Emma: An Extensible Multimedia Architecture

Chris Marrin chris@emma3d.org
Rob Myers rob@emma3d.org
Murat Aktihanoglu murat@emma3d.org

8/18/2005

Abstract

Personal computers and the World-Wide Web have reached a point where it is possible to bring a wide range of rich multimedia experiences to a mass web audience. Simple web-style authoring is essential in the production of this content. Emma is a language and runtime system which is easy to author and delivers a compelling range of multimedia experiences. We will show how, using open source technology, Emma serves a wide variety of application areas and makes this rich tool set available to every web participant.

Emma is a language and runtime system which is easy to author and delivers a compelling range of multimedia experiences. Using open source technology, Emma serves a wide variety of application areas and makes this rich tool set available to every web participant. We will show how Emma makes a leap from a traditional declarative media system to an interface-based object-oriented system where prototypes can communicate with each other through their 'definition of capabilities', existing content can utilize prototypes by introspection and each content can become a template for the prototypes it uses.

Introduction

Rich media[1] on the web

Over a decade-plus of rapid evolution, the World Wide Web has moved from its original hypertext document roots towards an increasingly rich, multimedia presentation experience. Today, driven by the video game revolution, a new quantum of graphics processing power has emerged across laptop and desktop computers. This occurrence presages a new era in the Web encounter.

Towards a cinematic experience

In an increasingly aggressive attention economy, more sophisticated media are called upon to reach a post-literate audience more effectively. Game-class graphics enable a new class of interactive narrative material, a new "steerable media" delivering the visual impact of film and video, which can be nimbly and continuously driven where the audience wants to take it.

This caliber of "non-game" multimedia content offers the compelling engagement of a video game experience, with potential to deliver a more meaningful payload. Such active first-person encounters with the subject matter compete well for the attention of the ipod generation. At the same time, the new graphics hardware muscle allows this content to transcend the "cartoon" appearance of earlier generations; with care, today's systems are capable of real-time video synthesis[2] indistinguishable from a video broadcast.

While synthetic video is much more compact than the best video compression, and thus well-suited to internet distribution, it does require intensive computational resources to render. This concern has been addressed with the current generation of personal computers, all of which embed some form of hardware acceleration to accommodate this process

The chief virtue of steerable media, however, is the ability to go beyond the linear format of video presentation and involve continuous viewer interaction. Instead of presenting a static web page which waits for the user to make a choice, a steerable media approach opens with a full motion experience, and launches directly in to a cinematic exposition on the topic at hand. At any point, however, the viewer can exert navigational and interactive control over the presentation, whether to pursue a guided exploration, or to drill in for "cut to the chase" answers.

Within this medium, concrete data visualizations and live simulations can bring web-based information to life. For example, the content applications driving our rich media work include:

- An interactive Digital Earth, where viewers fly into live data insets within a global geographic context.
- An interactive Digital Human, where visitors explore health exhibits inset deep within the context of a comprehensive anatomical model of the human body.

Development in Web time

The real problem with producing such game-quality experiences for the web is that no web producer has the 18 months, cadres of specialists, and millions of dollars it takes to build a hit game title today. The key to bringing this new class of content to the web is to harness the

---

[1] Rich media combines 2D and 3D graphics with images, video and audio, and is fully animated and user-interactive.

[2] Video synthesis is the process of compositing multiple rich media assets together at 30 frames per second, producing a video signal generated on the fly.

power of advanced graphics behind more accessible mainstream web skills, tools and methodologies.

The web opportunity calls for episodic content, continuously and incrementally developed to keep the experience fresh and impactful. This approach requires a content development process which enables rapid time to market, continuous innovation, and data driven solutions. It should be possible to republish fresh material through previously established formats and templates, and to introduce new narratives, new views, and new insets into the experience as needed over time. In the same way, it is vital to integrate third party content and its evolution into an ongoing unified, seamless, unfolding experience.

Finally, and most challenging of all, the web's rapidly growing communities of user-generated content such as Wikipedia suggest that this new class of rich media content should be editable in place and online, directly within the presentation engine.

