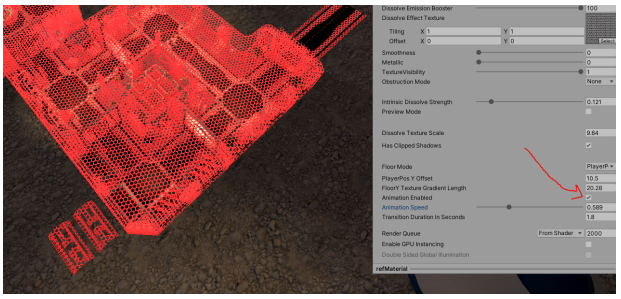
FloorY Texture Gradient Length off	**FloorY Texture Gradient Length on**

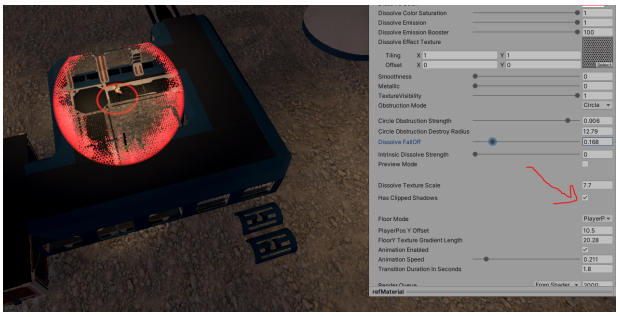
	
	

19 Animation Enabled:
Dissolve texture animation enabled or disabled

Picture/s 19 Animation Enabled

On	**On**



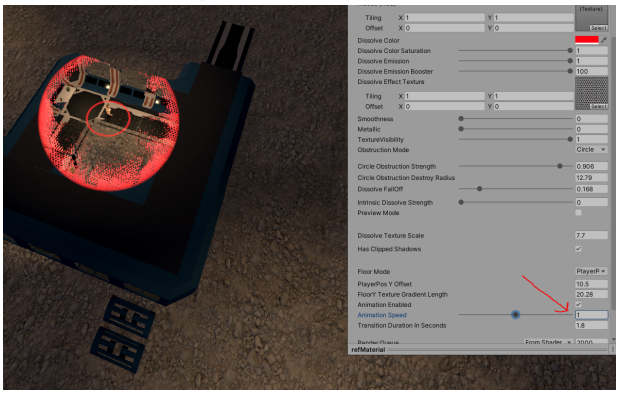


---

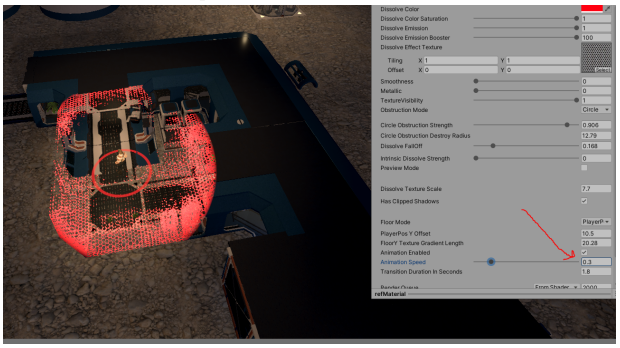
20 Animation Speed:
Dissolve texture animation enabled or disabled

Picture/s 20 Animation Speed

Animation speed 1



Animation speed 0.3

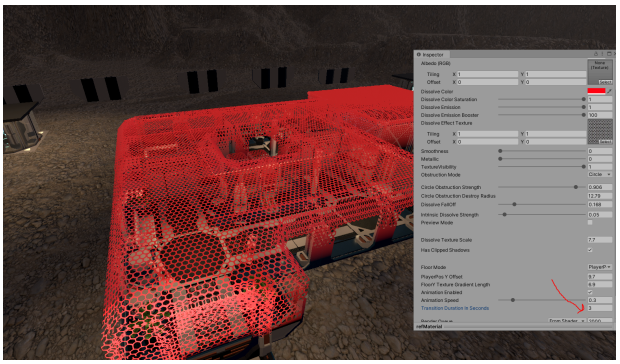


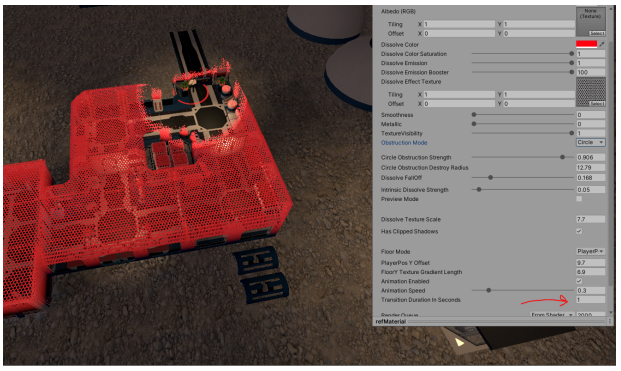
21 Transition Duration in seconds:
Length of the transition animation effect when entering and exiting a building.

Picture/s 21 Transition Duration in seconds

Transition duration in seconds setting

Set to 3



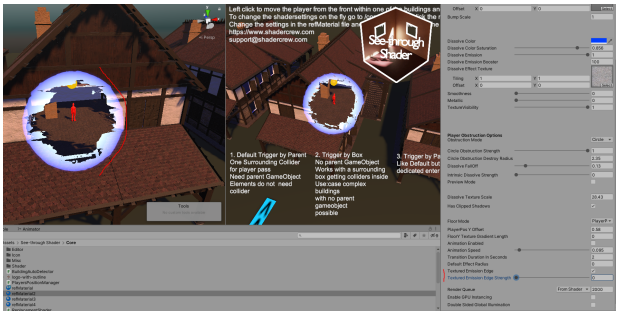


22 Textured Emission Edge Strength:

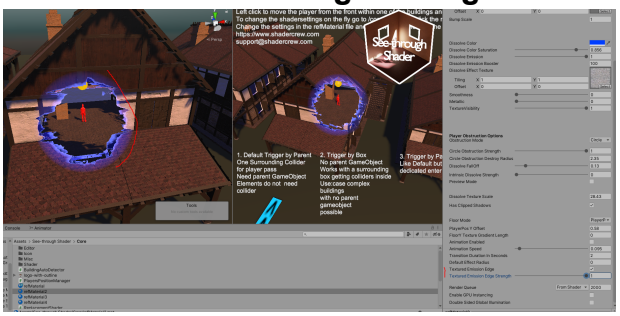
Textured emission allows the dissolve texture to have influence on the emission shape.

