PhotonBlog

Flash Game Dev Tip #6 – Setting-up FlashDevelop and Flixel for the first time

22nd Mar 2011 24

By Rich

Tags
flashdevelop, flex sdk, flixel, noobs

Posted in
Flash Game Dev Tips

24 comments

Print / PDF

Tip #6 – Setting-up FlashDevelop and Flixel for the first time

Last updated April 14th 2011 for FlashDevelop version: 3.3.4 and Flixel version: 2.5

Hello World

The popularity of Flixel attracts a lot of new developers. Often they are coming into it not from Flash, but from other languages and game making tools. There's nothing wrong with this, but it's all too easy to fall at the first hurdle: simply getting a clean working development environment set-up. One which you can then hit "compile" and see the all-magic "Hello World" appear, rather than a blank white window or an error message.

There are lots of tutorials on setting-up FlashDevelop (see "Further Reading" for some), and a few on getting started with Flixel. But they are mostly out of date, and none of them marry the two things together. So I hope to address that in this tip. I'll keep this tip updated as major new versions of FlashDevelop and Flixel are released.

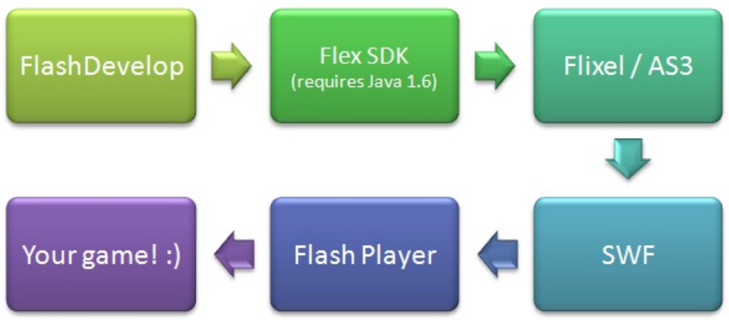
OS X dudes, take a different path

First things first. You need to be using Windows. FlashDevelop IS coming for OS X soon. You can also run it under Parallels. And there are builds in development that you can download from the forum, but I'm not covering it here. Sorry if that excludes you. You're welcome to come back and join us at the point where we integrate Flixel, as that will still be relevant to you.

Windows people, let's continue …

The Development Process

It's important you understand what each part of your development environment does. It's easy to get mixed-up with terminology, and not really knowing where in the chain a certain piece fits. The following diagram shows the build process:



FlashDevelop is our editor, our IDE. It's where you'll type in all the code, and spend the majority of your time.

When it comes time to test your code, FlashDevelop will run a program called *mxmlc.exe* which is part of the **Flex SDK**. It's the job of this program to grab everything it needs, and fire it off to the compiler. The main compiler is written in Java, hence the requirement for Java 1.6 to be installed on your PC.

At this point it's very easy to get confused by the word "Flex" or even the involvement of Java. Traditionally Flex has been associated with Flex Builder, the Adobe application used for creating mostly dull and dry… sorry, for creating *rich* internet applications. Flex Builder uses MXML files to control UI layout amongst other

search

GET LATEST NEWS & STUFF
Photon Storm Newsletter
Enter your email address sign up

Amiga MOD & XM Replay in AS3 / JS by Christian Corti

YOU CAN PLAY DROPLET HERE

Licence our games for your website on FlashGameLicense.com

OurGames
Chickaboom
Cat Astro Phi
Droplets
Kyobi
Quartet
Starfish

Our Content
ActionScript3
- 3D
- Experiments
- Flixel
- PixelBlitz
- Tutorials

Art
- Animation
- Pixel Art
- Tutorials

Cool Links
Demoscene
Flash Game Dev Tips
Game Development
- Book Reviews
- Business

things. Adobe since renamed it to FlashBuilder (I guess they get a kick out of confusing people). But you don't need to care about this. All you need to know is that the **Flex SDK** contains the compiler, and that is what FlashDevelop uses. As for the presence of Java in this equation: the compiler is written in Java. But it doesn't *create* java apps.

What the compiler does is to take your **Flixel / ActionScript 3 source code** and process it.

Providing there were no errors, the compiler will have turned your code into machine code, and output a **SWF file**.

SWF files need to be loaded into **Flash Player** to be viewed. Most people are familiar with experiencing flash within their web browser, but you can (and will!) also download a stand-alone Flash Player to run SWF files from Windows, without needing a browser.

Once your SWF is loaded into Flash Player, your game will begin!

This whole process is repeated over and over during development. So let's create this environment locally, and put together a set of flixel template files that will allow you to roll-out future projects quickly and easily.

Downloading spree

Java 1.6

As we've seen above the Flex SDK requires Java 1.6+ in order to compile. This isn't standard with Windows, so unless you are a Java developer and have a Java IDE installed such as Eclipse or Netbeans, you'll need to download and install Java first.

Go here: http://www.oracle.com/technetwork/java/javase/downloads/index.html

You need **Java Platform SE** (SE standards for "Standard Edition"). The current release is Java SE 6 Update 24, but any newer version should work just as well. Download the JRE (Java Runtime Environment) version. Get the 32-bit version, even if you run a 64-bit version of Windows. Install it and disable all of the annoying Sun Java update alerts.

