WHAT'S NEW IN LM2
9/12/2001

This is an overview of the new features in LiveMotion 2, beginning with changes to design and animation features, then covering the addition of player scripting and automation scripting/Live Tabs. Please send feedback to John Nack (design/animation; jnack@adobe.com) or Henry Lee (scripting; hlee@adobe.com).

DESIGN/ANIMATION

::> Photoshop integration

You can drag a layered Photoshop document directly into LiveMotion and animate it as a regular object. If it contains layers, you can convert layers into objects or into a sequence using Object-> Convert Into. If it contains layer sets from Photoshop 6, you can convert each layer set into an object; selecting a layer set and running Object->Convert a second time allows you to make it into individual objects or a sequence. Layer masks and effects should be preserved no matter what; blending modes should be preserved relative to other layers in a given layer set if the set is not converted. Photoshop text and vector objects should export to SWF as vectors unless they have a bitmap-causing effect applied.

The PSD will remain editable in Photoshop even after you convert into objects and/or animate. To edit, select the PSD and choose Object->Edit Original. You'll be working on a copy of the original (stored inside your LIV source file as soon as you drag it into LM), and LM will update it when you save in Photoshop and return to LM. Creating or removing layers in Photoshop during Edit Original can cause unpredictable results.

::> Illustrator integration

LiveMotion 1 could bring in Illustrator artwork and allow it to be re-edited externally, but it had a number of serious shortcomings: only AI8 files and below could come in, manipulating artwork was often slow, layers converted to objects lost their layer names and all had the same size bounding box, and objects would often be rasterized when exported to SWF. All these problems are remedied in LM2.

Illustrator is integrated into LM2 much like Photoshop. You can drag a .AI file from Illustrator versions 7-10 directly into LM and animate it or convert the layers into objects or sequences. You can convert the hierarchy of AI objects as far as you want (even to individual paths) by running Object->Convert repeatedly on selected artwork. Try making a couple of layers with sub-layers and see what happens. If you convert down to the path level, you can manipulate the path with LM's pen tools, but you won't be able to return to Illustrator. Also, creating or removing layers in Photoshop during Edit Original can cause unpredictable results.

LM2 preserves AI effects (which work like Photoshop filters) and lets you re-edit them via Edit Original. Unlike LM1, we now preserve transparency information, layer/object names, and correct bounding boxes from AI files. We also preserve AI gradients and text as SWF-native equivalents, and we rasterize only non-SWF-friendly things like shadows, blurs, gradient meshes, etc. Alternative blending modes may also be discarded.

AI files must be saved with PDF support enabled, and AI symbols will not be recognized by LM unless the objects have not been scaled, rotate, faded, etc. LM should be able to recognize the artwork in any PDF, so you can put content in from any tool that can print to PDF via Acrobat (Word, PowerPoint, etc.). LM will recognize only the first page, but you will be able to break it into objects and keep it editable in Illustrator.

We plan to document ways to integrate LM into a business/Acrobat workflow and welcome your thoughts. With the help of Live Tabs (see scripting section), LiveMotion could be an excellent presentation tool "when PowerPoint just isn't enough." We will ship a PowerPoint import plug-in that makes it easy to stick LM SWF content back into PPT.

::> After Effects integration

LM2 will be able to import AMX (Adobe Motion Exchange) files created by After Effects 5.5 (due to ship around the same time as LM2). It should preserve keyframes, objects, nested comps, masks, sound, and path text. This feature is not yet ready for testing.

LiveMotion's QuickTime export, in conjunction with the ability to automate any set of tasks in LM using JavaScript, should also make LM an attractive way to feed content into After Effects.

::> GoLive integration

GoLive allows you to place a native LIV file directly onto your HTML page and have LM create a SWF. When this SmartObject is updated, all SWFs based on it are updated. GoLive 6 adds the ability to detect replacement tags in the LIV file and change them via a custom GL interface. So, you could easily tag a text object and background in LM, and GL would allow you to swap the text, change the link, apply a new animation style, etc., without touching LM. A single LIV file can be linked to numerous derived SWFs throughout the GL site.

