Endogine

– a rapid multimedia and game development toolset –

Background

When I first encountered Macromedia Director in 1994, I was impressed by the capabilities and the toolkit that came with the program, version 4 at the time. Playing with it on and off for a couple of years, I started doing real work with Director in 1996. It had then reached version 5, which was a vast improvement over version 4 (MOA, Xtras, debugger, watcher, multiple casts and more). 1997 saw Director 6 (better sprite engine, behaviours were introduced, better Shockwave support), which finally, after a number of patches and bug-fixes, became a stable product. I never touched Director 7, released in 1998. I didn't really need the new features, and the word was that it was very slow and crash-prone, so after a long wait of 3 years I got Director 8 in 2000. Stable, much faster than version 7, and sporting a Lingo interface to image manipulation. Then, just a year later, 8.5 was released, with the 3D engine.

This is the version I'm still using. Director MX, released in 2002, had a new GUI that terrorized the user (and had few other features), and although MX04 in 2004 seems to have fixed most of that, the new features aren't that useful in what I do. The money is not an issue in itself; I just don't want to pay for mediocre products.

But I do need new features, and an updated sprite engine. So I go to Plan B, which is to not count on Macromedia to release the upgrade that I need, but write it myself.

Creating my own system allows me to port it to other platforms and IDEs. I've done a lot of work in C# the last year, including database apps, web applications and tools such as a file renaming utility, and a sound recording app, and I really like the language. The high-level multimedia capabilities and tools are missing though, so I'm going to port the engine to C# eventually (making it available to other languages supporting CLR – VB.NET, J# and managed C++). I truly believe the .NET platform is the way of the future, especially now that it exists on OSX and Linux as well.
When the .NET install base is large enough in a couple of years, and when you can run CLR in your browser, and the Universal 3D format is established, Director is dead. I even think that Flash (at least as an IDE, maybe not the format) will eventually go that way too.

This is my stepping stone to the future.

The systems

Here are the systems I've been working on, a little about their usage and what I plan to add.

Animation system

Create an animation object, load/create an animation chart, map its output to an object's parameter, and you're off.
You can animate things like a sound's volume, a sprite's loc, a 3D models size, or even another animation object's speed.

Interpolation between animation keys is made with splines, so when you lower an animation's speed, you still get a smooth movement. TODO: Add Bezier curves handles and time weighting for better user control over interpolation.

Use the **Score Ripper** to automatically convert score data to a format which the animation objects can use.
Then use the **Clip Player** to automatically play that whole score within the Plan B system. The Clip Player automatically creates/destroys sprites as it goes along, and creates their animation objects for them. You still have full control over the sprites created from the Clip Player - you can remove their animation objects at any time, add new ones etc. TODO: add animation import functions to support other formats, e.g. 3D animations such as MoCap data. Modify the Clip Player to play these "3D clips"

There is a form of simple **animation scripting language**, with "goto", "gosub", "wait", "randomValue", "randomGoto", "randomGosub" and other commands. This makes it easy to create animations that aren't the same all the time, thus keeping the viewers' interest longer.

TODO: implement support for the **SVG animation format**, both import and export.

Sprite engine

The engine doesn't use Director's sprite engine at all, but imaging lingo. If some behavior of the engine doesn't suit your need, all code is available for modification. (Although I might create an Xtra someday to take care of the most CPU-intesive parts. That would unfortunately make it less easy to modify for people without C++ skills)

Some features

Sprites can use images directly, not just members

Hierarchical sprite structure. Parent/child nodes. TODO: implement XML/DOM commands for accessing nodes.

Pixel perfect mouse detection, regardless of "ink"

Keyboard input, with proper detection of which key was released on keyUp, is propagated to sprites.

Lots of new inks, from jbCopyPixels Xtra. Source code for it is available, so you can create your own inks as well, with just a little C++ knowledge.

Easy creation of **custom sprite classes** by deriving from the Sprite class (or another class deriving from it).

Text editing/rendering. Proprietary text editing engine works like your OS's (TODO: OSX), not like Directors. Copy+paste works with other applications, Ctrl-arrow keys jump word-wise, etc. (This is a class derived from the Sprite class)

Render2Image. Draw to an image instead of the stage. If this image has an alpha channel, the merging of sprites will be done just like Photoshop does when merging layers.

Render2Plane. Sprites can render to Planes in a 3D world, which can give excellent performance. Rotation and other warping effects look much better as well. TODO: optimizations

Supersampling. Make sprites move in sub-pixels steps, get nicer-looking rotation etc. Individual setting (on/off, amount) per sprite, since it's CPU hungry.

TODO: **Render2Iso**. X,Y and Z coordinates are used to render into an Iso grid.

