Urban Data Visualisation in a web browser

Jérémy Gaillard, Alexandre Vienne, Rémi Baume, Frédéric Pedrinis, Adrien Peytavie, Gilles Gesquière

jeremy.gaillard@liris.cnrs.fr
Introduction

- Web technologies mature enough to allow 3D application on the web
- Sharing virtual city models
  - Collaborative urban management
  - Simulation
  - Training
- Use cases need both semantic and geometric information
- Importance of standards
Introduction

- Challenges:
  - Manage gigabytes of data
  - Aggregate data from multiple data stores
  - Allow multiple representation of the data depending on the user’s need
Introduction

Our contributions:
1. Fluid visualisation
2. Standard-based solution
   - Access to multiple sources of open data
3. Access to semantic information
4. Data loading prioritisation
5. Progressive textures
1. Related works
2. Urban data viewer framework
   a. General architecture
   b. Data preparation
   c. Client architecture
   d. Scheduler
   e. Additional data
3. Results
4. Conclusion
Related works
Related work

3D rendering on the web

3D rendering on the web: declarative approach vs imperative approach

Related work

3D rendering on the web

3D rendering on the web: declarative approach vs imperative approach

Related work

3D GIS data visualisation

Related work

3D GIS data visualisation


Our solution
Urban data viewer framework
Overview

Urban data viewer framework

a. General architecture
b. Data preparation
c. Client architecture
d. Scheduler
e. Additional data
Urban data viewer framework

General architecture

Our framework’s architecture

Implementation of the 3D Portrayal pending standard

3DP standard working group: http://www.opengeospatial.org/projects/groups/3dpswg
CityGML is not an efficient format for city visualisation
CityGML is not an efficient format for city visualisation

- JSON:
  - compact
  - easy to parse in JavaScript
CityGML is not an efficient format for city visualisation

- **JSON:**
  - compact
  - easy to parse in JavaScript

- **DDS:** texture format natively read by the GPU
  - Progressive textures with mipmaps
Urban data viewer framework

Data preparation

3DUSE: LIRIS’ software to read and process CityGML
3DUSE: LIRIS’ software to read and process CityGML
Urban data viewer framework

Data preparation

3DUSE: LIRIS’ software to read and process CityGML
Urban data viewer framework

Detailed client architecture

- Tile Manager
- Scheduler
- Texture Manager
- Server
Urban data viewer framework

Detailed client architecture

- Camera position
- Tile Manager
  - Tile choice
  - Add task: Load geometry
- Scheduler
- Server
- Texture Manager
Urban data viewer framework

Detailed client architecture

- Tile Manager
  - Tile choice
  - Load geometry
  - Add mesh to scene

- Scheduler
  - Add task: Load geometry
  - Trigger: Load geometry

- Server
  - Camera position
  - Geometry request
  - JSON geometry

- Texture Manager
Urban data viewer framework

Detailed client architecture

Tile Manager
- Tile choice
- Load geometry
- Add mesh to scene
- Create texture manager

Scheduler
- Add task: Load geometry
- Trigger: Load geometry

Texture Manager
- Material creation

Server
- Camera position
- Geometry request
- JSON geometry
- Add task: Load texture
Urban data viewer framework

Detailed client architecture

Tile Manager
- Tile choice
- Load geometry
- Add mesh to scene
- Create texture manager

Scheduler
- Add task: Load geometry
- Trigger: Load geometry

Geometry request
- JSON geometry

Server
- Camera position

Texture Manager
- Material creation
- Load texture
- Add texture to material

Texture request
- DDS texture

Geometry request
- DDS texture
Three priority types:
- Top priority: data unloading
- High priority: depends on strategy
- Low priority: depends on strategy

Strategy decision according to:
- Task nature
- Layer type
- Distance to tile
- etc.
Urban data viewer framework

Additional data

- WFS streams with openlayers
- Point cloud (LiDAR)
Results
Demonstration
Results

Data size and loading time for a single tile:

<table>
<thead>
<tr>
<th></th>
<th>JSON Size</th>
<th>Compressed size</th>
<th>Geometry download</th>
<th>Parsing</th>
<th>Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEM</td>
<td>2.2 Mb</td>
<td>0.341 Mb</td>
<td>113 ms</td>
<td>12 ms</td>
<td>57 ms</td>
</tr>
<tr>
<td>Buildings</td>
<td>2.5 Mb</td>
<td>0.401 Mb</td>
<td>84 ms</td>
<td>14 ms</td>
<td>32 ms</td>
</tr>
<tr>
<td>Remarkable buildings</td>
<td>1.2 Mb</td>
<td>0.187 Mb</td>
<td>47 ms</td>
<td>7 ms</td>
<td>15 ms</td>
</tr>
<tr>
<td>LiDAR</td>
<td>21 Mb</td>
<td>2.5 Mb</td>
<td>2520 ms</td>
<td>92 ms</td>
<td>17366 ms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Texture download</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution 0</td>
</tr>
<tr>
<td>Size</td>
</tr>
<tr>
<td>DEM</td>
</tr>
<tr>
<td>Buildings</td>
</tr>
<tr>
<td>Remarkable buildings</td>
</tr>
</tbody>
</table>
Results

