Interactive Software Maps for Web-Based Source Code Analysis

Daniel Limberger \(^{(a)}\) Benjamin Wasty \(^{(b)}\) Jonas Trümper \(^{(a)}\) Jürgen Döllner \(^{(a)}\)

\(^{(a)}\) Hasso-Plattner-Institut, University of Potsdam, Germany
\(^{(b)}\) Software Diagnostics GmbH, Potsdam, Germany
Agenda

- Concept & Design
- 3D Software Maps
- Data Provisioning
- Rendering Details
CONCEPT & DESIGN
Software Visualization

The visualization of artifacts related to software and its development process (a wide definition).¹

Providing pictures of abstract objects, their attributes and relations, within a context.

¹ Diehl, S. 2007. Software Visualization
Motivation of Software Visualization

Problem: Increasing size of software-projects results in management overhead. Companies struggle with essential information gaps between management and development.

Software visualization can reduce those information gaps.
Motivation of Software Visualization

Problem: Increasing size of software-projects results in management overhead. Companies struggle with essential information gaps between management and development.

Software visualization can reduce those information gaps.
Aspects for introducing software visualization to software-engineering processes:

- Identification of use cases that benefit from SoftVis.
- Development of the visualization concept itself.
- Elicitation of required information.
- Selection of appropriate rendering and interaction techniques.
- Provisioning of the visualization.
Provisioning of Software Visualizations to Software-Engineering Processes

Integration
into preexisting tool sets
without imposing usage overhead.

Customization
to project specific process,
requirements, and constraints.

Teaching
developers and managers how to use,
read, and interpret the visualization.
Provisioning of Software Visualizations to Software-Engineering Processes

**Integration**
into preexisting tool sets
without imposing usage overhead.

**Customization**
to project specific process,
requirements, and constraints.

**Teaching**
developers and managers how to use,
read, and interpret the visualization.
Provisioning of Software Visualizations to Software-Engineering Processes

**Integration**

into preexisting tool sets without imposing usage overhead.

**Customization**

to project specific process, requirements, and constraints.

**Teaching**
developers and managers how to use, read, and interpret the visualization.
Integration
into preexisting tool sets
without imposing usage overhead.

Customization
to project specific process,
requirements, and constraints.

Teaching
developers and managers how to use,
read, and interpret the visualization.
A **software map** links a rectangular 2.5D-Treemap with a software system structure and software metrics mapped to visual variables (i.e., height, area, and color).
HEIGHT MAPPED
Perspectives specify a mapping of attributes to visual variables.

Dashboards communicate relevance and context for software maps.
<table>
<thead>
<tr>
<th>X1</th>
<th>Y1</th>
<th>Z1</th>
<th>Z4</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2</td>
<td>Y2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Face Vertices**

<table>
<thead>
<tr>
<th>Vertex X</th>
<th>Vertex Y</th>
<th>Vertex Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>float32</td>
<td>float32</td>
<td>float32</td>
</tr>
</tbody>
</table>

**Color**

<table>
<thead>
<tr>
<th>ID (uint24)</th>
<th>Color</th>
<th>Threshold (uint16)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R0 G0 B0 T0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R0 G0 B0 T0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R0 G0 B0 T0</td>
<td></td>
</tr>
</tbody>
</table>

**Box Attributes**

- **Normal** (uint8)
- **Binar** (uint8)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Considerations on Data Provisioning

Visualization should not interfere with its exploratory nature:
- Fast client-side initialization times.
- Interactive frame rates, even with large software maps.

Considerations for software maps on web clients:
- Well known geometry structure.
  Alternatives for triangle based streaming?
- Use of perspectives reduces number of layout changes.
- Without any geometry rendered, no source code related information is required.
Considerations on Data Provisioning

Visualization should not interfere with its exploratory nature:
- Fast client-side initialization times.
- Interactive frame rates,
  even with large software maps.

