Screen Transitions Pro v1.3

OVERVIEW

Screen Transitions Pro contains a collection of high quality full screen transition effects to skyrocket the quality of your games to the next level! Screen Transitions Pro makes it easy to create professional looking screen transitions that are perfect for loading screens, battle transitions, stage intros, outros and much more!

This document will give you a general overview of this package and will tell you everything you need to know to start using it out-of-the-box.

Installation

Screen Transition Pro is installed using Unity's standard .unitypackage format. Installation is very straightforward: simply unpack the entire package into your project, then wait for the editor to compile all the scripts - **Screen Transition Pro** will then be installed.

Do note that it is not advisable to change the internal structure of the **Screen Transition Pro** folder itself or to rename the folder in any way. Future updates might create duplicates of scripts and assets, which could create some conflicts.

Package contents

This package contains a total of 100+ ready-to-use screen transitions of 5 different types:

- Fade
- Simple
- Displacement
- Single focus
- Multi focus

Screen Transition Pro root folder contains the following folders:

- **DEMO** - here you will find demo scenes for each transition type with example scripts, materials that are used on these scenes, art assets from Unity's 2D Game Kit free package and examples of Animation Clips.
  If you are already familiar with this package or are confident that you can figure it out yourself, feel free to delete this folder.
- **Documentation** - contains a copy of this document.
- **Integrations** - this folder contains 3 additional unity packages for 3rd party integration:

    - **PlaymakerIntegration** - contains 13 custom actions, 1 example scene and 1 example material.
    - **iTweenExample** - contains 1 example scene, 1 example script and 1 example material.
    - **DOTweenExample** - contains 1 example scene, 1 example script and 1 example material.

      Note that example scenes, materials and scripts will be unpacked into the DEMO folder. If you need to check them, make sure that you didn't delete the DEMO folder, because these scenes use art assets from it.
- **Materials** - contains 11 different transitions materials. It is highly recommended that you use them as a reference and make copies of the materials that you need to use in your project. Otherwise, if you use original materials, future package updates might override all of your local changes.
- **Scripts** - here you will find 11 C# scripts to control different screen transitions. You will need to attach one of them to your game camera to perform a screen transition.
- **Shaders** - here are listed 11 image effect shaders that are used for different screen transitions.
- **Textures** - here you will find 2 sub-folders
    - **Displacement** - contains 29 textures to control displacement screen transitions.
    - **Simple** - contains 83 grayscale textures to control simple screen transitions.

      For reasons of consistency all displacement textures have import settings set to: *2048x2048, Clamp, Point (no filter), RGBA 32 bit* and for simple textures to: *2048x2048, Clamp, Point (no filter), Format Automatic, Compression High Quality*.
      Some textures (especially simple ones) can be reduced in size without losing a significant level of detail. However, it is not recommended to change import settings of displacement textures since it can significantly damage transition quality. Do it at your own risk when it is really needed.

Screen Transitions Pro and Post-Processing Stack v2

Please note that **Screen Transitions Pro** is **NOT** an extension of **Post-processing Stack v2**. All screen transitions will be applied **after post-processing**, including anti-aliasing. If you want apply anti-aliasing to a screen transition, you need to use image effect based anti-aliasing, e.g. FXAA Fast Approximate Anti-Aliasing.

TRANSITIONS EXPLAINED

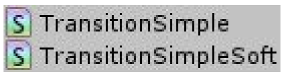
Screen Transition Pro contains 4 different transition types. In this chapter you will find detailed explanations of how each transition works, which properties they have and what you can do with them.

In general, all transitions have 3 main properties:

- **Transition material** - material, based on image effect shader, that will be applied to a rendered image.
- **Transitioning** flag - defines if a transition is active. In other words, it determines whether should transition material be applied to a rendered image or not. Most of the time this flag should be unset to avoid unnecessary calculations. Set it to *True* at the moment when you want to start a screen transition and set it back to *False* when transition is finished.
- **Cutoff** - the current progress of the transition. The rule of thumb here is:
  0 - No transition.
  1 or more - Full transition to background color.

  Actual values can vary depending on the transition type (see below).

Also each transition (except Fade) has 2 variations (2 separate shaders):


TransitionSimple
TransitionSimpleSoft

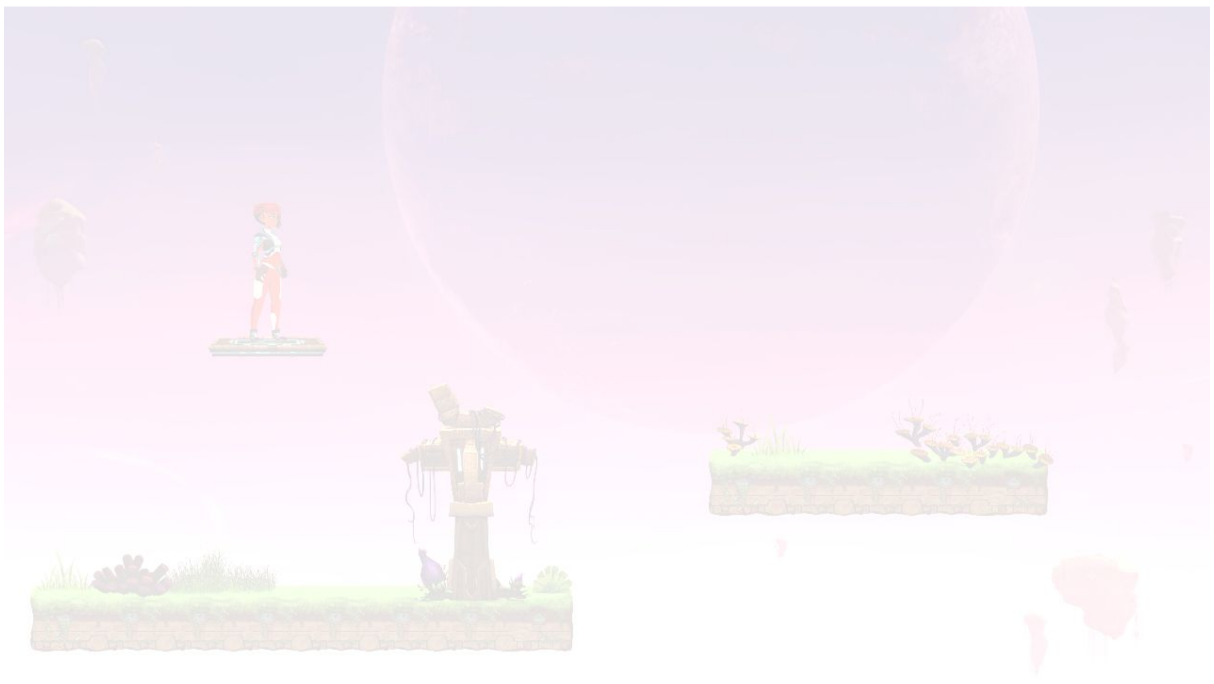
- **Normal** - with sharp edges
- **Soft** - with smooth falloff applied to blur sharp edges

Single Focus and *Multi Focus* transitions also have **Noise** variations which have noisy falloff instead of the smooth one.

The soft version is more performance-heavy, so if you target low-end devices it is recommended to use Normal variation instead.

Now let’s deep dive into each transition type.