Picture/s 22 Textured Emission Edge Strength

Textured Emission Edge Strength 0



Textured Emission Edge Strength 1



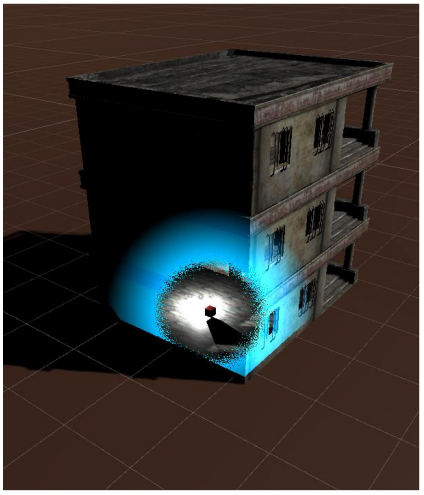
Mode related settings:

For Cone , Cylinder and Circle mode:

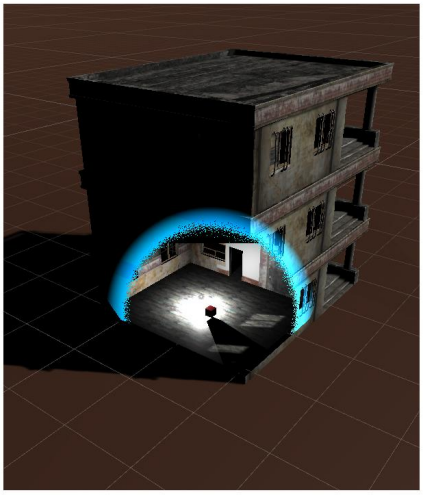
Dissolve Falloff:

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.

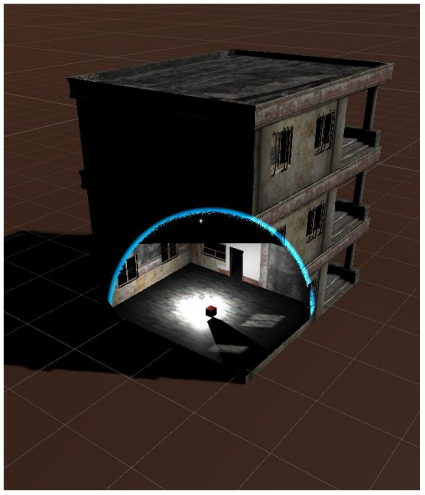
Picture/s Dissolve falloff:



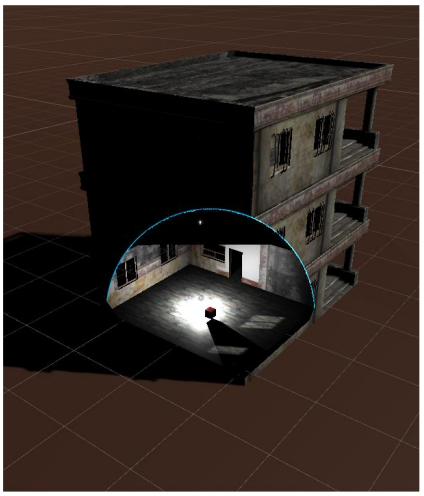
Dissolve FallOff: 0



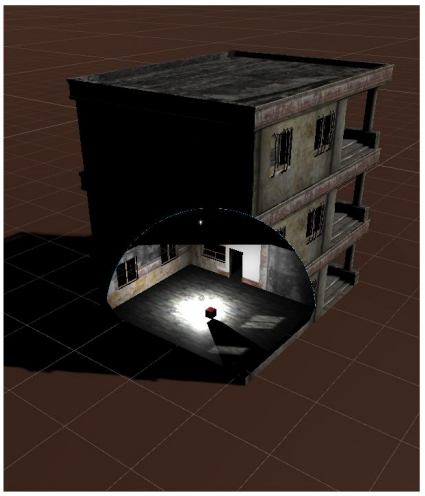
Dissolve FallOff: 0.25



Dissolve FallOff: 0.5

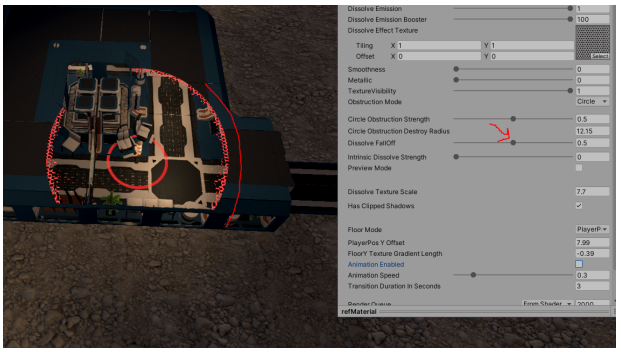


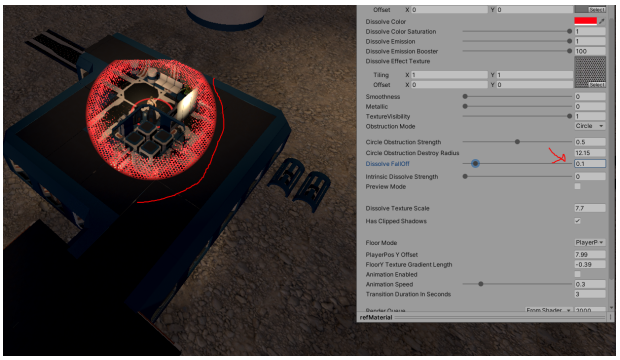
Dissolve FallOff: 0.75



Dissolve FallOff: 1

Dissolve falloff value 0.5 (circle mode)	**Dissolve falloff value 0.1 (circle mode)**



