Declarative 3D Approaches for Distributed Web-based Scientific Visualization Services

Yvonne Jung
Fraunhofer IGD
Darmstadt, Germany
yjung@igd.fhg.de

Johannes Behr
Fraunhofer IGD
Darmstadt, Germany
jbehr@igd.fhg.de

Timm Drevensek
Fraunhofer IGD
Darmstadt, Germany
tidreven@igd.fhg.de

Sebastian Wagner
Fraunhofer IGD
Darmstadt, Germany
sewagner@igd.fhg.de

ABSTRACT

Recent developments in the area of efficient web-service architectures and the requirement to provide applications not just for a small expert group lead to new approaches in the field of web-based (scientific) visualization. The just emerging support for GPU-supported and therefore high-performance 2D and 3D graphics in modern web-client implementations and standards provide new application environments, which are especially interesting for the demands of scientific visualization solutions. Thus, in this paper we present a web application deployment architecture that aims at supporting decision making processes more efficiently. We also show that current approaches in the field of declarative 3D techniques are useful for client-side rendering as well as for a large number of processing and visualization aspects.

Categories and Subject Descriptors

H.3.5 [**Information Systems**]: Online Information Services—*Web-based services*; I.3.6 [**Methodology and Techniques**]: Standards—*Languages*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Virtual Reality*

Keywords

X3D, HTML5, WebGL, DOM, Web Integration, Transcoder, Web Services, Service-oriented Architectures, X3DOM

1. INTRODUCTION

With the advent of 3D Internet, there is a strong trend towards 3D data and documents (like 3D scanners and printers, geo-referenced data, and Augmented Reality) as well as a convergence of application platforms (e.g. W3C WebApps, HTML5, etc.) towards web-based contents. However, we still have various operating systems with very different soft- and hardware requirements such as smart phones (e.g. iOS, Android, RIM, WindowsPhone), different desktop systems, cloud bases etc., which typically make cross-platform development complicated and time consuming. In this regard, web technologies incl. web-service architectures are rather common and can overcome these issues by interconnecting all these diverse approaches. Moreover, web browsers are converting more and more towards full runtime environments for whole applications and with WebGL or Stage3D, 3D-capable browsers are now broadly available, too.

However, this does not solve the often intricate processes to develop applications, where one can distinguish between two main types. For one thing, we have interactive processes with creative and individual components such as games with their manual or semi-automatic tool chains (e.g., DCC tools like 3ds Max and game editors like Unity3D) that focus on isolated applications. For another, there are automatable processes that typically are complex and distributed, which can be realized by means of a web-service architecture. In the context of this paper, we focus onto the latter approach by presenting a fully automated Web Service Portal, which thereby allows generating e.g. 2D/3D mash-ups for design review, city planning or similar decision making tasks.

Generally spoken, distributed data-centered applications are one of the common implementation concepts for scientific visualization solutions today to process huge data sets. Therefore, utilizing web-service architectures or even cloud-based solutions are just the next logical step. Recent developments in the area of high-performance web-service architectures and the requirement to provide applications not just for a small expert group lead to new approaches in the field of web-based visualization. The emerging support for hardware-accelerated and therefore high-performance 2D and 3D graphics in modern web clients and standards provide a new application environment, which is especially interesting for the demands of scientific visualization solutions. Therefore, some visualization packages (like ParaView with ParaViewWeb from Kitware [12]) already use this opportunity to move the established application model to a web and cloud-based solution. Deploying current application models to a new environment is just the first step. However, this field is not yet explored to utilize its full potential.

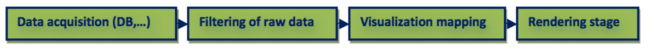


Figure 1: Coarse sketch of the visualization pipeline.

Thus, the web-application deployment component presented in this paper helps supporting decision making processes more efficiently, and the benefits include the following four major aspects:
– leveraging new GPU-accelerated 2D and 3D client APIs;
– achieving interactivity while combining traditional client- and server-side rendering techniques;
– providing new user experiences while considering scalability and security in open web-based environments;
– automating and optimizing data preparation processes for web-specific environments through web server infrastructures.

2. RELATED WORK

For the design of the proposed web-service architecture several domains and technologies are relevant. Corresponding work is thus briefly described within this section.