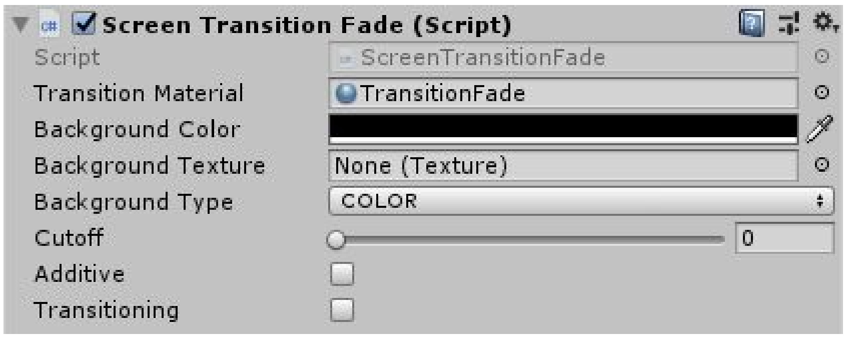
Fade Transition



During **Fade transition**, pixels of a rendered image gradually turn into the transition's **Background Color** or **Background Texture.** You can also optionally set the flag for "Additive" effect - background color will be gradually added to the rendered texture during transition.

To use Fade transition just attach ***ScreenTransitionFade.cs*** script to your game camera.

Properties:

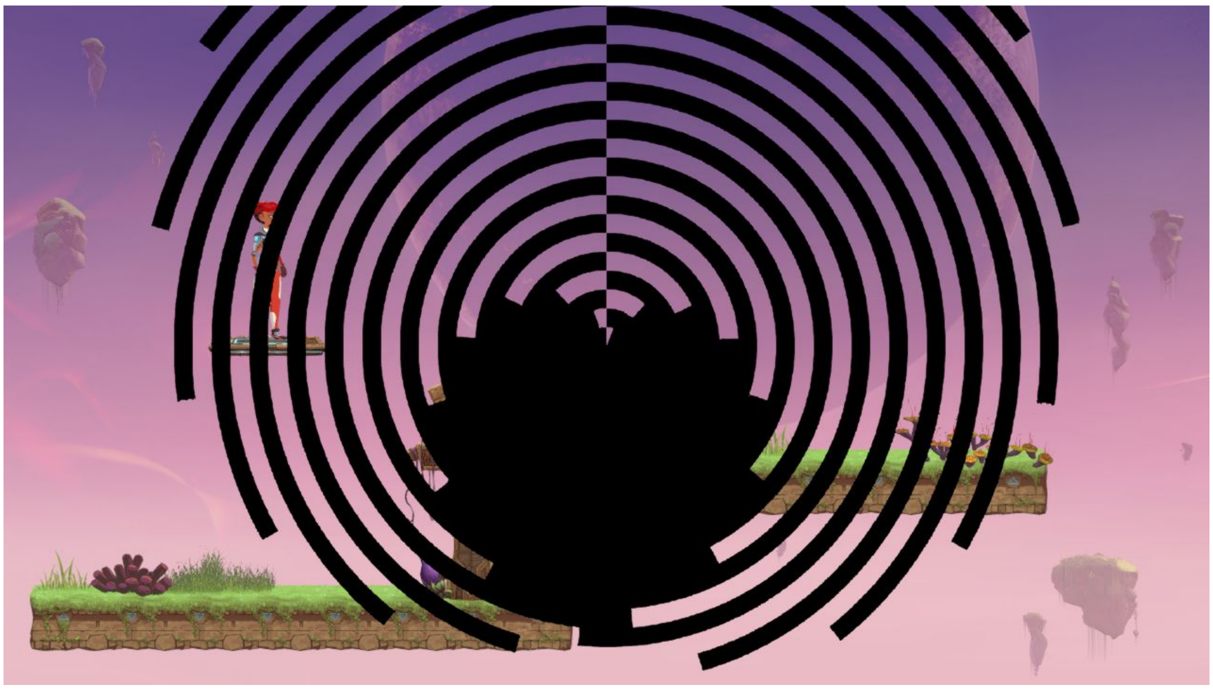


Screen Transition Fade (Script)

Script	ScreenTransitionFade
Transition Material	TransitionFade
Background Color	■
Background Texture	None (Texture)
Background Type	COLOR
Cutoff	0
Additive	☐
Transitioning	☐

- **Transition Material** - material that will be applied to rendered image during transition.
- **Background Color** - background color that will be used during transition.
- **Background Texture** - texture that will be used as the background during transition (Render Textures allowed).
- **Background Type** - you can choose what kind of background you want to use for the transition.
  - **COLOR** - Background color will be used as the background.
  - **TEXTURE** - Background texture will be used as the background.
- **Cutoff** - represents current progress of the transition.
- **Additive Color** - defines if transition color should be progressively added to the rendered image during transition. Works best with white background color.
- **Transitioning** - flag that tells Unity to process transition. Set this flag at the beginning of the transition and unset it at the end to avoid unnecessary calculations and to save some performance.

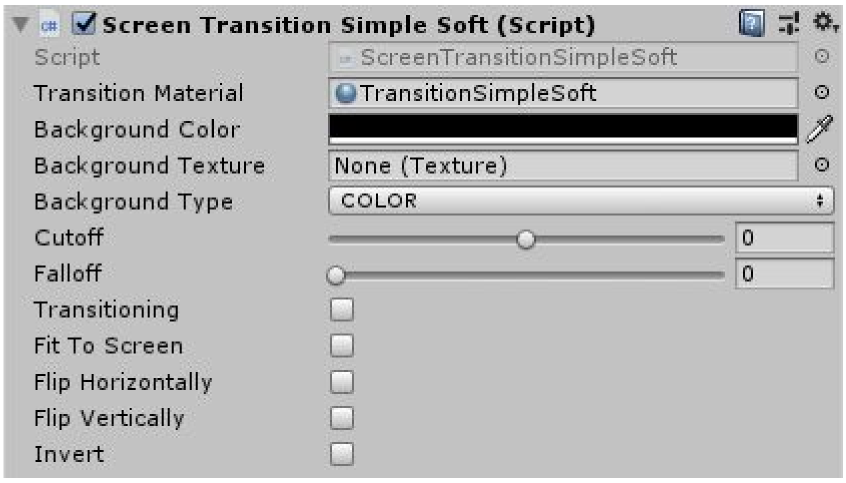
Simple transition



During **Simple transition**, pixels of a rendered image gradually turn into the transition's **Background Color** or **Background Texture** based on the pattern provided by the transition texture. The shader uses only *Blue channel* of the transition texture to perform calculations.

To use **Simple transition** just attach ***ScreenTransitionSimple.cs*** or ***ScreenTransitionSimpleSoft.cs*** script to your game camera.

Properties:



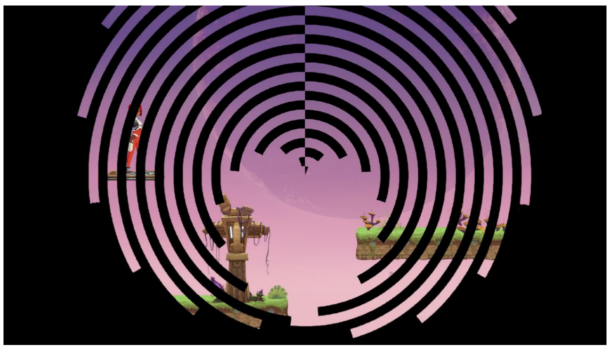
Screen Transition Simple Soft (Script)

Script	ScreenTransitionSimpleSoft
Transition Material	TransitionSimpleSoft
Background Color	■
Background Texture	None (Texture)
Background Type	COLOR
Cutoff	0
Falloff	0
Transitioning	☐
Fit To Screen	☐
Flip Horizontally	☐
Flip Vertically	☐
Invert	☐