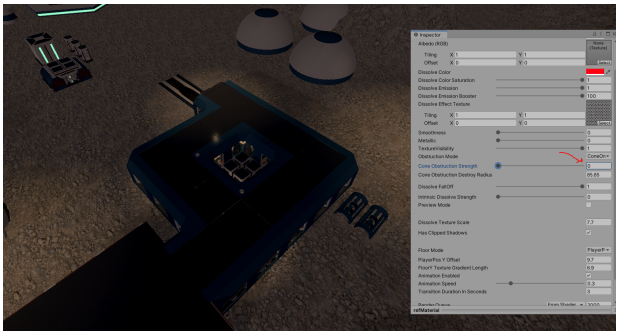


ConeOnly mode:

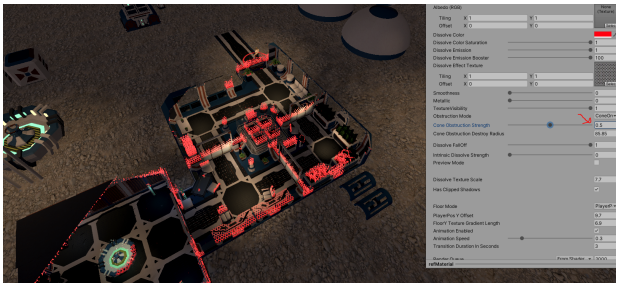
Cone Obstruction Strength:
Strength of clearance , the highest value 1 clears the most.

Picture/s Cone Obstruction Strength:

Cone Obstruction Strength value 0



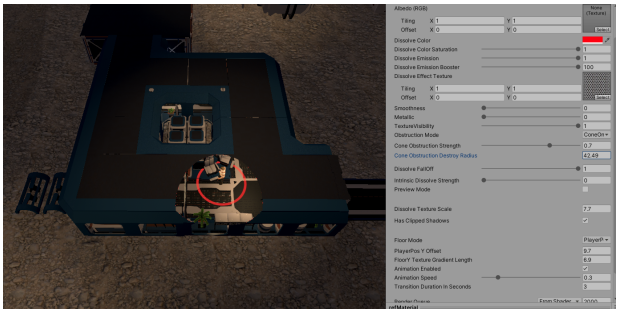
Cone Obstruction Strength value 0.5



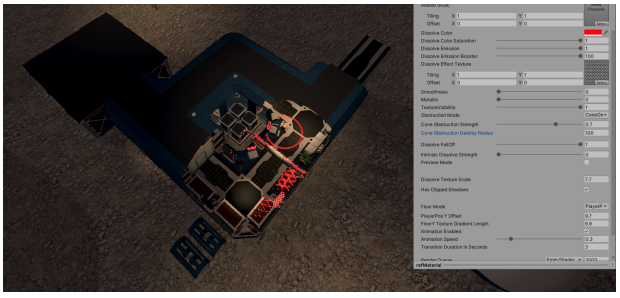
Cone Obstruction Destroy Radius:
Radius of cone within this the mesh will be dissolved

Picture/s Cone Obstruction Destroy Radius:

Cone Obstruction Destroy Radius 42



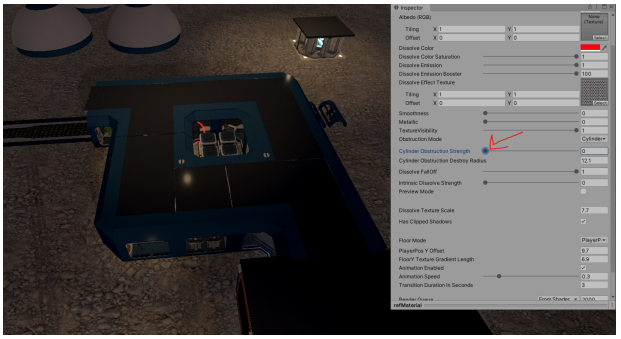
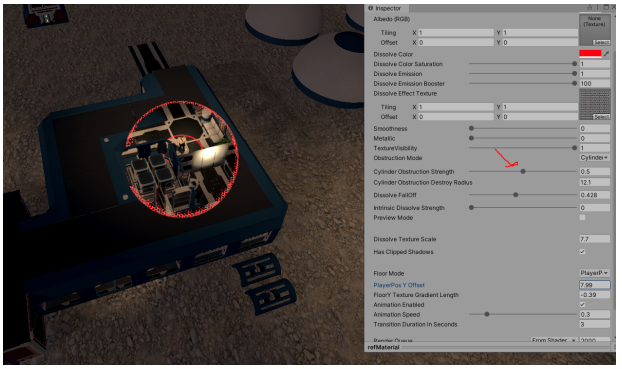
Cone Obstruction Destroy Radius 100



CylinderOnly mode:

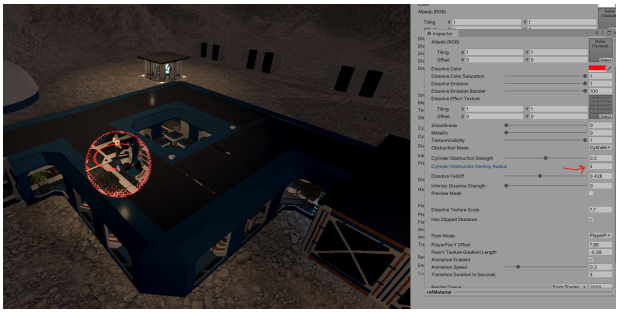
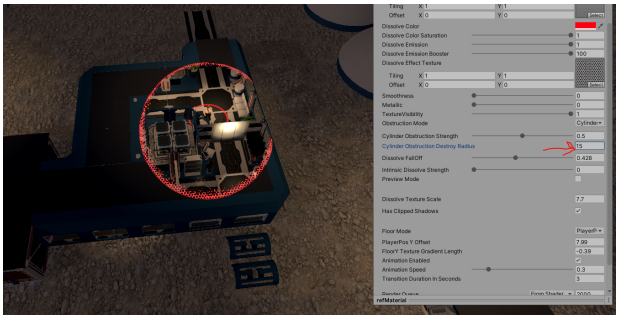
Cylinder Obstruction Strength:
Strength of clearance , the highest value 1 clears the most.

