Plugin & Scripting for Flash Applications

Gabriele Farina & Alessandro Crugnola
alittleb.it

Today's menu

- What is an extensible application and why to use it
- Limits
- Plugins
- Scripting
- Small examples

Why extensibility?

Why extensibility?

- Add more features without compile the core application every time.

Why extensibility?

- Add more features without compile the core application every time.
- Better maintenance and easier updates

Why extensibility?

- Add more features without compile the core application every time.
- Better maintenance and easier updates
- Better team organization

Why extensibility?

- Add more features without compile the core application every time.
- Better maintenance and easier updates
- Better team organization
- Open your application to the world

Limits

Limits

- API must be stable and consistent

Limits

- API must be stable and consistent
- More complexity for the development

Limits

- API must be stable and consistent
- More complexity for the development
- Security may be hard to achieve

Real world applications

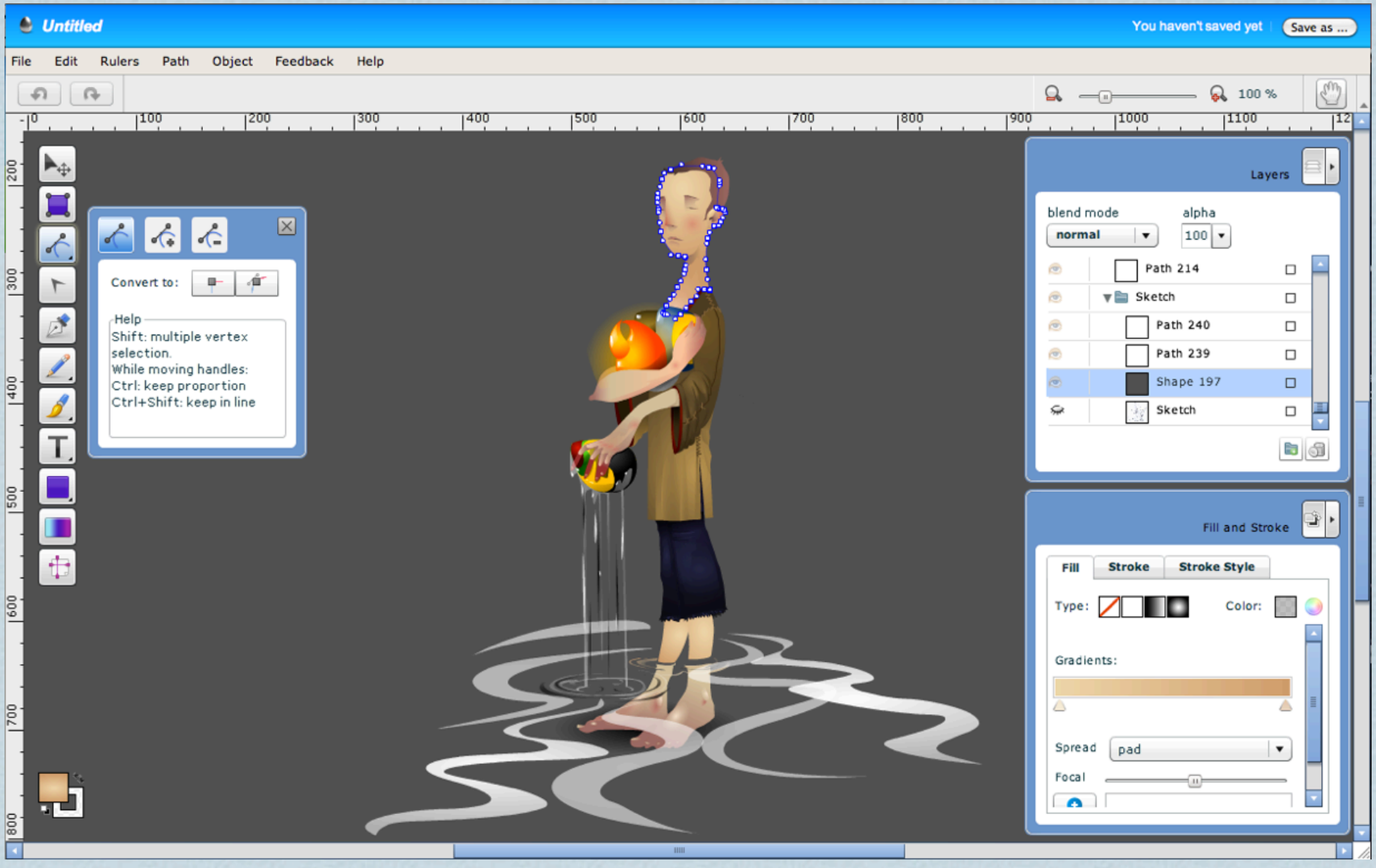
Sliderocket

http://www.sliderocket.com

One of the first public project to use a real plugin engine

Aviary

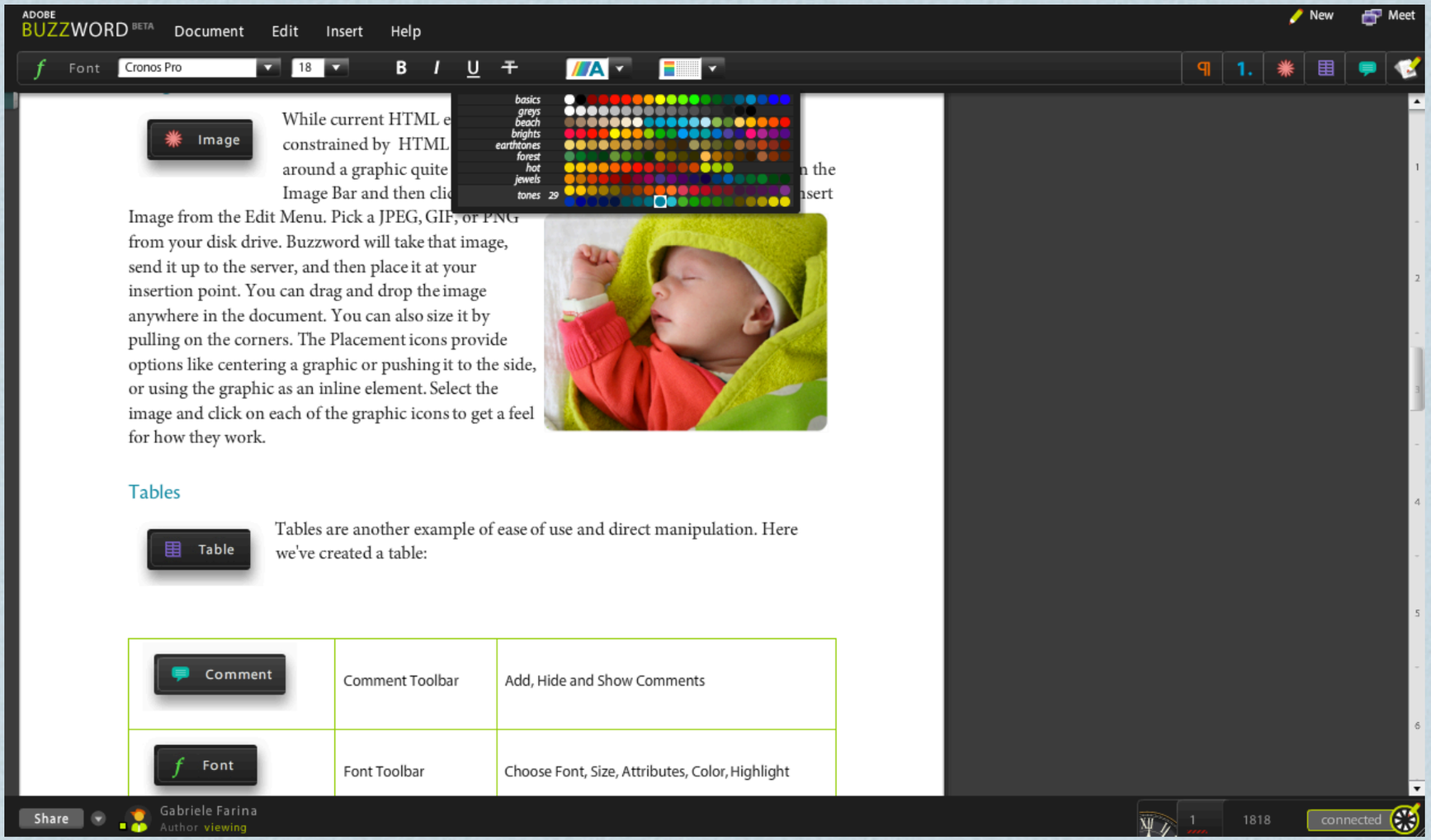
http://aviary.com



Our current project. It uses both plugins and scripting

Buzzword

http://www.buzzword.com



Background and delayed flex modules

Hobnox

http://www.hobnox.com

Audio sequencer that uses advanced plugins

Plugins

Plugins

- What is a plugin?

  Usually it's a runtime loaded SWF file which contains additional functionalities to extend our application.
  Sometimes simple archive files are used instead, to pack together a plugin and its resources (mozilla).
  They may be loaded from a local or remote server at startup or on demand.

Plugins

- Advantages

