WebGL-based Interactive Rendering of Whole Body Anatomy for Web-oriented Visualisation of Avatar-centered Digital Health Data

Y. Zhao, X. Zhao, F. Dong, G. J. Clapworthy, N. Ersotelos, E. Liu

Abstract — **The visualisation of whole-body anatomy has a variety of applications in health-related analysis and simulation. However, the rendering of complex 3D human anatomy models is generally performed by standalone applications rather than via a web interface, as rendering large 3D models has always been a weak spot of traditional web browsers. Consequently, online access to, and exploration of, the human anatomy in 3D has not been feasible in the past. With the advent of WebGL and HTML5, high performance OpenGL rendering seamlessly integrated with the web interface is now within reach, and this opens the possibility of visualising avatar-centered health data via a web interface. In this paper, a WebGL-based prototype for rendering whole-body anatomy is introduced, and the technical details are presented.**

I. Introduction

While health-related analysis, simulation, visualisation and education has long had the need for visualising the whole body, the complexity of human-anatomy models has meant that, up to now, only specialised standalone applications have been used for this purpose.

The Internet has become an inseparable part of people's lives, and the challenge of transforming traditional medicine and healthcare to accommodate the Internet era has gained increasing momentum in the research community and industry. There is a noticeable trend for standalone applications to migrate to a web-based environment, and recent technological developments make it feasible for human body visualisation to follow this pattern.

The interactive visualisation of large complex 3D models has not been well supported by traditional web browsers, so online access to, and exploration of, 3D human anatomy has not previously been feasible. However, with the advent of WebGL and HTML5, high performance OpenGL rendering seamlessly integrated with a web interface is now possible, which unfolds the potential of visualising avatar-centered health data via a web interface.

MyHealthAvatar [1], funded by the EC, is an initiative to offer access to, and the collection and sharing of, long-term consistent digital personal health data using the latest ICT technology. The data will be used by both the medical practitioner and the patient; particular aims are to provide patients with greater control of their own data and to engender patients' interest in active participation in their own healthcare.

A 4D avatar is proposed which functions as an interface to support the collection of, and access to, the complete medical information relating to individual citizen's longitudinal health status, gathered from both internal and external sources. Visual analytics will be applied to extract clinically meaningful information from the heterogeneous data of individual/shared avatars.

Related information will be visualised in a body-centric view around the avatar, which will thereby offer significant assistance to both patients and doctors. In this model, web-based interactive visualisation of the whole body anatomy is the key for information visualisation and is the central part of its web interface.

The benefits of using a 3D human body model rather than 2D pictures in an avatar web interface include, but are not limited to the following:

- a 3D human model is more realistic and intuitive
- a 3D human model is a natural representation of an avatar
- with user interactions, a 3D human model can represent and gather more information than 2D pictures do
- the user or patient can learn better about his/her health conditions and disease as well as human anatomy.

In this paper, a WebGL-based prototype for the rendering of whole body anatomy is introduced and the technical details are presented.

II. Related Background

A. WebGL

From the early days of the Internet, Web-based 2D and 3D graphics attracted much interest. Java [2] started as applets in the browser to show interactive graphics. Today, Flash players dominate most browsers' provision of interactive 2D graphics. VRML [3] plugins were previously popular for rendering 3D objects in web browsers, but with the evolution of OpenGL [4] and the advent of HTML5 [5] and WebGL (Web Graphics Library) [6], the 3D graphics capability of

This work was partially supported by the European Commission under Grant FP7-ICT-9-5.2-VPH-600929 within the MyHealthAvatar project..

Y. Zhao is a research fellow with University of Bedfordshire, LU1 3JU, UK (phone: +44-1582-743717; e-mail: youbing.zhao@beds.ac.uk).

X. Zhao is a research fellow with University of Bedfordshire, LU1 3JU, UK (e-mail: xia.zhao@beds.ac.uk).

F. Dong is a professor with University of Bedfordshire, LU1 3JU, UK (e-mail: feng.dong@beds.ac.uk).

G. J. Clapworthy is a professor with University of Bedfordshire, LU1 3JU, UK (e-mail: gordon.clapworthy@beds.ac.uk).

N. Ersotelos is a research fellow with University of Bedfordshire, LU1 3JU, UK (e-mail:nikolaos.ersotelos@beds.ac.uk).

E. Liu is a senior lecturer with University of Bedfordshire, LU1 3JU, UK (e-mail: enjie.liu@beds.ac.uk).

978-1-4799-3163-7/13/$31.00 ©2013 IEEE

web browsers has increased considerably.