Picture/s Cylinder Obstruction Strength:

Cylinder Obstruction Strength value 0	Cylinder Obstruction Strength value 0.5
	

Cylinder Obstruction Destroy Radius:
Radius of cone within this the mesh will be dissolved

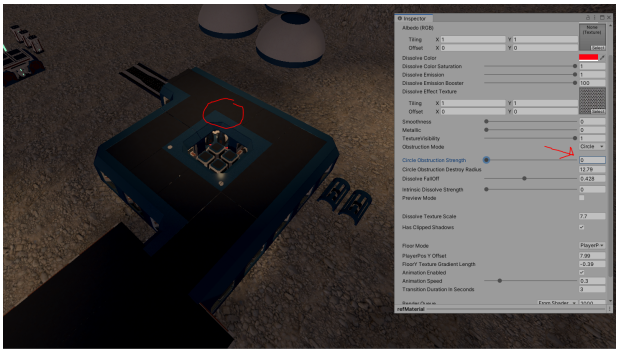
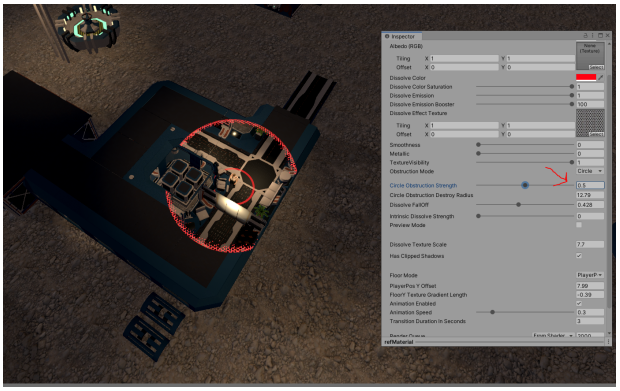
Picture/s Cylinder Obstruction Destroy Radius:

Cylinder Obstruction Destroy Radius 5	Cylinder Obstruction Destroy Radius 15
	

Circle mode:

Circle Obstruction Strength:
Strength of clearance , the highest value 1 clears the most.

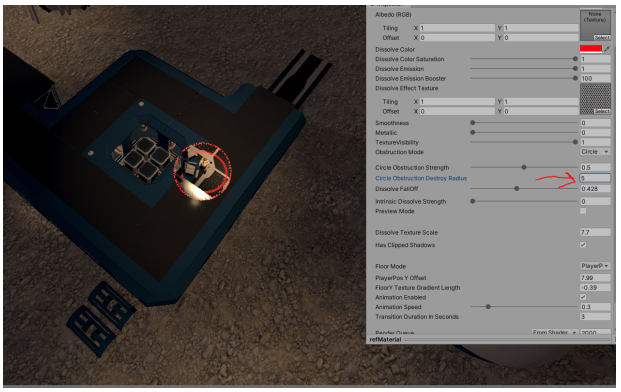
Picture/s Circle Obstruction Strength:

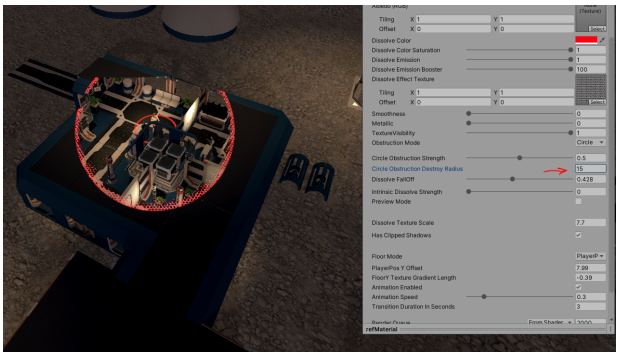
Circle Obstruction Strength value 0	Circle Obstruction Strength value 0.5
	

Circle Obstruction Destroy Radius:
Radius of cone within this the mesh will be dissolved

Picture/s Circle Obstruction Destroy Radius:

Circle Obstruction Destroy Radius 5	Circle Obstruction Destroy Radius 15





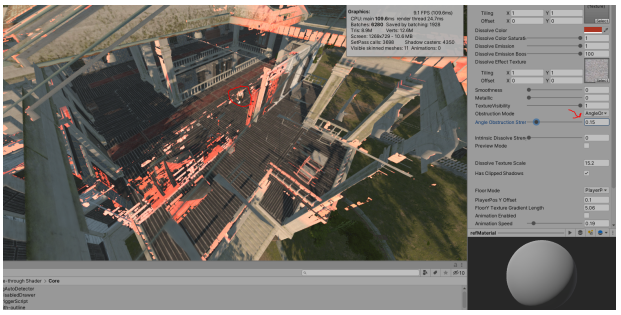
Angle mode:

Angle Obstruction Strength:

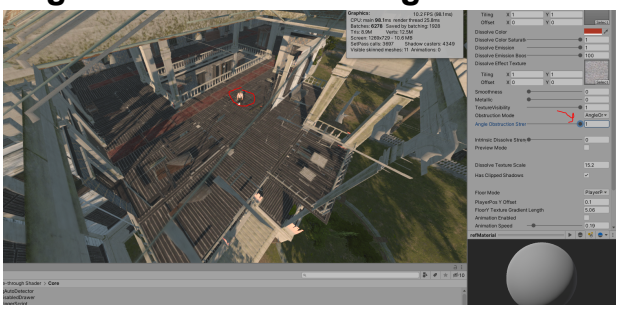
Strength of clearance when seen the player from the angle of the camera

Picture/s Angle Obstruction Strength:

Angle Obstruction Strength value 0.15



Angle Obstruction Strength value 1



Step by Step Guide:

Pre-requisites:

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 /core/refMaterial file you are good to go.