Since GL and LM are both fully JavaScriptable and can pass each other scripts, this is just one of many possibilities. We welcome other ideas.

::> Flash integration

LM and Flash can't read each others' files, but they can load each other's SWFs via Load Movie. The process should work just as it does in Flash (loading SWFs into movie clip targets, sending and receiving data from them, etc.) without regard to the application that created the SWF. We will try to make our SWFs import more easily into Flash, and we will investigate shipping a SWF importer for LiveMotion (due after we ship 2.0 itself). We would appreciate your testing LM SWFs and Flash SWFs together.

To repurpose Flash ActionScript for LM, it's easiest to use #include files. About 90-95% of the code will come through without incident, though LM had different rules for supporting certain functions like eval. See below.

::> Timeline enhancements
::> Time stretching
::> Scrubbable movie clips

The LM timeline has been brought more into line with After Effects. LM2 supports basic AE

keyboard shortcuts (P to reveal position keyframes, OTSU, etc.). These shortcuts function only when the timeline in the foreground

You can drag objects to re-order their z-depth & you can drag the separator between the names are and the main timeline. Objects can now be locked, hidden (not visible on canvas and won't export to SWF), or made shy (not visible on timeline but will export to SWF). You can use the hand tool to scroll the timeline vertically or horizontally; the easiest method is to hold down H while using any tool, adjust the timeline, and release. To rename an object, select it in the timeline and hit Enter. (This name also functions as the instance name for scripting.)

Time stretching is one of the coolest things in LM2. You can redistribute the frames of an object by Opt-dragging the end of its duration bar. To test, animate an object, duplicate it, offset the duplicate by Cmd-Opt-dragging it (offsets the whole motion path at once), and Opt-dragging the end point of the duplicate. The duplicate should now run the same animation slower or faster than the original. You can time-stretch a movie clip to proportionally affect all the objects it contains, and you can Opt-drag one end of the duration bar through the other to reverse the animation.

You can scrub movie clips by holding down the shift key while dragging the current time indicator (CTI). To test, create an object, animate its position, duplicate it, offset the duplicate, select both, hit Cmd-Shift-G to make a movie clip, and drag normally. You'll see only one frame of the clip as you would in Flash. Now hold down shift while dragging. We also dim other objects when you dive into a movie clip (by double-clicking it on the timeline) to edit it.

::> Photoshop text engine

The text engine in LM2 is based on the one found in Photoshop 6. It allows on-canvas creation of point text and paragraph text objects that mix fonts, sizes, kerning, etc. One limitation vs. Photoshop is that LM supports one color per text object, but text objects can have multiple object layers. The text can be converted into objects (Object->Convert), each of which appears as a named object, or to paths (Object->Convert after the first conversion). Text will export natively to SWF and therefore be much lighter than when exported from LM1. Text attributes like tracking can be animated via the stopwatches under Object Attributes. Anti-aliasing options affect only bitmap output (the Flash player anti-aliases all embedded fonts and leaves all non-embedded fonts crisp).

LM2 also supports variable text fields for Flash forms, dynamic data interfaces, etc. Selective font embedding is coming soon to the export palette, so you'll be able to embed just the characters you need for a particular text field. LM SWFs will also render simple HTML (same tag set as Flash), including URLs.

Note also LM's Maintain Alignment feature. Create some text, place it over an object (e.g., a rounded rectangle, group the two, and choose Object->Maintain Alignment. LM will now adjust the other grouped objects to align them with the text object. This is quite powerful in conjunction with automation scripting for batch production (style guides, localization, etc.).

::> Streaming sound

It was difficult to synchronize sound and animation in LM1, but the addition of streaming sound allows LM compositions to force the Flash player to drop animation frames in order to keep up

with audio. To use streaming audio, place a sound on the timeline and choose Streaming from the properties palette. This approach has exactly the same advantages and disadvantages as it does in Flash (notably that looping streaming audio increases SWF size with each loop, and that MP3 compression is problematic for streaming sounds inside movie clips).