  Easy maintenance.
  Loading can be delayed.
  Updates does not require core recompilation.
  Core updates may not require plugins recompilation (if well designed).
  Multiple versions.

Plugins

- Disadvantages

  Increase development complexity.
  API must be well designed.
  Risk of compatibilty issues: external content cannot be trusted as long as a development error could break the main application core.
  Security check is complex (where required).

Plugins

- When

  Applications that required to be extended by external developers.
  There are a lot of features not required at startup.
  Large and fragmented team.

Plugins

- Security

  Sometimes is really hard to limit the plugin functionalities (swf loaded with loadBytes).
  Possible solutions are:
  - static bytecode analysis
  - compile time preverification (custom tools)
  - bytecode preverification: dynamic execution

Scripting

Scripting

- Brief introduction
- Compiled and interpreted languages
- Tools (Lexing and parsing. antlr, yacc etc.)

Scripting

```
clip.fadeIn()
when ( clip.alpha is greater than 99 ):
    clip.backgroundColor = red
    clip.fadeOut()
```

Source

```
clip.fadeIn()
when ( clip.alpha is greater than 99 ):
    clip.backgroundColor = red
    clip.fadeOut()
```

Lexing

Tokens

```
clip.fadeIn()
when ( clip.alpha is greater than 99 ):
    clip.backgroundColor = red
    clip.fadeOut()
```

Parsing

AST

Compiler → **SWF**

Interpreter

Scripting

- Why?

  You create the language: perfect for specific solutions.
  A language can be used by NON developers.
  You can simulate missing features of the host environment.

Scripting: interpreted

- Advantages

  The VM can be developed quickly.
  A DSL can be significantly shorter and quicker to write.
  Full control over the functionalities.
  They can be dynamically genered at runtime.