FlashDevelop

Download and install FlashDevelop. I'm not going to screen shot the whole process, because to be fair if you can't click "Next" a couple of times then you're probably not yet ready to be coding games. The only important thing to note is this: **When FlashDevelop asks you if you want to download and install the Flex SDK – SAY YES!**

Flash Player

Flash Player is typically installed as a plug-in in most web browsers. This means if you drag a SWF file onto your browser, it should play. This is a less than ideal way to test your games though, so we'll download the stand-alone Flash Player in two versions: Release and Debug.

Go here: http://www.adobe.com/support/flashplayer/downloads.html

and get these two files:

Download the Windows Flash Player 10.2 Projector content debugger (EXE, 6.36MB) and
Download the Windows Flash Player 10.2 Projector (EXE, 5.37MB)

If a later version of Flash Player is available when you read this, get those instead.

You will now have 2 EXE files. Copy them somewhere you are not likely to accidentally delete them!

Debug Player vs. Release Player

If you drag a SWF file onto either of these new Flash Player exes it will open and play. The reason for getting both is that one is the **Debug Player**, and the other is the **Release Player**. The Debug Player (the one with _sa_debug in its name) contains the ability for your games to throw run-time errors, view redraw regions and report back to the Flash Debugger (fdb) that FlashDevelop uses. This last part is vital for your sanity, as it will report and catch errors during testing back to you (where-as Release Player typically fails silently).

You should always develop using the Debug Player, and then test it on the Release Player before you go live. Release Player will give you a free speed-boost, as it doesn't contain all the debugging baggage, so it runs quicker. It's also what the majority of people seeing flash on the web are using. If your game is running a little slow under Debug, it may well be fine under Release.

Configure Flash Player to open SWFs

By default if you double-click a SWF file in Windows it'll probably ask you to select a program to launch it with, or it may launch your browser. To change this behaviour so it opens SWFs with the Debug Player we just downloaded do the following:

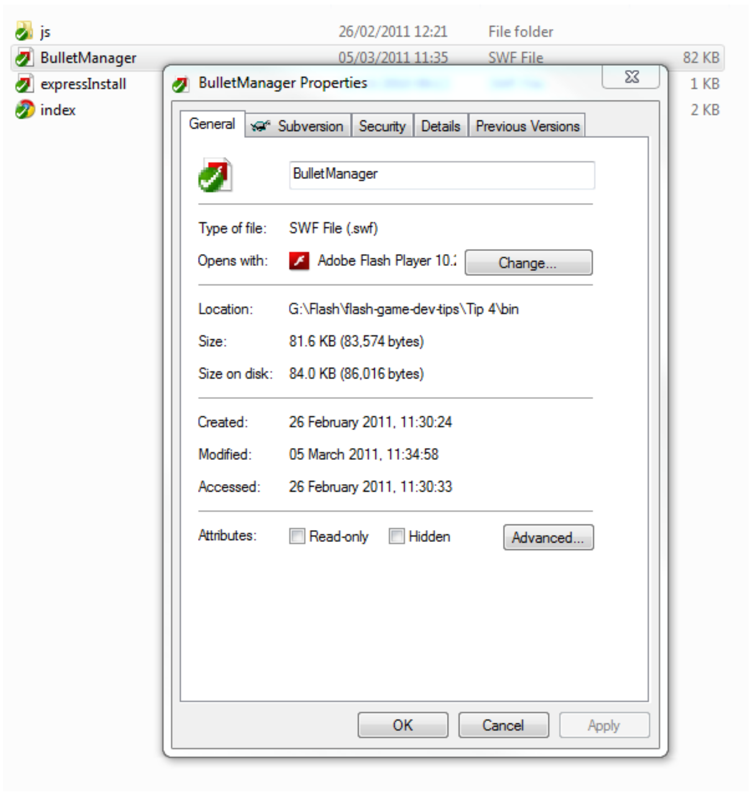
Find a SWF file on your hard drive, **right-click** it and select **Properties**. You'll see a properties window similar to this:

Blogroll

8-bit Rocket	Allen Chou
AltDevBlogADay	Bits of Bas
Codeflow	Freelance Games
Gaming Your Way	Hidden Resource
JActionScripters	JayC Santos
Mike Cann	Mr.doob
Point of Impact	The Jam Factory
Vortix Games	WordPress Plugins

Great Sites

A Game A Week	Dark Bit Factory
Demoscene.tv	Flash Bookmarks
Flashtro	Little Green Desktop
Nectarine Radio	Pixel Prospector
Retro Computers	The Joy of Sticks
We Are Back	



js 26/02/2011 12:21 File folder
BulletManager 05/03/2011 11:35 SWF File 82 KB
expressInstall 1 KB
index 2 KB

