3DCIS: A Real-time Browser-rendered 3D Campus Information System Based On WebGL

Nils Hering[1], **Martin Rünz**[1], **Lubosz Sarnecki**[1] **and Lutz Priese**[1]
[1]Institute of Computational Visualistics, University of Koblenz-Landau, Koblenz, Germany

Abstract— *Most of the current real-time 3D web applications are only available with plug-ins as Flash or additional software as Java. Avoiding this drawback, the new WebGL technology provides hardware accelerated computer graphics for web browsers without requiring plug-ins. Using Blender, WebGL, the WebGL-expanding framework GLGE, and an in-house developed exporter B2G from Blender to GLGE we have realized the cutting-edge web application 3DCIS based on a complex 3D model of our campus. With 3DCIS one is able to explore the campus interactively and to become acquainted with local persons and institutions. Textual information about buildings, rooms and persons is linked with 3D model information to enhance the intuitive experience of 3DCIS.*

Keywords: Web 3D, WebGL, GLGE, export, information system

1. Introduction

It is state of the art to use plug-in based systems to create 3D web applications. Well-known and widely spread examples are Adobe Flash, Unity3D or Microsoft Silverlight. Their main handicap is the user's obligation to install the corresponding plug-in. Another common software system to provide 3D applications in the web browser is the Java applet concept. Java applets do not require an explicit plug-in but an installed Java virtual machine.

The relatively new 3D web technology WebGL [9] avoids those drawbacks. A user may navigate in a 3D environment solely in a WebGL compatible web browser. This comfort is payed by the developer of a WebGL application as WebGL provides only a rudimentary application programming interface (API). On the other hand, development of 3D models is rather simple in Blender. Therefore, we had to build an exporter from Blender to WebGL that we will introduce in this paper. We have released the exporter under the GNU GLPv3 free software license to the community. As an application of B2G we present the campus information system **3DCIS** of our university. An interested visitor may navigate in 3DCIS through a 3D model of our campus and interactively gather further information about persons, rooms, facilities, etc. from the application. 3DCIS uses the probably largest online WebGL model in a real-time application world-wide at present (May 2011). The latest version of 3DCIS is currently available at `http://explore.uni-koblenz.de`.

2. Related work

The WebGL API provides capabilities in developing 3D web content which were previously the exclusive domain of the desktop environment. Leung and Salga deal in [10] with the question how mid level APIs can help the developer to create unique 3D web content. They emphasize the fact that WebGL gives the chance to not just replicate desktop 3D content and applications, but rather to exploit other web features to develop richer content and applications.

DeLillo presents in [5] the WebGLU development library for WebGL. WebGLU provides an API which allows the developer to implement WebGL based content easier and more clearly than with the WebGL API only.

In [6] Di Benedetto et al. introduce SpiderGL, a JavaScript library for developing 3D web applications. The aim of SpiderGL is to simplify the use of WebGL and to provide some extra features like e.g. the handling of asynchronous data loading. SpiderGL is used in [4] where Callieri et al. present a WebGL and SpiderGL based method for the building of interactive 3D visualization schemes for scientific data produced by molecular and cellular biology research.

In [11] Niebling and Becker deal with a web extension of the COVISE visualization environment [13]. To reach this goal they deploy WebGL to provide a web rendering component which allows to explore e.g. post-processed simulation results in a web browser.

Esnault et al. present a flexible framework for the 3D visualization of data in the web in [8]. They mention WebGL as one of the technologies to present the Web 3D result scene to the user.

In [2] Behr et al. introduce a scaleable architecture that implements the HTML5/X3D integration model X3DOM [1]. This architecture offers a single declarative interface to application developers while it provides several render backends. One of those backends is WebGL supplemented by a scenegraph.

Di Cerbo et al. deal with the extension of their e-learning platform DIEL in [7]. By using X3D and X3DOM in combination with WebGL to create a Web 3D interface they achieved to render 3D content on the web without requiring additional software.

In [12] Sons et al. present XML3D, a new technology for the support of interactive 3D content in mixed 2D/3D documents. A portable implementation of XML3D is based on JavaScript and WebGL.