2.1 Scientific Visualization

To provide important features for scientific visualization (cf. e.g. [6] and [15]) in a specialized web application, the application must be able to handle visualization-specific representations that consist of registered and merged point, surface, and even volume data as well as the corresponding meta information. But having the raw data to be visualized readily available is only the very first step, as is shown in Figure 1 with a rather simplified visualization pipeline. Usually, after data acquisition the information is re-sampled onto a structured regular grid before filtering the data accordingly (a discussion of common data representations including multi-resolution and adaptive resolution representations for large data sets can be found in [15]). Then, the scalar, vector, or tensor values of the data set are mapped to visual representations that can be rendered [6]. Well-known examples here are volume rendering or flow visualizations in CFD [18].

For multi-dimensional, multi-variate/ multi-modal, and probably even time-varying data, visualization is much more intricate, since first the registration between different data sets including the thereby introduced uncertainties has to be handled as well as data fusion aspects and the calculation of derived quantities (e.g. the strength of correlation between two different variables) to improve the visualization [6]. In this regard, not only the choice of an appropriate visualization technique but also the possibility to interactively explore the data set (e.g. by changing the value range or transfer function) is of great importance for grasping and analyzing the information and therefore requires real-time capable mapping and rendering methods, which nowadays is often achieved using GPU-based methods [18].

2.2 3D Graphics in the Web

Direct volume rendering in general is an alternative form of visual data representation compared to the traditional polygonal form. In [7] the implementation of a direct volume rendering system for the Web is presented. In this publication the authors discuss their WebGL-based volume rendering approach using ray-casting in two different but challenging application domains: medical imaging and radar meteorology. Performance, scalability, accuracy, and security are some of the many challenges that must be solved for 3D web applications, esp. since WebGL is still based on the old Shader Model 2.0 (like on the NVidia GeForce 5900 series introduced in 2003).

Another alternative method for representing 3D graphics is using point clouds, which is a set of points in 3D space with attributes. This is especially of interest, since 3D scanning devices such as LiDAR equipment and sonar scanners deliver their data as point clouds. One recent framework to simplify the streaming and rendering of point clouds based on JavaScript and WebGL [16] is "XB PointStream" [21].

In [23], the authors propose a WebGL-based framework for representing reflectance information via Bidirectional Texture Functions (BTF), which allows for the progressive transmission and interactive rendering of digitized artifacts consisting of 3D geometry and reflectance information. This is achieved by employing a novel progressive streaming approach for the huge BTF data set that allows the smooth interactive inspection of a steadily improving model during download. Analogously, the X3DOM framework [3, 4] currently follows a similar approach for lightweight geometry compression and transmission via so-called image geometries that likewise utilizes image compression techniques.

2.3 Declarative (X)3D in HTML5

The open ISO standard X3D [8] provides interactive 3D graphics for the web and is the only standardized 3D deployment format. It differs from other 3D formats like the interchange format Collada [1] in that it also includes the scene's runtime behavior. The X3D standard already provides point and surface primitives. Typically, the visual elements are described by their boundary representation, e.g. via an "IndexedTriangleSet" node. Volume rendering will be part of the next spec revision. Therefore, the Web3D medical working group has presented a sample implementation [11] and an ISO/ IEC PDAM extension for a volume rendering component in X3D, which also has answers to DICOM requirements for n-D presentation states [20].

In addition, since most geo-referenced data is provided in a geodetic or projective spatial reference frame, the X3D Geospatial component includes conventions that are defined by the Spatial Reference Model and thereby provides support for geographic and geospatial applications [8, 17]. With surface, volume, and geo-spatial components X3D already provides a solid foundation for scientific visualization tasks and thus enables an automated connection of existing data with e.g. atmospheric, oceanographic, or geological data to be visualized in the Web.

However, all existing and proposed X3D components only extend X3D for various application and visualization scenarios but do not address one of the essential issues of X3D right now, namely that X3D is still bound to a plugin-based integration model that has major usability and performance issues especially for large data sets [3, 4]. All plugin-based systems have two major drawbacks. First, they are only plugins and not installed by default on most systems. Therefore, the user has to deal with plugin installation, security, and browser or OS incompatibility issues, not to mention the lack of accessibility. Second, the presented systems define an application and event model inside the plugin, which is decoupled from the DOM content. Developers, who try to develop integrated web applications that use both, the DOM and the plugin-model, have to deal with small plugin-specific interfaces and synchronization problems.