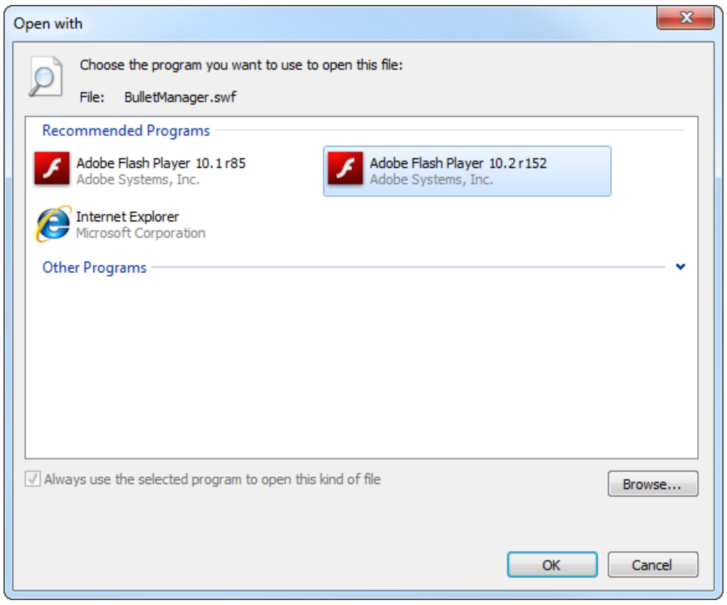
BulletManager Properties

General | Subversion | Security | Details | Previous Versions

BulletManager

Type of file: SWF File (.swf)
Opens with: Adobe Flash Player 10.2
Location: G:\Flash\flash-game-dev-tips\Tip 4\bin
Size: 81.6 KB (83,574 bytes)
Size on disk: 84.0 KB (86,016 bytes)
Created: 26 February 2011, 11:30:24
Modified: 05 March 2011, 11:34:58
Accessed: 26 February 2011, 11:30:33
Attributes: Read-only Hidden

Click the "Change …" button to get this:



Open with

Choose the program you want to use to open this file:
File: BulletManager.swf

Recommended Programs
Adobe Flash Player 10.1 r85 — Adobe Systems, Inc.
Adobe Flash Player 10.2 r152 — Adobe Systems, Inc.
Internet Explorer — Microsoft Corporation
Other Programs

Always use the selected program to open this kind of file

Click on the "Browse …" button and then select the Flash Player Debugger EXE you just downloaded. Make sure that "*Always use the selected program to open this kind of file*" is checked and click OK. Once you're done it will look like the above window. And what's more, double-clicking SWF files will now automatically launch them into Flash Player. Don't worry, this action won't change the way your web browser runs Flash. That will still work in the same way.

flixel

Adam (the creator of flixel) keeps the source code on github.com – a social source sharing site. There are 3 branches of flixel at the moment: Master, Dev and Beta. Currently version 2.5 of flixel is in dev and beta, but will soon be pushed to Master (where the final release versions sit). For now let's get the 2.5 release from the beta branch.

So point your browser to https://github.com/AdamAtomic/flixel/tree/beta

.. and click the big **Downloads** button, and pick the .zip package.

You'll have a new zip file ready for use in a moment.

Project Folder

You now need to decide where you want to create all of your Flash projects.

I personally have a folder called "Flash" on one of my drives, which is then broken down into "Games", "Demos", and various other things. For the sake of this tutorial I'll assume you have picked somewhere to keep all of your work, and I'll refer to that place as your *Projects Folder*.

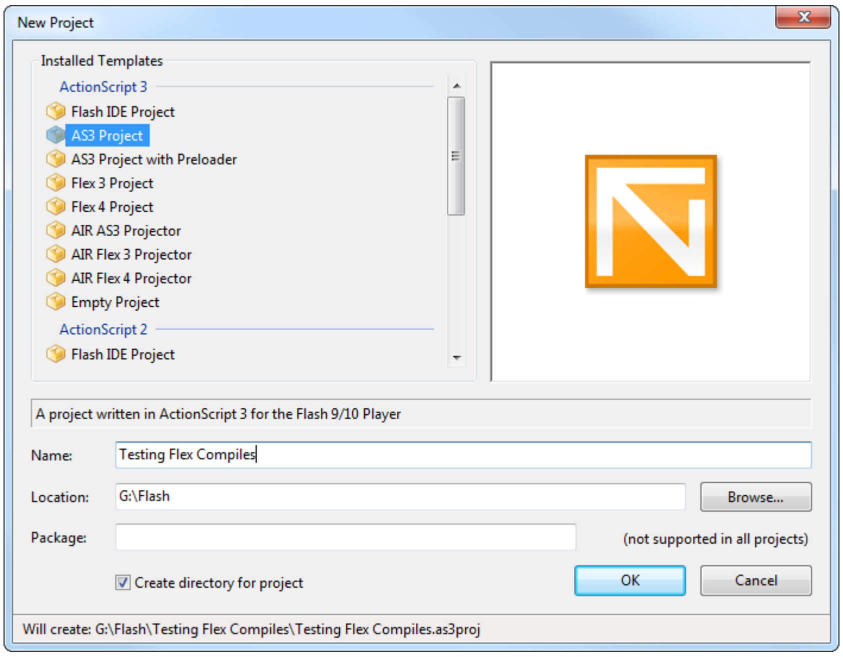
Extract the flixel zip file you just downloaded into your *Projects Folder*. It will create a new folder called **AdamAtomic-flixel-719aabe** – rename it to "**flixel 2.43**" (for your sanity more than anything). You've now got the latest release of flixel. In the **docs** folder you'll find documentation. In the **org** folder is everything we'll need in a moment when we create our first flixel project.