3. WebGL and GLGE

WebGL is a platform independent API used to create 3D content for web browsers. It utilizes the HTML5 *canvas* element and therefore requires a web browser capable of processing both HTML5 and WebGL. WebGL and HTML5 are still in further development by different consortia. WebGL is based on OpenGL ES 2.0, a popular 3D API designed for embedded devices. WebGL simply enables users to draw 3D primitives on a 2D canvas. But it has no scene graph and is not able to load and instantiate textured meshes. WebGL is already supported by a variety of browsers. The future of WebGL is promising, especially as it is a standard since March 2011.

A lot of WebGL frameworks are currently in development. 3DCIS uses GLGE [3]. GLGE is a programming library based on WebGL, providing more comfortable handling of WebGL features. The main benefit of such a library is that much of the low-level operations of WebGL are wrapped by higher-level functionality. This means easier access to WebGL's scope and more comfort in developing usable 3D web applications.

GLGE provides text rendering and offers fast picking of objects. It is open source and written in JavaScript. GLGE is able to load Collada scenes (widely used Khronos standard) as well as scenes stored in it's own XML structure.

4. Blender-to-GLGE exporter

The desired workflow is to maintain 3D models in Blender and to export them to GLGE in such a smooth way that changes in the model can be studied in WebGL immediately, without manual modifications. Blender is free software, productive and available for all major operating systems. It possesses a powerful Python API.

4.1 Blender export situation

Since Blender 2.5 was still in it's beta stages during the beginnings of our work, there was, and in places still is, a lack of community supplied plug-ins like exporters. Fortunately, Blender 2.4 and Blender 2.5 file formats are cross compatible in both directions. This means that a workflow using Blender 2.5 for content creation and 2.4 for export is possible. However, this would result in using two different program instances simultaneously.

Another possibility to export Blender 2.5 data is Collada. Both, Blender 2.5 and GLGE support this XML based format, but lack feature completeness. The Collada export in Blender is implemented with OpenCOLLADA in C++, an MIT licensed Collada API, also available for Maya and 3DStudio. At the start of our work Blender and OpenCOLLADA had to be compiled from source in order to export Collada which is harder to deploy on an artist's system than a plain Python script.

We thus decided to write a new exporter from Blender to the GLGE XML format to have more control over the final result and appearance of the scene. Therefore, two exporters have been implemented. An early one running in Blender 2.49 and our advanced **B2G** (Blender-to-GLGE) running in Blender 2.5x. Both are written in Python and produce GLGE-readable data.

4.2 Early exporter

The early exporter allows to either export the complete Blender project, the complete scene, or the current selection such as separate buildings or objects. It was written for Blender 2.49b in Python 2.6. Mesh objects with position, normal and UV (texture mapping) information are exported. Also camera objects are exported correctly. The material information only holds the texture name but no additional shading information, so only flat shading is possible.

4.3 GLGE scene format and B2G

The new Blender 2.5 Python API uses Python 3.1 and can access the whole range of scene data available in Blender. The aim has been that the resulting GLGE export looks like the Blender 2.5 scene (see the comparison in figure 1). It should be easily modifiable by an artist without editing the exported markup.

The following listing describes a basic GLGE scene XML structure.

```
<glge>
  <material/>
  <mesh/>
  <scene>
    <object/>
    <light/>
    <camera/>
  </scene>
</glge>
```

The root element of the GLGE document format is *glge*. Inside the root element meshes and materials are defined with *mesh* and *material* tags. The mesh to material relation and the instancing of the meshes is defined inside the *scene* tag. A scene can contain 3 different types of children with transformation information. An *object* tag, which is a mesh linked with a material definition, and also *camera* and *light* tags which define the scene's cameras and lights. Furthermore, there is a possibility to group objects which is currently unused by B2G.

The scene tag itself contains an ambient color, a background color and fog settings as attributes. All this information can be found in the world settings in Blender and is exported.

B2G is able to export mesh objects with mixed triangle and quad faces. Beside the vertex position, also the UV coordinates for texture mapping and face normals are exported. The artist can decide if he uses smooth, interpolated normals or solid normals per face in Blender. The normal information is required by the renderer for advanced shading techniques with a Phong model and normal mapping.