Data size and loading time for a single tile:

<table>
<thead>
<tr>
<th></th>
<th>JSON Size</th>
<th>Compressed size</th>
<th>Geometry download</th>
<th>Parsing</th>
<th>Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEM</td>
<td>2.2 Mb</td>
<td>0.341 Mb</td>
<td>113 ms</td>
<td>12 ms</td>
<td>57 ms</td>
</tr>
<tr>
<td>Buildings</td>
<td>2.5 Mb</td>
<td>0.401 Mb</td>
<td>84 ms</td>
<td>14 ms</td>
<td>32 ms</td>
</tr>
<tr>
<td>Remarkable buildings</td>
<td>1.2 Mb</td>
<td>0.187 Mb</td>
<td>47 ms</td>
<td>7 ms</td>
<td>15 ms</td>
</tr>
<tr>
<td>LiDAR</td>
<td>21 Mb</td>
<td>2.5 Mb</td>
<td>2520 ms</td>
<td>92 ms</td>
<td>17366 ms</td>
</tr>
</tbody>
</table>

Texture download

<table>
<thead>
<tr>
<th></th>
<th>Resolution 0</th>
<th>Resolution 1</th>
<th>Resolution 2</th>
<th>Resolution 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Size</td>
<td>Download</td>
<td>Size</td>
<td>Download</td>
</tr>
<tr>
<td>DEM</td>
<td>0.68 Mb</td>
<td>76 ms</td>
<td>2.05 Mb</td>
<td>347 ms</td>
</tr>
<tr>
<td>Buildings</td>
<td>1.92 Mb</td>
<td>3280 ms</td>
<td>5.75 Mb</td>
<td>4998 ms</td>
</tr>
<tr>
<td>Remarkable buildings</td>
<td>1.21 Mb</td>
<td>2563 ms</td>
<td>3.63 Mb</td>
<td>3624 ms</td>
</tr>
</tbody>
</table>

Intel© i5 4590 @ 3.3GHz CPU, NVidia GTX970 GPU

Low parsing time
Results

Data size and loading time for a single tile:

<table>
<thead>
<tr>
<th></th>
<th>JSON Size</th>
<th>Compressed size</th>
<th>Geometry download</th>
<th>Parsing</th>
<th>Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEM</td>
<td>2.2 Mb</td>
<td>0.341 Mb</td>
<td>113 ms</td>
<td>12 ms</td>
<td>57 ms</td>
</tr>
<tr>
<td>Buildings</td>
<td>2.5 Mb</td>
<td>0.401 Mb</td>
<td>84 ms</td>
<td>14 ms</td>
<td>32 ms</td>
</tr>
<tr>
<td>Remarkable buildings</td>
<td>1.2 Mb</td>
<td>0.187 Mb</td>
<td>47 ms</td>
<td>7 ms</td>
<td>15 ms</td>
</tr>
<tr>
<td>LiDAR</td>
<td>21 Mb</td>
<td>2.5 Mb</td>
<td>2520 ms</td>
<td>92 ms</td>
<td>17366 Ms</td>
</tr>
</tbody>
</table>

Texture download

<table>
<thead>
<tr>
<th></th>
<th>Resolution 0</th>
<th>Resolution 1</th>
<th>Resolution 2</th>
<th>Resolution 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Size</td>
<td>Download</td>
<td>Size</td>
<td>Download</td>
</tr>
<tr>
<td>DEM</td>
<td>0.68 Mb</td>
<td>76 ms</td>
<td>2.05 Mb</td>
<td>347 ms</td>
</tr>
<tr>
<td>Buildings</td>
<td>1.92 Mb</td>
<td>3280 ms</td>
<td>5.75 Mb</td>
<td>4998 ms</td>
</tr>
<tr>
<td>Remarkable buildings</td>
<td>1.21 Mb</td>
<td>2563 ms</td>
<td>3.63 Mb</td>
<td>3624 ms</td>
</tr>
</tbody>
</table>

Intel® i5 4590 @ 3.3GHz CPU, NVidia GTX970 GPU

High time disparity
Conclusion and future works

Introduced a framework allowing:

- Efficient visualisation of data from different sources using standards
- GPU-friendly progressive textures
- Data loading prioritisation

Future works (joint works with Oslandia):

- Investigate other texture compression formats
- Webworker integration
- Individual building picking
Urban Data Visualisation in a web browser

Jérémy Gaillard, Alexandre Vienne, Rémi Baume, Frédéric Pedrinis, Adrien Peytavie, Gilles Gesquière

jeremy.gaillard@liris.cnrs.fr