The XML3D Architecture

Kristian Sons*[1,2,3], Felix Klein†[1,2], Jan Sutter*[1,3], and Philipp Slusallek*[1,2,3]

[1]Saarland University [2]Intel VCI [3]DFKI

1 Introduction

Graphics hardware has become ubiquitous: Integrated into CPUs and into mobile devices and recently even embedded into cars. With the advent of WebGL, accelerated graphics is finally accessible from within the web browser. However, still the capabilities of GPUs are almost exclusively exploited by the video game industry, where experts produce specialized content for game engines.

XML3D aims answering the question of how we could make 3D graphics available to a broader audience. The approach is to target web developers, making 3D application development as similar as possible to web application development. Unlike previous approaches such as VRML, XML3D takes the capabilities of modern graphics hardware into account, without tying it to a specific rendering algorithm.

The novel concepts of XML3D, including its seamless integration into existing web technologies and its approaches for dynamic effects, programmable materials, and instancing of configurable assets are spread across various publications. The poster depicts the overall architecture of XML3D based on the mark-up of an actually running example scene (see Figure 1).

2 Our Approach

XML3D [Sons et al. 2010] is an extension to HTML5 that allows describing interactive 3D graphics in any web page. XML3D is renderer-independent and contains elements for geometry, lights, etc. It supports event attributes such as *onclick* and reuses existing HTML elements, CSS and other concepts wherever possible.

XML3D has a generic approach to data where users can define arbitrarily named parameters that can be used e.g. as mesh attributes, material parameters, or as input of our dataflow approach. These parameters can be clustered in data blocks which can be reused, specialized and composed from multiple sources. Structured 3D data (assets) can be externally defined and instanced multiple times within the scene [Klein et al. 2014]. Each asset can be specialized during instantiation by overriding values from within the asset.

With Xflow [Klein et al. 2013], data blocks can be composed in a graph. By attaching an operator to a data block, the graph can be transformed into a dataflow processing graph. Such dataflows can be used for geometry processing (e.g. skinning and morphing), animations, image processing (e.g. post-processing and AR), etc. Its declarative approach allows mapping computations to the GPU and other parallel processors via various supported APIs (e.g. WebGL, WebCL, ParallelJS, SIMD.js).

*e-mail: {kristian.sons, jan.sutter, and slusallek}@dfki.de

†e-mail: klein@intel-vci.uni-saarland.de

SIGGRAPH 2015 Posters, August 09 – 13, 2015, Los Angeles, CA.
ACM 978-1-4503-3632-1/15/08.
http://dx.doi.org/10.1145/2787626.2792623

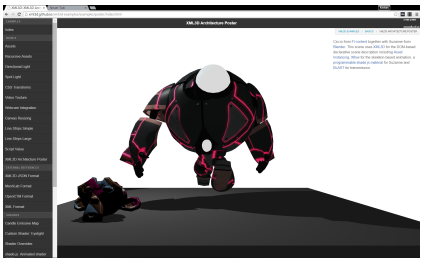

Figure 1: *A screenshot of the scene illustrated in the poster using XML3D as scene description running in Google Chrome.*[1]

Using shade.js [Sons et al. 2014], developers can write portable materials using JavaScript. It is renderer agnostic, adaptive and compiles to GLSL for forward and deferred rendering via OpenGL but can also compile to e.g. OSL for ray tracing and global illumination.

In XML3D, geometry data, lights and materials, but also generic data and dataflow graphs are resources which can be defined in the same document or by external resources, e.g. referenced documents or services. Those resources can be streamed to the client using Blast [Sutter et al. 2014], a novel format for the transmission of structured binary data. Additionally, Blast offers a flexible approach to compression based on a code-on-demand approach.

Despite its high abstraction level, XML3D offers mechanisms such as data flow processing and programmable shading in order to expose the flexibility of programmable GPUs. The polyfill implementation *xml3d.js* uses JavaScript and WebGL to emulate XML3D in standard browsers. We plan to examine the usability of XML3D in future work. For more information refer to: http://www.xml3d.org.

References

KLEIN, F., SONS, K., RUBINSTEIN, D., AND SLUSALLEK, P. 2013. XML3D and Xflow: Combining Declarative 3D for the Web with Generic Data Flows. *IEEE Computer Graphics and Applications 33*, 5.

KLEIN, F., SPIELDENNER, T., SONS, K., AND SLUSALLEK, P. 2014. Configurable Instances of 3D Models for Declarative 3D in the Web. In *Proceedings of the 19th International Conference on 3D Web Technologies*, ACM.

SONS, K., KLEIN, F., RUBINSTEIN, D., BYELOZYOROV, S., AND SLUSALLEK, P. 2010. XML3D: Interactive 3D Graphics for the Web. In *Proceedings of the 15th International Conference on 3D Web Technologies*, ACM, no. 212.

SONS, K., KLEIN, F., SUTTER, J., AND SLUSALLEK, P. 2014. shade.js: Adaptive Material Descriptions. *Eurographics: Computer Graphics Forum 33*, 7.

SUTTER, J., SONS, K., AND SLUSALLEK, P. 2014. Blast: A Binary Large Structured Transmission Format for the Web. In *Proceedings of the 19th International Conference on 3D Web Technologies*, ACM.

[1]http://xml3d.github.io/xml3d-examples/examples/poster/index.html