WebGL (a JavaScript binding for OpenGL ES 2.0 in the Web Browser [16]), Adobe's Flash 11 with Stage3D [24], and Microsoft's Silverlight 5 [19] now all provide access to the native GPU layer without any further plugin installation process, but the issue of the missing DOM integration still exists. Hence, what is still missing is a better integration with open architectures, which integrate well with existing web technologies like CSS(3), HTML(5), JavaScript, DOM scripting and Ajax. In this regard, with the X3DOM project recently a DOM-based integration model for declarative (X)3D in HTML5 was proposed [3, 4], that allows a seamless integration of 3D contents into the HTML document model by utilizing standard web APIs for integrating content and interactions.

2.4 Service-oriented Architectures

Finally, we have to shortly review service-oriented architectures (SOA). An SOA is a software architecture paradigm for structuring and using distributed functionalities that are managed by various owners. Comparable to business transactions an SOA composes several low-level services to more complex services with a higher level of abstraction [9]. Here, web-services help supporting the collaboration between different applications running on different platforms by utilizing REST, JSON, and XML-based standards like UDDI, WSDL, and SOAP as web-service protocols. In this regard, one prominent application of web-services are the standardized geo-services (such as WCS [2]) provided by the Open GeoSpatial Consortium (OGC) to make spatial geo-data accessible in a structured way.

Hence, some 3D rendering systems already make prototypical use of REST and JSON for scene-graph manipulation and distributed rendering [22] within a service-oriented platform. To demonstrate the full potential of their web-service interface, the authors additionally included a second visualization backend that provides global illumination algorithms that are evaluated on a server farm whereas the web frontend only shows the final image and the UI elements. Likewise, in [5] a full workflow pipeline from data acquisition up to interactive web-based visualization in the Cultural Heritage domain was proposed. Remote or server-based rendering in general is a very active research discipline for the last 20 years. Visualization servers or services are build to overcome the performance issues with client solutions or provide specific systems to protect the 3D dataset while not or only partially distributed to the client. Koller et al. [13] provide an interesting proposal to the protection system with a remote rendering service that not just only transfers images to the client but also includes a number of active defense method to guard against 3D reconstruction attacks.

A streaming-based remote visualization for mobile devices was presented in [14], where complex 3D models are rendered on a cluster of PCs and transferred to the client via MPEG video streaming. Smart environments consisting of several interconnected devices, which can change dynamically whenever mobile devices enter or leave the environment, are specifically addressed in [25]. To utilize such environments efficiently for information visualization, the authors also propose a service-oriented architecture. Although their work still focuses on the specificities of certain display devices, it nevertheless already also addresses issues such as caching strategies and compression methods.

3. SYSTEM ARCHITECTURE

Within this section, we focus onto automatable processes for web application development by designing and implementing a fully automated Web Service Portal, which provides services that automatically combines application templates and raw data into interactive 3D visualizations for the web.