Considerations for software maps on web clients:
- Well known geometry structure.
  Alternatives for triangle based streaming?
- Use of perspectives reduces number of layout changes.
- Without any geometry rendered,
  no source code related information is required.
Report, Hierarchy and Geometry Data

Report

- JSON structure.
- Meta information like project name, number of items, authors, revisions, the name of the perspective, metric names and explanations, ids of all labeled items and modules.
- Hierarchy part of the top items per metric.

Geometry

- Binary AJAX request (Array-Buffer).
- Pre-layouted 2D bounding boxes.
- Attributes color, diff threshold, height.
- Not required: id, normal, binary height.

Hierarchy

- JSON structure
- Remaining Hierarchy

Note: we choose to decode geometry data and retrieve vertex attribute arrays on the client.
Report 1

- JSON structure.
- Meta information like project name, number of items, authors, revisions, the name of the perspective, metric names and explanations, ids of all labeled items and modules.
- Hierarchy part of the top items per metric.

Geometry 2

- Binary AJAX request (Array-Buffer).
- Pre-layouted 2D bounding boxes.
- Attributes color, diff threshold, height.
- Not required: id, normal, binary height.

Hierarchy 3

- JSON structure
- Remaining Hierarchy

Note: we choose to decode geometry data and retrieve vertex attribute arrays on the client.
Report ¹

- JSON structure.
- Meta information like project name, number of items, authors, revisions, the name of the perspective, metric names and explanations, ids of all labeled items and modules.
- Hierarchy part of the top items per metric.

Geometry ²

- Binary AJAX request (Array-Buffer).
- Pre-layouted 2D bounding boxes.
- Attributes color, diff threshold, height.
- Not required: id, normal, binary height.

Hierarchy ³

- JSON structure
- Remaining Hierarchy

Note: we choose to decode geometry data and retrieve vertex attribute arrays on the client.
Vertex Attribute Arrays

WebGL limited to GLushort vertices requiring chunked drawing:
20 vertices with attributes, 30 indices per box yields 3276 modules and items per draw call.

- constant index array
- two static arrays (constant vertex attributes)
  - normal as face attribute per-face
  - binary height per-vertex
- chunk specific attribute arrays
  - vertex
  - ID per-box
  - color per-box
  - threshold per-box
Encoding Normals

Axis aligned, five-sided boxes only require five distinct normals.

Encoded by indices:

$$\pm 2 \rightarrow \pm x, \pm 1 \rightarrow \pm z, \text{ and } 0 \rightarrow +y.$$ 

Efficient normal decoding without branching:

```cpp
vec3 decodeNormal(in float i) {
    i -= 2; // i assumed to be scaled in [0,4]
    float a = sign(i);
    return vec3(i - a, 1 - a * a, 2 * a - i);
}
```

Note: no dynamic access to constant arrays within shaders is possible in WebGL.
Encoding Normals

Axis aligned, five-sided boxes only require five distinct normals.

Encoded by indices:

$$\pm 2 \rightarrow \pm x, \pm 1 \rightarrow \pm z, \text{ and } 0 \rightarrow +y.$$ 

Efficient normal decoding without branching:

```cpp
vec3 decodeNormal(in float i)
{
    i -= 2; // i assumed to be scaled in [0,4]
    float a = sign(i);
    return vec3(i - a, 1 - a * a, 2 * a - i);
}
```

Note: no dynamic access to constant arrays within shaders is possible in WebGL.
Stylization for Enhanced Perception

Composition of various deferred, image-based enhancements:

- Shading with light from camera, with fixed altitude.
- Ambient occlusion or unsharp masking the depth.
- Edge enhancements.
- In progress: hybrid rendering
Wrap Up

Software Maps for code analysis in collaboration platforms.

Provisioning of the visualization (integration, customization, teaching).

Provisioning of the data (separation of hierarchy and geometry).

Client data decoding and efficient rendering using WebGL.
More Details within the full Paper

- Image enhancements aiding visual evaluation.
- Importance based LOD-filtered annotations.
- Suitable navigation & interaction techniques.
- Details-on-demand features of individual items.
Gracias por su atención!

