## What is Screen Fader?

This easy-to-use script will allow you to fade in or out your screen with only one line of code, so all you need is to write:

```
Fader.Instance.FadeIn();
```

But on the other hand, you will get powerful possibilities. You can setup colors, transparency, speed of effect and delays before it starts in the Inspector panel.

You can subscribe on events and get notifications when effects will be started or completed.

This works well on Free and Pro Unity, suitable for Web, Standalone, Android and iOS platform. And all this takes less then 50kb on your drive and costs less then your morning starbucks coffee.

And one more thing: you'll get 2 extra scripts that will allow you to fade your screen with squares or stripes effect, and of course you can also setup their additional parameters, such as number of stripes or squares and direction of effect.

## Quick Start

As mentioned above, to fade-in screen, open you script and type JUST ONE LINE OF MAGIC CODE when you want to fade your screen in:

```
Fader.Instance.FadeIn();
```

That is all. Now you can try to make different combinations of it. For example - fade-in, than pause for 10 seconds, and fade-out after this:

```
Fader.Instance.FadeIn().Pause(10).FadeOut();
```

or - fade-in, call coroutine method, and fade-out when conoutine method will be breaked:

```
Fader.Instance.FadeIn().StartCoroutine(this, MoveCamera()).FadeOut();
```

or somthing more complicacy:

```
Fader.Instance
      .FadeIn(0)
      .StartCoroutine(this, LoadPlayerInfoFromWebServer())
      .StartCoroutine(this, SetupPlayerCharacter())
      .StartCoroutine(this, ShowGUI())
      .StartAction(new PlayBackgroungMusic(), "ambience.mp3", defaultVolume)
      .FadeOut(2);
```

# API Help

## Table of content

- FadeIn
- FadeOut
- Pause
- Flash
- LoadLevel
- SetColor
- StartAction
- StartCoroutine
- StopAllFadings

### FadeIn( time )

Fade-in screen in 'time' seconds. 'time' argument is optional, default value - 1.

```
Fader.Instance.FadeIn();
Fader.Instance.FadeIn( 3 ); /// Fade screen in in 3 seconds
```

### FadeIn( GameObject, time )

Fade-in gameobject. GameObject's material should support transparency. 'time' argument is optional, default value - 1.

```
/// Hide characterGO in 3 seconds
Fader.Instance.FadeIn( characterGO, 3 );
/// Change character’ hemlet
Fader.Instance.FadeIn( HemletOfWarion ).FadeOut( HemletOfKing );
```

### FadeOut( time )

Fade screen out in 'time' seconds. 'time' argument is optional, default value - 1.

```
Fader.Instance.FadeOut();
Fader.Instance.FadeOut( 3 ); /// Fade screen out in 3 seconds
```

### FadeOut( GameObject, time )

Fade gameobject out. GameObject's material should support transparency. 'time' argument is optional, default value - 1.

```
/// Hide wall and show Enemy Boss character.
Fader.Instance.FadeOut( Wall ).FadeIn( EnemyBoss );
```

## Pause( time )

Make pause in chain of fadings or actions.

```
Fader.Instance.FadeIn().Pause().FadeOut();    /// pause 1 second
Fader.Instance.FadeIn().Pause( 3 ).FadeOut(); /// pause 3 second
```

## Flash( inTime, outTime )

Make flash - quick fade-in and fade-out. Arguments inTime and outTime are optional. Default values are inTime = 0.075, outTime = 0.15

```
Fader.Instance.Flash();
Fader.Instance.Flash( 0.1, 3 ); /// Quickly fade screen in and slowly out
```

## LoadLevel( name )

Load new scene by its name

```
Fader.Instance.LoadLevel( "IntroScene" );
Fader.Instance.FadeIn().LoadLevel( "Scene-1" ).FadeOut();
```

## LoadLevel( index )

Load new scene by its index

```
Fader.Instance.LoadLevel( 1 );
Fader.Instance.FadeIn().LoadLevel( 1 ).FadeOut();
```

## SetColor( color )

Default color of fading is black, but this method allow changing of color at runtime.

```
Fader.Instance.SetColor( Color.white ).FadeIn().LoadLevel( 0 ).FadeOut();
Fader.Instance.SetColor( Color.red ).Flash(); /// Damage - red flash
```

## StartAction( IAction )

Execute a custom action. Custom action is an instance of class implementing IAction interface. Here is an example:

```
HidePrincessAction action = new HidePrincessAction();
Fader.Instance.StartAction( action );

/// Hide princess while screen is faded, and then fade screen out.
Fader.Instance.FadeIn().StartAction( action ).FadeOut();
```

IAction interface, defines Execute() method that will be called and Completed property, that you should set to 'true' when action will completed:

```
public interface IAction
{
    bool Completed { get; set; }
    void Execute();
}

public interface HidePrincessAction: IAction
{
    public bool Completed { get; set; }
    public void Execute()
    {
        GameObject.Find("MarioPrincess").GetComponent<Renderer>().enabled = false;

         /// It lets screen fader know that action is completed
         /// and next action could started.
        this.Completed = true;
    }
}
```

### StartAction( IParametrizedAction, object[] )

Execute a custom action that accept some parameters. Here, custom action is an instance of class implementing IParametrizedAction interface. Let's improve our example:

```
ChangeCharacterAction param_action = new ChangeCharacterAction();
/// Hide Mario and show Princess
Fader.Instance.StartAction( param_action, "Mario", "Princess" );
/// Hide Princess and show Mario
Fader.Instance.StartAction( param_action, "Princess", "Mario" );
```

IParametrizedAction's Execute() method can get any number of parameters. See the implementation of ChangeCharacterAction:

```
public interface ChangeCharacterAction: IParametrizedAction
{
    public bool Completed { get; set; }
    public void Execute( params object[] args )
    {
        GameObject.Find( args[0] ).GetComponent<Renderer>().enabled = false;
        GameObject.Find( args[1] ).GetComponent<Renderer>().enabled = true;

         /// It lets screen fader know that action is completed
         /// and next action can be started.
        this.Completed = true;
    }
}
```

### StartCoroutine( MonoBehaviour, Coroutine )

Start coroutine method in series. Next fading tasks will be starter after coroutine finish. MonoBehaviour is an game object where coroutine ( IEnumerator method ) located. NOTE: This Conoutine's methods have to have 'yield break' operator!

```
/// Show 'Game Over' for 5 sec and fade-in screen.
Fader.Instance.StartCoroutine( this, ShowGameOver() ).Pause( 5 ).FadeIn();

...

IEnumerator ShowGameOver()
{
    text.text = "Game Over";
```

```
        yield return new WaitForSeconds( 5 );
    }
```

## StartCoroutine( MonoBehaviour, methodName, methodParameter )

Start coroutine method in parallel and pass parameter to it. Next fading tasks will be starter right after coroutine was started.

```
    /// Show 'You Game is Over, Mario' text for 5 sec and fade-in screen.
    Fader.Instance.StartCoroutine( this, "ShowGameOver", "Mario" ).Pause( 5 ).FadeIn();

    ...

    IEnumerator ShowText( string name )
    {
        text.text = "You Game is Over, " + name;
        yield return new WaitForSeconds( 5 );
    }
```

## StopAllFadings()

Brakes all fadings immediately.

```
    Fader.Instance.StopAllFadings();
```