WebGL is a JavaScript API for rendering interactive 3D graphics in web browsers. WebGL elements can be mixed with other HTML elements and composited with other parts of the page. The latest versions of Firefox, Chrome and Safari support WebGL, though Internet Explorer will only support WebGL from version 11.

B. Three.js and SceneJS

With WebGL one can write OpenGL programs that render in a web page. However, since version 3.1, OpenGL no longer supports backward compatibility, which implies that the old programming paradigms in OpenGL 1.x and 2.x are no longer favoured, and WebGL users need to write shaders by themselves. This is inconvenient for some programmers who know little about OpenGL shaders.

Three.js [7] is an open-source javascript library, based on WebGL, that is designed to fill the gap by simulating the old OpenGL style programming paradigms such as lighting, material, camera, etc. It is easier to use and does not require knowledge about shaders.

SceneJS [8], another open-source javascript library based on WebGL, provides a JSON-based scene graph API and supports scene graph management. It was created for the efficient rendering of large numbers of objects.

C. Web-based Human Body Visualisation

There are two major 3D human body websites, one is *Zygote Body* [9] and the other is *BioDigital Human* [10]. Both provide interactive visualisation of the whole-body male and female anatomy. The model quality of Zygote Body is better than that of BioDigital Human. Zygote Body shows the whole body after model loading, while BioDigital Human displays a skeleton initially and lets the user select other parts to show. The memory management of Zygote Body also outperforms BioDigital Human, and while the latter provides some special visualisations of conditions, none of these aims at health data visualisation.

The user interfaces of Zygote Body and BioDigital Human are shown in Figures 1 and 2 respectively. BioDigital Human uses SceneJS as its WebGL engine.

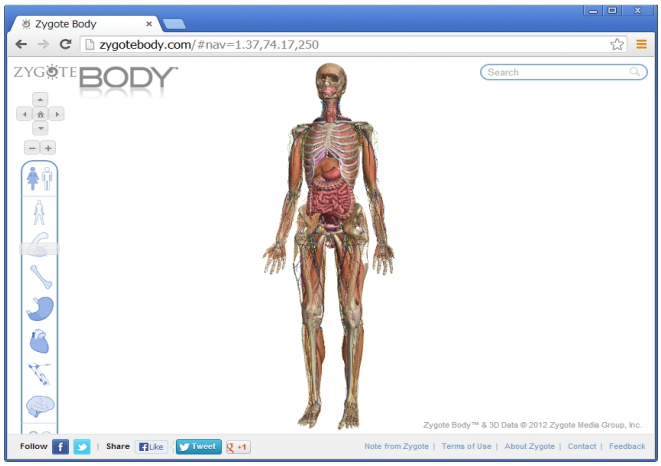

Fig.1. The user interface of Zygote Body

Fig.2. The user interface of BioDigital Human

III. OUR WORK

A prototype of web-based human body visualisation using three.js has been implemented and can be accessed online at http://myhealthavatar.ccgv.org.uk. The current work supports rendering, interactive viewing, part selection and model picking; both Firefox and Chrome have been tested to ensure that it works properly.

In the current prototyping phase, we use three.js for web-based 3D graphics as it provides many examples and is easy to start with. In addition, three.js also comes with loaders to load Wavefront OBJ format directly. We also use JQuery [11] for web user interface.

A. Model Preparation

We obtained a 3D surface-model set of the whole human body of both the male and the female from an online model vendor. The model set comprises 56 3D Studio Max [12] surface models with textures, each consisting of anatomical sub-object groups identified by their Latin names. The quality of the model is medium but the structure is well organised.

The resolution of the model is too high for direct use in web applications. The loading of the unsimplified models will consume a great deal of memory and will result in the browser freezing or crashing. Consequently, the models have been simplified by processing with the ProOptimizer in 3D Studio Max; sometimes, multi-pass mesh reduction or interactive reduction is needed as automatic processing may not fulfil the task. After simplification, most of the models are reduced to 5-10% of their original size, some even lower. For example, the number of vertices in the skull model is reduced from 222k to 8k.

B. Model Loading

We use Wavefront OBJ as the model format as plain text OBJ files are easy to read and it has many importers and exporters. Materials and textures are desirable for realistic anatomy rendering, and it supports these well. Three.js provides an OBJ loader with ObjMTLLoader.js and MTLLoader.js. In addition to texture mapping, the code can

be easily adapted to support bump mapping, but to save memory we decided not to use bump mapping as it almost doubles the texture memory consumption.