Automatic or trigger mode ?

Now 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 mostly buildings simple 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 position of the player relative to the building .
This will work with simple to middle complex buildings and with most of them.

As these two modes are 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 are the triggers that will trigger when the player passes dedicated enter / exit colliders.
This will be more precise as for example the trigger by parent will iterate over all items beneath a parent gameobject.
As in the nature of unity and already mentioned 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.

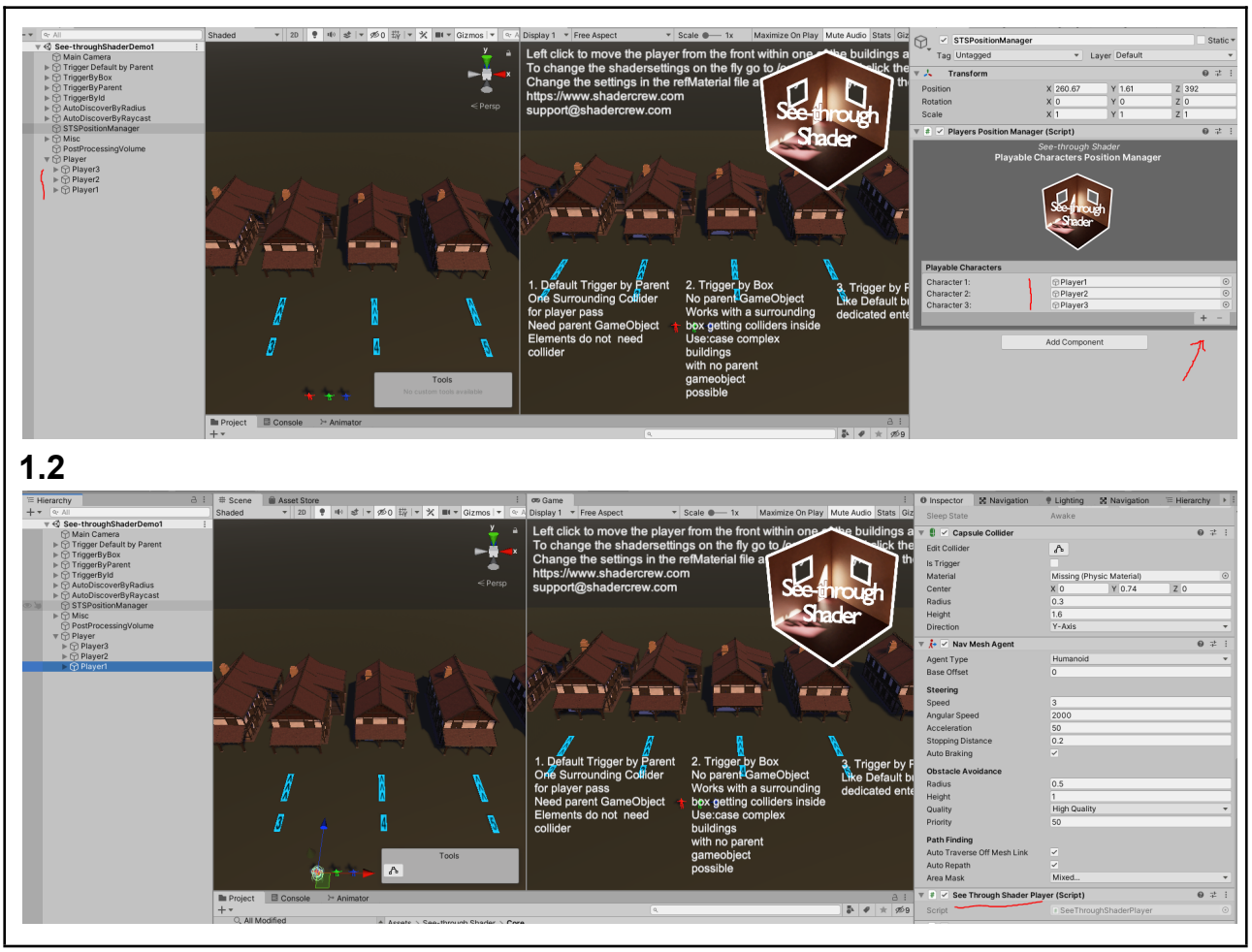
TriggerById trigger script works where no collider and no parent gameobject is available , the game designers assigns a unique id or name to every building element so they can be found by the trigger.

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

You are totally free to combine the automatic mode and the trigger mode together to best fit your game project
The trigger modes work with colliders to trigger the effect so the player must have a rigidbody (can be set to isKinematic=true) to collide.

Step 1 Always needed:
1 Add the “PlayerPositionManager” script to a 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.
1.2 Add the SeeThroughShaderPlayer script to each playable Character