Create your first FlashDevelop Project

Note: If you know that FlashDevelop works fine on your PC, skip this step.

If all went well earlier you should have a shiny new FlashDevelop icon on your desktop. Double-click it to get started and the "New Project" window should appear (if it doesn't click on the *Project* menu item and pick *New*)

You'll be presented with this:



A project written in ActionScript 3 for the Flash 9/10 Player

Name: Testing Flex Compiles

Location: G:\Flash

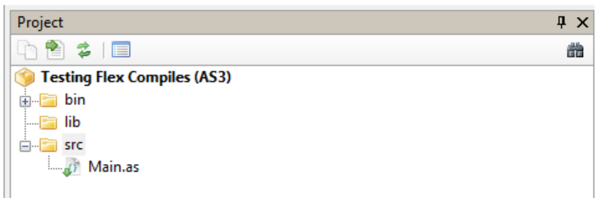
Package:

☑ Create directory for project

Will create: G:\Flash\Testing Flex Compiles\Testing Flex Compiles.as3proj

Select "AS3 Project", give it a Name (I used "Testing Flex Compiles" above) and set the Location to be your Project Folder. Have "Create directory for project" ticked and click OK. It may now prompt you for some details, such as your name, fill them in once and it will remember them.

A big blank empty FlashDevelop project will open. In the Project window you should see 3 folders like below:



The folders are:

bin – Where the final SWF file gets compiled to
lib – You can put external compiled library files here, such as SWCs (don't put AS3 source code here)
src – The source folder, where all of the project source code lives

I automatically create 3 new folders at the same level:

assets – Where all assets that the game requires are placed, this is sub-foldered into *graphics*, *music*, *maps* and *sound*
dame – When creating level tilemaps I always use the map editor DAME, and store my tilemaps here
psds – High-res and working graphics are stored here. I call it psds only because I use Photoshop

Obviously you don't have to create these 3 folders, but I would strongly recommend you do create an *assets* folder. The difference between *assets* and *psds* is that *psds* for me contains all of the work in progress graphics and Photoshop files. Where-as *assets* contains the PNGs exported out of Photoshop, for final use in my game.

If you expand the *src* folder you'll find a file called **Main.as** inside. Double-click it to edit it.

This file is created automatically by FlashDevelop and it's where your journey with flash coding begins. We want to see if the Flex compiler is working, so we'll add in a couple of lines of code. Modify the init function so it looks like this:

```
private function init(e:Event = null):void

{
 removeEventListener(Event.ADDED_TO_STAGE, init);


 var redbox:Sprite = new Sprite;

 redbox.graphics.beginFill(0xFF0000, 1);

 redbox.graphics.drawRect(0, 0, 100, 100);

 addChild(redbox);

}
```

What we are doing here is creating a new Sprite called redbox, and making it look like a 100×100 pixel sized red square. We then add it to the display list so it renders.

Press F5 and FlashDevelop will compile and try to run your SWF.

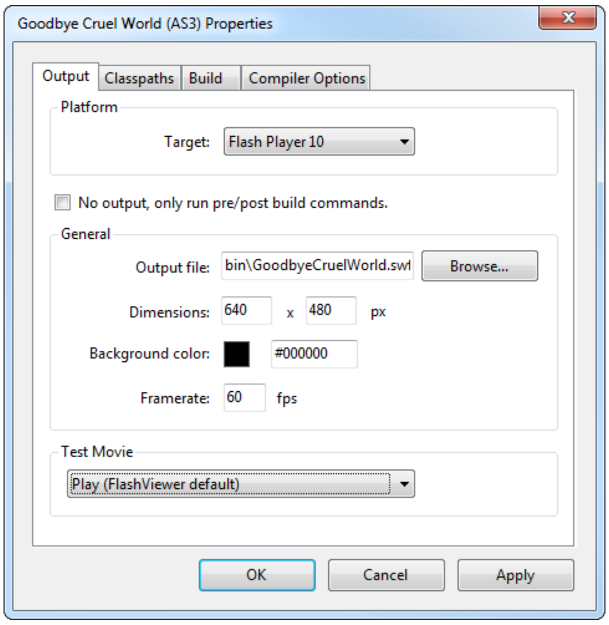
If all has gone well (no typos, no set-up mistakes, etc) then Flash Player will open with an 800×600 sized white window, with a red square in the top left of it. Ok so it's hardly Super Mario, but everyone has to start somewhere right?!

Your first flixel project

Create a new FlashDevelop project, but this time pick the Project type "AS3 Project with Preloader", call it "Goodbye Cruel World" and click OK.

The same 3 folders as before will be created. Create any extra ones you need (hint: assets!)