::> MP3 import

LM should support import of MP3 audio files, though the current build does not. MP3 compression should be respected on export to SWF unless you
specify different settings.

::> QuickTime export

LM2 will export QuickTime video files. Its bitmap handling should be much better than what you find in Flash itself, and it should support alpha channels depending on the codec you choose. QT export will finally provide a good environment for the use of LiveMotion's ability to animate bitmap characteristics like softness, distortion filters, etc. We won't support native QuickTime interactivity (wired sprites and filters) in LM2, though we've discussed adding it in a future release.

::> Off-canvas objects
::> Zoom below 100%
::> Larger maximum size for objects

These three changes smooth some rough edges from LM1. Now if you position objects off the edge of the canvas, you'll see their bounding boxes and an indication of orientation. You can reduce magnification below 100%, useful for off-canvas work and large compositions. And LM can now render objects larger than 1024 pixels wide and/or high. The new upper limit is
2048 pixels.

::> Workflow support

LiveMotion has built-in WebDAV check-in/-out capabilities on par with those in Illustrator 10. In addition, its Library, Styles, and Sounds palettes can be pointed at local folders that reference WebDAV servers. Therefore when a user adds an element (e.g., a script style), it is added or updated on the server and can be seen by other team members with the same setup. If GoLive 6 Workgroup Server or another management tool is in place, the files will be checked into the workflow.

::> Limitations

* Because of all the architectural changes necessary for scripting, Carbonization, etc., we had to drop support for Photoshop filters in LM2. We found that most people were doing their filtering work outside LM, so we focused on preserving Illustrator effects and Photoshop layer effects.

* When you zoom in on objects in LM, they will have rough edges even if they will export to SWF as vectors. To know whether an object will export as vector or bitmap, turn on Active Export Preview in the view menu and select an object. LM will display a vector or bitmap indicator icon at the bottom of the composition window, along with size when exported to the selected format.

* LM2 won't feature a Flash-style per-file library (useful for managing assets in large projects). This will be a top feature of LM3; in the meantime, please emphasize the collaboration possibilities in our library and styles palettes.

* Batch Replace HTML as a feature (which could mass produce bitmap images based on an HTML source) is gone, replaced by automation scripting (far more flexible and interesting) and GoLive integration.

* We would like to be able to place QuickTime video inside LM and export the frames to bitmaps in SWF, but this feature won't ship with LM 2.0. It may ship later as a plug-in. SWF import could also happen post-2.0 as a plug-in.

* Finally, we won't support Flash shape morphing or Generator templates.

SCRIPTING

In brief:

There are two main incarnations of scripting in LiveMotion 2: player scripting, which is the creation of scripts that control the Flash player (basically, Flash 5 ActionScript); and automation scripting, which is the creation of scripts that control the LiveMotion authoring environment (doing anything that a user could do manually). Automation scripts can be controlled by Live Tabs, our name for LiveMotion files being run as palettes that control automation script-based tasks (adding interactivity, modifying artwork, pulling in remote data, etc.). Alpha tour feedback indicates that automation scripting is *the* killer feature for developers in LM2.

Player scripts and automation scripts can both be previewed internally and analyzed by LM's source-level debugger (lets users set breakpoints, watch variables, step through line by line, etc.). By way of comparison, Flash 5 is not scriptable and contains only a rudimentary debugger that displays values over time.

The sharing of LM scripts on the Adobe Xchange will let us build community around the application.

In depth:

::> Player scripting

We've replaced all the behaviors from LM1 with JavaScript-based scripting. Through scripting, you should be able to recreate all the interactive content you've seen in Flash 5. Scripting can be applied rapidly by drag-and-drop (scripts can be stored in Styles alongside animation, rollovers, and layer effects), cut and paste, and automation scripts/Live Tabs.

The scripting user interface remains in flux (download a build of LM to see it in action), but it contains some key features:

* Movie Clip browser: displays all movie clips. Double clicking on the movie clip will jump you into the script for that movie clip. If you are coding for multiple movie clips, it saves you from having to go to the composition/timeline to switch between objects.

