Getting Started Guide

Contents

1. Download

The Flixel Power Tools are a set of plugins for Flixel 2.5 that provide extra functionality and features for your games.

Ensure you are using the latest version of **Flixel** by downloading it from:

https://github.com/AdamAtomic/flixel

Then make sure you have the latest version of the Flixel Power Tools:

https://github.com/photonstorm/Flixel-Power-Tools

They are both updated constantly, so if this file is more than 6 months out of date you would be strongly advised to see if there are any upgrades!

If you don't have git installed locally use the big "Download" button on the top-right of the github pages to grab a zip of the source instead.

2. Installation

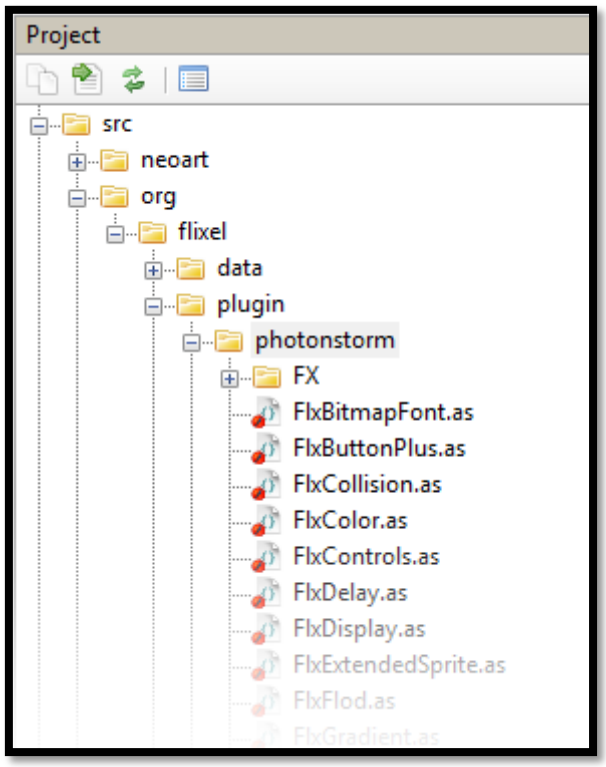
Note: You must be using Flixel 2.5 or above.

Unzip the contents of the *photonstorm.zip* archive into the following location:

```
src/org/flixel/plugin
```

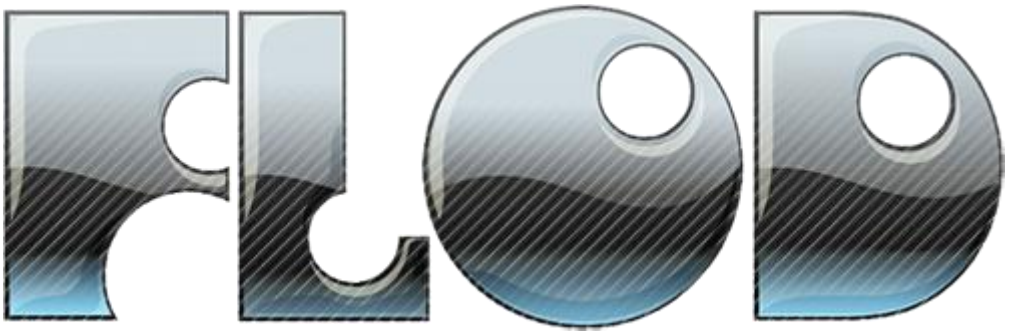
Where `src` is the root source folder of your Flixel project.

This should create a folder called *photonstorm* inside the flixel *plugin* folder. The final structure will look this:



This screen grab is taken from within FlashDevelop, but the structure should be identical no matter what you view it in (Flash Builder, Windows Explorer, etc)

3. Flod - AS3 Amiga Soundtracker Replay Class



The Flixel Power Tools include support for integrating the Flod replay library directly with Flixel, so the music is controlled by the native Flixel volume changing and mute.

You can download Flod from:

http://www.photonstorm.com/flod

There is also a Getting Started guide on the same page.

Be sure to check out the license agreement before using it in your games.

Once downloaded copy the folder `neoart` into your `src` folder.

If you have downloaded the full Flixel Power Tools then you will find this folder here:

```
Flixel-Power-Tools\Test Suite\src\neoart
```

4. How to use the Flixel Power Tools

Note: Always make sure you import the plugin at the top of your class:

```
import org.flixel.plugin.photonstorm.*;
```

In this example we'll use the **Health Bar** tool to create a floating health bar above our player sprite. The Health Bar works by hooking to the health value of an FlxSprite, and as the sprite health is updated (via a call to FlxSprite::hit or directly by changing FlxSprite.health) the bar is visually updated in real-time.

