UW GAME DEV CLUB

Making games with:
Haxe (The language)
OpenFL (The library)

HAXE

One language, everywhere.

OpenFL

UW GAME DEV CLUB

Pros:

-Exports to multiple platforms (flash, html5, windows, android, ios, etc)

-Similar language (Haxe) and API (OpenFL) to actionscript3/flash

-Light weight and open source

Cons:

-Not so great documentation (relatively speaking)

-Kinda complicated to set up (definitely not one click or one step)

-Android/ios builds are slooooooooow

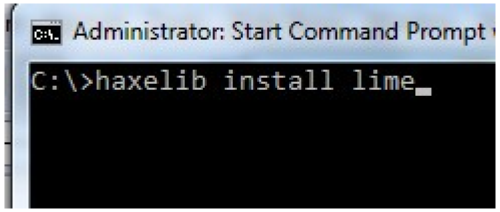
NICKELODEON

TiVo

UW GAME DEV CLUB

Setup:

-Download+Install Haxe
(http://haxe.org/download)

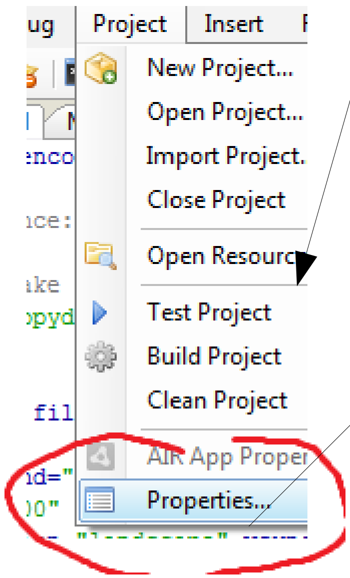


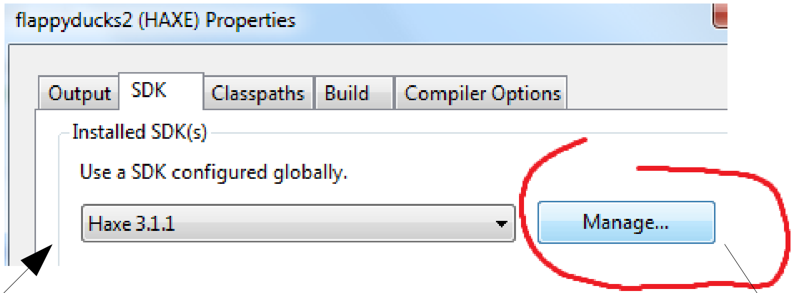
```
C:\>haxelib install lime_
```

-Install lime & openfl
(“haxelib install lime”, “haxelib install openfl”)

-Download+Install FlashDevelop
(http://www.flashdevelop.org/)
(If you want an IDE like eclipse, though you can code/compile command line if you really want)

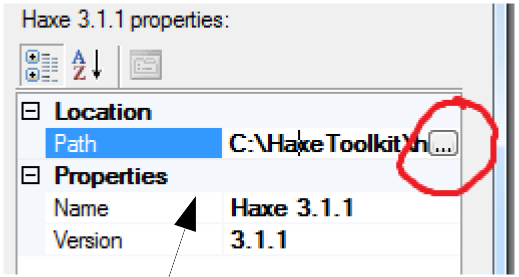
-Link to the Haxe installation in Flashdevelop



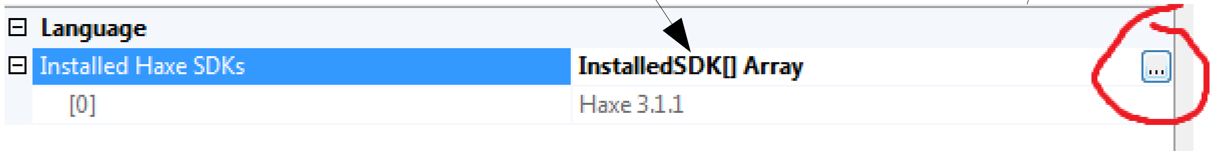


Use a SDK configured globally.

Haxe 3.1.1


Haxe 3.1.1 properties:

Location	
Path	C:\HaxeToolkit\h
Properties	
Name	Haxe 3.1.1
Version	3.1.1

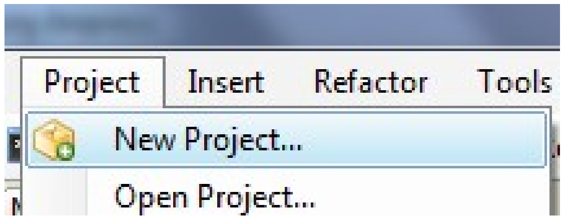


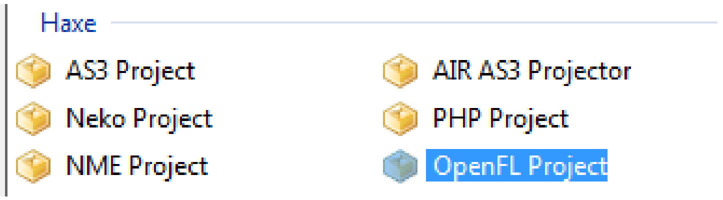
Language	
Installed Haxe SDKs	InstalledSDK[] Array
[0]	Haxe 3.1.1