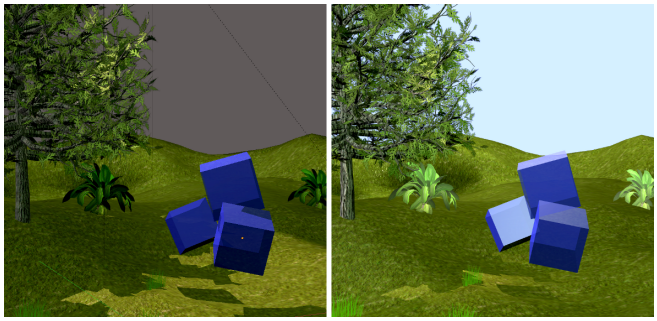

Fig. 1: Comparison between the scene in the Blender 2.55 editor (left) and the exported GLGE scene in the Chromium 9 web browser (right). The sky color is not rendered in Blender's GLSL view. Differences in the specular highlight are also noticeable, since GLGE and Blender use different shading algorithms.

A common technique is the multiresolution modifier where a tessellation can be applied to the mesh data. Blender's mesh modifiers are also supported by B2G.

There are three different types of light objects in GLGE. Point lights, spot lights, and directional lights. Each of these lights can have a different attenuation and color. Constant, linear and quadratic attenuation is implemented in GLGE. For spot lights GLGE renders shadows using shadow mapping. This requires a direction and some extra options. Spotlights are exported, thus shadow mapping in a scene is possible. This slows down performance since the scene has to be rendered one more time for each spot light. Also a shadow map texture has to be stored for each spotlight.

The GLGE material holds the information relevant for the object's shading. Common shading values like specular, shininess and emit are possible. Those options are exported from the Blender material. A material can have a uniform color or can take the information from a color texture map. Additionally normal maps can be set in Blender to gain bump mapping in GLGE. All textures have to have a width and height which is power of two as specified in the OpenGL ES Specification. To render plants B2G handles transparent textures. It is also possible to set the transparency of whole objects which is used to visualize rooms in the campus application. The GLGE scene graph takes care of sorting the transparent objects for every frame.

The GLGE scene file can include further XML files. B2G utilizes that feature and writes the scene, materials and meshes into separate files to maintain readability in a text editor. Opening the big mesh files in an editor is slow and unstable and may result in editor crashes. The exporting of readable XML files is optional, so B2G strips newlines and indentations if desired.

5. 3DCIS

The goal of 3DCIS (3D campus information system of our campus) is to provide a virtual web-based access to the campus and consequently the possibility to explore the campus online. Besides the model-provided local architectural information 3DCIS also visualizes information about the university's departments, employees, facilities and institutions. This information is valuable for students and employees, but 3DCIS shall also attract interested scholars. For the 3DCIS' realization the open source web application framework Django was used with a MySQL database. It is running on an Apache server on Ubuntu Server 10.04 LTS.

The GUI of 3DCIS in the browser consists of three areas: a left vertical navigation bar, an upper horizontal information bar and the central 3D application (see figure 2).

Fig. 2: The GUI of the German 3DCIS version without the browser parts.

5.1 Basic 3D model

Originally, we had created a model of our campus in SketchUp in a previous project and exported it to Blender. So the starting point was a highly detailed 3D model **HPCM** (High-poly campus model) with more than 1 470 000 polygons.

Since in WebGL the model has to be transferred to the client on each page view, a much smaller number of polygons is required. For this reason, we had to downscale HPCM into **LPCM** (Low-poly campus model) in Blender. LPCM uses only 5590 polygons. Figure 3 shows a snapshot of HPCM in Blender, while figure 4 gives a snapshot from LPCM in the web browser.

LPCM simply has flat surfaces where HPCM uses geometry for doors and windows (see figure 5). Consequently, the textures used in LPCM have to contain more information than those in HPCM. On that account a technique called texture baking has been used. Texture baking allows one to render lighting data and geometry data to a texture. As a result, flat surfaces in LPCM reproduce the shape of HPCM

Fig. 3: The HPCM rendered with the Blender 2.5 internal renderer using ambient occlusion and ray tracing.