The following code is put directly into your FlxState, but it could also go into an FlxGroup.

First create a local variable for the health bar:

```
private var playerHealth:FlxHealthBar;
```

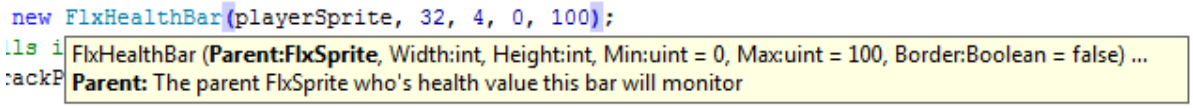
Then in the *create* function instantiate it:

```
playerHealth = new FlxHealthBar(playerSprite, 32, 4);
```

The parameters above are:

- playerSprite - the FlxSprite the Health Bar is linked to
- 32 - the width in pixels of the Health Bar
- 4 - the height in pixels of the Health Bar

The important part to remember is that if you have installed the plugins correctly then most AS3 editors will provide you with context-sensitive tips as you type the parameters. This means as you type you'll see something like:



All classes and functions in the Flixel Power Tools have documentation built into them, so they will populate context-sensitive help.

Hint: In FlashDevelop If you ever want to bring the floating help window back-up again just make sure you are inside the braces ( ) and press CTRL+SHIFT+SPACE together.

With the Health Bar created we will tell it to track the player sprite:

```
playerHealth.trackParent(0, -5);
```

The parameters are the x and y offsets (in pixels) from the top left of the sprite it is tracking. So in this case it will be -5 pixels above the sprite. Tracking means that as the sprite moves around the game world, so the health bar will follow it, sticking loyally to the sprites location. This is of course optional, the Health Bar can also work in a HUD panel, not tracked to the coordinates of the sprite at all. You'll find that lots of the Power Tools are flexible like this.

trackParent() is just one of the functions available in the FlxHealthBar class, but there are many more. The best way to learn is by looking at the code in the **Test Suite** and by opening up the `FlxHealthBar.as` file and looking at the functions inside it, and reading the documentation in there.

Finally we add the health bar to the display so it's rendered:

```
add(playerHealth);
```

Health Bars are just extended FlxSprites. So anything you can do normally with a FlxSprite, such as collision, rotation, scaling, alpha, overlap checks, etc you can do with the Health Bar. This technique is used a lot across the Power Tools - and where possible if the tool can create FlxSprites then it does so.

This example may be focused just around the Health Bar, but the majority of tools work in the same way.

5. The Test Suite

The Flixel Power Tools come with a comprehensive Test Suite. This was created as a way to visually demonstrate how the different tools work, and also as a means for you to learn from - you can open up the source code for each test, and quickly see how the tool is used. The tests have comments in the source to guide you through what is going on.

You can find the Test Suite in the `Test Suite` folder. Inside you'll see a FlashDevelop project, an `assets` folder that contains all the graphics used in the tests, a `bin` folder where the Test Suite SWF is, a `lib` folder containing the TweenMax SWC library, a `psds` folder containing PSD versions of some assets and finally the `src` folder.

If you've never seen the Test Suite before then open the `FlixelPowerTools.swf` file that is in the bin folder. If you've got Flash Player installed locally (and you should!) then it will open into that and run. If you don't then open `index.html` into a web browser.

1. It's all in the src

Inside the `src` folder is a folder called `tests`.

The tests folder is broken down into sub-folders, one per tool, and contain one or more test classes.

So if you want to learn how to use the Bitmap Fonts then open:

```
src/tests/bitmapfont/
```

.. and you'll find several tests which show exactly how to do that.

If you have FlashDevelop installed then you are strongly recommend to open the `Flixel Power Tools Test Suite` FlashDevelop project. You can then see all of the source, tweak something, compile it and see the results instantly.

2. Compiling with FlashBuilder

If you are using FlashBuilder you can re-create the FlashDevelop project by pointing it to the `src` folder. The only extra step you need to take is to ensure that the SWCs inside the `lib` folder are linked into your project.

6. Have fun!

I created the Flixel Power Tools to make my game-making life easier. Hopefully they will do the same for you, allowing you to focus on the game content itself, rather than spending time re-inventing the wheel.

1. Bugs?

If you find a bug please ensure you are using the latest version of the class, and then report it on the github issues tracker at:

```
https://github.com/photonstorm/Flixel-Power-Tools/issues
```

2. Support

Support is available via the official Flixel forums. When posting be sure to say which version of the class you are using (every Power Tool has a version number at the top of the source code)

```
http://flixel.org/forums/
```

3. Contact

If you would like to help contribute to the Power Tools then please get in touch. Or if you've used them in a game you've made then I would love to see it!

You can email me:

Richard Davey

rdavey@gmail.com

But most of all - have fun :)