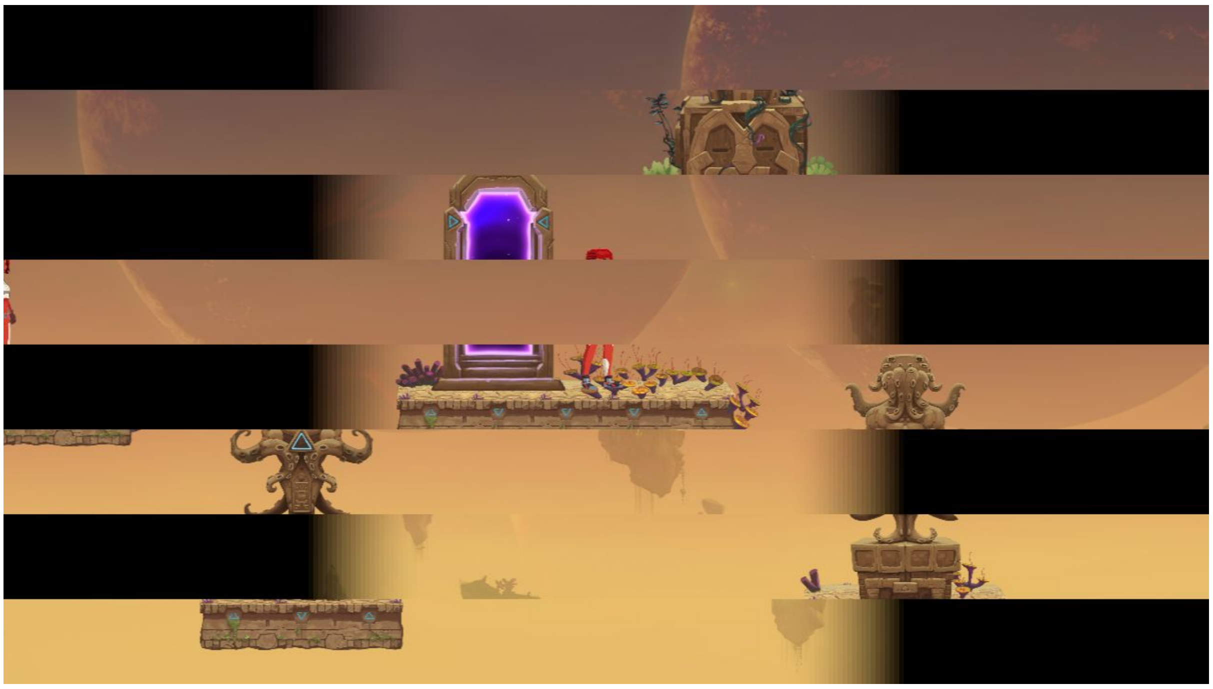
- **Transition Material** - material that will be applied to rendered image during transition.
- **Background Color** - background color that will be used during transition.
- **Background Texture** - texture that will be used as the background during transition (Render Textures allowed).
- **Background Type** - you can choose what kind of background you want to use for the transition.
    - **COLOR** - Background color will be used as the background.
    - **TEXTURE** - Background texture will be used as the background.
- **Cutoff** - represents current progress of the transition.

	No transition	Full transition
Normal	0	1
Soft	[-1; 0] - no transition (depends on the falloff size)	1

- **Falloff** - **FOR SOFT VERSION ONLY**. Smooth blend between rendered texture and background color. 0 - no blend (sharp border). 1 - max blend (size depends on the gradient in the *Blue channel* of the transition texture).
- **Transitioning** - flag that tells Unity to process transition. Set this flag at the beginning of the transition and unset it at the end to avoid unnecessary calculations and to save some performance.
- **Fit To Screen** - set this flag if you want transition texture to fit the screen. If unset, the transition texture will maintain 1:1 aspect ratio and will fit the screen horizontally if screen width is greater than screen height or it will fit the screen vertically if screen height is greater than screen width.
- **Flip Horizontally** - set this flag if you want to flip transition texture horizontally.
- **Flip Vertically** - set this flag if you want to flip transition texture vertically.
- **Invert** - set this flag if you want to invert transition texture. It will swap rendered image with background color/texture.



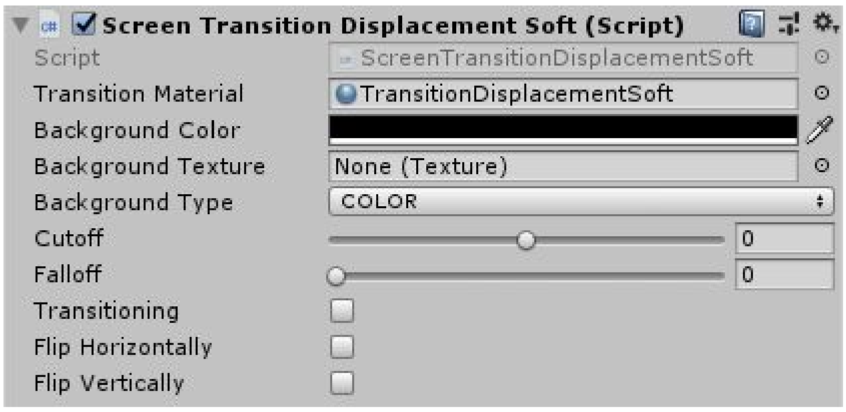
Displacement transition



The main difference between **Simple transition** and **Displacement transition** is that **Displacement transition** actually displaces pixels of the rendered image. The transition pattern is “pushing” the rendered image outside the screen. This transition is more performance-heavy and utilizes all 4 texture channels.

To use **Displacement transition** just attach ***ScreenTransitionDisplacement.cs*** or ***ScreenTransitionDisplacementSoft.cs*** script to your game camera.

Properties:



Screen Transition Displacement Soft (Script)

Script	ScreenTransitionDisplacementSoft
Transition Material	TransitionDisplacementSoft
Background Color	■
Background Texture	None (Texture)
Background Type	COLOR
Cutoff	0
Falloff	0
Transitioning	☐
Flip Horizontally	☐
Flip Vertically	☐

- **Transition Material** - material that will be applied to rendered image during transition.
- **Background Color** - background color that will be used during transition.
- **Background Texture** - texture that will be used as the background during transition (Render Textures allowed).
- **Background Type** - you can choose what kind of background you want to use for the transition.
    - **COLOR** - Background color will be used as the background.
    - **TEXTURE** - Background texture will be used as the background.
- **Cutoff** - represents current progress of the transition.

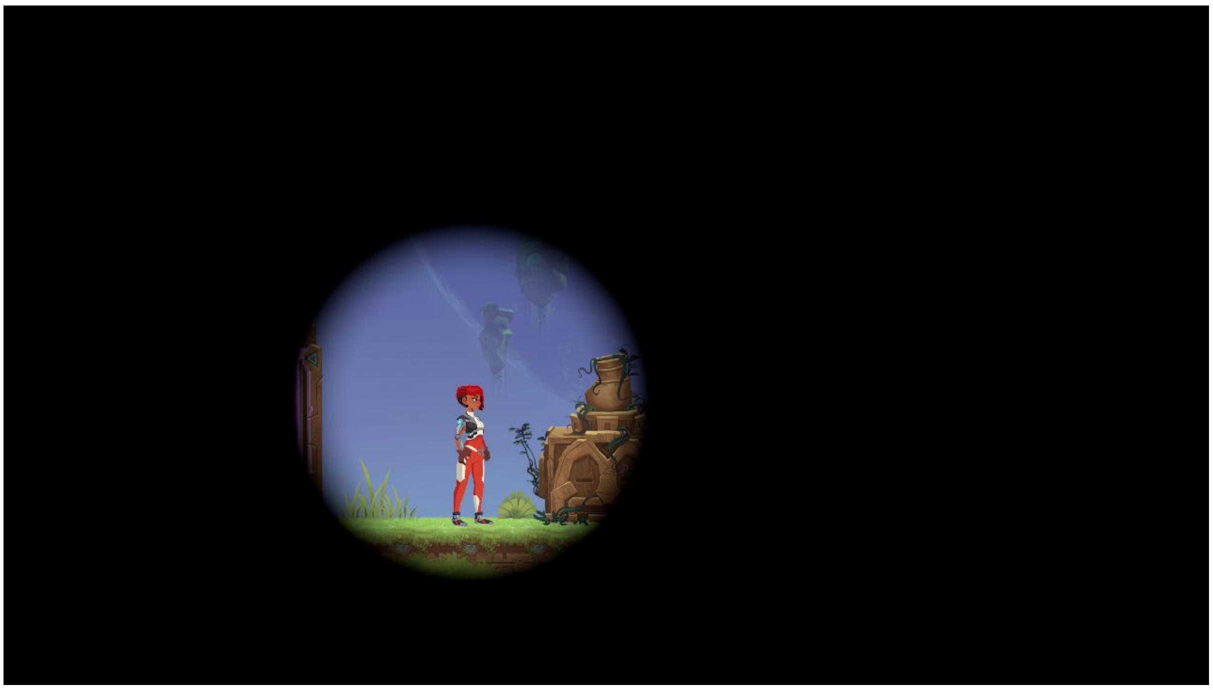
	No transition	Full transition
Normal	0	1
Soft	[-1; 0] - no transition (depends on the falloff size)	1