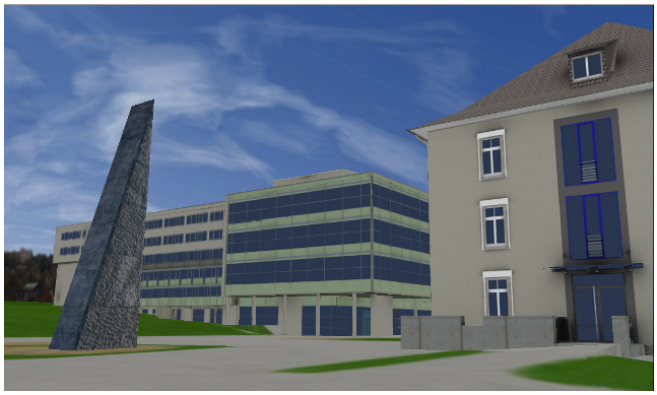

Fig. 4: The LPCM used in 3DCIS.

and contain fixed lightning information. By UV mapping the web application is aware of how to apply textures to objects.

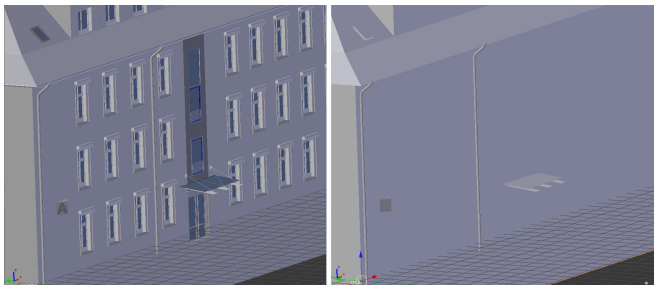

Fig. 5: Comparison of high-poly geometry in HPCM (left) and low-poly geometry in LPCM (right).

To increase the usability of 3DCIS, annotations have been added to LPCM. These annotations imply the locations of rooms, stairways, and elevators which are stored as an instance of a cube mesh as depicted in figure 6. These annotations are used to join the 3D model information with additional information of persons in 3DCIS.

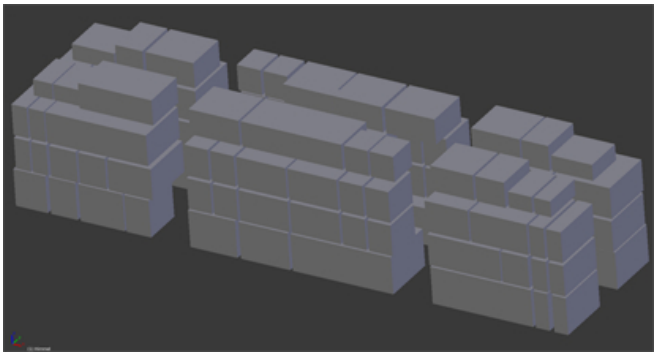

Fig. 6: Annotated rooms (as cubes) in the LPCM Blender model.

5.2 3D user interaction

The handling of 3DCIS is oriented towards 3D computer games to get students interested in our application. The entire campus is presented in WebGL and explorable in a way known from computer games with first-person view.

As usual in 3D applications the view of the user is represented by a camera. This camera indicates e.g. the user's position and view directions. By pressing an arrow key the camera position moves on the ground plane. Shifting the mouse cursor on the 3D application changes the view direction of the camera. A camera-synchronized mini map adapts all the user's movements and alleviates orientation in 3DCIS. The motion speed can be modulated via an options menu. This menu also allows the activation and deactivation of simple billboard texture vegetation.

As there are differences in altitude in the campus model the camera's height has to be set correctly. Using the arrow keys just affects the camera's x- and y-position but not the camera's height. In order to receive height information of the modeled area, a height-map was generated. Our height-map is a simple grayscale image where dark pixels (0 means black) represent lower altitudes and lighter pixels (255 means white) represent higher altitudes (see figure 7).

To move the camera correctly the z-value of its current and aimed position is queried from the height-map. A high gap between those two values indicates that the user is passing an incline. If the gap is too high the terrain is not passable and further movement is prevented. 3DCIS contains a collision detection that prohibits movements into solid objects using the same height map.

There are many possibilities to get information within 3DCIS. The easiest way to receive information is to click on buildings. When a click on a building occurs the building gets highlighted in blue and additional information about this building is displayed. Buildings are also accessible via the navigation bar or via search requests as we will explain in the next subsection.