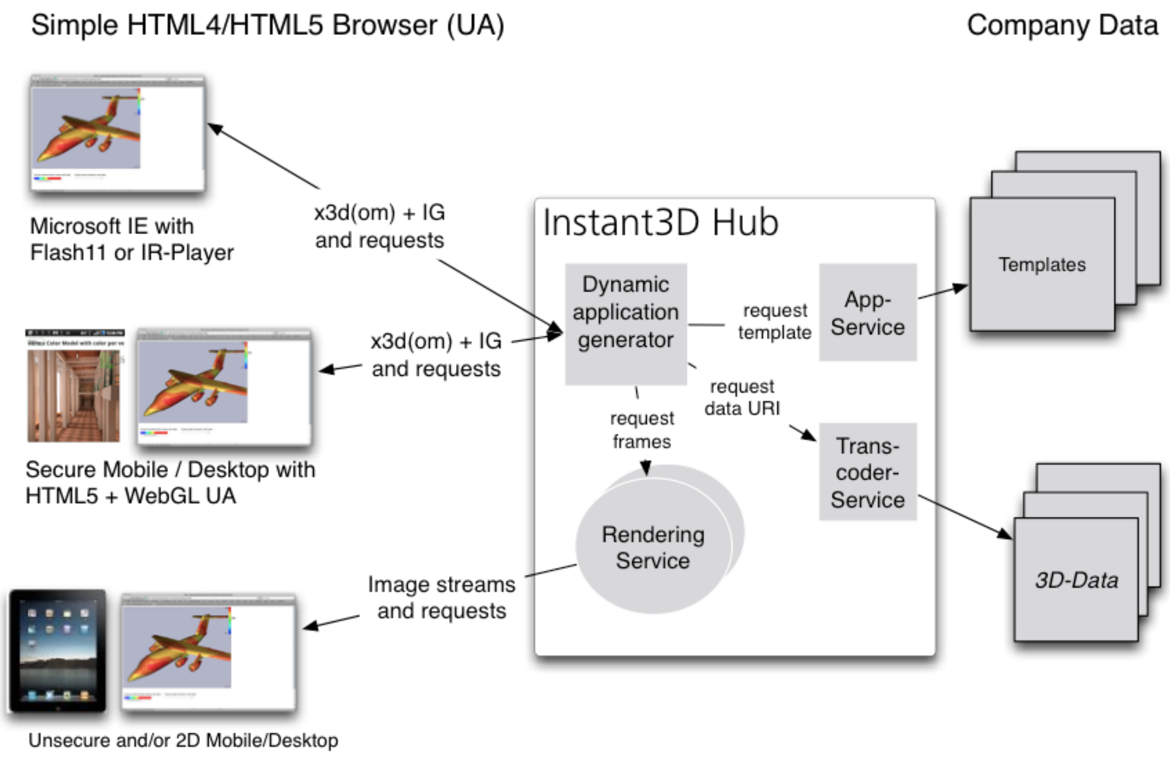
3.1 Use Cases and Design Considerations

The goal here is to provide web services that automatically convert raw 3D data (e.g. geo-spatial data, simulation results, and architectural models) into interactive 3D visualizations for the Web that can be delivered as a cloud service. For example, value ranges that can be interactively explored like a temperature or density distribution are automatically tagged, like e.g. it is often done in modern IDEs such as the CG FX Composer framework, to provide all data that is necessary for an appropriate user interface, like a slider with a corresponding value range as shown in Figure 4.

This enables the user (in our case typically a decision maker) to interactively explore the presented 3D data, which is achieved with the help of two approaches. On the one hand, certain data values and ranges are reflected in such a way, that the application service can directly derive appropriate UI elements (e.g. a slider from x meters to y meters that allows visualizing different water-levels in a disaster management scenario). On the other hand, the user must be able to directly interact with the visualization, e.g. by selecting a certain region for which he wants to obtain more information (which e.g. in a GIS scenario might be "show buildings and trees in marked area") or by clicking onto a certain POI, which in turn delivers other information from the processing backend.

Therefore, the visualization needs to be scalable in such a way that – depending on the desired application type – mobile and desktop machines with rather different computing and 3D capabilities are supported alike. For example this can be achieved via a hybrid approach that provides both, server-side streaming for low-end mobile clients (using InstantPlayer as render server [10]) and direct web-based 3D rendering for desktop machines via X3DOM. Moreover, cross-platform and cross-browser issues need to be addressed. Finally, especially when also CAD and PDM or highly classified data is used, security aspects are of high importance. Here again, streaming technologies help with information hiding in that only image streams but no 3D models are transferred over the network to the client for display.

As mentioned, the proposed software architecture is based on web services. As described in section 2.4, these are intended for being used in broader software ecosystems, which automatically exchange data. Usually, the service interfaces



Simple HTML4/HTML5 Browser (UA)
Company Data
Microsoft IE with Flash11 or IR-Player
x3d(om) + IG and requests
x3d(om) + IG and requests
Secure Mobile / Desktop with HTML5 + WebGL UA
Image streams and requests
Unsecure and/or 2D Mobile/Desktop
Instant3D Hub
Dynamic application generator
request template
App-Service
request data URI
request frames
Trans-coder-Service
Rendering Service
Templates
3D-Data

Figure 2: Instant3D Hub – scalable, secure, and efficient SOA for 3D visualizations on the Web.

are described by XML documents such as SOAP. Obviously, utilizing a unified standard for shared services and communication is essential here. Furthermore, declarative XML-based languages such as X3D are suited well for SOAs in that 3D content, such as CAD/ CAM data or a given WCS response in a GIS application, can be directly transformed, e.g. via an XSLT or similar transform, from one representation to another (such as an X3D world that can be rendered in real-time in the web browser using the X3DOM framework). As shown in Figure 2, this transformation is done with the help of a transcoder service, which also needs to consider the respective client capabilities (e.g. mobile device vs. desktop system) to provide an appropriate visualization.

