Getting Started Guide

Flixel is a well known open source and free game engine created by Adam "Atomic" Saltsman. Flixel has been used in hundreds of games, including IGF nominees, Adult Swim games, and avant-garde experiments.

The Flixel Power Tools are a package of classes designed to provide extra functionality to your Flixel games. Originally created by Richard Davey (aka Photon Storm), this is the **Flixel Community** fork of the library, striving to keep the library living and up to date with the latest version of Flixel.

This version of Flixel Power Tools also uses the Flixel Community fork of Flixel, but is also compatible with Adam Atomic's original Flixel version 2.5 and up.

This guide will help you to get up and running with Flixel Power Tools.

Contents

1. Download

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

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

https://github.com/FlixelCommunity/flixel

Or you can download the original version of Flixel by Adam Atomic:

https://github.com/AdamAtomic/flixel

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

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They are both updated constantly, so if this file is more than 6 months out of date you would be strongly advised to see if there are any upgrades!

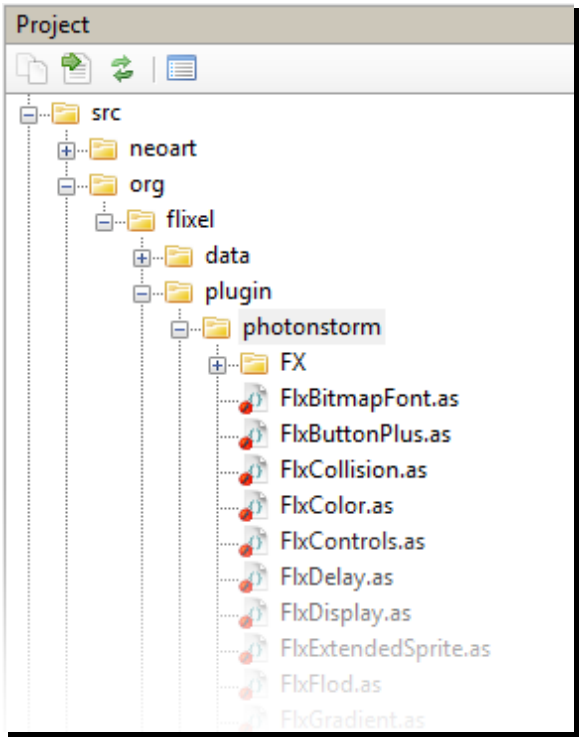
If you don't have git installed locally use the big "Download" button on the top-right of the github pages to grab a zip of the source instead.

2. Installation

Note: You must be using Flixel 2.5 or above.

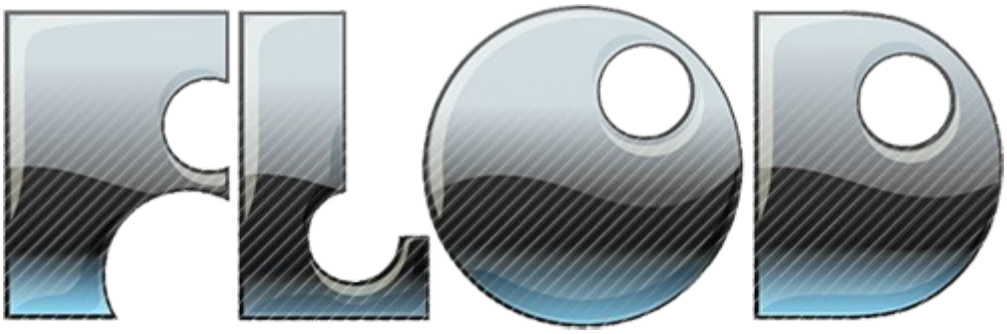
Just save the SWC (found in the bin folder) to your project's lib folder. The only extra step you need to take is to ensure that the SWCs inside the lib folder are linked into your project. Don't forget to also include either the source (or the SWC) for Flixel!

Or, if you prefer to use the source, copy the src folder from Flixel Power Tools into your own project's src folder. 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.

A SWC of the Flod library has been included in the lib folder, but again, if you prefer to work with the source, you can download it and copy it to the folder neoart inside your src folder.

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 **FlxBar** tool to create a floating health bar above our player sprite. The FlxBar 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 healthBar:FlxBar;
```

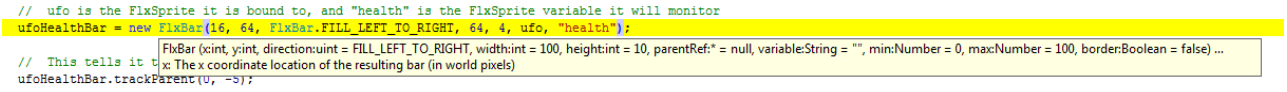
Then in the *create* function instantiate it:

```
healthBar = new FlxBar(16, 64, FlxBar.FILL_LEFT_TO_RIGHT, 64, 4, ufo, "health");
```

The parameters above are:

- 16 x 64 are the x/y coordinates
- FlxBar.FILL_LEFT_TO_RIGHT is the direction it will fill
- 64 x 4 is the visual width and height of the bar
- ufo is the FlxSprite it is bound to
- "health" is the value within the ufo FlxSprite it will track

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:

```
healthBar.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 FlxBar.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(healthBar);
```

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 full of demos, examples and mini-games. 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 all the helper libraries that Flixel Power Tools uses, 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. 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/FlixelCommunity/Flixel-Power-Tools/issues

7. 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/

8. Contributing

Flixel Power Tools is continously being worked on, including fixing old bugs and adding new features. Check us out on GitHub to see if there is anything you can do to help:

https://github.com/FlixelCommunity/flixel/blob/master/CONTRIBUTING.md

9. Contact

If there is anything else we can help you with Flixel Power Tools, then please get in touch. Or if you've used Flixel Power Tools in a game you've made then we would love to see it!

You can email Richard Davey (the original creator of Flixel Power Tools):

rdavey@gmail.com

Or you can contact one of us who maintain the Flixel Community fork:

contact@flixelcommunity.org

10. 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.

But most of all - have fun :)