Before we code anything we're going to configure the Project Properties. To do this click on the **Project** menu and pick **Properties**. Change the Dimensions to 640 x 480, the Background colour to black and the Framerate to 60 fps. Also ensure that the Target is Flash Player 10 and in the Test Movie panel it says "Play (FlashViewer default)". It should look like this:


Goodbye Cruel World (AS3) Properties

Output | Classpaths | Build | Compiler Options

Platform
Target: Flash Player 10

No output, only run pre/post build commands.

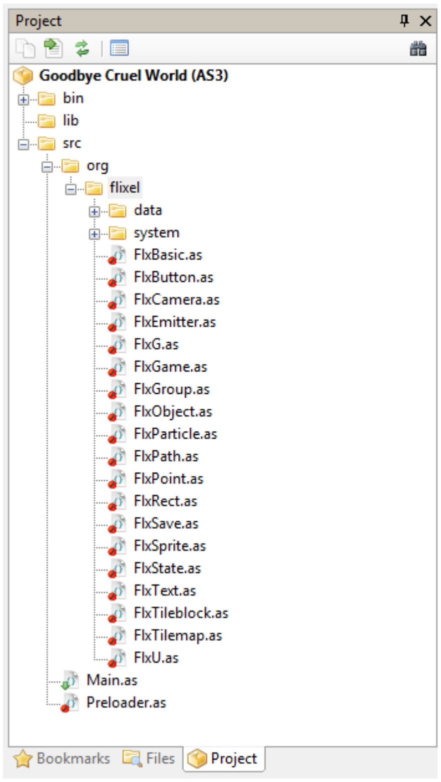
General
Output file: bin\GoodbyeCruelWorld.swf
Dimensions: 640 x 480 px
Background color: #000000
Framerate: 60 fps

Test Movie
Play (FlashViewer default)

This told FlashDevelop that we want our game to be 640×480 in size, have a black background by default, run at 60 frames per second and that we'll test it in the Flash Player associated with SWF files (which we set-up earlier).

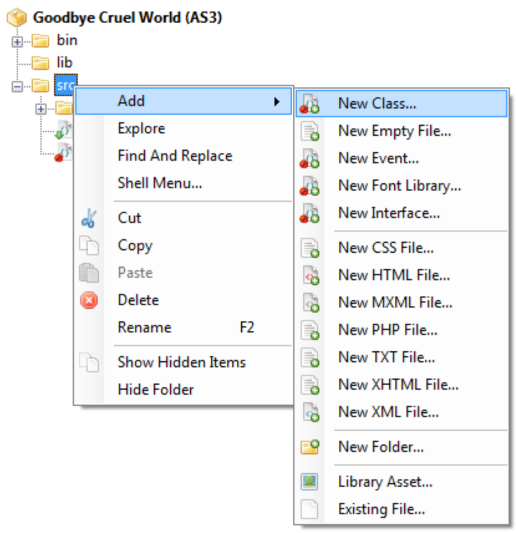
Now remember that zip file we downloaded from github? The one actually containing flixel? Go back to where you extracted it and copy the **org** folder into the **src** folder of your new project. Do NOT include docs, build_docs.sh, flx.py or license.txt. Just take the org folder and nothing else.

If you do this correctly the Project structure window in FlashDevelop will look like this:



It's very important to note the structure here. The **org** folder is sitting at the root (the top) of the **src** folder. This is vital. If you have it located anywhere else it'll fail to compile. In the shot above I've expanded out the **org/flixel** folder so you can see the contents. You'll also notice the **Main.as** file as before, and a new **Preloader.as** file.

Right-click the src folder and select "Add – New Class …":

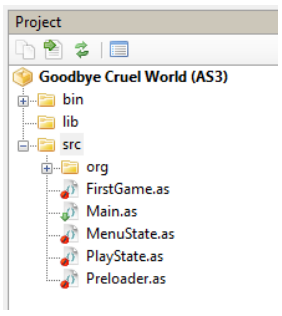


In the panel that appears ignore everything other than the **Name** (which will default to *NewClass*). Change it to **FirstGame** and click OK.

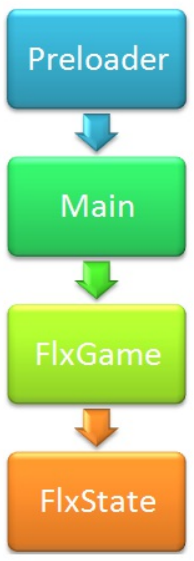
Repeat this process again, only instead of calling the class FirstGame, call it **MenuState**.

Repeat the process one final time, and call it **PlayState**.

By the end of this our Project window has the following files, all in the correct place:



Before we dive in, it's worth talking about how flixel structures itself. This is best illustrated with a diagram:



When the SWF is loaded into Flash Player the code in the Preloader will run. This will carry on running until the SWF has fully loaded.

At this point the class Main gets created.

Main in turn creates a new class: FlxGame – this is the start-up class for flixel, it tells flixel what size your game is, if it should be zoomed and which state to run by default.

FlxGame then calls the FlxState you nominated, and things kick off from here.