- **Falloff** - **FOR SOFT VERSION ONLY**. Smooth blend between rendered texture and background color. 0 - no blend (sharp border). 1 - max blend (size depends on the gradient in the *Blue channel* of the transition texture).
- **Transitioning** - flag that tells Unity to process transition. Set this flag at the beginning of the transition and unset it at the end to avoid unnecessary calculations and to save some performance.
- **Flip Horizontally** - set this flag if you want to flip transition texture horizontally.
- **Flip Vertically** - set this flag if you want to flip transition texture vertically.

Note that this kind of transition always fits transition texture into the screen. Otherwise it is possible that parts of the rendered image start moving before you see any transition (in case it starts outside of the screen).

This kind of transition also cannot be inverted due to technical limitations.

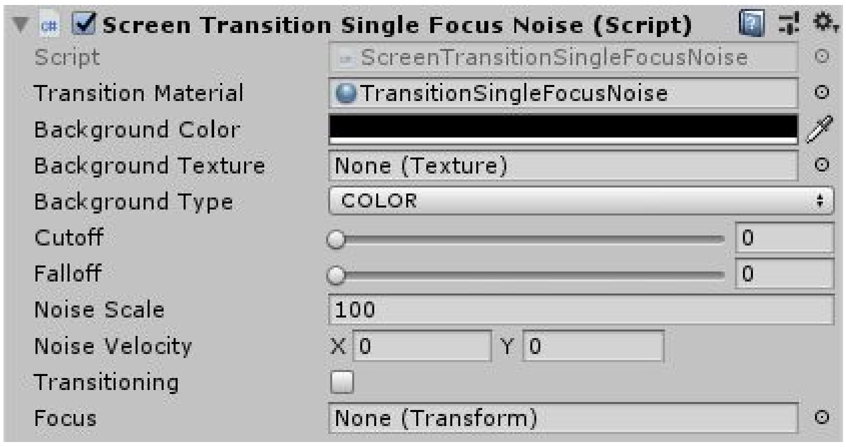
Single focus transition



This is the type of interactive screen transition that uses a scene object as the target (**Focus**). The transition represents the shrinking circular area. The center of this area will always match the position of the focus object on the screen.

To use **Single Focus** transition just attach ***ScreenTransitionSingleFocus.cs***, ***ScreenTransitionSingleFocusSoft.cs*** or ***ScreenTransitionSingleFocusNoise.cs*** script to your game camera.

Properties:



Screen Transition Single Focus Noise (Script)

Script	ScreenTransitionSingleFocusNoise
Transition Material	TransitionSingleFocusNoise
Background Color	■
Background Texture	None (Texture)
Background Type	COLOR
Cutoff	0
Falloff	0
Noise Scale	100
Noise Velocity	X 0 Y 0
Transitioning	☐
Focus	None (Transform)

- **Transition Material** - material that will be applied to rendered image during transition.
- **Background Color** - background color that will be used during transition.
- **Background Texture** - texture that will be used as the background during transition (Render Textures allowed).
- **Background Type** - you can choose what kind of background you want to use for the transition.
  - **COLOR** - Background color will be used as the background.
  - **TEXTURE** - Background texture will be used as the background.
- **Cutoff** - represents current progress of the transition.

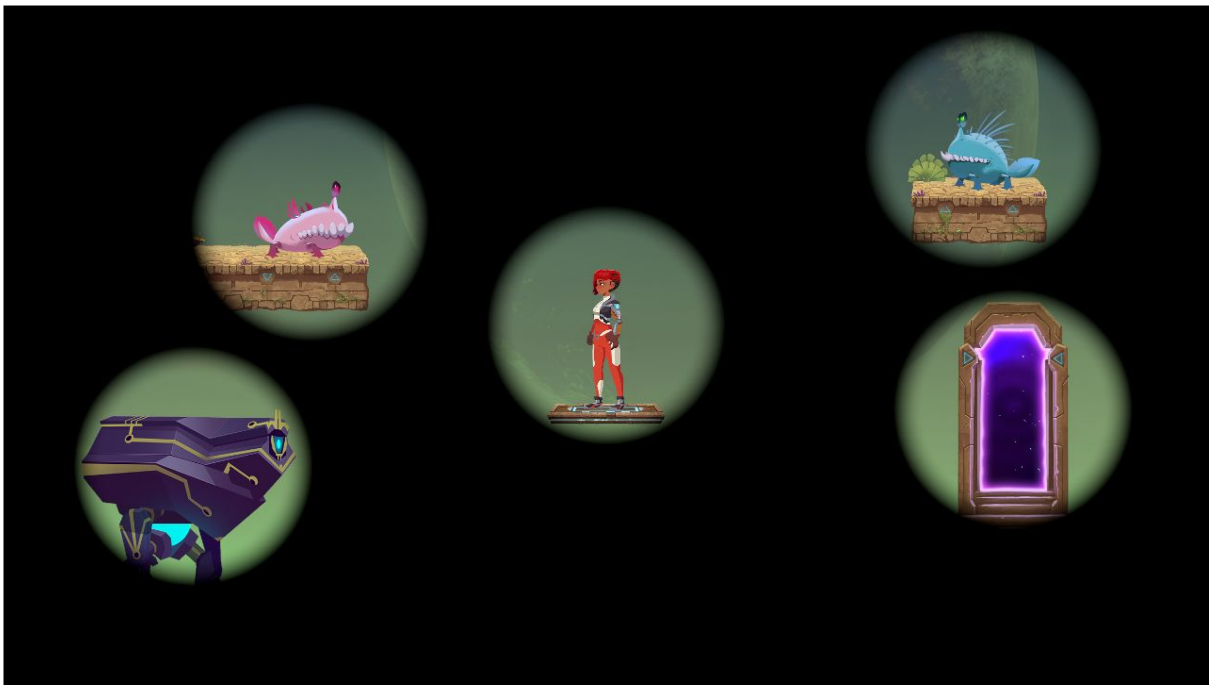
	No transition	Full transition
Normal	0	1.5
Soft	[0; 1] - no transition (depends on the falloff size)	2.5

- **Falloff** - **FOR SOFT VERSION ONLY**. Smooth blend between rendered texture and background color. 0 - no blend (sharp border). 1 - max blend.
- **Noise Scale - FOR NOISE VERSION ONLY**. Scale of the noise applied to the falloff. Values between 100 and 200 could be a good starting point.
- **Noise Velocity - FOR NOISE VERSION ONLY.** Horizontal and vertical speed of the noise applied to the falloff.
- **Transitioning** - flag that tells Unity to process transition. Set this flag at the beginning of the transition and unset it at the end to avoid unnecessary calculations and to save some performance.
- **Focus** - position of this *Transform* will be the center of the transition's circle.

