Integrating scale and space in 3D city models

Jantien Stoter
Hugo Ledoux
Martijn Meijers
Ken Arroyo Ohori
5D modelling project

• Full integration of 3D space, time and scale dimensions

• Into a 5D hypercube

• Goal is to implement one

• Advantages:
  1. Ensures consistency across all dimensions (eg no gaps in space/time)
  2. Querying across diff spaces
  3. Managing data more efficient
Capture step-wise changes of generalization process in a directed graph structure.

2D + scale  
Graph of generalisation process
2D + scale = 3D model

Variable scale
Variable scale | No predefined scales
Variable scale  |  No predefined scales
Variable scale  |  No predefined scales
Variable scale  No predefined scales
Principles of 5D modeling

Generalized Area Partitioning-tree (GAP)

One possible data structure
How to build and represent a 4D cube?
G-maps

\[ \phi_2 = \langle \alpha_0, \alpha_1 \rangle \]

\[ \phi_1 = \langle \alpha_0, \alpha_2 \rangle \]

\[ \phi_0 = \langle \alpha_1, \alpha_2 \rangle \]
G-maps

$\phi_0 = \langle \alpha_1, \alpha_2 \rangle$

$\phi_1 = \langle \alpha_0, \alpha_2 \rangle$

$\phi_2 = \langle \alpha_0, \alpha_1 \rangle$
Modification to G-maps to handle GIS data
3D city models + LODs in one hypercube

- Start with LOD3 and perform automatic generalisation to populate the cube
- Storing with G-maps data structure
- We can ensure consistency across the different LODs
- Slicing to obtain one 3D model
Non-horizontal slicing = mixed scale
Modelling of noise along the railway

More details closer to railway
4D space-time example

IFC file of building’s construction
Thank you.
What is needed then?

```c
struct Dart {
    Dart *involutions[n+1];
    Embeddings *embeddings[n+1];
};

struct Embedding {
    Dart *referenceDart;
    Embedding *holes[];
    int dimension;
    ...
    float red, green, blue;
};

struct PointEmbedding : Embedding {
    float x, y, z;
};

struct GMap {
    Embedding* embeddings[];
};
```
<table>
<thead>
<tr>
<th>$n$</th>
<th>object</th>
<th>darts</th>
<th>embeddings</th>
<th>pointers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>point</td>
<td>0</td>
<td>$&lt;1&gt;=1$</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>line segment</td>
<td>2</td>
<td>$&lt;2,1&gt;=3$</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>triangle</td>
<td>6</td>
<td>$&lt;3,3,1&gt;=7$</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>tetrahedron</td>
<td>24</td>
<td>$&lt;4,6,4,1&gt;=15$</td>
<td>222</td>
</tr>
<tr>
<td>3</td>
<td>$n$-simplex</td>
<td>$\prod_{i=1}^{n}(i+1)$</td>
<td>$2^{n+1} - 1$</td>
<td>$2e + 2d(n+1)$</td>
</tr>
<tr>
<td>2</td>
<td>Figure 1</td>
<td>18</td>
<td>$&lt;5,7,3&gt;=15$</td>
<td>138</td>
</tr>
<tr>
<td>3</td>
<td>Figure 2c</td>
<td>96</td>
<td>$&lt;12,20,11,2&gt;=45$</td>
<td>858</td>
</tr>
<tr>
<td>3</td>
<td>Figure 3</td>
<td>128</td>
<td>$&lt;9,20,12,4&gt;=45$</td>
<td>1114</td>
</tr>
</tbody>
</table>

Memory consumption
Difficulties to solve (efficiently)

• Construction (recreation of topology, comparison of two objects)

• Marking and storing temporary values

• Consistency and validity (geometry, topology, both)

• Keeping track of embeddings (e.g. deletion)

• Memory comsuption