Rooms can only be addressed via the information or navigation bar. Selected rooms are highlighted in blue and

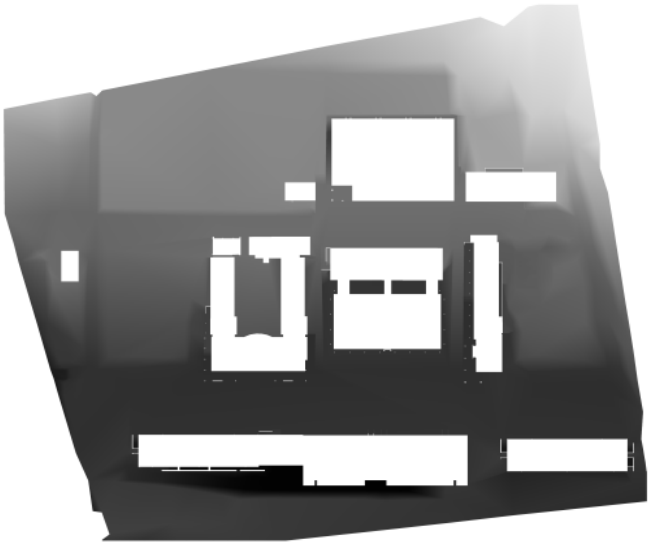

Fig. 7: Height and collision map in which the gray value of a location represents its height in the application.

Fig. 8: A model-annotated room visualized in 3DCIS by a blue cuboid.

their 3D position can be seen in the now transparent building, see figure 8. Rooms are represented by cuboids as described in section 5.1. In order to achieve these different graphic representations (ordinary, highlighted and transparent objects), different materials are in use.

A further interesting feature is the flight option. In addition to the room's visualization the user will experience an automatic flight to the building which contains the requested room. To generate these flights instantly, the 3D environment is divided in 1 216 nodes. Each node is either passable or not passable. The distribution of the nodes is stored in a bitmap which is working as the height-map.

The A* graph search algorithm is used to find the shortest way from the user's position to the room's position. At the end of the flight the camera pans to the room's position in the building.

5.3 Informational features

3DCIS combines the experience of moving through the 3D campus model with useful search features. It is possible to search for buildings, departments, institutions, persons, rooms etc. The search results provide as many information as possible of the linked data, see subsection 5.4. The search result of a person e.g. is visualized in an information window which shows the name and title, an image, telephone number, e-mail address, homepage url and room number. The information window is realized as an overlay over the WebGL content (see figure 2). By clicking the room number, the accordant room gets visualized as described in section 5.2. A click on the flight button next to the room number initiates a flight to the person's room as described in 5.2.

The provided information is accessible by search queries via a search bar on the larger information bar. The automatic search for matching search terms is extended by keyword mapping. The keyword mapping maps syntactical different search terms on semantically similar ones. Alternatively, most information is also available via accordion tabs on the navigation bar. Highest level entries divide the information in "persons", "rooms" and "for persons interested in studies".

5.4 Data mining

To provide the informational features 3DCIS has to have access to data that is contained in three main data sources. The first source is the university's content management system (CMS). Nearly all the university's employees, namely professors, scientific assistants, public employees or student assistants, are documented in the CMS's database. The CMS data is obtained by polling a current XML file regularly using a Python script. This script uses a XML Python module and is called up by a cronjob. The CMS person entries contain data such as name, title, picture, room, telephone number, e-mail address, homepage url etc.

Besides the university's CMS data its LDAP address book is also accessed. For the LDAP access an accordant Python LDAP module is used. The LDAP address book's person entries contain a unique ID, the person's name, room number, phone number and e-mail address. The third main data source consists of manual data. Information about e.g. whole buildings are not available in easy-accessible manner and had to be added manually.

The data is checked having regard to data privacy. E.g. names or e-mail addresses of students from the LDAP which are not published in the CMS, are not published or searchable in 3DCIS.

The collected data is combined using Django data-models. A Django data-model is a python class which automatically sets the data base schema. By using these models one is able to describe and organize a data structure in Python

code. Such models have been designed for *Person, Group, Room* and *Building* for example. The Room data-model, respectively the Room class, looks like this e.g.:

```
class Room(WCObject):
    name = models.CharField(...)
    room_type = models.CharField(...)
    telephones = models.ManyToManyField(...)
    building = models.ForeignKey(...)
    tags = models.ManyToManyField(...)
```

The above code listing is not valid Python code. The dots in brackets are place holders for further descriptions.