In the current prototype, all models are preloaded. The model names and storage location are retrieved from a database dynamically. There are multiple model files to be loaded and all of them are downloaded in parallel by the OBJMTLLoader. The loading time of the whole model set is dependent on the internet connection and the local cache; it usually ranges between seconds and several minutes. A progress bar is displayed to show the loading progress. After all model parts are loaded the skeleton is displayed and the anatomy structure is shown in the model tree in the right panel, as shown in Figure 3. Both the tabbed panel and the tree are implemented using a JQuery plugin - jstree [13].

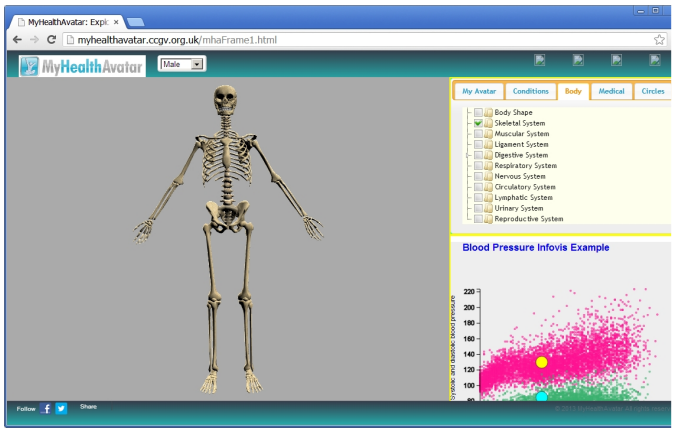

Fig.3. A skeleton after all model parts have been loaded

C. *Interactive Model Rendering*

The core components in three.js for rendering are the scene, renderer, camera and lights. All the models that are shown in the browser are stored in the scene; when an object is to be hidden, it is removed from the scene. The renderer is of type THREE.WebGLRenderer and accepts a scene and a camera as parameters to perform the rendering. Lights and camera settings are very similar to those in the old-style OpenGL programming. Trackball controls are attached to the camera to support interactive model exploration with rotating, zooming and panning.

D. *Adding/Removing Models and Gender Change*

To explore the human anatomy model set efficiently, selective rendering is desired as there are too many body parts and many of them occlude each other. The anatomy tree which supports node selection can be used for part selection. When the user checks a node, a javascript event is trigged and the selected model is added to the scene; when a node is unchecked, the corresponding model is removed from the scene.

When the user wishes to change the gender of the model, for ease of handling, all existing models, including the models shared by male and female, are removed from the scene and the model parts of the changed gender are added to the scene.

Figure 4 shows a female model with the skeleton and internal organs after user selection.

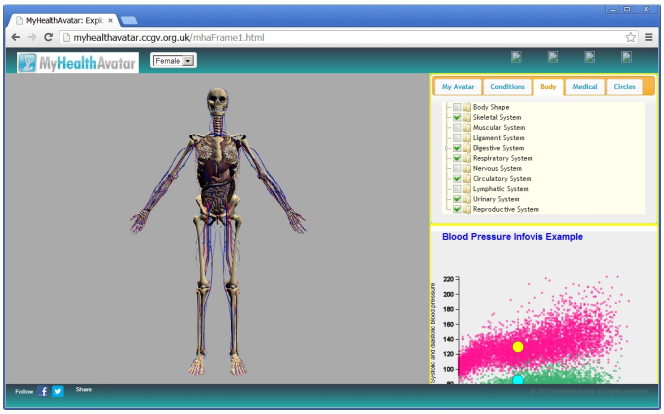

Fig.4. Part selection shows the skeleton and internal organs of the female model

E. *Picking*

For information visualisation of the digital avatar, displaying body-related health data along with the corresponding body parts is an intuitive and efficient way for both patients and doctors. For example, the colour of a dysfunctional organ can be shown in a colour different from its normal colour. When the patient or doctor picks the part, related information automatically displays. Consequently, for interactive health information exploration, visualisation and analysis, picking is a key step in the human-computer loop.

In our prototype, picking is implemented by the raycaster provided by three.js. All the models in the display are stored in a list which is used by the raycaster together with the position of the user clicks. The raycaster calculates all ray intersections and the first one is the foremost one being picked. A popup message box will show up, which can be used to display health related information; in the current prototype it shows the name of the picked part. Figure 5 shows the stomach is picked by the user.