Technically Main and FlxGame can be bundled into the same class file. I don't like to do this personally as I'll explain in a future tutorial, but there's nothing wrong in doing so.

Right, time to put some code into those files we created!

Edit Main.as

Open Main.as in FlashDevelop and edit it to this:

```
package
{
 import flash.display.Sprite;
 import flash.events.Event;

 [Frame(factoryClass="Preloader")]
 public class Main extends Sprite
 {
  public function Main():void
  {
   if (stage)
   {
    init();
   } else {
    addEventListener(Event.ADDED_TO_STAGE, init);
   }
  }

  private function init(e:Event = null):void
  {
   removeEventListener(Event.ADDED_TO_STAGE, init);

   var game:FirstGame = new FirstGame;

   addChild(game);
  }
 }
}
```

This is the default file as created by FlashDevelop, with two extra lines. It waits for the pre-loader to have finished and then it creates a new local variable called game, which is an instance of the FirstGame class. This is then added to the display list.

In short, it's just made your flixel game visible.

Edit FirstGame.as

Open FirstGame.as in FlashDevelop and edit it to this:

```
package
{
 import org.flixel.FlxGame;
 public class FirstGame extends FlxGame
 {
  public function FirstGame()
  {
   super(320, 240, MenuState, 2);
  }
 }
}
```

Quite simply this creates our FlxGame instance. The call to super takes 4 parameters. The width and height of our game, the FlxState to call when FlxGame is ready, and the zoom level we're running at. v2.5 of flixel also added 2 new parameters: the frame rate of your game, and the frame rate to set Flash Player to. It's fine to leave the defaults for now.

In this case our SWF dimensions were 640×480, but the size set here is 320×240 with a zoom level of 2. So basically every pixel is going to appear doubled-up in size, for that 8-bit retro style 🙂 MenuState is the FlxState that will be called first.

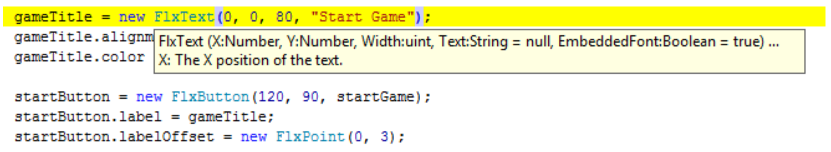
Edit MenuState.as

Open MenuState.as in FlashDevelop and edit it to this:

```
package
{
 import org.flixel.*;

 public class MenuState extends FlxState
 {
  private var startButton:FlxButton;

  public function MenuState()
  {
  }

  override public function create():void
  {
   FlxG.mouse.show();
   startButton = new FlxButton(120, 90, "Start Game", startGame);
   add(startButton);
  }

  private function startGame():void
  {
   FlxG.mouse.hide();
   FlxG.switchState(new PlayState);
  }
 }
}
```

This looks more complicated, but all it's actually doing is creating a button that the user can click to start the game. FlxStates have a function called create which you must over-ride and then fill with your code. In our case we've told flixel to show the mouse pointer and create a start game button.

Look at the flixel documentation for details on what all of these things do. Or, as you're now cooking with the power of FlashDevelop you can get context-sensitive help on most flixel functions by simply clicking inside of the braces and pressing CTRL+SHIFT+SPACE. You'll see a pop-up like this, and as you move along through the parameters it'll tell you what each of them do.



```
gameTitle = new FlxText(0, 0, 80, "Start Game");
gameTitle.alignm
gameTitle.color

startButton = new FlxButton(120, 90, startGame);
startButton.label = gameTitle;
startButton.labelOffset = new FlxPoint(0, 3);
```

FlxText (X:Number, Y:Number, Width:uint, Text:String = null, EmbeddedFont:Boolean = true) ...
X: The X position of the text.

Edit PlayState.as

Open PlayState.as in FlashDevelop and edit it to this:

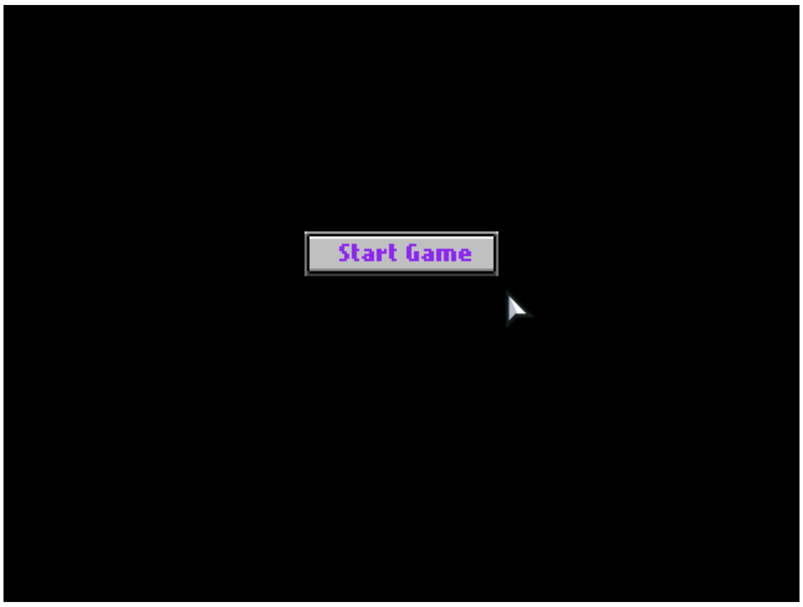
```
package
{
 import org.flixel.*;

 public class PlayState extends FlxState
 {
  public function PlayState()
  {
  }

  override public function create():void
  {
   add(new FlxSprite);
  }
 }
}
```

Here we create one single FlxSprite and add it to the display. As we've not loaded in any graphics this will appear as a small flixel logo. It's not much, but it lets us know that our MenuState is hooked to our PlayState correctly.