Scripting: interpreted

- Disadvantages

  Long development time for the front end.
  They may be pretty slow.
  Development requires good skills and experience in compiler implementation and programming language design.

Scripting: interpreted

- When

  You want the full control over the extension.
  People who will use the language may not be a developer.
  Target environment can change without changes to the front end.

Scripting: interpreted

- Security

  High security. If well designed you have the full control over the security while the script is loaded and when it is executed.

Scripting: compiled

In flash writing a compiled scripting language usually means to generate ABC code (to be injected into a swf file) at runtime. Since flash player 10 pixel bender bytecode is used too.

"Compiled scripting is the natural evolution of the interpreted scripting".

Scripting: compiled

- Advantages

  Execution is faster because it's performed by the native VM.
  In general you have the same advantages of the interpreted scripting.

Scripting: compiled

- Disadvantages

  Complex design.
  Complex development.
  Slower compilation time.
  Hard maintenance.

Scripting: compiled

- When

  The same as interpreted

- Security

  The same as interpreted but more complex

A plugin is not a SWF

- What I have to expose to my plugin system?
- They're usually interface/protocol based

A plugin is not a SWF

- Services

  Singletons exposes functionalities that can be packed conceptually together. They are a good approach if you want to be able to switch between different implementations for the same behavior.

A plugin is not a SWF

- Components

  Similar to services but they are meant to be used as classes.
  Components are usually implementations of abstract visual components or interfaces that can be instanced more than once.