TODO: **Space partitioning** for optimizing rendering

TODO: fast line-based **collision detection** (space partitioner -> rect -> convex hull -> concave hull)

TODO: Implement **dirty rects** (pretty easy)

TODO: Implement **relational property definitions**. Like "button's right edge = text's width + 10". This makes auto-fitting very easy, like Menubar buttons clinging to the right-hand border of a window, and if you have a product that supports several languages, the GUI will adapt automatically to the new texts. Not a trivial task – I must calculate the order in which sprite's rects should be calced, and see to it that relations aren't circular (sprite 1 links to sprite 2 which links to sprite 1) – but I've done it in another engine. Some parts of my current sprite script must be altered for this to work, though.

TODO: **Flash support**. Some work has been done, but it's far from complete. The goal ist to be able to read/render shapes, bitmaps and convert animation data to my own format. ActionScript execution could be implemented, but I don't find that a perticularly interesting field
Flash could be used for content creation/layout by artists who like the tool, then the coding is done in Director. Note that 3D rendering of Flash content would come practically free, and that Flash clips/shapes can be freely mixed with other sprites (e.g. stick a sprite between two shapes in the Flash movie)
Flash can be used for the animation work, but when run from within Endogine, the objects being animated can be other things than shapes – 3D models, for example.
The Flash parser can also be used by people who want to write Flash generators (e.g. a server-side shockwave movie that generates Flash files for viewing in the user's browser), or batch tools for Flash files.
SVG support wouldn't be hard to add, once Flash is in place.

Editors/Tools

The editors can be brought up at any time. In Authoring and Projectors (TODO: as MIAWs) or in Shockwave.
TODO: a **data management system** that automatically connects to a cvs server, or SourceSafe. All resources in the system are external. When you load a movie, resources are copied/checked out from the cvs. Edit something (scripts or other resources), and the resource is checked in. If someone else updates a resource while your editing, you're notified and you can choose to download it immediately.

vCorse

Versatile Score editor. Edit 2D sprites, 3D models, or just about any object/property over time.

- Multiple vCorse windows at the same time - different aspects can be viewed/edited simultaneously
- Create groups of objects for easier editing
- TODO: Use SQL queries to find specific objects in the score
- Spline editor (for animations)
- Objects with children: Doubleclick to open editors for that space (just like Flash, with clips within clips)
- The whole score data is available for editing in list or XML form, making batch processing easy.

Painter (TODO)

Paint directly onto the sprites on the stage! Each sprite acts much like a layer in Photoshop. You can also open a separate window for painting if you want to concentrate on a specific sprite/bitmap.

Prototype and Resource Manager

The member/castlib replacement.
Protoype. An assembly of member(s) and a script. Not like the member script, though! It's more like a script with default settings, including pointer(s) to graphics. It's a bit like a behavior, but instead of dragging a member to the stage, then drag a script to it, you just drag the prototype directly. The prototype definition itself is a little struct of settings. This makes it easier for non-programmers – "editors" – to create scenes and GUIs. For example, you need 4 direction buttons for navigation in a movie. The programmer creates the button script, imports the graphics, and defines 4 prototypes, one for each button.
Of course, the sprite and script created with the prototype can be accessed just like any other sprite.
Technically, a prototype is much like a member type, but the programmer has full control over them, and can create new "member types" without delving into the C++/Xtras realm.

Prototype manager. There are many ways in which you may want to group your prototypes. By type, by name, or by some user-defined property. The prototype manager lets you do this easily, and you can search for prototypes with SQL queries if needed. The manager also supports hierarchical structures, so you can browse your groups like from within Explorer or Finder, with a collapsible folder structure on the left pane, and the actual prototypes on the right. 041203: Let models in 3D members appear in the manager as well, just as any other resource. The same with Flash elements?

Property Inspector

Under construction. All properties of a sprite are accessible here. When you create your own, derived sprite classes, the new properties that you have added will also show up here.

Window/Widgets/Controls library

These classes are derived from the Sprite class (or from another Widget class). It's easy to create new classes and expand the library. Automatic TAB control focus, user settable. Currently implemented classes are

- Frame
- Dragbar
- Button
- Window
- Scrollbar
- UpDown control

Roadmap

At the moment, I'm making rapid progress on several aspects of the engine, much because I need the features in a current game project. The core sprite engine is very functional, but there are a few design decisions about it haven't really made yet. Until I do, people should probably not use it for real projects, as its behavior may change.
When I release the Sprite engine, it will be open source. I hope that this can expand the functionality, by people writing new Widgets, new inks, improvements of the Animation object and Clip Player, better 3D support etc.

Completing the editors will be a big job, which I probably will have to do myself. I suspect I may have to make them commercial in order to stay out of financial trouble.

Suggestions or comments? Mail me at
info@endogine.com

/Jonas