For being able to present even large-scale big data, suitable compression schemes are required for both, the client (esp. when considering mobile devices) and the server (where esp. latency is an issue). In our implementation this is handled by utilizing image-based compression and delivery algorithms that can encode geometric as well as material data. In addition, appropriate server-side caching strategies allow for faster delivery of 3D contents.

3.2 Web-Service Architecture

Figure 2 coarsely shows the concept of the proposed service-oriented architecture. The core component is the "Instant3D Hub" that acts as application provider, whereas the application identification is resolved via a URI. Thereto, the application service takes a request from the so-called Hub and extracts an application template, which specifies the data and application characteristics. For data preparation, the template provides at least the HTML page, the necessary meta-information of the respective application, as well as descriptions of all required data containers, e.g. in the declarative X3D format for 3D contents.

After that, the Hub invokes the appropriate transcoder service for providing the concrete data (like a 3D world with a certain camera pose). Therefore, the transcoder service translates and, where required, caches the 3D data from already existing descriptions by converting them from various resources and formats, which are not yet suitable for real-time web presentation, to a declarative deployment format for final presentation. Here we use X3D in combination with HTML5, since declarative formats nicely fit to SOA pipelines. The result is then provided under a new URI. This basic architecture tries to fulfill three basic requirements that are outlined next.

Scalability The central HUB recognizes the browser criteria (e.g. mobile or desktop) and can use those to ask for a device- and/ or context-specific application UI. The application uses a scene-graph manipulation layer, which maps the actual changes to browser local (e.g. DOM content) or remote scene representations.

Efficiency The overall structure is build to produce very interactive and pleasant user experiences. All main UI pages should get delivered immediately and all further data sets should be provided with a flexible caching infrastructure on request. The whole service architecture is build asynchronous and does not wait at any point for any data conversion process at all. The client and server side rendering processes are both built to be able to stream efficient binary updates to the server for highly responsive environments.

Flexibility The architecture is not bound to a specific application model and very open through the web-service infrastructure. It provides an environment useful to a large number of applications by harmonizing three basic steps: application-UI retrieval, continuous data-preparation and a hybrid visualization approach, which can map to client and server side rendering.

3.3 Application and UI retrieval

The goal of the framework is to deliver context- and browser-specific web applications. The application service provides templates for those applications, which are retrieved on request. Those templates consist of the following parts:

HTML documents HTML documents at first can be seen as the building ground for all application-specific structures. Those documents are provided by the application service for a specific browser and therefore device (e.g. tablet or desktop).

Data references The application includes usually a list of external resources, which should be delivered to the browsers while running. This could be resources like XML documents or images, which do not need any further processing but also link to data sets that need to be converted before viewed (e.g. a specific 3D data set) or transformed for a specific device class.

Metadata The metadata describes criteria for the conversion and transcoding process. This can e.g. control whether optimizations on the graph are allowed or not and how identification of objects should be performed.

The hub requests this information from the application service, requests then data sets from the transcoder service with the data references and metadata settings and delivers the final HTML document with the not yet existing external references.

3.4 Transcoder

The transcoder service converts the data of a given resource to a new single resourc or set of resources while considering the guidelines provided by the application metadata. The resulting URI's are provided immediately as result of the transcoding request even so the data may not be available yet. The basic HTTP return code is used to communicate the availability of a specific resource while requesting.

200 OK: the request has succeeded and the entity corresponding to the requested resource is sent in the response.

404 Not Found: the conversion job could not be completed successfully and the sources will not be available without it.

503 Service Unavailable: The service is not yet available since the conversion process is still running. The HTML head includes a standard Retry-After element that defines when the resources is expected to be available.

The transcoder also includes an efficient caching and updating mechanism, which allows to convert the data only once for similar requests, but also updates the converted data as soon as new data sets are available.

3.5 Rendering

The converted data is fetched to a real-time rendering system, which consists of a scene-graph whose elements can be identified and modified during runtime. This scene-graph is manipulated through a client-side abstraction layer that allows mapping those changes to local or remote representations of the graph.