When **Focus** object is *Null* or placed behind the game camera, the center of the screen will be used as the target instead.

This type of transition cannot be flipped or inverted.

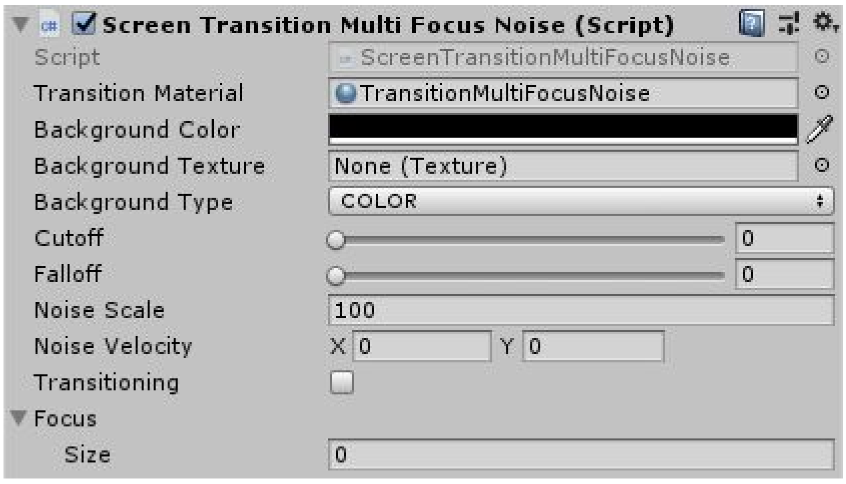
Multi focus transition



This is another interactive transition, but it can focus on up to 5 different objects on the game scene. Five circular areas will match positions of the focus objects on the screen.

To use **Multi Focus** transition just attach ***ScreenTransitionMultiFocus.cs***, ***ScreenTransitionMultiFocusSoft.cs*** or ***ScreenTransitionMultiFocusNoise.cs*** script to your game camera.

Properties:



Screen Transition Multi Focus Noise (Script)

Script	ScreenTransitionMultiFocusNoise
Transition Material	TransitionMultiFocusNoise
Background Color	⬛
Background Texture	None (Texture)
Background Type	COLOR
Cutoff	0
Falloff	0
Noise Scale	100
Noise Velocity	X 0 Y 0
Transitioning	☐
Focus	
Size	0

- **Transition Material** - material that will be applied to rendered image during transition.
- **Background Color** - background color that will be used during transition.
- **Background Texture** - texture that will be used as the background during transition (Render Textures allowed).
- **Background Type** - you can choose what kind of background you want to use for the transition.
    - **COLOR** - Background color will be used as the background.
    - **TEXTURE** - Background texture will be used as the background.
- **Cutoff** - represents current progress of the transition.

	No transition	Full transition
Normal	0	1.5
Soft	[0; 1] - no transition (depends on the falloff size)	2.5

- **Falloff** - **FOR SOFT VERSION ONLY**. Smooth blend between rendered texture and background color. 0 - no blend (sharp border). 1 - max blend.
- **Noise Scale - FOR NOISE VERSION ONLY**. Scale of the noise applied to the falloff. Values between 100 and 200 could be a good starting point.
- **Noise Velocity - FOR NOISE VERSION ONLY.** Horizontal and vertical speed of the noise applied to the falloff.
- **Transitioning** - flag that tells Unity to process transition. Set this flag at the beginning of the transition and unset it at the end to avoid unnecessary calculations and to save some performance.
- **Focus** - positions of the *Transforms* from this list will be used as the centers of the transition's circles.

This transition will take into account only the first 5 entries of the **Focus** array. All other entries will be ignored. Empty entries (Null references) will be ignored.

When all **Focus** objects are *Null* or placed behind the game camera, the center of the screen will be used as the target instead.

This type of transition cannot be flipped or inverted.

GETTING STARTED

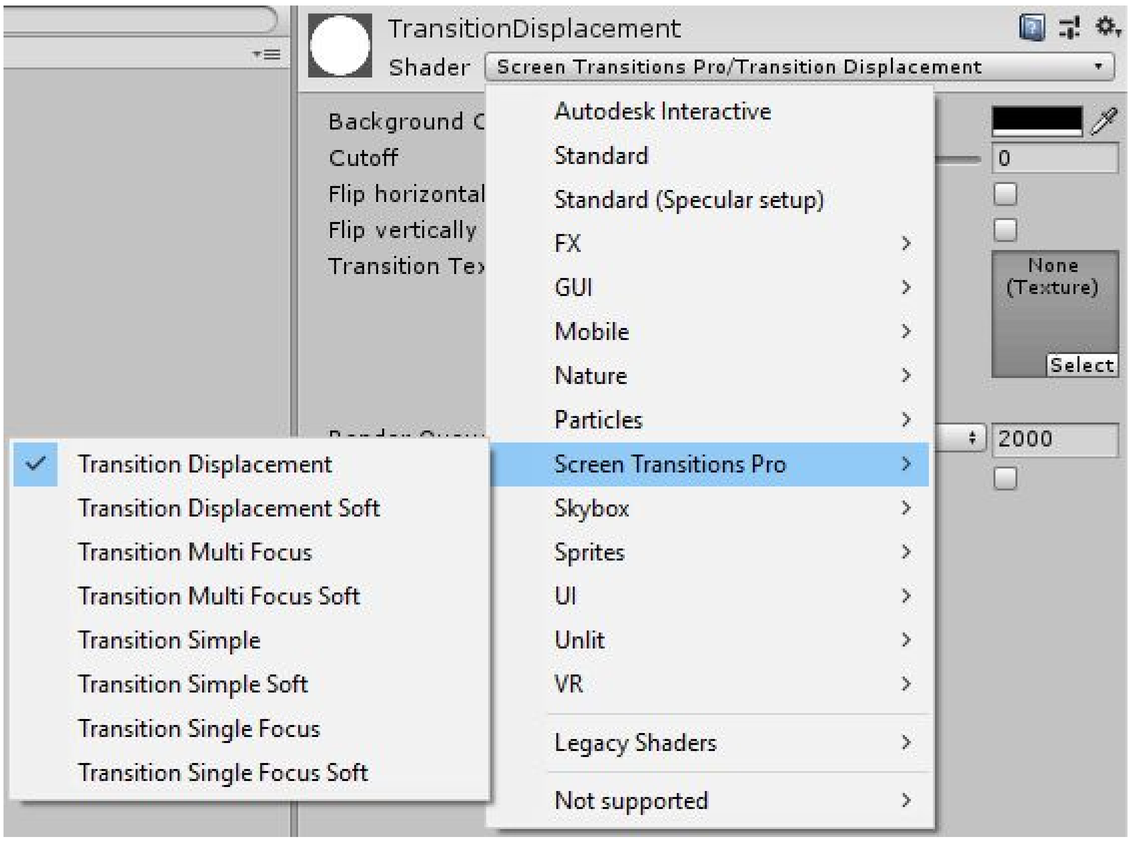
To start using screen transitions from this package you need to perform several simple steps.

1. Import and unpack package to your project and wait until Unity compiles all the scripts.

2. Choose the transition type and attach corresponding transition script from **/Scripts** folder to the game camera.

3. Get transition material. There are 3 ways to do so:

1. Duplicate example material from **/Materials** folder.
2. Right-click on the shader in **/Shaders** folder and select *Create* -> *Material* from context menu to create new material based on the selected shader.
3. Create new material and select **Screen Transitions Pro** shader from the drop-down menu.