If you have an invalid movie clip name (one that uses spaces), they will be highlighted in red. The name is only invalid in terms of scripting. In JavaScript, object names cannot have spaces.

* Backward and Forward buttons: These will function similar to the Forward and Backward button in a web browser, navigating the most recent script blocks. The buttons are able to navigate to other movie clip views if this was recorded in the history. They only navigate for a single composition, however. For example, if I was on the onEnterFrame script of object one, then switch to the down state of object two. The backward button would allow me to jump back to the onEnterFrame script instead of having to go and find the object again and select the correct event handler.

* Handler/State script/Key frame view buttons: Clicking on each button sets the View options menu drop down. The menu displays all the possible event handlers/states/key frames for which you can write code. For example, clicking on the handler button and selecting the onMouseMove from the menu jumps you into writing code for the onMouseMove event.

The only state available for a movie clip is "normal" unless you create more states in the rollover (to be named states) palette. Once those states have been created, the respective states will appear in the drop down menu.

The same constraint for states is true of key frame scripts. In order to have scripts appear for key frames, you have to go into the timeline and click on the scripts button at the current time marker. Once you've created a key frame (even if you don't write any scripts), the key frame will be available from the drop down. The script indicator (blue corner on the button) indicates if there's a script in any of the views that's associated with that button.

* Player DOM browser: displays all properties and methods available in scripting. See below about syntax browsers.

* Target Browser: displays all of the movie clips. clicking on one and clicking ok (eventually just double click) will drop the path to that movie into your script.
* State Browser: displays all movie clips' states. clicking on one and clicking ok will drop in the name of the state. used for certain methods in scripting.
* Label Browser: similar to state browser, instead it displays the labels on the timeline of a movie clip.

[Note: recently the Target/State/Label browsers have been merged into a single entity.]

* Automation DOM Browser: Similar to the Player DOM Browser, this is for writing scripts into Live Tabs. See below about Syntax Browsers.

* Find/Replace: This brings up the Find/Replace dialog.
* Syntax highlighting: This switches syntax highlighting on and off.
* Negative view: This feature will change the text from black on white to white on black.

On the left hand column of the script window, you'll notice that when you click on it, it creates a red circle. That's a break point. It's used in debugging to pause playback at a certain point and bring up the debugger.

::> Syntax Browsers

There is a syntax browser for both ActionScript and the automation DOM. Through this browser you can open the twisty for each object type, and it will display the functions/properties associated with that object type. By double clicking on the displayed function/property it will copy it into the script editor window. When you have a property/function selected you'll also notice that there's a description of it in the bottom windowpane. When you drop in a function, it'll also put in all the parameters as well so you'll know what to fill out.

To try this, open the script editor. Click on the Player DOM browser. With the browser open, select MovieClip functions. Open up the twisty. Double click on getURL(). In the script window, you'll see that script now contains getURL("string",_target,"string"). In the DOM browser it says what the parameters are. In the script window it tells you the type of the parameter it requires.

Note: double clicking on the function/property will not work in builds previous to 47. In those earlier builds, select the function/property and click the OK button to have the code snippet dropped into the script window.

::> Preview Mode

Clicking on the little finger icon at the bottom of the tools palette activates The Preview mode. The preview mode in LiveMotion 2 can execute essentially all the scripting/animation as the Flash SWF player/plug-in. The only limitations are things that work with external files such as loadMovie(). However, this does not mean that it cannot interact with data sources. You can still preview loadVariables and XML sockets.

The question has come up as to why we have a Preview mode in LiveMotion when you can just preview in a browser. The Flash player is great in that Macromedia made the player/plug-in to be forwards compatible. Though this is terrific for the web, this feature is very limiting when it comes to debugging. The player simply ignores anything it doesn't understand or errors. The LM2 preview mode does not ignore errors. Instead, if it runs across an error it will bring up the debugger. In addition, you can use the debugger and step through you code and see each line of code get executed on your composition.

::> Debugger

The debugger allows you to view the code as it gets executed. It also alerts you to any errors that may have occurred during code execution. The same debugger is used in preview mode, automation script, and Live Tabs.