The dynamic media architecture outlined in this whitepaper is motivated by the requirements of this new form of presentation experience.

Prior work

This perspective on a "full motion" interface for the web has been driven by the authors' experience in developing content and content delivery architectures over the last 20 years. Key waypoints in the evolution of this media vision include:

- The Full Service Network: interactive television utilizing a set top video/graphics media interpreter.
- Inventor: Silicon Graphics' rapid 3D construction kit.
- WebSpace: the Web's first 3D browser.
- Cosmo Player/Worlds: VRML 97 browser.
- Blendo: Sony's experimental Steerable Media engine for PlayStation2 and the web.

Overview

What is Emma?

Emma is a language that allows the construction of rich-media, content-driven applications as well as the runtime system used to present them to the user in real time. Using a declarative[3] form, the author creates scene descriptions which tell the runtime how to present audio-visual information to the user, how to animate that presentation over time, and how to respond to input from the user.

Evolution

Emma has evolved from many new and existing technologies. It shares features of HTML, VRML, MPEG4-BIFS, SVG and SMIL. It also builds on 6 years of generative media research at Sony's U.S. Research Labs with the *Steerable Media* project and development of the Blendo video synthesis engine.

Emma provides several novel innovations related to these technologies. It is easily authored like HTML, and provides a rich vocabulary of multimedia tools for the author to choose from. It has the spatial richness of VRML, adding recent innovations in graphics technology and interaction. It has features from MPEG4 BIFS, such as layering and 2D integration. It also integrates the rich 2D vocabulary of SVG into its description language. And it has many of the media sequencing features of SMIL, applied to user interaction and simulation in addition to animation.

Extensibility

The Emma system consists of a core runtime and a set of objects the author applies to construct an application. One of Emma's primary attributes is its *extensibility*. The core set of objects in the engine is quite small. It contains no 2D or 3D rendering, no media processing, and no interactive capabilities. All of these are added through extensions. Fortunately, all of the functionality mentioned above is available in standard extensions that ship with Emma. They are preloaded and behave as though they are built-in to the engine. But they can be replaced with or enhanced by other extensions to customize Emma to the task at hand.

Extensions can be written in native C++ code, which can then be shipped with a custom Emma application, or extensions can be written in the Emma language itself, to be downloaded along with the content.

---

[3] With a declarative form, as opposed to a programmatic form, the author specifies objects and their relationship to each other in time and space rather than specifying a sequence of operations to achieve a given presentation.