F5 = The moment of truth

Press F5 to test your game.

FlashDevelop will invoke the compiler and if all was well, it'll launch Flash Player with your SWF in. Hopefully you'll see this:



And clicking Start Game will show a black screen with a small white square in it.

Success! You've just compiled and launched your first flixel project

You've also got a good FlashDevelop template here to use for future games. Just copy the whole project folder, rename it and you're good to go once more.

Before you know it you'll be creating games like this too:

If you have any trouble with any part of this tutorial, please post about it on the official Flixel forums: http://flixel.org/forums

Further Reading

To learn about the Flex 3 compiler design there is a great white-paper on the Adobe site.

To understand how mxmlc works, read this article by Senocular.

A Beginners Guide to FlashDevelop is a great article if you want to learn more about what FlashDevelop can do.

And the Flash Game Dev Tips Google Code Project page contains lots of other tips you can use with your new-found flixel skills.

Share 27 Tweet 5 share 0 share 42 Like 12 +1

Posted on March 22nd 2011 at 10:55 pm by Rich.
View more posts in Flash Game Dev Tips. Follow responses via the RSS 2.0 feed.

« Flash Game Dev Tip #5 – Configure your games in real-time

Kingdums Public Beta – Please join in! »

24 Responses

Leave a comment

Marza March 23rd 2011 at 10:54 am

Wow, great tutorial, I've been learning java for almost a year now, and wanted to start trying flash for ages! (I've aslo done some tinkering in Game Maker)

I haven't read your earlier dev tips as closely, but I think I will soon, now that I have a starting points for making some of them games :]

Keep up the good work and stuff! : D

Marza March 23rd 2011 at 11:11 am

Oh, sorry for double post, but I have a little question…
In the installation notes for FlashDevelop it says you need TortoiseSVN or TortoiseGit for using the source control plugin. I didn't get it but was just wondering if it's a good idea to get, and what the "source control plugin" is for?

rich March 23rd 2011 at 11:34 am

Hi Marza – This explains what Source Control is pretty well http://en.wikipedia.org/wiki/Revision_control and if you then decide you want to use it (and I'd strongly recommend it, although it does take some time getting in to) then go with TortoiseSVN and install that. It's free, and you can create your own accounts on Google Code or a SVN hosting service such as beanstalkapp.com (which is who I use)

Frank March 23rd 2011 at 10:41 pm

Great, that's surely a big help for all FlashDevelop / Flixel beginners!
A nice shortcut I'd like to add is to install a FlashDevelop Flixel template like the one posted by LeO on bliip: http://www.bliipstudio.com/as3/flashdevelop-flixel-template-starting-template/
Just unpack the archive into your FlashDevelop\Projects directory, and then instead of creating an "AS3 Project", create an "AS3 Flixel Project". Saves you several steps of this tutorial, like downloading and unpacking Flixel source code and creating the typical game classes. And it can easily be repeated for new projects!

Victor Grunn April 5th 2011 at 5:00 am

Thanks. This was a fantastic tutorial from start to finish.

Ruber Eaglenest April 7th 2011 at 11:13 pm

It's the time. Thanks a lot!

Daniel April 9th 2011 at 10:12 pm

Great stuff! Something is missing though: The tutorial doesn't mention how the Preloader class got there or what code goes in it. Just thought I'd mention that 😃

rich April 10th 2011 at 1:31 am

Hi Daniel – Glad you liked it. It does say to "but this time pick the Project type "AS3 Project with Preloader" and it then goes on to explain that there will be a Preloader.as file created as a result of picking this. You don't need to put any code in it (at this stage) because FlashDevelop fills it out for you. When you come to customise the preloader, then you need to change it (but that's a little beyond the scope of this tutorial specifically).

Daniel April 10th 2011 at 8:30 am

Oops, missed that part :p sorry. I read the part just after that part several times and and couldn't figure out how the Preloader class just got there all by itself and I also assumed it was a subclass of FlxPreloader… That'll teach me to read stuff more carefully in the future 😉

Adrian April 14th 2011 at 10:42 pm

I seem to be having a problem with two lines in MenuState…

FlxG.state = new PlayState;

Throws an error because FlxG.state appears to be read-only. And…

startGameButton.loadText(new FlxText(0, 0, 100, "Start Game"));

I couldn't find loadText in the documentation, so I tried scrapping the line and just using the label param. Could these be caused by using a different Flixel version? I'm pretty sure I'm on 2.43

Adrian April 14th 2011 at 10:52 pm

Quick follow up!