3.5.1 Client-side Rendering

Depending on data security aspects and the respective web browser capabilities, the Hub generates a concrete visualization application from a given pre-prepared scenario template. For one thing, this can be realized via client-side rendering by utilizing X3DOM, which either uses JavaScript with WebGL or Flash 11 with Stage 3D for real-time rendering. Open research issues here are suitable caching strategies for fast content delivery as well as scalable methods for binary compression of high-end material data (compare [23]) and of big geometric data (i.e. vertex attributes).

The latter can be achieved by utilizing our image geometry approach. This method not only allows to asynchronously deliver and compress vertex attribute data but it also allows to nicely separate the structure of the 3D content from raw vertex data that usually only bloats the HTML document. The advantages of client-side rendering are a rather simple server infrastructure and highly interactive apps since everything is rendered on the client. Disadvantages are that the data-load can easily overburden the client, that lots of 3d data needs to be transferred, and security or IPR issues.

3.5.2 Server-side Rendering

The second option is server-side rendering. In this case, for instance an X3D-based runtime environment [10] is utilized for rendering and the rendered image frames are transferred (e.g. as an MJPEG stream) to the client. Since interactions are handled via WebSockets and XMLHttpRequest (XHR), the latency is much higher than for client-side rendering. Therefore, we are currently exploring suitable interaction methods and message protocols.

Another disadvantage here is the fact that a complex server infrastructure is required. However, since no 3D data but only images are transferred over the network, there are no IPR issues concerning the 3D data, and data security is inherently given as a nice side effect. Also, the visualization application can be executed on arbitrary clients, which is another advantage of utilizing server-side rendering.

4. RESULTS AND DISCUSSION

4.1 Application Examples

In this section a few use cases are exemplarily described to demonstrate the potential of the proposed service architecture. In the area of cultural heritage for instance, working

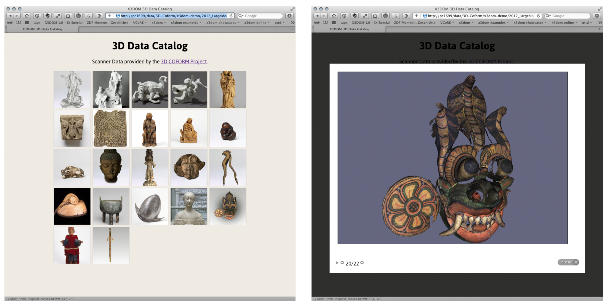


Figure 3: Presentation of laser scanned CH artifacts using our proposed web-service architecture.

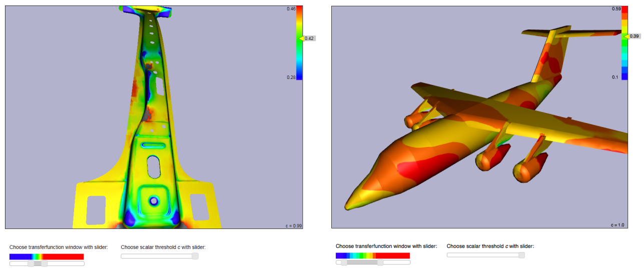

Choose transferfunction window with slider:
Choose scalar threshold c with slider:
Choose transferfunction window with slider:
Choose scalar threshold c with slider:

Figure 4: Picking of generic vertex and/or fragment properties (left: material thickness of a structural element, right: electromagnetic field strength).

with 3D scanner data for preservation is getting more and more common. Figure 3 shows an example of a web application for the visualization of scanned historical 3D objects, which was generated using the proposed transcoder architecture by automatically processing and converting the scanned data to create an interactive viewer.

Figure 4 shows a 3D CAE prototype for the visualization of simulation data in the web browser using X3DOM [3, 4], where two examples from the domain of sheet metal forming (left) and electromagnetic field simulation (right) are taken to allow the exploration of FEM results. These application prototypes have a huge potential for communicating and presenting e.g. simulation results towards decision makers or consumers, without distributing a whole application while providing the user with more information than a static image or a boring fact sheet.

The example to the left in Figure 4 shows the results of a sheet metal forming simulation in automotive environments. The material thickness after the forming process is color-coded and applied via a look-up texture. By clicking onto a colored region, the corresponding thickness value is obtained and marked with a yellow arrow and textual information in the top right of the application. The slider elements below are implemented using the JavaScript library jQuery (`http://jquery.com/`) and can be used to interactively modify the color-coding by setting offset, bias, and threshold.