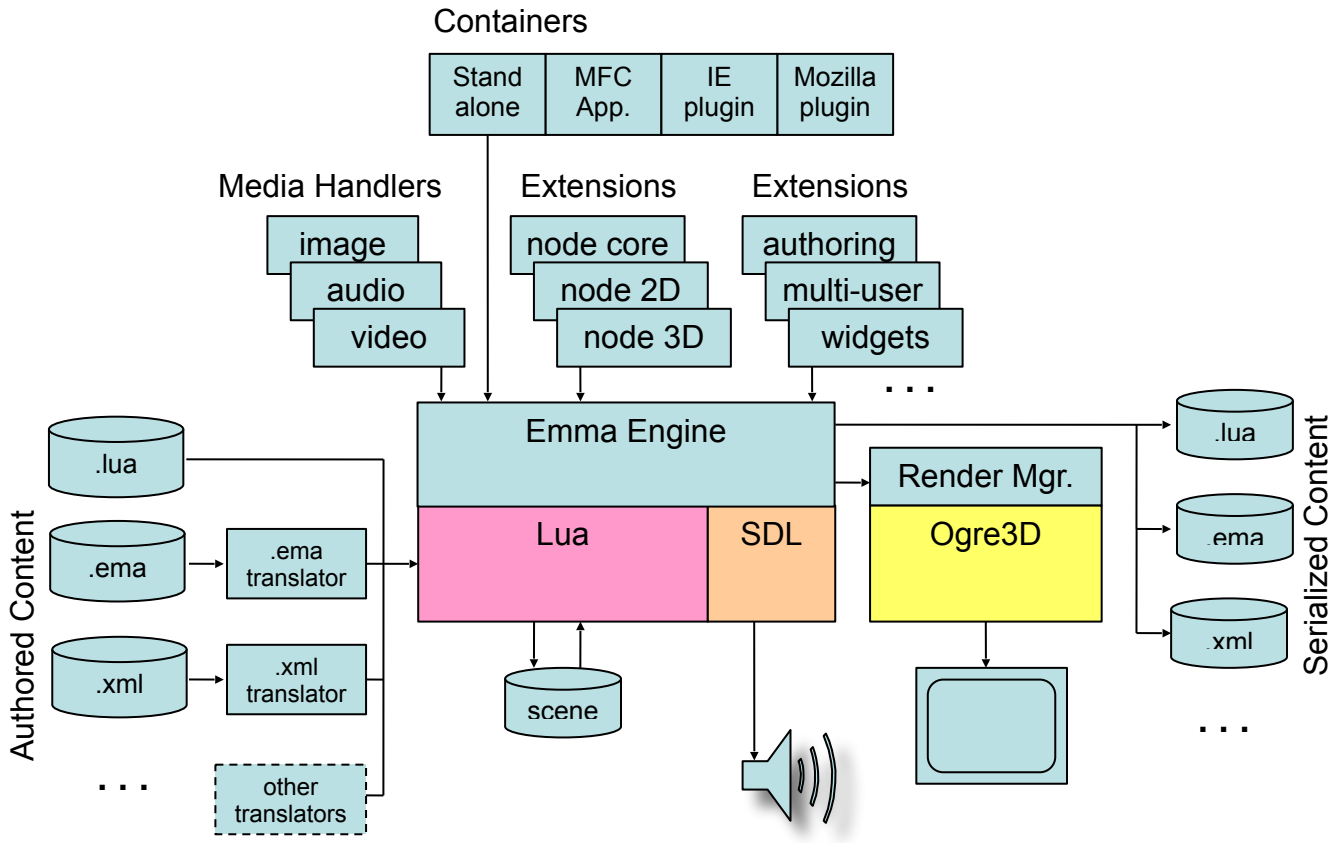

Figure 1. Emma's extensible architecture

Scripting

Another of Emma's central strengths is its use of Lua as an embedded scripting language. Declarative systems avoid much of the need for a programming language. But scripting provides the ability to handle new tasks not supported directly by the object set. Scripting is tightly coupled with the Emma runtime system. In fact, all Emma declarative content is ultimately translated into Lua scripts, after which it is loaded and run. Every object in the system, native or declarative, has a Lua aspect and is therefore accessible from scripts. And since the core of Emma contains an efficient C++-to-Lua interface, all objects are accessible from C++ as well.

Lua is significantly faster than most other scripting languages such as Javascript, Perl or Python. This allows Lua to handle more tasks, be they simple (like changing the color of a single object) or complex (like transforming every point of an elaborate mesh). Emma can render a 30,000 triangle surface, where each vertex is being modified every frame using Lua script expressions, achieving 60 frames per second on a 3GHz machine. Because Lua is integrated directly in the Emma language, it is extremely easy for authors to use whenever and wherever it is needed.

Language	CPU (sec)	Memory
C++	0.11	1148
Lua 5	**3.19**	**2712**
Python	6.22	4924
Perl	9.29	6068
Javascript	15.89	9984

Figure 2. Heapsort performance comparison (http://dada.perl.it/shootout/heapsort.html)

Lua provides many advanced features such as threading, true tail calls (making recursion of arbitrary depth possible), and closures. It also has an incremental garbage collection facility which will not impede demanding real-time rendering system performance.