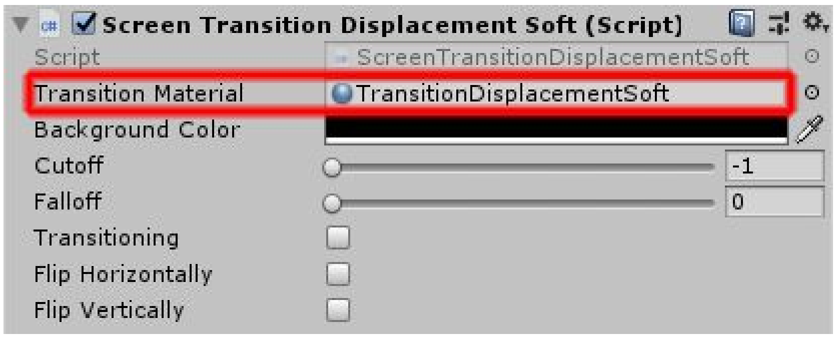

TransitionDisplacement
Shader: Screen Transitions Pro/Transition Displacement

IMPORTANT: to work correctly, shader and material types must match with the type of the screen transition script attached to game camera!

4. Set up the material from previous step.

- If you're working with **Simple** transition, assign transition texture from **/Textures/Simple** folder to the material.
- In case you're working with **Displacement** transition, assign transition texture from **/Textures/Displacement** folder to the material.

5. Assign material from previous steps as a **Transition Material** for the transition script attached to the game camera.

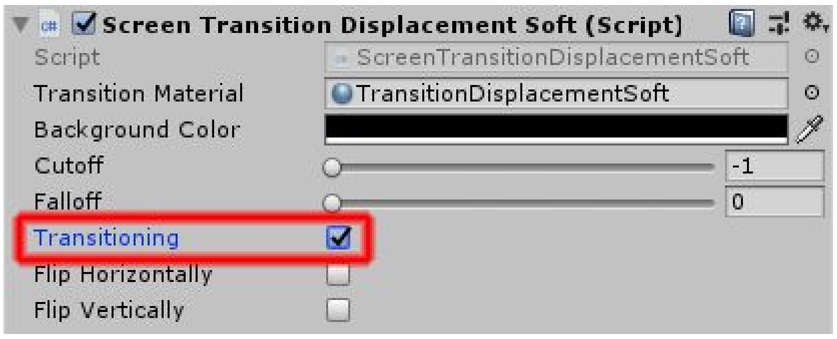


Screen Transition Displacement Soft (Script)

Script	ScreenTransitionDisplacementSoft
Transition Material	TransitionDisplacementSoft
Background Color	
Cutoff	-1
Falloff	0
Transitioning	☐
Flip Horizontally	☐
Flip Vertically	☐

6. If you're working with **Single Focus** or **Multi Focus** transitions, you can assign **Focus** targets in the transition script inspector or create and attach script that will assign them at runtime.

7. Set **Transitioning** flag to *True* in the transition script inspector.



Screen Transition Displacement Soft (Script)

Script	ScreenTransitionDisplacementSoft
Transition Material	TransitionDisplacementSoft
Background Color	
Cutoff	-1
Falloff	0
Transitioning	✓
Flip Horizontally	☐
Flip Vertically	☐

8. Move **Cutoff** slider back and forth between the minimum and maximum values. You should now see screen transition in your *Game View* (all screen transitions work in *Edit Mode*). If you don't see any transitions please check the Troubleshooting section of this document. Note: you can preview only transitions with background solid color. If you want to preview transition with background texture, you have to press "Play" button and move Cutoff slider in the Play Mode.

9. Move **Cutoff** slider to the extreme left position and unset **Transitioning** flag.

You screen transition is ready to use.
Now let's talk about how you can control and animate screen transitions.

HOW TO CONTROL TRANSITIONS

There are several ways to control screen transitions. The 2 main approaches are from code and using Animation Clips.

From code

To be able control screen transitions from code first you need to include namespace

```
using ScreenTransitionsPro;
```

to your script.

All properties of screen transition scripts are public, so you can access and modify them directly, like this:

```
ScreenTransitionSimple transition = GetComponent<ScreenTransitionSimple>();
transition.cutoff = 1f;
```

This implies that you know what type of transition you're dealing with. If you prefer a more generic approach, you can use **IScreenTransition** interface that provides you with a list of handy functions. All screen transition scripts implement this interface.

List of functions:

- SetTransitioning
- SetMaterial
- SetCutoff
- SetFalloff
- SetBackgroundColor
- SetBackgroundTexture
- SetFitToScreen
- SetHorizontalFlip
- SetVerticalFlip
- SetInvert
- AddFocus
- RemoveFocus
- SetNoiseScale
- SetNoiseVelocity

If you try to set an unsupported value (e.g. **Invert** to a **Displacement** transition) this value will be ignored and a Warning will appear in the Unity's console.

Now let’s take a look at 2 ways for controlling transition from code.

Update loop

It’s probably not the best way to do it, but it is quite simple and straightforward, so even beginners can implement it. We just assign transition values every frame. Something like this:

```
void Start()
{
    _transition = GetComponent<ScreenTransitionDisplacement>();
    _transition.transitioning = true;
}

void Update()
{
    _transition.cutoff = Mathf.Abs(Mathf.Sin(Time.time));
}
```

For more details please check the example scene located at
/DEMO/Scenes/DisplacementTransition_UpdateAnimation.unity
and the example script located at
/DEMO/Scripts/TransitionUpdateAnimation.cs

Coroutines

A better way to control screen transitions from code is to use coroutines.

```
void Start()
{
    _transition = GetComponent<ScreenTransitionSimple>();
}

void Update()
{
    if (Input.GetKeyDown(KeyCode.Space))
        StartCoroutine(FadeOut());
}

IEnumerator FadeOut()
{
    _transition.transitioning = true;
    while (_transition.cutoff < 1f)
    {
        _transition.cutoff += Time.deltaTime;
        yield return 0;
    }
}
```

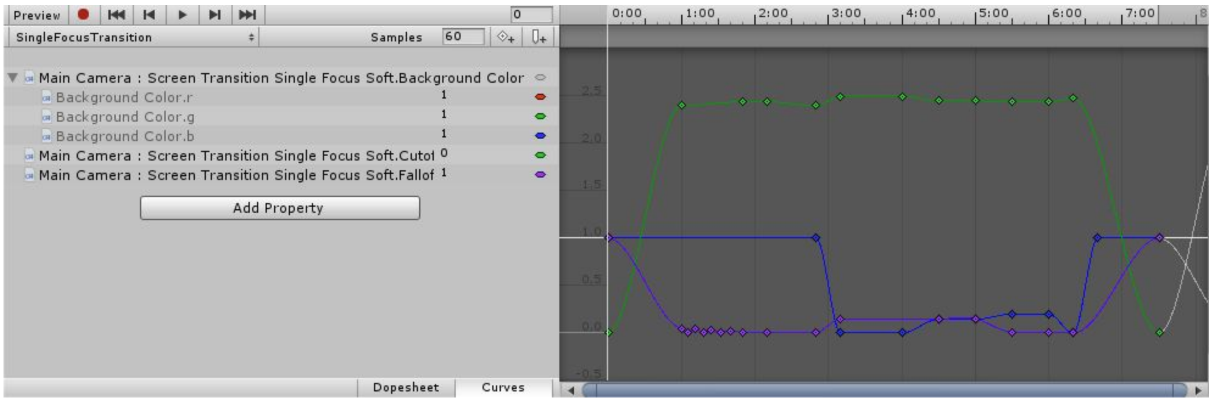
With this approach code looks much cleaner and more structured. Also it performs calculations only when needed and not every frame.

For more details please check the example scene located at
/DEMO/Scenes/SimpleTransition_CorourineAnimation.unity
and the example script located at
/DEMO/Scripts/TransitionCoroutineAnimation.cs

For more info about Coroutines please check Unity's official tutorials.

Animation Clip

Another great way to control screen transition is to use Animation Clips. By using animation curves you can animate any property of the transition and create not only linear or eased transitions, but more complex ones: shaky and wabbly animations, multi-staged transitions, flipping and inverting in the middle of the transition to create glitch effect and much more! You can also call Animation Events from the transition animation.