4.2 Standardization Aspects

Another point is the lack of open standards in scientific visualization and 3D graphics as well as in mobile and interactive systems. Although current visualization systems are often highly specialized and rather sophisticated, they still utilize proprietary formats and methods that are neither compatible in their concepts of operation nor in their supported data formats. On the one hand this prevents a harmonization of data from different sources and thereby hinders its distribution and utilization. On the other this also leads to parallel developments of incompatible and isolated technologies.

In this context, the open ISO standard X3D [8] is the only standardized 3D deployment format that also defines the runtime behavior. Hence, there are ongoing efforts [20] to develop an open interoperable standard for the representation of volumetric data based on input from a wide variety of modalities. The proposed X3D volume rendering component [8] therefore aims at the exchange and interactive exploration of volumetric data and at industrial applications that use X3D as interchange format, but can link to proprietary databases and hardware, too.

A service for open, flexible, and scalable access and processing of Earth data is the OGC Web Coverage Service (WCS) 2.0 Standard, which now allows providing a comprehensive portion of Earth science data categories through one coherent and implementation-independent interface [2]. The coverage model of WCS 2.0 transcends pure raster data and includes almost all relevant categories, such as irregular and curvilinear grids, general meshes, trajectories, surfaces, solids, and point clouds.

In this regard, X3D already incorporates basic means for point rendering, as well as a geospatial component. Since most geo-referenced data are provided in a geodetic or projective spatial reference frame, X3D therefore provides support for a number of nodes that can use spatial reference frames for modeling purposes. However, there are still several drawbacks like the lack of well-defined terrain rendering etc., which for instance were addressed in [17]. In addition, with the X3D Earth working group there is a strong collaboration of the Web3D Consortium with the OGC, because as mentioned, X3D provides a solid foundation here and is a good starting point for further standardization efforts to enable an automated connection of existing data sources for further visualization.

Hence, a pro-active participation in several standardization bodies (ISO, W3C, Khronos) helps preventing parallel developments of isolated technologies by harmonizing the general understanding. Therefore, the “Declarative 3D for the Web Architecture” W3C Community Group was founded in 2011 with the objective to standardize a declarative approach to interactive 3D graphics as part of HTML documents.

5. CONCLUSIONS

Traditional scientific visualization approaches are extremely demanding in regards to software and hardware requirements. Thus, today’s visualization solutions are tailored for specific data representations and application scenarios to cope with several soft- and hardware limitations, which led to extremely specialized software packages that are built and run by professionals in high-cost scenarios. However, recent developments in the area of web applications and the requirement to provide applications not only for a small group of experts lead to new approaches for web-based visualization. Morever, since data-centered, distributed applications were one of the common concepts to process huge data sets for visualization, web-service architectures and cloud-based solutions are just the next step. Additionally, the emerging support for GPU-accelerated high-performance 3D raster-graphics in modern web clients and standards provide an application environment that is especially interesting for the demands of scientific visualization solutions.

In this regard, the approach we presented in this paper does not only implement an established application model in a new web- and cloud-based environment, but it also introduces a web-service architecture as core of the application distribution and deployment infrastructure. Instead of simply delivering data to a rigid viewer like in conventional applications, our proposed architecture delivers a user experience that is tailored for specific user scenarios and the actual context. Thereby, the final environment is able to derive and deliver a dynamic application with regards to user, system, security, and data requirements.

This allows the system to provide dynamic technical solutions and to automatically choose a client, server, or even hybrid visualization method for an unchanged application at runtime, while also considering the device capabilities as well as IPR and security concerns. What is even more important, besides the technical aspects, is the architecture's ability to provide very specific applications for certain user groups, since the system is not built as a rigid application model for a small user group. Thus, the presented service architecture enables new application scenarios, including scientific visualization for decision support or visual analytics applications, as well as e.g. corresponding web applications for public information scenarios.

6. REFERENCES

[1] R. Arnaud and M. Barnes. *Collada*. AK Peters, 2006.

[2] P. Baumann. Beyond rasters: introducing the new ogc web coverage service 2.0. In *Proceedings GIS '10*, pages 320–329, New York, NY, USA, 2010. ACM.

[3] J. Behr, Y. Jung, T. Drevensek, and A. Aderhold. Dynamic and interactive aspects of X3DOM. In *Proceedings Web3D 2011*, pages 81–87, New York, USA, 2011. ACM Press.