UW GAME DEV CLUB

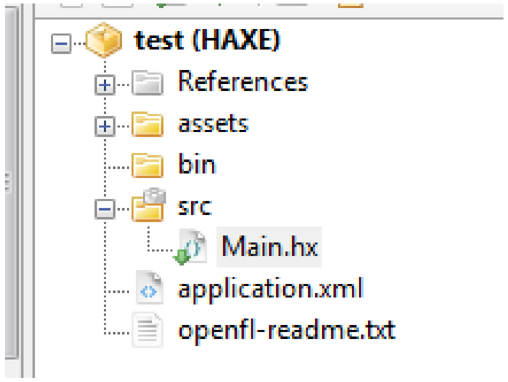
Making a new project in FlashDevelop

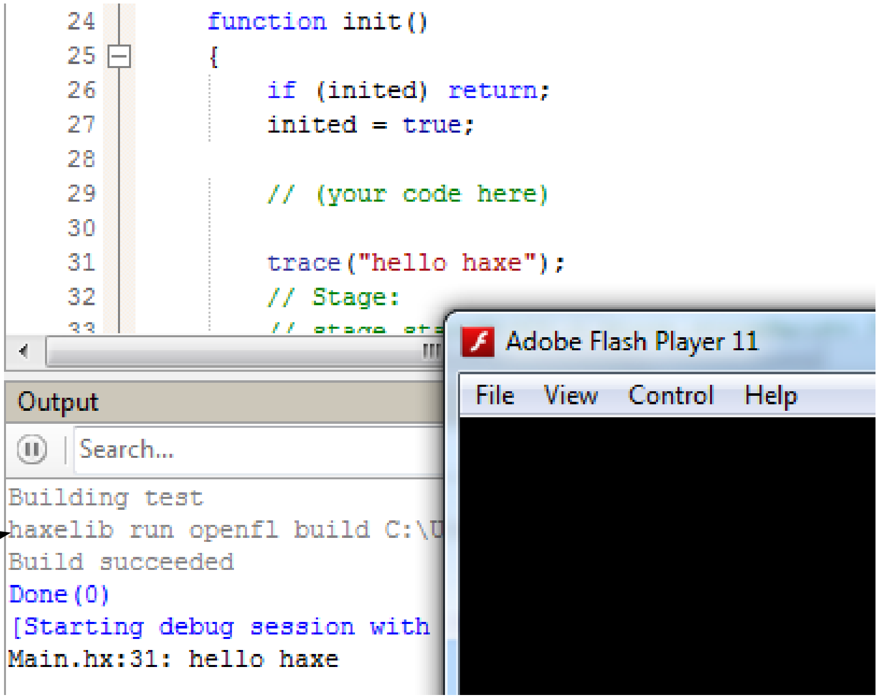
Create a new OpenFL project to be able to use the OpenFL library.





The starting point for the code is in “Main.hx”

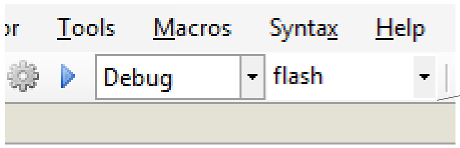




```
24      function init()
25      {
26          if (inited) return;
27          inited = true;
28
29          // (your code here)
30
31          trace("hello haxe");
32          // Stage:
```

```
Building test
haxelib run openfl build C:\U
Build succeeded
Done(0)
[Starting debug session with
Main.hx:31: hello haxe
```

Assuming everything is set up properly, running and compiling (with this setup) will bring up a blank swf.



UW GAME DEV CLUB

Haxe the language

Very similar to Actionscript 3 (think structure like java, code like javascript with types)

```
package ;
import flash.display.Sprite

class TestClass extends Sprite {

    public var field1:Int = 1;
    public var field2:String = "string field";

    public function new() {
        trace("constructor");
        this.testmethod();
    }

    public function testmethod():Bool {
        trace(field2);
        return true;
    }
}
```

Documentation:
http://haxe.org/ref

UW GAME DEV CLUB

OpenFL (flash) the library

Stage (the screen, the root of the display tree, etc)

```
stage.addEventListener(eventType,onEvent)
```
for
`TouchEvents, MouseEvents, KeyboardEvents, EnterFrame,` etc

```
stage.addChild(new Sprite())
```

Sprite (an object on the screen)

```
sprite.x
sprite.y
```
(Float position)

```
sprite.graphics.beginFill(0xFF0000);
sprite.graphics.drawCircle(x,y,radius);
```

Graphics (draw things on this sprite)

Official API (kinda shitty)
http://www.openfl.org/documentation/api/flash/display/

Adobe's flash API (pretty good, different language but exact same classes/etc)
http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/index.html

UW GAME DEV CLUB

Game setup

Create a new “FlappyGame” class (that extends flash.display.Sprite) and add it to the stage in Main in init()

(In Main.hx)

```
function init() {
    if (inited) return;
    inited = true;
    stage.addChild(new FlappyGame());
}
```

In the FlappyGame constructor, add a event listener to this for Event.ADDED_TO_STAGE. Call your own initializaton method in the callback.