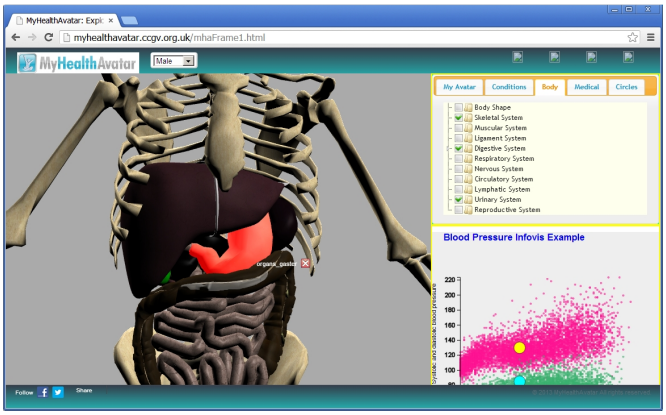

Fig.5. Highlighted stomach and the pop-up message box after user picking

F. *Memory Management*

Most current web browsers do not automatically release all

the model memory allocated during reloading and page refreshing, which makes browser memory consumption accumulate and leads to slow-down, freeze or even crash of the browsers.

Memory management in our work comprises two parts: WebGL memory release of geometry, textures and materials by calling the dispose() method in Three.js; model release by setting the relevant model variables to null while page reloading. Test results of an automatically refreshing page in Chrome (Figure 6) show that memory consumption accumulation has been removed. However, total memory consumption in Chrome and Firefox is still high when all parts of the body are displayed, which implies further model reduction may be needed.

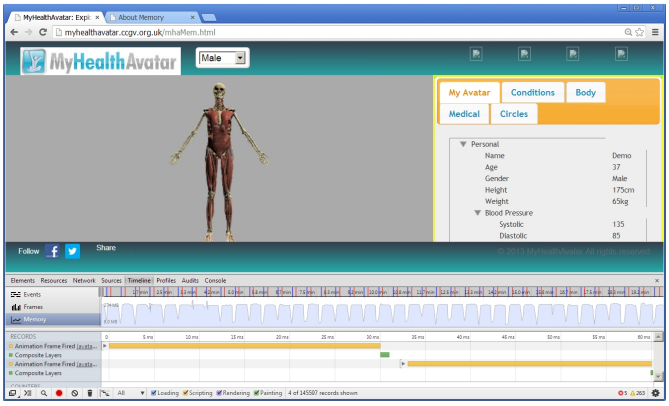

Fig. 6. Timeline tracking of memory consumption in Chrome for an automatically refreshing human rendering page

G. Tools

3D Studio Max is used to view, edit, simplify models and export them to OBJ format. FireBug is used to debug javascript in Firefox. The Web server we use is Apache. To accelerate OBJ model downloading, the GZip option is turned on for .obj files, which could reduce the downloading size to 1/3 ~ 1/4 of its original size.

IV. Conclusion

Our prototype shows that, with the support of WebGL, the Internet has become a feasible platform on which to visualise large 3D models such as the complete human anatomy. Embracing this powerful 3D rendering capability in web pages may provide opportunities for a variety of web-based applications, especially avatar-centered health applications, to present more effective and intuitive visualisation of their data to their end users across the Internet.

V. Future Work

Memory optimisation is still the most important task for the future. Compared to Zygote Body and BioDigital Human, our prototype has a larger memory footprint, and too much memory consumption may cause the browser to freeze, or even crash.

In *MyHealthAvatar*, the targeted users are diverse in their requirements. Providing only a standard human body model for all users ('one size fits all') is not the best way to achieve effective visualisation. We intend to migrate our previous work on musculoskeletal deformation [14] to the web platform and provide the users with a customised avatar body model.

Acknowledgment

This work is partially supported by the European Commission under Grant FP7-ICT-9-5.2-VPH-600929 within the *MyHealthAvatar* project.

References

[1] MyHealthAvatar, http://www.myhealthavatar.eu/
[2] Java, http://www.java.com/
[3] VRML, http://www.w3.org/MarkUp/VRML/
[4] OpenGL, http://www.opengl.org/
[5] HTML5: http://www.w3.org/html/wg/drafts/html/master/
[6] WebGL, http://www.khronos.org/webgl/, http://en.wikipedia.org/wiki/WebGL
[7] Three.js, http://threejs.org/
[8] SceneJS, http://scenejs.org/
[9] Zygote Body, http://www.zygotebody.com/
[10] BioDigital Human, https://www.biodigitalhuman.com/
[11] JQuery, http://jquery.com/
[12] Autodesk 3D Studio Max, http://www.autodesk.com/products/autodesk-3ds-max
[13] Jstree: http://www.jstree.com/
[14] Y Zhao, G J Clapworthy, J Kohout, F Dong, Y Tao, H Wei, N McFarlane, *Laplacian Musculoskeletal Deformation for Patient-Specific Simulation and Visualisation*, Proceedings of Information Visualisation 2013 (IV13), London, July 2013 (electronic)