SingleFocusTransition Samples 60

Main Camera : Screen Transition Single Focus Soft.Background Color	
Background Color.r	1
Background Color.g	1
Background Color.b	1
Main Camera : Screen Transition Single Focus Soft.Cutof	0
Main Camera : Screen Transition Single Focus Soft.Fallof	1

For more details please check the example scenes located at
/DEMO/Scenes/MultiFocusTransition_AnimationClip.unity
and
/DEMO/Scenes/SingleFocusTransition_AnimationClipAdvanced.unity

This document is not intended to teach you how to work with Animation Clips or how to use Animation Panel. For more information about these topics please refer to the original Unity Tutorials.

Tween engines

Tween engines is a great way to create smooth eased animations from code. This fits perfectly with the screen transitions concept.

Screen Transitions Pro package contains examples of how you can control screen transition using 2 of the most popular tween engines from **Unity Asset Store**: iTween and DOTween.

iTween

To animate transition using iTween you need to use **iTween.ValueTo** function with callback that will set tweened value to the transition:

```
private IScreenTransition _transition;

void Start()
{
    _transition = GetComponent<IScreenTransition>();
}

void Update()
{
    if (Input.GetKeyDown(KeyCode.Space))
    {
        _transition.SetTransitioning(true);
        iTween.ValueTo(gameObject, iTween.Hash(
            "from", 0f,
            "to", 1f,
            "time", 1f,
            "onupdate", "SetCutoff",
            "easetype", iTween.EaseType.linear));
    }
}

void SetCutoff(float c)
{
    _transition.SetCutoff(c);
}
```

For more details please import package located at
/Integrations/iTweenExample.unitypackage

The example scene is located at
/DEMO/Scenes/Integrations/iTweenExample.unity
and the example script is located at
/DEMO/Scripts/TransitionITweenAnimation.cs

DOTween

To animate transition using DOTween you need to use **DOTween.To** function with callback that will set tweened value to the transition. Syntax is a bit more complicated than for iTween and involves lambda expressions. You will also need to store additional variable for tweened value:

```
private IScreenTransition _transition;
private float _cutoff = 0f;

void Start()
{
    _transition = GetComponent<IScreenTransition>();
}

void Update()
{
    if (Input.GetKeyDown(KeyCode.Space))
    {
        _transition.SetTransitioning(true);
        DOTween.To(() => _cutoff, x => _cutoff = x, 1f, 1f).OnUpdate(SetCutoff);
    }
}

void SetCutoff()
{
    _transition.SetCutoff(_cutoff);
}
```

For more details please import package located at
/Integrations/DOTweenExample.unitypackage

The example scene is located at
/DEMO/Scenes/Integrations/DOTweenExample.unity
and the example script is located at
/DEMO/Scripts/TransitionDOTweenAnimation.cs

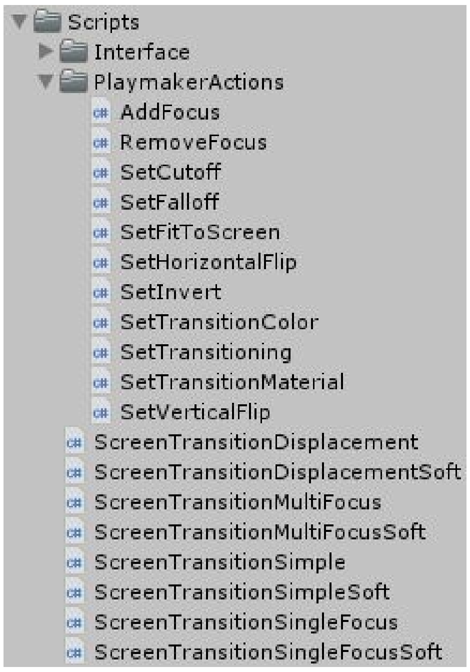
PLAYMAKER INTEGRATION

Screen Transition Pro is completely integrated with the Playmaker. Using custom actions from the integration package you will get the full control over screen transitions without writing a single line of code.

Installation

Make sure that you have Playmaker imported to your project before installing integration package.

Installation of the Playmaker integration package is pretty straightforward. Just unpack Unity package located at **/Integrations/PlaymakerIntegration.unitypackage** to your project and wait until Unity compiles all the scripts.
After installation you will see imported actions in you **/Scrips/PlaymakerActions** folder.
Playmaker will automatically detect new actions, you will be able to find them in Action Browser under **Screen Transitions Pro** header.

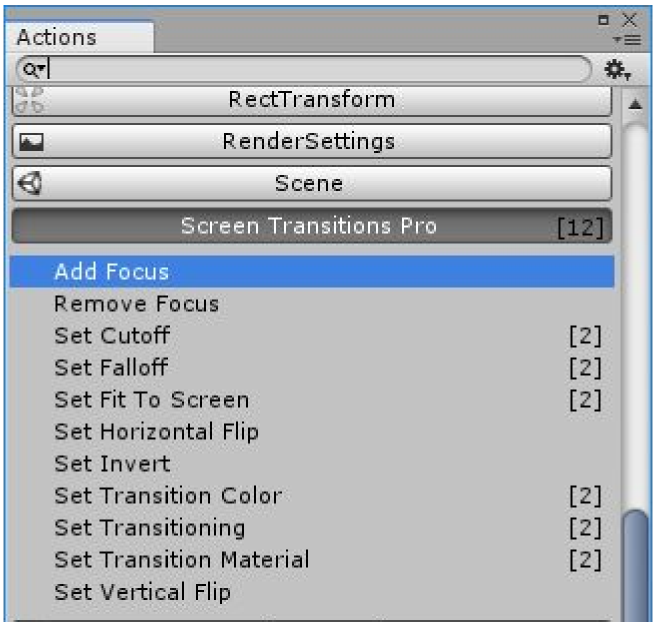


```
▼ Scripts
  ▶ Interface
  ▼ PlaymakerActions
      AddFocus
      RemoveFocus
      SetCutoff
      SetFalloff
      SetFitToScreen
      SetHorizontalFlip
      SetInvert
      SetTransitionColor
      SetTransitioning
      SetTransitionMaterial
      SetVerticalFlip
    ScreenTransitionDisplacement
    ScreenTransitionDisplacementSoft
    ScreenTransitionMultiFocus
    ScreenTransitionMultiFocusSoft
    ScreenTransitionSimple
    ScreenTransitionSimpleSoft
    ScreenTransitionSingleFocus
    ScreenTransitionSingleFocusSoft
```

Actions

Integration package contains following custom Playmaker actions:

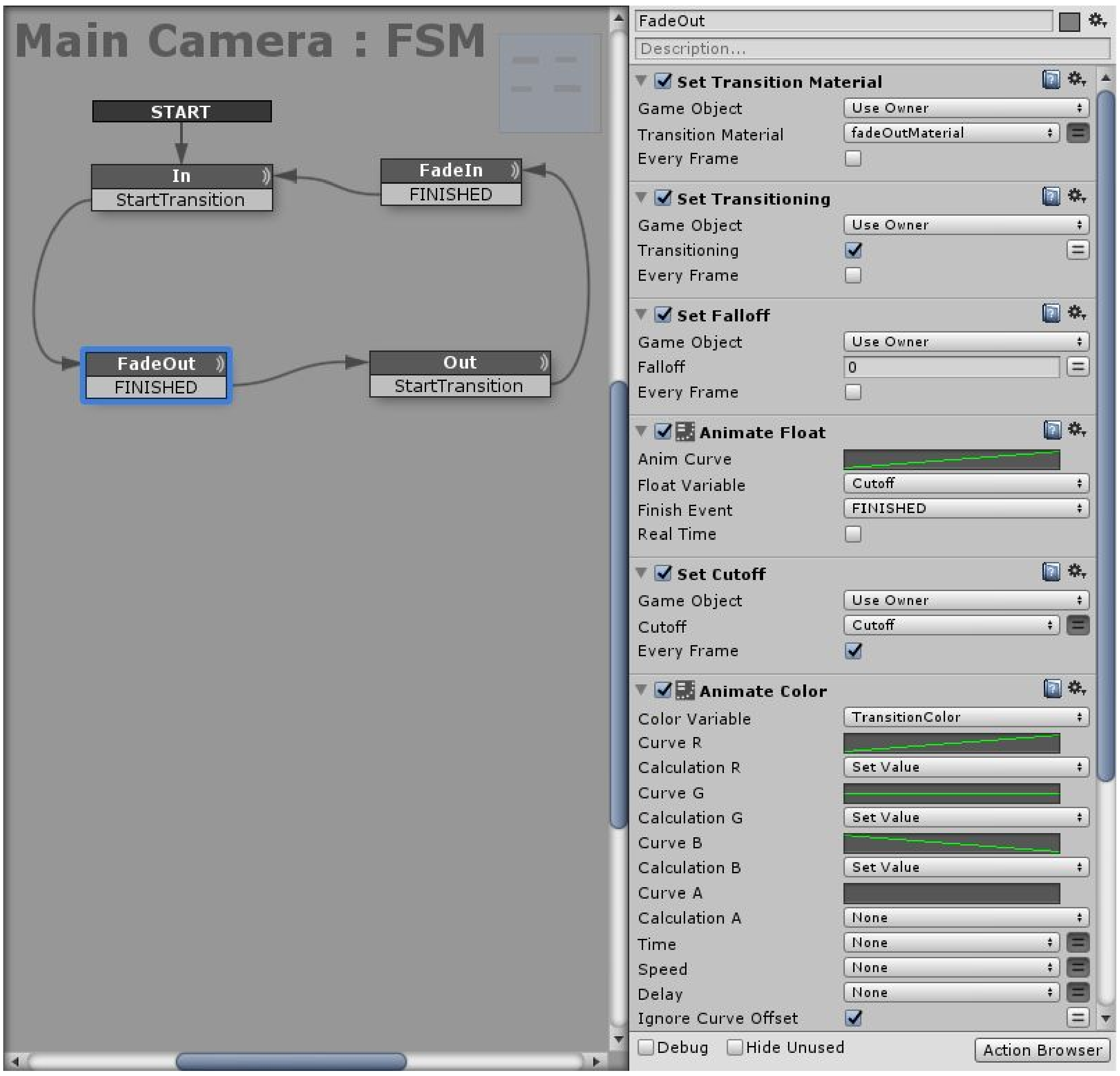
- Set Transition Material
- Set Transition Color
- Set Background Color
- Set Transitioning
- Set Cutoff
- Set Falloff
- Set Fit To Screen
- Set Horizontal Flip
- Set Vertical Flip
- Set Invert
- Add Focus
- Remove Focus
- Set Noise Scale
- Set Noise Velocity



```
Actions
RectTransform
RenderSettings
Scene
Screen Transitions Pro                 [12]
Add Focus
Remove Focus
Set Cutoff                             [2]
Set Falloff                            [2]
Set Fit To Screen                      [2]
Set Horizontal Flip
Set Invert
Set Transition Color                   [2]
Set Transitioning                      [2]
Set Transition Material                [2]
Set Vertical Flip
```

Example

Integration package also contains example scene with simple FSM attached to the Main Camera.



FadeOut

Description...

Set Transition Material

Game Object	Use Owner
Transition Material	fadeOutMaterial
Every Frame	☐

Set Transitioning

Game Object	Use Owner
Transitioning	✓
Every Frame	☐

Set Falloff

Game Object	Use Owner
Falloff	0
Every Frame	☐

Animate Float

Anim Curve	
Float Variable	Cutoff
Finish Event	FINISHED
Real Time	☐

Set Cutoff

Game Object	Use Owner
Cutoff	Cutoff
Every Frame	✓

Animate Color

Color Variable	TransitionColor
Curve R	
Calculation R	Set Value
Curve G	
Calculation G	Set Value
Curve B	
Calculation B	Set Value
Curve A	
Calculation A	None
Time	None
Speed	None
Delay	None
Ignore Curve Offset	✓

For more details please check example scene located at **/DEMO/Scenes/Integrations/PlaymakerExample.unity**

TROUBLESHOOTING

I move Cutoff slider but nothing happens.
Check to see if Transitioning flag is set. Without it the transition won't happen.

I have Transitioning flag set, but still nothing happens.
Check to see if you provided the correct Transition Material to screen transition script.

Transition Material is here, Transitioning flag is set, still no transition.
Check to see if Transition material has correct transition texture assigned.

Transition texture, material and flag are all set, still no transition!
Make sure that your target platform supports image effects.

Everything is set, image effects are supported, still no transition!!!
Check to see if you use HD or Lightweight Scriptable Render Pipeline. Current version of this package is not compatible with them.

If you still have an issue, please send me your scene setup and I will try to figure it out.

Error 'The type or namespace name 'ScreenTransitionWhatever' could not be found'
Make sure that you added the following line

```
using ScreenTransitionsPro;
```

at the beginning of your script.

Smooth falloff doesn't look smooth at all.
Smooth falloff is based on the Blue channel of the transition texture along with Cutoff, so on some very stretched gradients (like spirals) you might see actual steps of falloff gradient, because standard textures has only 256 shades per channel. The solution here is to not use smooth falloff on long gradients or to play transition quite fast so it will be barely noticeable.

Texture background does not appear in Edit mode.
Yes, it's true. In order to tell shader to use a texture instead of a solid color you have to press "Play" button in the Editor (command is sent to shader in the Start() function). Sorry about that.

UI and HUD displayed on top of screen transition.

Set your UI Canvas Render Mode to “*Screen Space - Camera*” and assign to Render Camera the same game camera that you use for the transition. Unfortunately, there is no way to fix it for Overlay canvases. This is how Unity works; Overlay canvas is always drawn after any image effects.

Screen transition is affected by other image effects.
If you have multiple image effects attached to the same game camera, make sure that the screen transition script is the last script in the inspector stack (at the bottom of the list, after all other image effect scripts).

CONTACT ME

If you have any questions or issues regarding **Screen Transitions Pro** feel free to contact me at **andrey.torchinsky[at]gmail.com** or post it on the dedicated forum thread.

I’m constantly working to add new transition patterns to the package. If you have a specific request send me a ref to the desired transition (video or gif would be perfect, but a simple image also works) and I will try to find a way to implement it in **Screen Transitions Pro**. All future updates will be free of charge.

I would also really appreciate if you could take a moment to send me examples of how you integrated and used **Screen Transition Pro** in your project. I might give it shout out in my Twitter.

The last thing that I would like to ask is that you rate this package in the Unity Asset Store and leave a comment with your experience. If you have a negative experience, please contact me first and I will do my best to solve the issue.

Thank you for purchasing **ScreenTransition Pro**, I hope it will be useful for you.

Best,
Andrey Torchinsky

CHANGE LOG

v1.3

- Added ability to set a texture instead of a solid color as a transition background
- Added Render Texture support
- Added 1 new demo scene that shows how to use Render Texture
- Added 1 new Playmaker action

v1.2

- Added **Noise** variation for *Single Focus* and *Multi Focus* transitions
- Added 1 new demo scene
- Added 2 new Playmaker actions
- Updated 3rd party integration packages

v1.1

- Added 1 new transition type - **Fade**
- Added 1 new demo scene
- Added 5 new **Simple** transitions
- Added 3 new **Displacement** transitions
- Added Tooltips in the Inspector

v1.0

Original version