See-Through System v1.4 manual

Contents

1. General setup

To enable See-Through System(STS), place “Image Effects/See-Through System” component on your camera.

Despite huge amount of possible settings, STS works “out-of-the box”, all you need to do is to designate Trigger, Obstacle and Background layers:

Trigger Layers designate objects that needed to be visible when obscured by Obstacles. Background Layers will be visible in the areas where triggers are obscured. Note that you can place your characters in the Trigger Layers but exclude them from Background Layers. That way your character will trigger transparency in obstacles but will not be visible themselves.

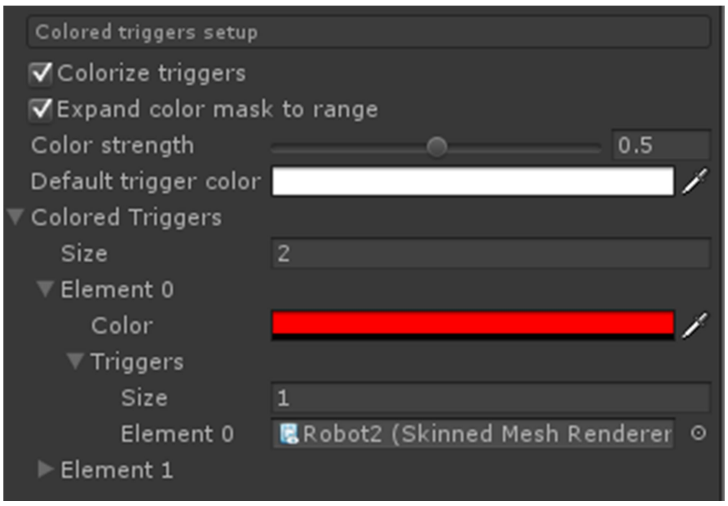
In most cases, Triggers are your characters, Obstacles are your walls and Background is all your layers except Obstacles.

2. Selective disabling of triggers and obstacles.

By placing objects in “disabled” arrays, you can tell STS to ignore them when calculating obscurance. You can use *DisableTrigger, DisableObstacle, EnableTrigger, EnableObstacle* methods to add renderers into these arrays at runtime.

Note that you need to put *Renderers*, **NOT** *GameObjects* in these arrays, and single character can consist of multiple Renderers and GameObjects. Use Component.GetComponentsInChildren to get all renderers of your character.

3. Colorized trigger



Colorized triggers allow you to change tint of transparent windows created by different triggers. To enable this effect, toggle **Colorize triggers** checkbox.

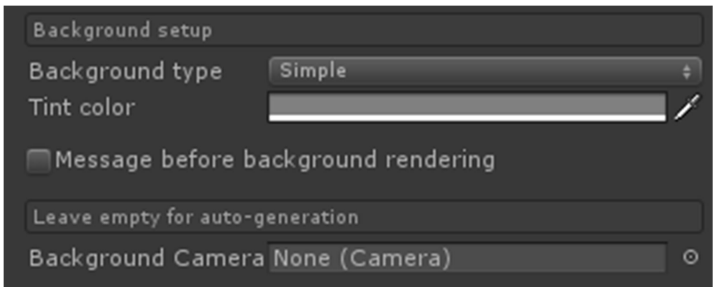
Expand color mask to range defines whether only the trigger itself will be colored or the area around it will be affected too.

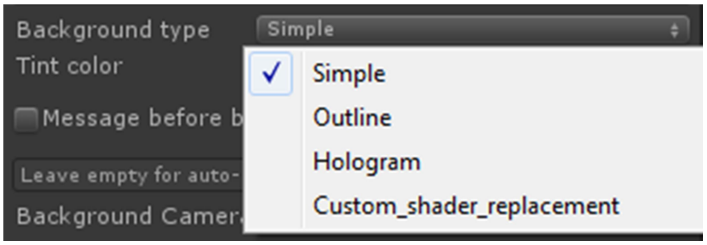
Color strength defines color coefficient, value of 1 will provide complete coloring, value of 0 will disable any coloring.

Default trigger color can be set up so that any triggers with undefined color will be colored with it.

Colored triggers array contains color and renderer definitions for colored triggers. You can fill it in editor or use *ColorizeTriggerObject* and *DecolorizeTriggerObject* methods at runtime.

4. Background rendering setup





Background type defines how background will be rendered. "Custom_shader_replacement" allows you to designate your own shaders and tags to be used with BackgroundCamera.RenderWithShader method.