Rendering

Unlike many common 3D web runtimes, Emma features advanced rendering capabilities, since it is built on the powerful and highly popular Ogre3D rendering engine. This open source project provides an object-oriented package of 3D graphics functions that are cross-platform

and support all major graphics hardware on the market. It provides access to the latest graphics features, such as multitexturing, multi-pass rendering, shaders and particle systems. Emma gives direct access to the powerful tools provided by Ogre, making it easy to create realistic 3D, animated objects without having to understand low-level graphics concepts.

Figure 3. An Emma scene with animated characters, particle systems and background image cyclorama.

Ogre3D is packed with advanced rendering features:

- skeletal animation, including blending of multiple animations, variable bone weight skinning, and hardware-accelerated skinning
- real-time shadowing
- particle systems (smoke, rain, fireworks, etc.)
- full Shader support (Cg, HLSL, GLSL, assembly)
- multitexture, multi-pass blending materials
- .bmp, .cut, .dcx, .dds, .ico, .gif, .jpg, .lbm, .lif, .mdl, .pcd, .pcx, .pic, .png, .pnm, .psd, .psp, .raw, .sgi, .tga, .tif, .wal, .act, and .pal texture support
- volumetric textures, cubemaps
- 1d textures, compressed textures
- automatic mipmap generation
- projective texturing
- skyboxes, skyplanes and skydomes
- Direct access to hardware vertex, index, pixel buffers
- High dynamic range rendering (HDR)
- Full Scene Anti-Aliasing (FSAA)
- Direct3D and OpenGL support

Emma loads Ogre file formats for 3D meshes, animations and appearances. So the many authoring tools that support these formats can be used in the Emma authoring toolchain (like 3DS Max). And we are creating more tools (such as VRML-to-Ogre and Collada-to-Ogre translators) to make it even easier to get high-quality 3D artwork into Emma.

2D Rendering

In addition to the 3D subsystem, Emma has a high quality 2D renderer. Rather than converting 2D to flat 3D shapes and rendering the results, Emma generates a high-quality, pixel-perfect, anti-aliased 2D mask. It then uses the 3D engine to layer this into the scene as the mask for a solid color, gradient, or complex multi-texturing operation.

The 2D mask is generated from a spline description containing multiple independent cubic and quadric curves, and straight lines. The 2D subsystem borrows from the SVG standard for functionality and form. This does not make Emma SVG compatible, but it does make it easy to convert content from SVG to Emma. TrueType fonts are also supported and can be used to render high-quality text strings or as outlines for textured shape rendering.

Unlike most 2D packages, which can only display screen aligned shapes, Emma can apply a full 3D transformation to the shapes. This is done while maintaining the pixel accuracy of the anti-aliasing, resulting in high-quality 2D shapes in the 3D scene.

How Emma works

Every Emma application consists of at least one *content file* in a supported format. Files in the `.lua` format (Lua source code) are supported directly by the core, while other formats are loaded using extensions. A handler for the `.ema` format (Emma declarative source) is preloaded with the standard distribution, enabling `.ema` scene description files to be read as well.

A loaded file contains descriptions of objects and property settings for these objects. These are created and structured into a *scene graph*. This graph is examined, or traversed, then graphics commands are sent to the graphics subsystem, animation commands are sent to the animation subsystem, and so on. The result is a visually appealing presentation which is animated and accepts user input from any supported input device, whether it is a keyboard, mouse, joystick or custom device.

How a content file is read into Emma depends on how it is being used. For instance, if running Firefox, you simply drag a .ema or .lua file into the window and the Emma runtime starts up and loads the content. Several other containers are available for running Emma applications. See ***Embedded or Standalone*** for more details.

A simple example

Emma is used to build a scene graph of nodes, each of which provides certain functionality. For instance, a `Shape` node is used to present geometry to be rendered. One example of geometry is the `MeshInline` node, which reads the filename (url) of a `.mesh` file. `.mesh` is an Ogre format that can be produced by a number of mainstream modeling tools. So, in order to display a mesh object you would simply write:

```
Shape {
    geometry MeshInline {
        url "robot.mesh"
    }
}
```

When this is loaded you will see a robot displayed. It will be textured because all of its material information is contained within the mesh file.