Debugger window
There are six buttons at the top of the debugger.
* Play: executes the code until the next error or break point
* Stop: stops execution of the code
* Kill: stops and exits the code and debugger
* Step: execute the next line of code. It considers function calls as one line of code.
* Step Into: executes the next line of code. If it references a function, step into that function.
* Step Out: If you are inside of a function or loop, this will button will jump you out of the function or loop. Stepping out returns you to the code that called the function or jumps to the end of the loop.

There are three windows in the debugger. The bottom windowpane is the script. An arrow

indicates which line is currently being executed. The left hand window is the stack. It will help you identify which function or event handler you are currently debugging. The right hand window is the variables watch list. You'll be able to view properties and variables here.

Above the right hand window is an input field. You can enter in variable names in here and have that added to the variables watch list. Enter the variable name and click on the little bug button.

In the Scripts menu, there are three options for the debugger.
* Don't Debug
* Debug on Errors
* Debug at Start

These options determine when LiveMotion will bring up the debugger. The debugger will also come up if you had previously set a break point in your script.

::> Automation scripting

LM2, the application, is scriptable via JavaScript through both external files and through the script editor below. To bring up the script editor, either create a new file by selecting File/New JavaScript (CTRL+ALT+N) or open a .js extension file from File/Open.

Automation script editor
The Automation script editor behaves just like the player script editor. The editor has the DOM/Syntax browser, syntax coloring, and negative view. Any script that is created here can be saved out as a .js file.

There are two new buttons at the upper left. The first from the left is the play button. Clicking on this button executes all the script in the editor window. The second button only executes the script that is highlighted.

To create an automation script and make it available from the Scripts menu, copy or save your automation script to the LiveMotion 2.0/Scripts folder with a .js extension. The next time you start up LiveMotion, it will parse through the folder and display the script in the menu. Whatever folder structure you create in the Scripts folder will be pulled into LiveMotion.

