Lively HTML

Seminar Web-based Development Environments

David Jaeger and Robert Krahn

Hasso-Plattner-Institut, Potsdam
{firstname.lastname}@student.hpi.uni-potsdam.de

Abstract. This is an abstract ...

1 Introduction

The Lively Kernel is a collaborative and self-sustaining development and runtime environment for Web applications. It provides a rich and interactive graphics system by implementing the Morphic user interface. Currently the rendering of Lively is based on SVG graphics integrated into an XHTML document. The generated XHTML document can subsequently be displayed in the Web browser. When doing day-to-day tasks with Lively, the current rendering system works quite smoothly. However, performance glitches arise when a user wants to display a larger amount of text in a text field or has loaded hundreds of entries in a list element.

The reason for this negative performance impact can be seen in the Web browser, which is normally optimized for HTML rendering rather than SVG graphics. Furthermore, with the HTML specification, a Web browser has to provide a set of natively implemented input elements. These can be used as a counterpart to the SVG graphics lacking performance. Essentially, some SVG graphics have to be replaced with HTML widgets to achieve a better performance for text fields and lists. Unfortunately, an SVG scene graph accommodating all SVG graphics, as employed in Lively, does not play well with HTML widgets. Consequently, the only way to integrate HTML elements into Lively is to switch the whole rendering subsystem from SVG to HTML with a complete HTML scene graph.

In addition to better performance for text content, an HTML scene graph can bring more significant benefits.

Compatibility First, the usage of HTML promises higher compatibility between Web browser, as HTML is more standardized and accepted in browsers than extensions like SVG. This should also allow to run Lively in the Internet Explorer.

HTML Integration Lively can integrate HTML elements residing in a surrounding HTML document, by *livelifying* the HTML element and inserting it into its scene graph.

When HTML rendering can really bring the mentioned advantages, there might come up the question, why SVG has been chosen at all at the beginning of the Lively project. An the reason is that until now, it was impossible to implement the Morphic functionality with pure HTML, not even with the help of CSS. To be more precise, it was impossible to rotate or scale elements and draw complex shapes like ellipses or polygons. However, this has changed when the W3C[1] has proposed new versions for HTML and CSS, namely HTML5 and CSS3. HTML5 provides a new *canvas* HTML element on which complex shapes can be drawn and CSS3 allows to apply transformations on HTML elements.

Dan Ingalls, developer of the Lively Kernel, has already finished a prototype[2] rendering an entire Lively world in a single *canvas* element. Nevertheless, this prototype does not support the integration of HTML widgets, since no scene graph for attaching a widget is available. In the following sections, we propose a new approach for HTML rendering in Lively. We provide an explicit HTML scene graph. Furthermore, we make use of established HTML elements where possible and only use advanced HTML5 and CSS3 features when really needed.

We first start with a short overview on Lively's rendering model in section 2. After that, we introduce our approach for the integration of HTML rendering into the existing code. In section 4, we provide more details on the concrete implementation and show how it was possible to change the rendering system while we still needed a rendering system for development. Section 5 presents results of performance comparisons between SVG and HTML rendering. In 6, we introduce related work to HTML rendering in Lively. The last section concludes this report and provides some hints for future work.

2 Lively's Rendering Model

3 Concept

[3,4]

3.1 Replacing the Shapes

3.2 Implementation Issues

HTML Representation

Transformations

Serialization

Text

[1] World Wide Web Consortium

3.3 Event System

4 Implementation

In the following, we describe the implementation of Lively HTML.

4.1 Mapping

In the existing SVG implementation of the Lively Kernel the shape classes implemented in the module lively.scene are responsible for interacting with the Web browser's document object model to create, modify, and remove SVG nodes.

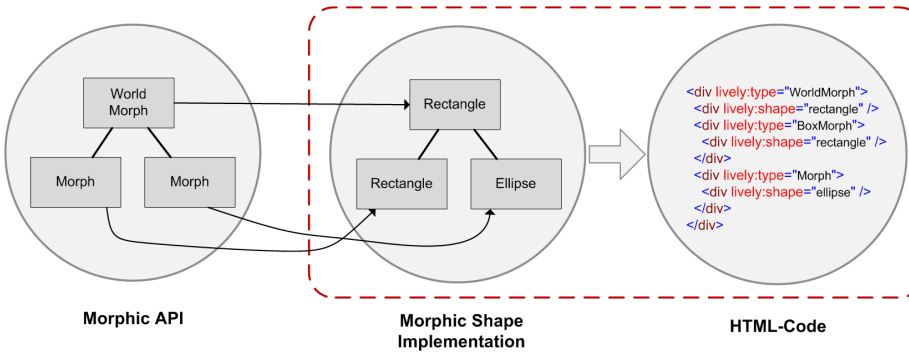

Fig. 1: The rendering system in Lively Kernel uses the Bridge Pattern [1] to separate the Morphic interface from its implementation.

4.2 Using ContextJS

As was mentioned before, implementing HTML-based rendering means to modify Lively's shape class hierarchy. In the existing SVG implementation shapes are responsible for interacting with the Web browser's document object model

to achieve - bootstrapping HTML - extension of the existing system without breaking SVG - load HTML and SVG world together on one page. We were then able to modify the HTML world using tools running in the SVG world

4.3 Widgets?

5 Evaluation

6 Related Work

7 Conclusion

This is a conclusion...

7.1 Future Work

References

1. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley (1995)
2. Ingalls, D.: Lively Canvas Implementation. Source Code (2009), http://www.lively-kernel.org/repository/lively-wiki/lively/CanvasExpt.js
3. Maloney, J.H.: Morphic: The Self User Interface Framework. Sun Microsystems, Inc. (1995), ftp://ftp.squeak.org/docs/Self-4.0-UI-Framework.pdf
4. Maloney, J.H., Smith, R.B.: Directness and Liveness in the Morphic User Interface Construction Environment. In: UIST '95: Proceedings of the 8th annual ACM symposium on User interface and software technology. pp. 21–28. ACM, New York, NY, USA (1995)