We can add a simple spin animation to this object by adding a `Timer` node. This node fires its `fraction` field on every frame with a value that increases from 0 to 1 over the set `interval`. A Shape node has a `rotation` field that can be manipulated by this fraction value:

```
DEF MyShape Shape {
    geometry MeshInline {
        url "robot.mesh"
    }
}

Timer {
    loop true
    interval 5
    fraction DO {
        TOP.MyShape.rotation = {0,1,0,
            Math.PI*2*self.fraction }
    }
}
```

The `fraction` field has a `DO` statement attached to it; a `DO` script will be run whenever its field value changes. This script computes a rotational angle that completes a full circle as fraction goes from 0 to 1. The Y axis (0,1,0) is specified as the axis of rotation. A name is assigned to the Shape node, so it can be accessed by the Lua script. Note that the 4 values (0,1,0 and the computed angle) are surrounded by braces. This creates a Lua table to be passed to the rotation field of the Shape. Generally Lua primitive values and tables can be passed to Emma field types and the expected thing will be done. The same is true for copying between field types. For instance, an MFFloat array with 3 elements can be assigned to a SFVec3f.

Features of Emma

Emma is an *object-oriented*, *declarative*, *extensible* language and system. *Object-oriented* means that it is made up of objects which establish its functionality. Some of these objects are built-in. The fact that Emma is *extensible* means that new objects can be added to the system to provide new functionality. This could be in the form of libraries of generally useful capabilities for a particular application (like common shapes, or e-commerce forms). Or it could be a set of extensions created specifically for a given application. Because Emma is *declarative*, objects can be combined into an application using an easy-to-use, HTML-like syntax. And extensions can be written in this declarative form as well as the application itself. This makes Emma very good at creating large applications, consisting of many components that change over time, and are maintained by many different developers with diverse skill sets.

Objects, Fields and Events

The Emma runtime system is made up of a set of objects. Some are built-in, others are loaded as part of the standard distribution and others are loaded by authors as part of an application. Each object appears to Lua as a table containing the properties of that object class. But like other *prototype delegation* languages (like Self and Javascript), many of these properties are not present in the object, but in its prototype. It is only when the property is set that a copy of it is made in the specific object. This saves space and allows an entire class of objects to share common initial state.

The properties of an object can contain Lua values: numbers, strings, tables, functions, etc. They can also contain Emma *field objects*. These are special objects which can have events associated with them. A field object behaves differently from a normal property when it is being set. A normal property will simply replace the value. But a field object will replace the contents of the field object with the value being set. For instance, if you assign a table with 3 numeric values to a property contaning an SFVec3f object, the 3 numbers will be placed in the x, y, and z elements of the SFVec3f.

You can also associate an event in the form of a Lua function with the field object. When the assignment occurs, the function is executed, passing a reference to the node which contains the field. This is a very powerful feature which gives Emma its ability to declaratively assemble nodes and fields.

Rendering, Animation and Behavior

The main activities of an Emma application have to do with *rendering*, *animation* and *behavior*. The runtime is a *continuous rendering system*. That means the display is fully refreshed at a real-time frame rate, typically 30 or 60 frames-per-second. The author can make use of many facilities to try to maintain this frame-rate, including background processing and scene simplification. Emma provides feedback to the author, including the current frame rate itself, to aid the author in making these choices.

Animation is key to the Emma system. Three separate animation styles are available. Since Emma consists of a scene hierarchy, each level of that hierarchy has an associated 3D transformation. Even 2D objects can be transformed in 3D space. And each transformation can be manipulated over time to move objects within the scene. This can be done programmatically using Lua scripting, as shown in the example in the previous section. Or a VectorInterpolator node can be used. This is a keyframe animation node similar to the PositionInterpolator or OrientationInterpolator nodes of VRML. After the keys

are setup, a fraction value is routed from a Timer to the VectorInterpolator and its output value is routed to the destination transformation. Many VectorInterpolators can be routed to different transformations, or to different parts of the same transformation, resulting in continuous animation of scene elements.

