**MOTIVATION**

Often, users of visualization applications do not have access to high performance systems for the computationally demanding visualization tasks. Rendering the visualization remotely and using a thin client (e.g., a web browser) to display the result allow users to access the visualization even on devices that do not target graphics processing. However, the flexibility to manipulate the data interactively suffers in thin-client configurations. This makes a meaningful interaction with data sets that contain many objects difficult. This is especially true in in-situ visualization scenarios, where direct interaction with the data can be challenging.

We tackle this problem with an approach that employs a deferred visualization pipeline to divide the visualization computation between a server and a client. Our thin client is built on web technologies (HTML5, JavaScript) and is integrated with the D3 libraries to enable interactive data-driven visualizations. An intermediate representation of objects is introduced which describes the data that is transferred from the server to the client on request. The server side carries out the computationally expensive parts of the pipeline while the client retains extensive flexibility by performing object modifications tasks without requiring an expensive re-rendering of the data.

**OUR SYSTEM**

We introduce a novel intermediate representation for deferred visualization. The Volume Object Model (VOM) consists of metadata and pre-rendered visualizations of each object in a data set.

The architecture of our system. The server generates the Volume Object Model and transmits it to the web client on request through a socket-based streaming solution. In deferred rendering, intermediate results of the rendering pipeline are written to buffers which are taken into account in a later stage for the generation of the final image. Previous deferred visualization pipelines generated proxy images as a deferred representation for volume data [3]. Different types of proxies allow for a deferral of specific operations on the volume, such as relighting or limited viewpoint changes. Our approach proposes a pipeline that defers object-level manipulations to the client-side by transmitting an intermediate volume representation.

**INTERFACE & RESULTS**

On the client side, each object is rendered to an HTML5 canvas element, which can be modified through JavaScript. Our system enables the user to script modifications on the object level. Objects can be addressed by their unique ID or by classes, which can be assigned to objects. This permits users to modify individual objects as shown in the examples above, write functions to query, filter, and apply modifications to groups of objects. In summary, our approach allows for the execution of fully interactive object-related visualization tasks in a web browser without triggering an expensive re-rendering on the server.

Reconstructed visualizations of a decorated Christmas tree. On the left, we draw the borders of each object in the visualization. In the middle, we sort and arrange the objects according to their size. We can create complex visualizations from the objects and their metadata, e.g., arranging objects into a scatterplot according to specific features as shown on the right. Here, we plot voxel count vs. density and use each object as a glyph in the scatterplot. Our system allows the user to program object level manipulations on the client side with only a few lines of code.

**REFERENCES**


This thesis was written in cooperation with the King Abdullah University of Science and Technology (Thuwal, Saudi Arabia).