Daniel Limberger
daniel.limberger@hpi.uni-potsdam.de

Computer Graphics Systems Group
Prof. Dr. Jürgen Döllner

hpi3d.de  youtube.com/hpicgs  @hpi3d

Software Diagnostics GmbH
softwarediagnostics.com
Exemplary Post Processing
## Numbers in Tables on Loading

<table>
<thead>
<tr>
<th></th>
<th>Android</th>
<th>Customer Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Units #</td>
<td>30,853</td>
<td>90,089</td>
</tr>
<tr>
<td>Vertices #</td>
<td>617,060</td>
<td>1,801,780</td>
</tr>
<tr>
<td>Triangles #</td>
<td>308,530</td>
<td>900,890</td>
</tr>
<tr>
<td>geometry.bin</td>
<td>964 KiB</td>
<td>2,815 KiB</td>
</tr>
<tr>
<td>compressed*</td>
<td>404 KiB / 19 ms / 3 ms</td>
<td>1,175 KiB / 56 ms / 11 ms</td>
</tr>
<tr>
<td>compression ratio</td>
<td>0.42</td>
<td>0.42</td>
</tr>
<tr>
<td>report.json</td>
<td>4 KiB</td>
<td>16 KiB</td>
</tr>
<tr>
<td>compressed*</td>
<td>2 KiB / 0 ms / 0 ms</td>
<td>7 KiB / 0 ms / 0 ms</td>
</tr>
<tr>
<td>compression ratio</td>
<td>0.51</td>
<td>0.45</td>
</tr>
<tr>
<td>hierarchy.json</td>
<td>1,513 KiB</td>
<td>5,429 KiB</td>
</tr>
<tr>
<td>compressed*</td>
<td>480 KiB / 21 ms / 6 ms</td>
<td>1,587 KiB / 70 ms / 19 ms</td>
</tr>
<tr>
<td>compression ratio</td>
<td>0.32</td>
<td>0.29</td>
</tr>
<tr>
<td>GPU Memory</td>
<td>12.082 MiB</td>
<td>34.679 MiB</td>
</tr>
</tbody>
</table>

*gzip (zlib) compression level 1, with compress and decompress times.*
### Numbers in Tables on Rendering

<table>
<thead>
<tr>
<th></th>
<th>Android</th>
<th>Customer Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmission at 1.5 Mbps (synthetic):</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Geometry + Report</td>
<td>5.04 s</td>
<td>14.75 s</td>
</tr>
<tr>
<td>Hierarchy</td>
<td>7.88 s</td>
<td>28.28 s</td>
</tr>
<tr>
<td>Transmission at 7.6 Mbps (synthetic):</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Geometry + Report</td>
<td>1.00 s</td>
<td>2.91 s</td>
</tr>
<tr>
<td>Hierarchy</td>
<td>1.55 s</td>
<td>5.58 s</td>
</tr>
<tr>
<td>Local initialization on i5-3337U:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interface</td>
<td>0.24 s</td>
<td>0.24 s</td>
</tr>
<tr>
<td>Geometry + Report</td>
<td>0.48 s</td>
<td>2.13 s</td>
</tr>
<tr>
<td>Hierarchy</td>
<td>1.11 s</td>
<td>4.27 s</td>
</tr>
</tbody>
</table>

Averaged frame rendering time:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel HD Graphics 4000</td>
<td>5.62 ms</td>
<td>28.06 ms</td>
</tr>
<tr>
<td>Intel Core i5-3337U at 1.80 GHz (WIN)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NVIDIA GeForce GT 650M</td>
<td>1.92 ms</td>
<td>13.62 ms</td>
</tr>
<tr>
<td>Intel Core i7 at 2.80 GHz (OS X)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NVidia GTX 680</td>
<td>2.20 ms</td>
<td>16.04 ms</td>
</tr>
<tr>
<td>Intel Xeon W3530 at 2.80 GHz (WIN)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
System Running on an Ultrabook
System Running on a Tablet

Interactive Software Maps for Web-Based Source Code Analysis  Daniel Limberger  20th of June, Web3D2013