Ogre meshes also provide built-in skeletal animation, which can be activated from within the MeshInline node. A Timer node is added to control performance of the animation. A separate MeshAnimation node is added to name each animation activated from among the suite of animations available within this mesh. These MeshAnimation nodes are used to blend together contributions from each animation named.

Behavior gives users the ability to interact with the application. Emma has the ability to do traditional click navigation familiar to HTML content. But it can also give authors the ability to add continuous navigation functions. This can include anything from choosing from a set of animations, to full immersive walkthroughs. User input can come from the keyboard, mouse, joysticks or a number of other input devices.

Media

In addition to the rendering of 2D and 3D graphics, Emma supports a full set of audio and video media types. Media codecs are plugins and Emma is preconfigured with a common set. But codecs can be created for custom media types as well. Emma uses a streaming model of decoding, so large media files or live feeds can be decoded easily, as long as the connection has sufficient bandwidth.

Audio can be ambient, with its volume and placement under full control of the author, or spatial, made to emanate from a given position within the scene. Media uses a controlling Timer to enable variable playback rates, pause, rewind, and even scrub effects. These special effects are dependent on the media source. For instance, live video feeds typically cannot use 'trick play' effects.

In addition to audio and video, Emma treats content files as media streams. This means that codecs can be written to load content files into the engine. These codecs are really just translators which convert the source format into Lua scripts that the engine then loads and executes. But implementing them as media streams means that Emma can load literally any kind of file format directly into the engine.

Embedded or Standalone

The core of the Emma runtime is a DLL, shared or static library. Therefore it can be easily embedded in other applications. Four examples of this are shipped with the Windows version of Emma. Emma comes with plugins to allow it to be embedded in Firefox or Internet Explorer. This allows multiple windows of Emma content in a single HTML page. There are also two standalone applications that ship with Emma. One is specific to Windows called MFCApp and allows you to open many Emma scenes in an MDI interface. The other is a simple application called Emma which allows you to open a file or drop a file onto it to be run. You get all these applications, ready to run, when you install Emma on your system.

Built-in authoring

Because of its extensibility mechanisms, authoring tools can be built on top of the Emma runtime itself. Authoring extensions will allow a scene to be loaded, modified, then written back out. With Emma's GUI extension, widgets specific to a given node can be written to control that node's functionality.

Emma supports the notion of *smart media objects*. These are normal Emma Prototypes, but with authored techniques concerning how to edit and interconnect them. The editing methods can be anything from simple generic property sheets to a fully custom GUI widget, or even in-scene direct manipulation editing techniques.

Emma also takes advantage of the large number of debugging aids for the Lua language. Syntax directed editors can be used to write Lua scripts, and debuggers can be used to set breakpoints, single step the application and examine Lua variables. Since the entire Emma runtime environment is available to Lua, this allows the debugging of all aspects of the runtime.

External communications

Emma applications can communicate with other elements outside the runtime engine. This includes interactions with the HTML page in which the content is embedded, and access to server-side databases. Incoming data can be XML formatted, or XMLRPC can be used to structure it. Outgoing data can be formatted as simple strings, XML or XMLRPC calls.

Emma can also store data locally in a persistent object store. This allows applications to save user preferences or state information for the next time the application is run or to pass this information on to other applications.

Open source

Emma makes extensive use of open source software. Ogre, Lua, SDL, expat, curl and wxWidgets are all used. These were chosen because they are cross-platform libraries, running on Windows, Macintosh and Linux systems. The use of these well supported libraries enhances Emma's stability and makes available a large support network. With Emma's extensibility model, these libraries also make it possible to incorporate additional functionality designed to work with them. Lua especially has a large array of support libraries, giving socket access, web server functions and many others.

Emma is also in open source, which allows it to be moved into other areas of interest, limited only by the energy of those in the open source community.

Current status

Emma is a work-in-progress. The current version is 0.3 and source or binaries can be downloaded by going to http://www.emma3d.org. This includes much of the functionality mentioned here. But a large amount of functionality is still to come.