::> Live Tabs

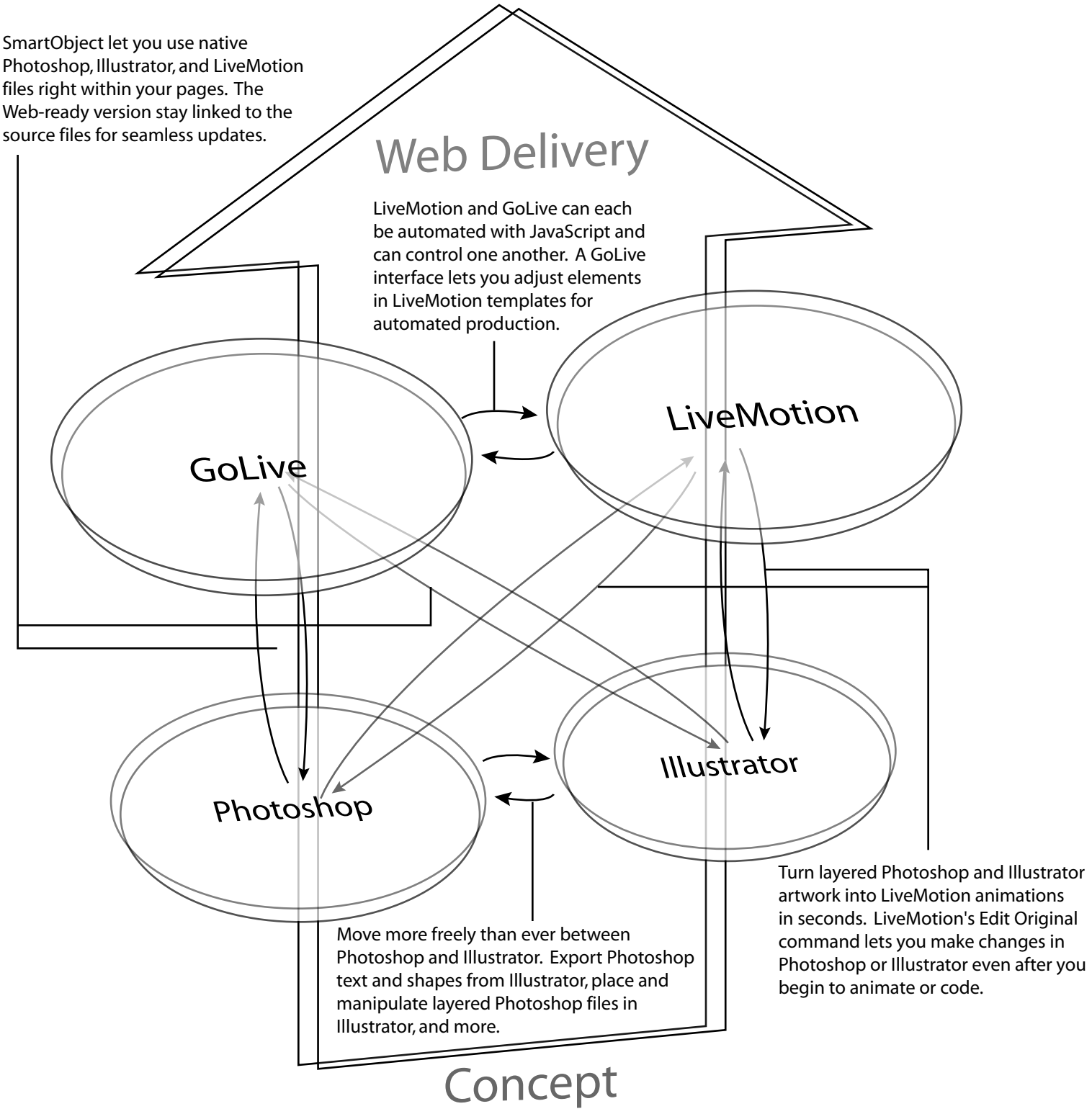
LM2 contains a preview engine that can execute essentially all the scripting and animation supported by the Flash player. Normally you use preview (activated by the little finger icon at the bottom of the toolbar) for testing and debugging your SWF compositions. But our engineers have also managed to use the preview engine to drive the interfaces of custom palettes called Live Tabs. Just as you could open a normal .LIV file and preview its animation, scripting, etc., you could load a Live Tab .LIV (by placing it in your Live Tabs folder, launching LM, and choosing it from the Live Tabs menu) and see its scripting and animation previewed the same way.

So, to create the interfaces for a Live Tab, you just need to know how to create a Flash interface; buttons, sliders, text fields, etc. work the same way (or will, once they're fully implemented). The difference is that when a user interacts with these controls, they're programmed to send commands to the LiveMotion authoring tool instead of just to the Flash player. That is, the controls in the Live Tab send out automation script commands. You use JavaScript to build the interface and send the commands, but you refer to a different set of objects in each case (movie

clip, button, etc. for the interface, pen tool, keyframes, etc. for the commands you send out to LM).

Since the underlying preview engine is the same, Live Tabs can do the same things as the Flash player (pulling in XML and other data, parsing it, etc.). This should allow some interesting things--for example, a Tab that allows me (the designer) to pull in a data feed, pop open the nodes of interest, and attach them to dynamic text fields, then export to SWF. The Live Tab would write the appropriate ActionScript onto my text field objects to locate and parse the data source.

The sharing of Live Tabs and automation scripts on the Adobe Xchange site will be critical to building community around LM2.


SmartObject let you use native Photoshop, Illustrator, and LiveMotion files right within your pages. The Web-ready version stay linked to the source files for seamless updates.
Web Delivery
LiveMotion and GoLive can each be automated with JavaScript and can control one another. A GoLive interface lets you adjust elements in LiveMotion templates for automated production.
GoLive
LiveMotion
Photoshop
Illustrator
Turn layered Photoshop and Illustrator artwork into LiveMotion animations in seconds. LiveMotion's Edit Original command lets you make changes in Photoshop or Illustrator even after you begin to animate or code.
Move more freely than ever between Photoshop and Illustrator. Export Photoshop text and shapes from Illustrator, place and manipulate layered Photoshop files in Illustrator, and more.
Concept