[4] J. Behr, Y. Jung, J. Keil, T. Drevensek, P. Eschler, M. Zöllner, and D. W. Fellner. A scalable architecture for the HTML5/ X3D integration model X3DOM. In S. Spencer, editor, *Proceedings Web3D 2010*, pages 185–193, New York, USA, 2010. ACM Press.

[5] R. Berndt, G. Buchgraber, S. Havemann, V. Settgast, and D. W. Fellner. A publishing workflow for cultural heritage artifacts from 3d-reconstruction to internet presentation. In *Proceedings of the 3rd intl. conf. on digital heritage*, pages 166–178, Berlin, 2010. Springer.

[6] R. Bürger and H. Hauser. Visualization of multi-variate scientific data. In *Eurographics 2007 State of the Art Reports*, pages 117–134, 2007.

[7] J. Congote, A. Segura, L. Kabongo, A. Moreno, J. Posada, and O. Ruiz. Interactive visualization of volumetric data with webgl in real-time. In *Proceedings Web3D 2011*, pages 137–146, New York, NY, USA, 2011. ACM.

[8] W. Consortium. Extensible 3d (X3D), 2011. http://www.web3d.org/x3d/specifications/.

[9] T. Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, 2005.

[10] FhG. Instant Reality, 2012. www.instantreality.org.

[11] N. John, M. Aratow, J. Couch, D. Evestedt, A. Hudson, N. Polys, R. Puk, A. Ray, K. Victor, and Q. Wang. MedX3D: Standards enabled desktop medical 3d. In *MMVR: Medicine Meets Virtual Reality*, pages 189 – 194, 2008.

[12] Kitware. ParaViewWeb, 2012. http://paraview.org/Wiki/ParaViewWeb.

[13] D. Koller, M. Turitzin, M. Levoy, M. Tarini, G. Croccia, P. Cignoni, and R. Scopigno. Protected interactive 3d graphics via remote rendering. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 695–703, New York, NY, USA, 2004. ACM.

[14] F. Lamberti and A. Sanna. A streaming-based solution for remote visualization of 3d graphics on mobile devices. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):247–260, 2007.

[15] D. R. Lipsa, R. S. Laramee, R. D. Bergeron, and T. M. Sparr. Data representation for scientific visualization: An introduction. In M. J. Davis, editor, *Computer Graphics*, chapter 6, pages 121–136. Nova Science Publishers, 2011.

[16] C. Marrin. Webgl specification, 2011. https://www.khronos.org/registry/webgl/specs/1.0/.

[17] M. McCann, R. Puk, A. Hudson, R. Melton, and D. Brutzman. Proposed enhancements to the x3d geospatial component. In *Proceedings Web3D 2009*, pages 155–158, New York, NY, USA, 2009. ACM.

[18] T. McLoughlin, R. S. Laramee, R. Peikert, F. H. Post, and M. Chen. Over two decades of integration-based, geometric flow visualization. *Computer Graphics Forum*, 29(6):1807–1829, 2010.

[19] Microsoft. Silverlight, 2011. http://www.microsoft.com/SILVERLIGHT/.

[20] N. F. Polys, N. W. John, D. Evestedt, and M. Aratow. Unifying iso standards for medical image presentation. In *Workshop on Medical Virtual Environments*. IEEE, 2010.

[21] A. Salga. XB PointStream, 2011. http://scotland.proximity.on.ca/asalga/demos/freecam/.

[22] T. Schiffer, A. Schiefer, R. Berndt, T. Ullrich, V. Settgast, and D. W. Fellner. Enlightened by the web – a service-oriented architecture for real-time photorealistic rendering. In *Tagungsband 5. Kongress Multimediatechnik Wismar*, pages 1–8, 2010.

[23] C. Schwartz, R. Ruiters, M. Weinmann, and R. Klein. Webgl-based streaming and presentation framework for bidirectional texture functions. In *Proceedings VAST 2011*, pages 113–120. Eurographics, 2011.

[24] A. Systems. Flash, 2011. http://www.adobe.com/products/flashplayer.html.

[25] C. Thiede, C. Tominski, and H. Schumann. Service-oriented information visualization for smart environments. In *13th Intl. Conf. on Information Visualisation*, pages 227–234. IEEE, 2009.