(In FlappyGame.hx)

```
public function new() {
    super();
    this.addEventListener(Event.ADDED_TO_STAGE,
        function(e:Event) {
            init();
        }
    );
}
```

In your initialization method, add an event listener for Event.ENTER_FRAME to this and call your update method in the callback.

This is your update cycle.

```
public function init():Void {

    //do other initialization stuff here

    stage.addEventListener(Event.ENTER_FRAME,
        function(e:Event) {
            update();
        }
    );

}
```

UW GAME DEV CLUB

Input

```
stage.addEventListener(MouseEvent.CLICK,
     function(e:MouseEvent) {
          //called when clicked
     }
);

stage.addEventListener(TouchEvent.TAP,
     function(e:TouchEvent) {
          //called touched (on mobile devices)
     }
);


stage.addEventListener(KeyboardEvent.KEY_DOWN,
     function(e:KeyboardEvent) {
          //called keys pressed
     }
);
```

Some of these only work on computers (when playing as a flash/html5/windows game), others only work when playing on a mobile device.

Have them all call the same functions.

UW GAME DEV CLUB

Game Plan - Moving bird

-Have a field `bird:Sprite` that is your player, move this sprite around.

-Have a field `bird_vy:Float` that is the current y velocity of your player

-In your update cycle, decrement your velocity (gravity) and apply your velocity on the player bird's position

```
bird_vy+=0.5;
bird.y+=bird_vy;
```

-Any time the player taps/clicks, set the velocity to some high positive number

```
bird_vy = -8;
```

-If the player's y position is less than 0 or greater than height, game over

-Make a game_start and game_end function (to start and end the game), as well as a boolean “if the game is over”. Stop the game when the game is over.

UW GAME DEV CLUB

Game Plan - Spawn Pipes

-Make a new class that extends `flash.display.Sprite`, “Pipe”. It should hold a width and height (as well as draw something with its graphics). In addition, give it a

```
public function hit_player(x:Float, y:Float):Bool
```

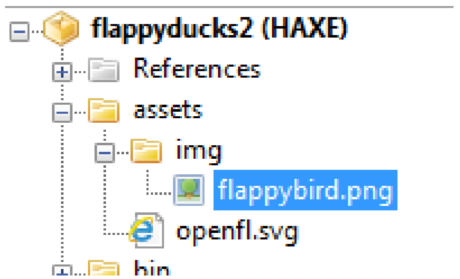
-Make an `Array<Pipe>` in your flappygame

-Every once in a while, add a new pipe randomly placed on the right side of the screen

-Every update cycle, loop through all the pipes
- -Check if the player is hitting the individual pipe, if so end the game
- -Move the pipe a little to the left

UW GAME DEV CLUB

Bonus - Adding an Image



Add your image file in “assets/img” folder

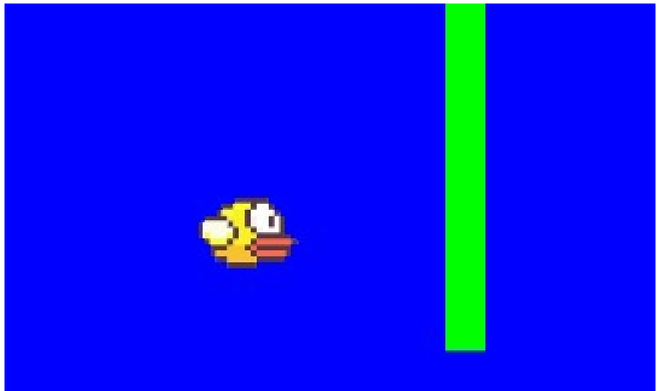
To access this image, call

```
openfl.Assets.getBitmapData("img/flappybird.png")
```

(This returns the “BitmapData” object representing your image)

To render the image onto the screen, do:

```
sprite.addChild(new Bitmap(Assets.getBitmapData(“yourimage.png”)))
```



QUALITY VIDEOGAME

UW GAME DEV CLUB

Big list of Haxe resources:

Flappyducks source code:
https://github.com/spotco/flappyducks

Reasons to use Haxe/OpenFL (especially if you're a flash developer!)
http://gamasutra.com/blogs/LarsDoucet/20140318/213407/Flash_is_dead_long_live_OpenFL.php

Papers, Please! (Popular recent game made with Haxe)
http://papersplea.se/

OpenFL API (kinda shitty)
http://www.openfl.org/documentation/api/flash/display/

Adobe's flash API (pretty good, different language but exact same classes/etc)
http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/index.html

Haxe Language Reference
http://haxe.org/ref