I resolved the issue with

FlxG.state = new PlayState;

by instead using

FlxG.switchState(new PlayState);

rich April 14th 2011 at 11:41 pm

Ok the whole article is now 100% v2.5 compatible, including all the screen grabs and source code 😀

Anonymous May 6th 2011 at 2:30 pm

Um, your changes to menuState.as are rather odd – doesn't add the title, and the title is "Start Game" while the button says "My Game"

dj rosenbaum May 11th 2011 at 11:03 pm

made it as far as Create your first FlashDevelop Project

any idea what this error means?

http://i.imgur.com/OSVUQ.png

Error: The private attribute may be used only on class property definitions.
private function init(e:Event = null):void

dj rosenbaum May 11th 2011 at 11:24 pm

Now it works.. at the "Create your first FlashDevelop Project"

the code should look like this

http://i.imgur.com/aWeGZ.png

rich May 11th 2011 at 11:41 pm

The tutorial does say "Modify the init function so it looks like this" (the key word being "Modify").

Hosting Best May 25th 2011 at 5:27 pm

I cant believe how long I was coding without FlashDevelop. The program is sweet!!!
Saves so much time and was worth the couple hours it took to learn and get used too

frankeinstein October 29th 2011 at 12:14 pm

how can i solve this?

[Fault] exception, information=TypeError: Error #2023: Class Preloader$ must inherit from Sprite to link to the root.

lyuba November 4th 2011 at 8:43 am

Can't you do this with Add Classpath?

Dean December 8th 2011 at 9:04 am

I'm having trouble adding the org folder to the src folder, could anyone help with a step-by-step? Thanks!

Anon December 19th 2011 at 1:55 pm

Dean- I'm just learning all of this so I'm not sure how much authority I have on the matter, but when it came to adding the org. to the src. folder I clicked on the src. and chose to explore it. Then once the program files came up I manually added the org. Hope that helps (: and slight issue with my "game" I get the black box and everything when I F5 but there's no button saying to start my game…coding matched up verbatim with what was in the tutorial. Anyone help?

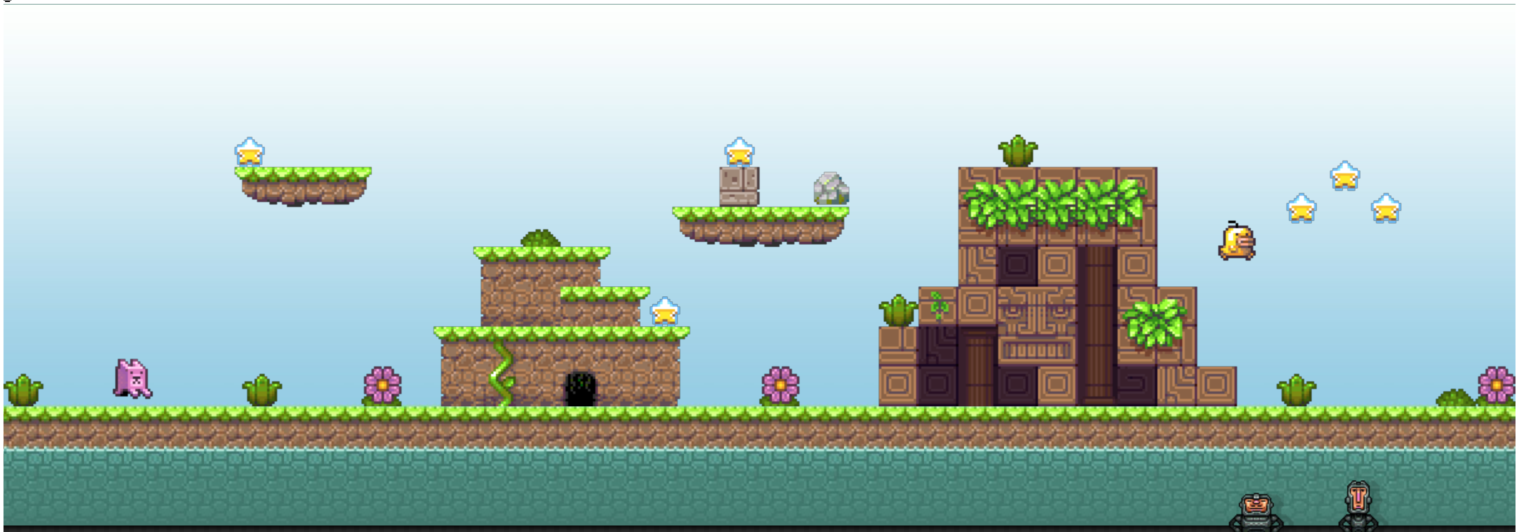
Make yourself heard

Name	
Email	
Website	
Comment	

Post Comment

Notify me of follow-up comments by email. ☐

Notify me of new posts by email. ☐



Photon Storm

PROGRAMMING by RICH DAVEY
PIXELS & SOUND by ILIJA MELENTIJEVIC