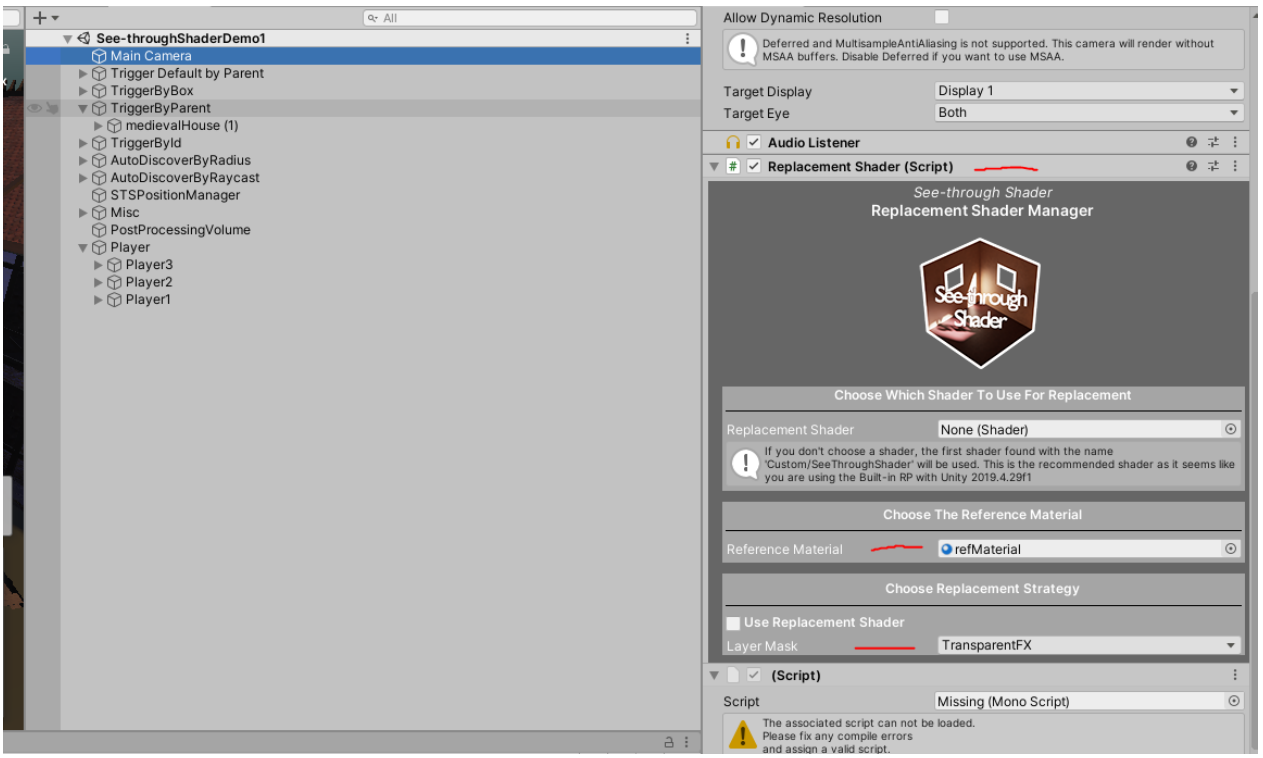
1.1



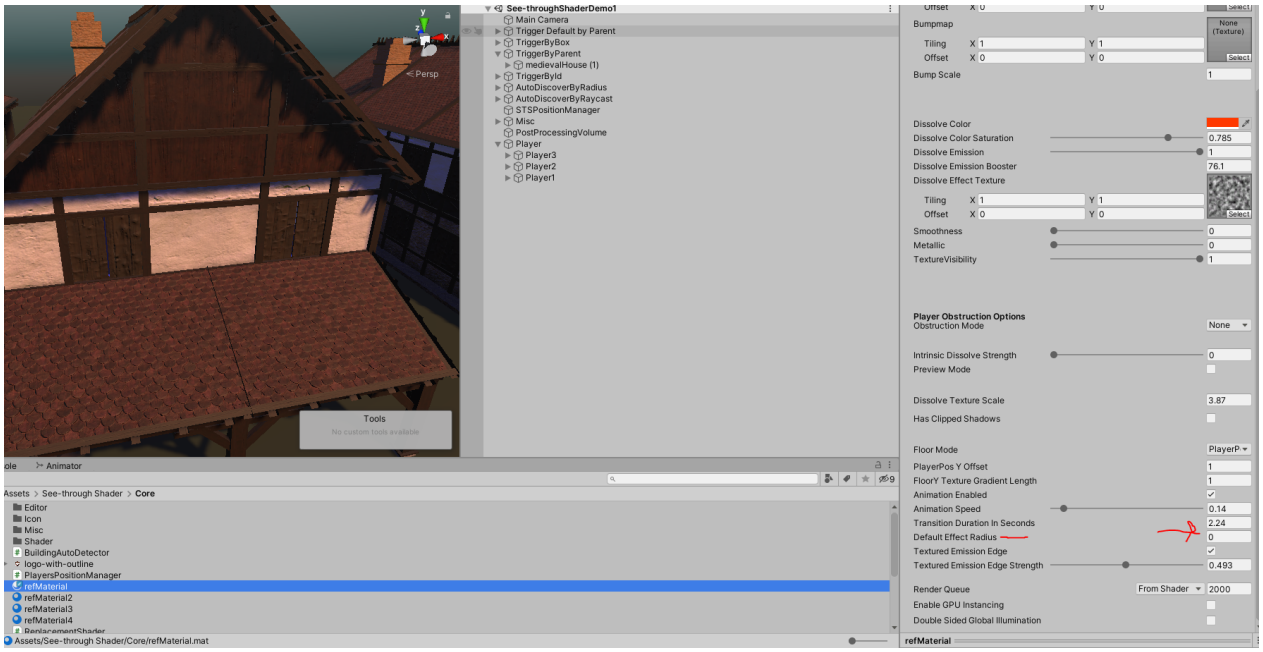
1.2

Step 2 automatic mode
2.1 Add the ReplacementShader script to the main Camera
2.2 Set the layer(s) you want the shader to operate on.
Multiple layers are possible.
2.3 If radius is used set the radius in the refMaterial or your template Material
2.4 If raycast mode is used , add the BuildingAutodetector to the player
Combination of 2.3 and 2.4 is possible. Press Play