A plugin is not a SWF

- Extensions

  Extensions are a good way to be able to give the implementation the ability to decide what to extend.
  For instance an application may define an entry point (like a menu bar) and expects the plugins to add functionalities (menu item) to it.
  A good example is the xul overlay system.

Example: Aviary API

```
1  var queue: JobQueue = new JobQueue();
2  var core: AviaryCore = AviaryCore.getInstance();
3
4  queue.addJob( core.initialize( new AviaryConfiguration() ) );
5  queue.addJob( core.pluginManager.loadPlugin( "storage.1.2.0.swf" ) );
6  queue.addJob( core.pluginManager.loadRemotePlugin( "http://sephiroth.it/plugins/eps_export.xml" ) );
7
8  var runner: JobRunner = new JobRunner( queue );
9
10 runner.addEventListener( JobEvent.COMPLETE, onJobComplete );
11
12 runner.run();
13
14 function onJobComplete( event: JobEvent ): void
15 {
16     var rookery_service: IRookeryService = plugin( "com.aviary.core.Storage" )
17         .getService( IRookeryService );
18     var eps_exporter: IFileExporter = new ( plugin( "it.sephiroth.aviary.EPS" )
19         .getComponent( IFileExporter ) )();
20     var eps: IBinaryFile = epx_exporter.export( rookery_service.getSession().activeDocument );
21
22     rookery_service.exportBinaryFile( "test.eps", eps );
23 }
```

Example: MMscript

```
resizeMainBillboardAnimationInNoRelated = fun $duration
{
    after ( 1, RelatedLeft.visible, @false );
    after ( 1, RelatedMiddle.visible, @false );
    after ( 1, RelatedRight.visible, @false );

    sequence ( { duration: duration },
        parallel ( { easing: Back_easeOut },
            parallel ( { target: MainBillboard },
                tween { property: scaleX, toValue: 0.15 },
                tween { property: scaleY, toValue: 0.15 },
                tween { property: y, toValue: 1180 },
                tween { property: x, toValue: 332.375 } ),
            parallel ( { target: Billboard },
                tween { property: scaleX, toValue: 0.15 },
                tween { property: scaleY, toValue: 0.15 },
                tween { property: y, toValue: 1180 },
                tween { property: x, toValue: 332.375  } ),
            parallel ( { property: alpha, toValue: 0 },
                tween { target: RelatedRayLeft },
                tween { target: RelatedRayMiddle },
                tween { target: RelatedRayRight },
                tween { target: BillboardBLL },
                tween { target: BillboardBRL } ) ) );
};
```

Example: MMscript

```
95
96  getXFor = fun   ( 1, @product1 )    368
97                  ( 2, @product1 )    353
98                  ( 2, @product2 )    383
99                  ( 3, @product1 )    338
100                 ( 3, @product2 )    368
101                 ( 3, @product3 )    398
102                 ( 4, @product1 )    323
103                 ( 4, @product2 )    353
104                 ( 4, @product3 )    383
105                 ( 4, @product4 )    413
106                 ( 5, @product1 )    308
107                 ( 5, @product2 )    338
108                 ( 5, @product3 )    368
109                 ( 5, @product4 )    398
110                 ( 5, @product5 )    428
111                 ;
112
113 centerProducts = fun    ( n, () )   ()
114                          ( n, ( ( name, _, Image ) :: T ) )
115                          {
116                              animate ( tween {   target: Image, property: x,
117                                                  toValue: getXFor ( n, name ),
118                                                  duration: 200, easing: Exponential_easeOut } );
119
120                              centerProducts ( n, T );
121                          }
122                          ;
123
```

Example: hscript

```
var script = "
    var sum = 0;
    for( a in angles )
        sum += Math.cos(a);
    sum;
";
var parser = new hscript.Parser();
var program = parser.parseString(script);
var interp = new hscript.Interp();
interp.variables.set("Math",Math); // share the Math class
interp.variables.set("angles",[0,1,2,3]); // set the angles list
trace( interp.execute(program) );
```

Bye bye!

- http://www.sephiroth.it
- http://aviary.com
- http://www.alittleb.it