Future work

Leveraging Emma's extensible framework, we aim to systematically integrate an open-ended range of high-quality media and dynamic behavior vocabularies into this full-motion interactive platform. Development items for the future of the Emma runtime include:

- Linux and MacOS X builds.
- Advanced media cache management, including extension of support for Internet-distributed Ogre assets.
- Integration of a broadcast-quality 2D rendering package, which brings SVG-style vector graphics, typography, layout, animation and anti-aliased rendering to both 2D screens and immersive 3D scenes.
- A full suite of multimedia components, including sound, image, and video codecs, mixers, and compositors. These will seamlessly integrate with the texturing and alpha-blending capabilities of Emma's architecture, and unify with the core animation playback and interactive event controls.
- Navigation kits for object viewing and for in-scene navigation, including policies based on collision detection and terrain-following.
- An extensible range of VectorInterpolators, including an efficient MeshInterpolator and extensibility for spline interpolators.
- Interactive triggers and scene sensors, advanced timer features, and scoring support for linear media segments, e.g., machinima-type applications
- Scene-embedded GUI components (counterpart to the existing external wxWidgets), plus layout kits to support automated and dynamic spatial layout for objects in 2D and 3D.
- Application Surfaces for embedding third-party rendering onto surfaces
- Physics kits for the efficient application of physics properties for in-scene objects.
- Multi-user kits provide a common infrastructure for avatar tracking and multi-user server interfaces.
- AI kits for writing objects that can algorithmically make decisions based upon scene feedback.

With authoring components plugged in, Emma grows to become a powerful and extensible platform for graphical media authoring, production, content debugging, and interactive simulation. Future investments in the authoring capability include:

- Serialization: any scene in the engine can be saved out into markup.
- Importer/exporter modules: Collada, Illustrator, and other file format conversions are slated to integrate Emma-based authoring with mainstream modeling, animation, illustration, and media tool suites.
- Debugger: Extensions to the core for debugging content structure, scripting, and routing are in the works.
- Authoring tool framework: We have already integrated WxWidgets to provide cross-platform support for tool interfaces. This work will be followed up by further investments in tool infrastructure, to assist the developer of Emma-based applications.
- Assembly editor: a GUI framework for interactive scene assembly and behavior composition.
- Composable production tools: Emma's framework supports rapid development of both custom and re-usable production tools throughout the production pipeline. A library of "mini-solutions" is expected to accrue over time.

References

The Lua Programming Language
http://www.lua.org/

The Ogre3D Project
http://www.ogre3d.org/

COLLADA - An open Digital Asset Exchange Schema for the interactive 3D industry
http://www.collada.org

VRML97 Specification (ISO/IEC 14772-1:1997)
http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-IS-VRML97WithAmendment1/

X3D Draft Specification
http://web3d.org/x3d/specifications/x3d_specification.html

Scalable Vector Graphics (SVG) 1.1 Specification
http://www.w3.org/TR/SVG/

XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition)
http://www.w3.org/TR/html/

The Great Win32 Computer Language Shootout
http://dada.perl.it/shootout/

wxWidgets Cross Platform GUI Library
http://wxwidgets.org/

The Expat XML Parser
http://expat.sourceforge.net/

libCurl Library
http://curl.haxx.se/

Simple DirectMedia Layer (SDL)
http://www.libsdl.org/

ISO/IEC 14496 (MPEG-4)
http://www.mpegif.org/mpeg4/

Open Inventor
http://oss.sgi.com/projects/inventor/

CosmoPlayer browser
http://www.karmanaut.com/cosmo/player/

Steerable media: interactive television via video synthesis
Chris Marrin, Rob Myers, Jim Kent, Peter Broadwell
Proceedings of the sixth international conference on 3D Web technology, 2001

Synchronized Multimedia Integration Language (SMIL 2.0)
http://www.w3.org/TR/2005/REC-SMIL2-20050107/

XML-RPC
http://www.xmlrpc.com/