6. Conclusion

We use Blender to develop and maintain 3D models. With our in-house developed Blender-to-GLGE exporter B2G we can export our Blender models to WebGL. B2G accomplishes that the GLGE export almost looks like the original Blender scene by exporting all relevant artefacts like e.g. light sources or object's shading information. The user may use WebGL models in the internet browser without any plug-in.

Our 3D campus information system 3DCIS is based on such a WebGL model. 3DCIS provides for example

- interactive navigation in our 3D campus model,
- a camera-synchronized mini map,
- different search options,
- visualization of the search results via
  - automatic flights,
  - correct display of 3D room locations.

Considering nowadays Web 3D conditions 3DCIS uses a rather large model (although we broke down the HPCM to the LPCM). Nevertheless, 3DCIS performs real-time rendering of the campus scene.

3DCIS can be used as an information system for university's employees. It is composed like a common 3D game to attract potential students.

This work is not completed yet. In the future we want to add several more features to 3DCIS. Besides a nicer campus scene, with plants and changing light conditions, we want to increase the number of user adaptable "gameplay" options, e.g. the model's level of detail. Additionally, an English version of 3DCIS is currently under construction. According to this step the data management parts of 3DCIS will be re-engineered and a change from Django to a PHP based system is likely.

Acknowledgments

We would like to thank everybody involved in this work for their commitment. Special thanks go to the student project "Interaktiver Campuswegweiser", Christian Fuchs, Markus Lohoff and Christian Schneider.

References

[1] J. Behr, P. Eschler, Y. Jung, and M. Zöllner. X3dom: a dom-based html5/x3d integration model. In *Proceedings of the 14th International Conference on 3D Web Technology*, Web3D '09, pages 127–135, New York, NY, USA, 2009. ACM.

[2] J. Behr, Y. Jung, J. Keil, T. Drevensek, M. Zoellner, P. Eschler, and D. Fellner. A scalable architecture for the html5/x3d integration model x3dom. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, pages 185–194, New York, NY, USA, 2010. ACM.

[3] P. Brunt. Glge api documentation, December 2010. http://www.glge.org/api-docs/.

[4] M. Callieri, R. M. Andrei, M. Di Benedetto, M. Zoppè, and R. Scopigno. Visualization methods for molecular studies on the web platform. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, pages 117–126, New York, NY, USA, 2010. ACM.

[5] B. P. DeLillo. Webglu development library for webgl. In *ACM SIGGRAPH 2010 Posters*, SIGGRAPH '10, pages 135:1–135:1, New York, NY, USA, 2010. ACM.

[6] M. Di Benedetto, F. Ponchio, F. Ganovelli, and R. Scopigno. Spidergl: a javascript 3d graphics library for next-generation www. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, pages 165–174, New York, NY, USA, 2010. ACM.

[7] F. Di Cerbo, G. Dodero, and L. Papaleo. Integrating a web3d interface into an e-learning platform. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, pages 83–92, New York, NY, USA, 2010. ACM.

[8] N. Esnault, J. Royan, R. Cozot, and C. Bouville. A flexible framework to personalize 3d web users experience. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, pages 35–44, New York, NY, USA, 2010. ACM.

[9] KHRONOS. Webgl specification, December 2010. https://cvs.khronos.org/svn/repos/registry/trunk/public/webgl/doc/spec/WebGL-spec.html.

[10] C. Leung and A. Salga. Enabling webgl. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 1369–1370, New York, NY, USA, 2010. ACM.

[11] F. Niebling, A. Kopecki, and M. Becker. Collaborative steering and post-processing of simulations on hpc resources: everyone, anytime anywhere. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, pages 101–108, New York, NY, USA, 2010. ACM.

[12] K. Sons, F. Klein, D. Rubinstein, S. Byelozyorov, and P. Slusallek. Xml3d: interactive 3d graphics for the web. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, pages 175–184, New York, NY, USA, 2010. ACM.

[13] A. Wierse, U. Lang, and R. Rühle. A system architecture for data-oriented visualization. In J. Lee and G. Grinstein, editors, *Database Issues for Data Visualization*, volume 871 of *Lecture Notes in Computer Science*, pages 148–159. Springer Berlin / Heidelberg, 1994.