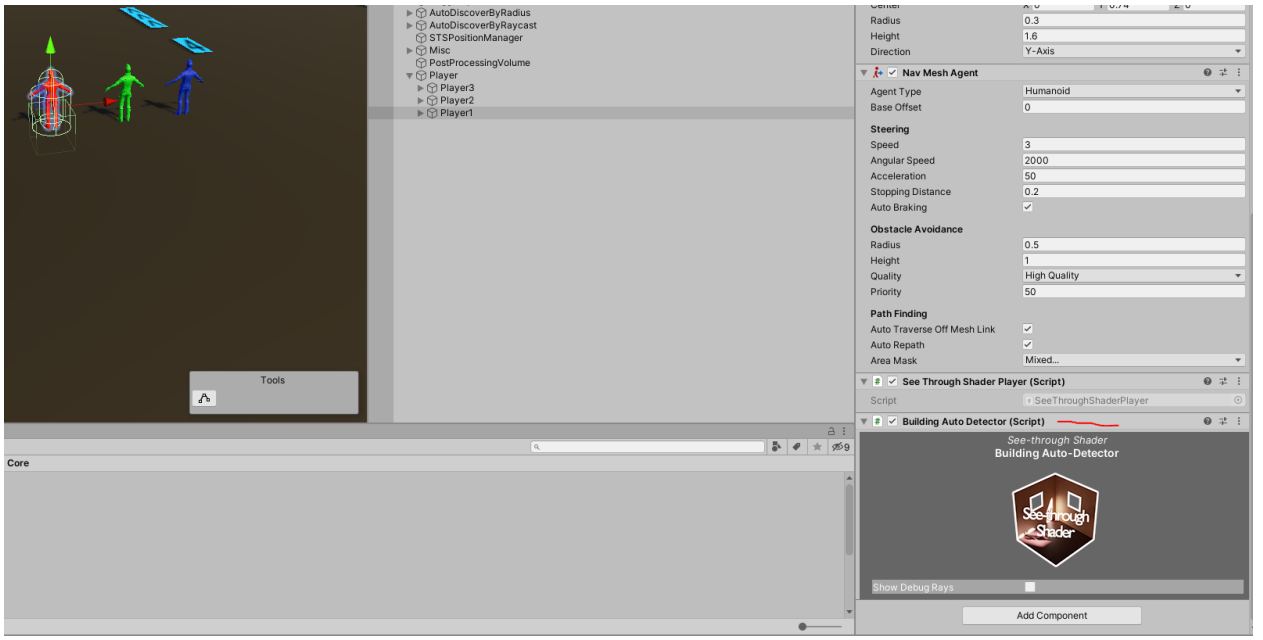
2.1 / 2.2


All
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
See-through Shader
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


See-throughShaderDemo1
Main Camera
Trigger Default by Parent
TriggerByBox
TriggerByParent
medievalHouse (1)
TriggerById
AutoDiscoverByRadius
AutoDiscoverByRaycast
STSPositionManager
Misc
PostProcessingVolume
Player
Player3
Player2
Player1
Persp
Tools
No custom tools available
Animator
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
Bumpmap
Tiling
Offset
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
refMaterial

2.4


AutoDiscoverByRadius
AutoDiscoverByRaycast
STSPositionManager
Misc
PostProcessingVolume
Player
Player3
Player2
Player1
Tools
Core
Radius
Height
Direction
Nav Mesh Agent
Agent Type
Humanoid
Base Offset
Steering
Speed
Angular Speed
Acceleration
Stopping Distance
Auto Braking
Obstacle Avoidance
Radius
Height
Quality
High Quality
Priority
Path Finding
Auto Traverse Off Mesh Link
Auto Repath
Area Mask
Mixed...
See Through Shader Player (Script)
Script
SeeThroughShaderPlayer
Building Auto Detector (Script)
See-through Shader
Building Auto-Detector
See-through Shader
Show Debug Rays
Add Component

Step 2 (trigger mode)

TriggerByParent:

Trigger by Parent has two trigger options:

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

Option 1 whole collider:

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

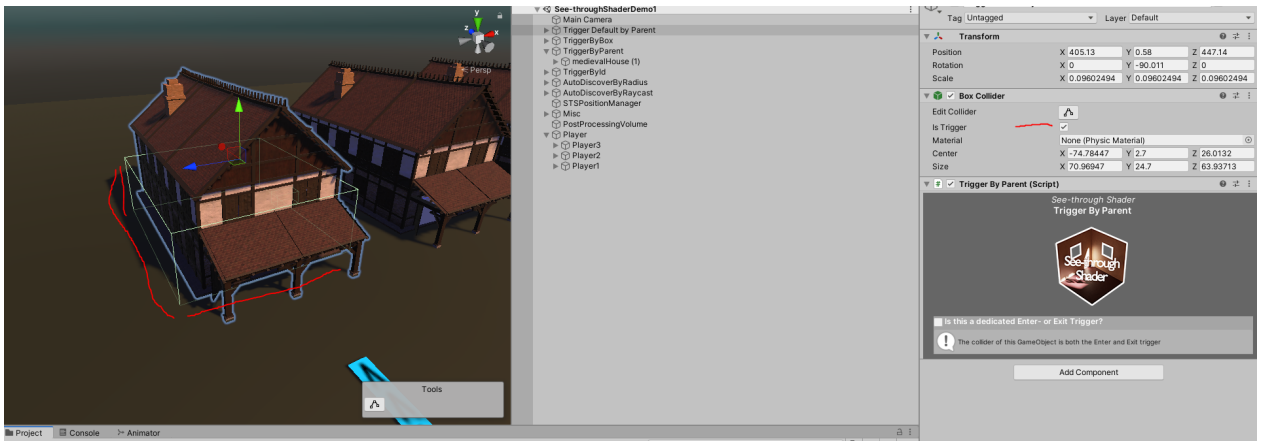
Add the TriggerByParent.cs script to the parent GameObject of the building

Drag the refMaterial file to the refMaterial field

Press play

TriggerByParent Option 1 whole enter / exit collider

2.1

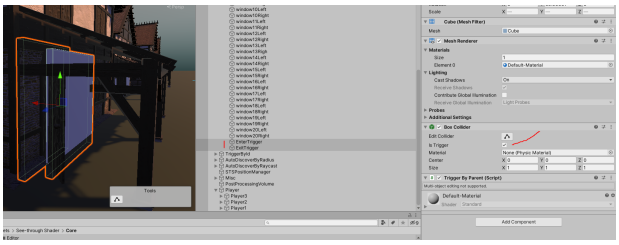


TriggerByParent Option 2 dedicated enter / exit triggers:

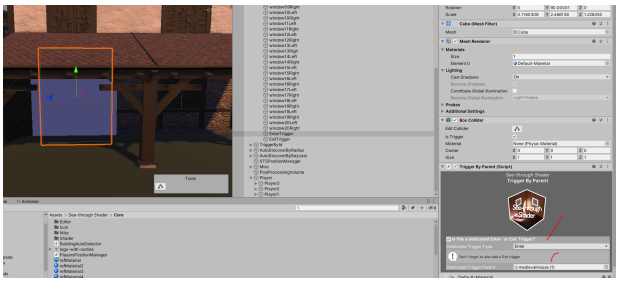
When the use case of your game is a building with a door or something like that where you want the player exactly to pass then you can create two cubes acting as the enter exit trigger.
The settings of both will be the same, just the IsEnterTrigger or IsExitTrigger will be different.

2.1 Create two cubes and place / scale them to fit your entrance , set 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 buildings elements.
2.4 Now for each trigger set the isEnterTrigger or IsExitTrigger flag
2.5 (OPTIONAL) Add the ShaderPropertySync script and drag the refMaterial or your template Material file onto the Material field and check the “SyncContinuus” Box.
This can be used in game as well 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.

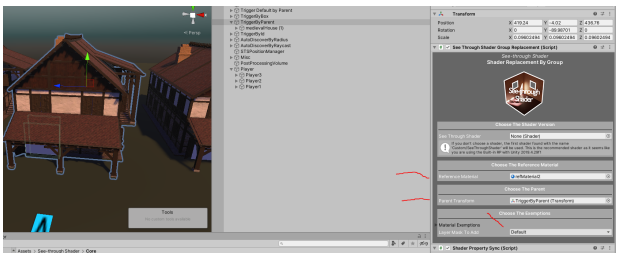
2.1



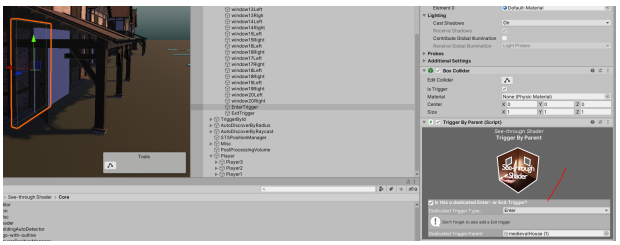
2.2



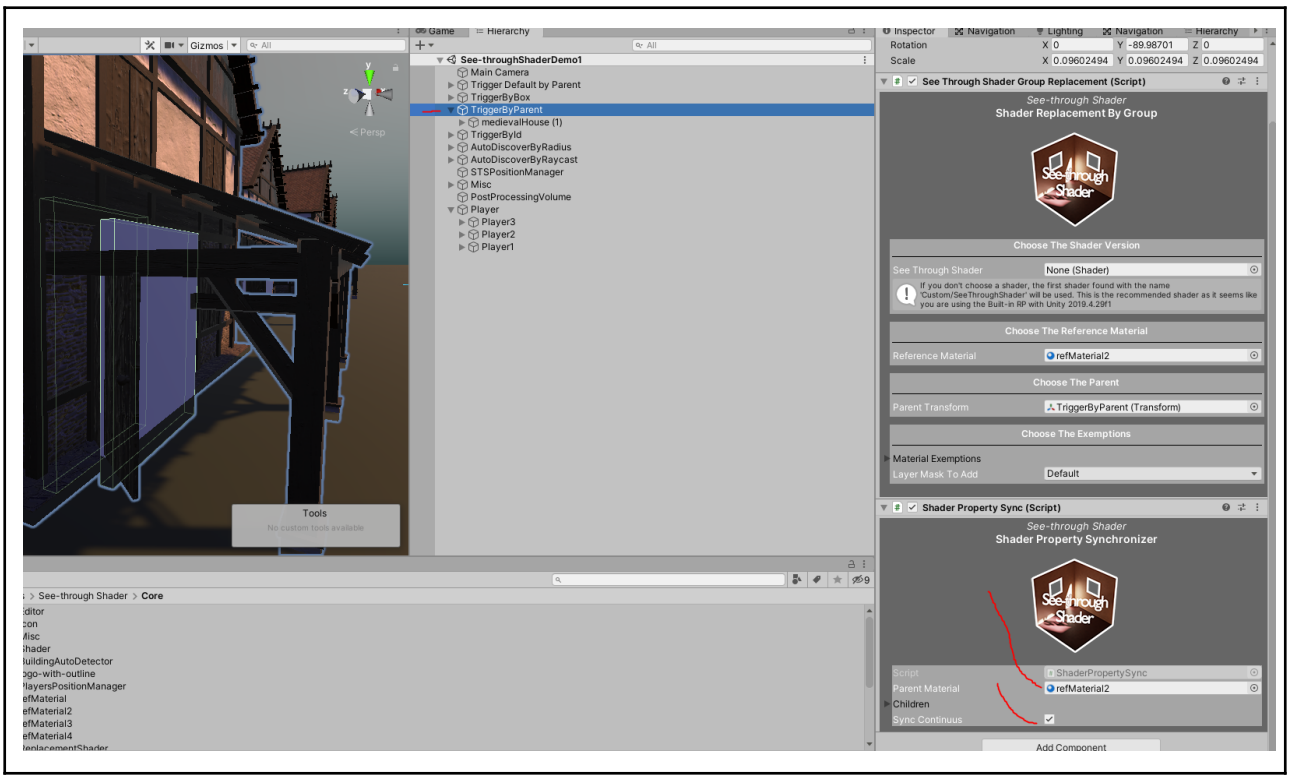
2.3



2.4



2.5



Thank you for using See-through Shader. Any feedback is welcome.
Shadercrew
October 2021
support@shadercrew.com
https:/www.shadercrew.com