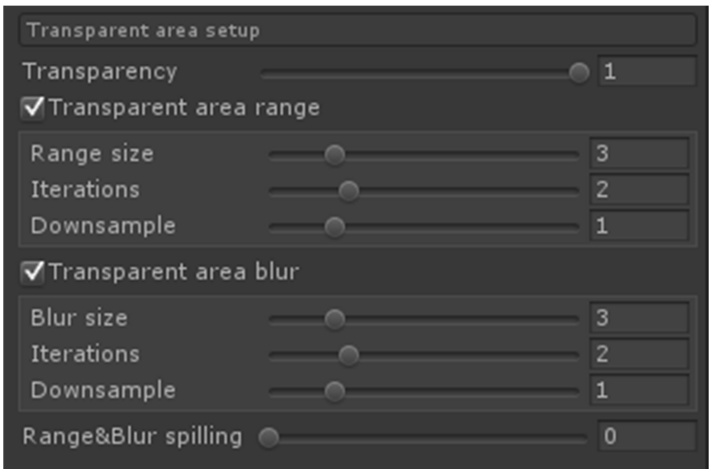
Tint color allows you to shift colors of final background image. For more precise color control, you can use standard color curves image effects on Background Camera.

Message before background rendering – if toggled on, it will send a "STS_BeforeBackRender" message to selected object with background camera as parameter. This allows you to change camera parameters by your own scripts.

Background camera – you can create your own background camera if you want to place image effects on it. If no camera selected, STS will automatically create simple camera for background rendering.

Outline and background colors – if you're using Hologram or Outline backgrounds, this option will allow you to customize them further.

5. Transparency windows setup



Transparency – this coefficient defines transparency of obstacles that obscured your triggers.

Transparent area range – this will generate transparent outline around your triggers.

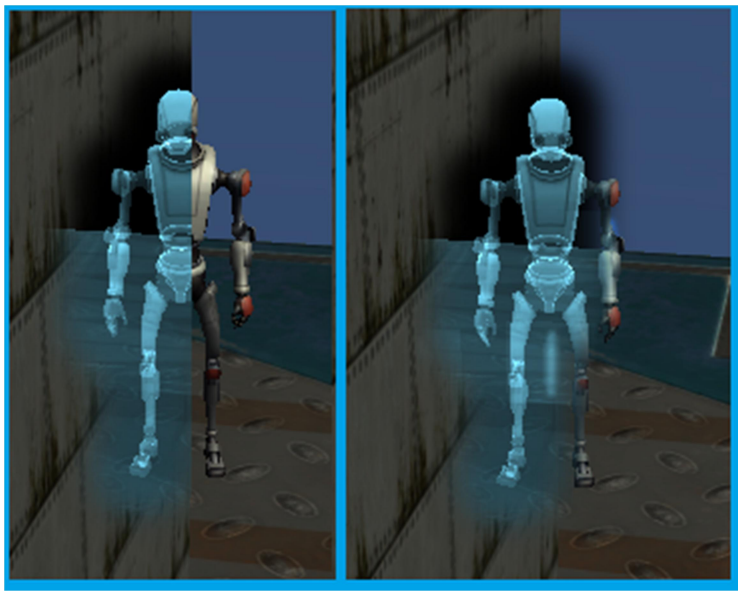
Transparent area blur – this will blur transparent outline.

Sliders control blur parameters and are similar to standard fast blur image effect.

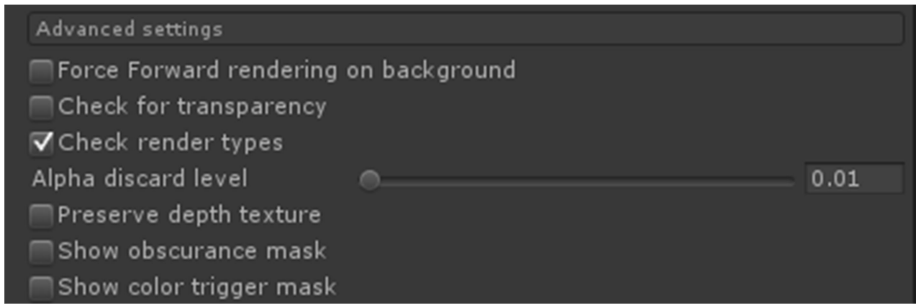
No range and blur: **Range:** **Range and blur:**

Range&Blur spilling – how much should special effects from transparent areas spill on non-obstacle areas of the screen.

Spilling = 0 **Spilling = 1**



6. Additional settings



Force Forward rendering on background – if toggled on, STS will always use Forward rendering when rendering parts of scene visible through triggered transparency windows. It's used for avoiding lighting issues when using Deferred lighting rendering path.

Check for transparency – if toggled on, STS will check if there's any obscured areas that need to be rendered before rendering background, also will optimize background rendering by designating areas that need to be rendered(works only with forward rendering). This can save a lot of GPU performance if there's a lot of fancy background, but it will cost some CPU time to make that check.

Check render types – if toggled on, STS will consider type of shaders on your triggers and obstacles when making trigger masks. Use it if you're heavily using transparent and cutout shaders (for example, all 2D projects using sprites should use this option). **Alpha discard level** determines transparency level of alpha-blended objects that's considered invisible for purposes of STS.

Preserve depth texture – STS will preserve original depth texture. Use it if you're using image effects that use depth textures down the line.

Show obscurance mask – Debug mode, will show mask that's been used to blend transparent and non-transparent areas.

Show color trigger mask – Debug mode, will show mask that's been used to colorize triggers. Note that color mask is inverted.