Remote Interactive Walkthrough of City Models

Jean-Eudes Marvie, Julien Perret, Kadi Bouatouch

Visibility streaming

Cell-to-object visibility relationships

- One file per object :
  - building, road, crossroad, park.
- One file per cell :
  - each cell refers to its potentially visible objects.

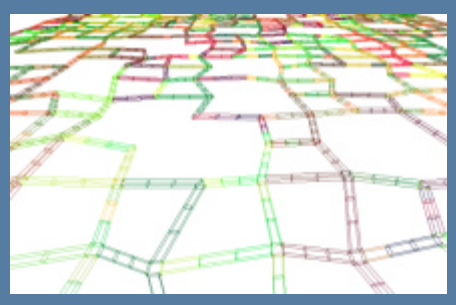
Streaming

1- First cell is downloaded
- potentially visible objects are also downloaded

2- Navigation starts

3- Future visited cells are pre-fetched
- missing objects are downloaded

Navigation space



Navigation is constrained to roads and crossroads (the cells)

Memory management

Client memory is limited
⇒ How to release some memory ?

1- Using the partial adjacency graph
- cells already downloaded

2- Remove the furthest cells & objects
- can be swapped on local disk

Compressed database

L-system based procedural models

One parameter file per building

```
#VRML V2.0 utf8

# Extern PROTO invocation i.e. building model
EXTERNPROTO Build01 [
      field MFVec3f footprint
      field SFInt32 floors
      field MFInt32 adjacentHeights
]
"Library/build01.wrl "

# model instanciation
Build01{
      footprint [
         12.2065,0,246.818, 32.1485,1.0,248.112,
         37.0815,0,228.059, 9.39851,1.0,229.303 ]
      floors  4
      adjacentHeights [ 0, 0, 3, 3 ]
}
```

(300 Bytes)

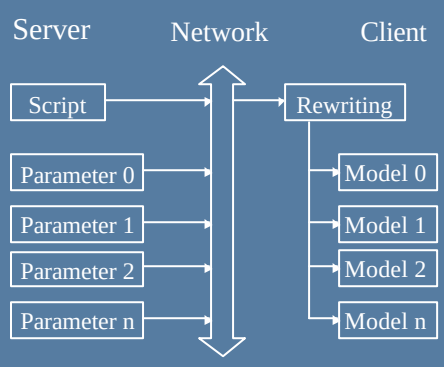
Procedural model script (L-system). Transmitted one time

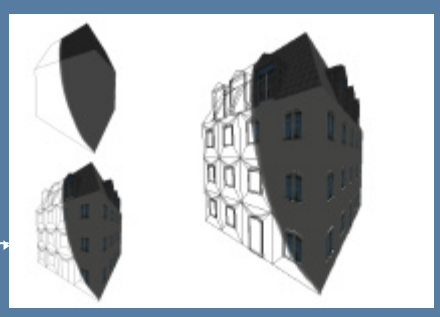
(12 KBytes)

Rewriting on the client side, in parallel

(20 ms)

Transmission mechanism



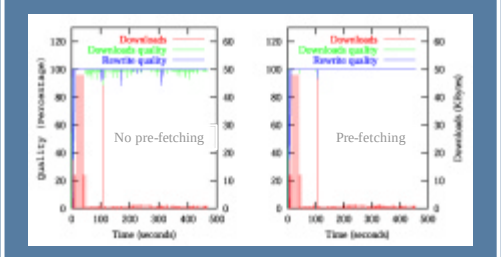


LODs generated for one building

Cell node example

```
DEF cell_1 ConvexCell {
  cellUrl [ "Cells/cell_100.wrl#cell_100" ,
            "Cells/cell_102.wrl#cell_102" ]
  children [
       SharelInline { url "Build/build07.wrl" },
       SharelInline { url "Build/build49.wrl" },
       … ]
  coverageHints [ 0.55, 0.17, … ]
  coord Coordinate {
    point [ 45.0203 0 305.857, 34.3379 0 305.329,
            41.9268 0 317.121, 41.9268 4 317.121,
            34.3379 4 305.329, 45.0203 4 305.857 ]
  }
  cellCoordIndex [ 0, 1, 2, -1, 3, 4, 5, -1,
                   1, 0, 5, 4, -1 , 2 ,1 , 4 ,
                   3, -1, 0, 2, 3 , 2 , -1 ]
}
```

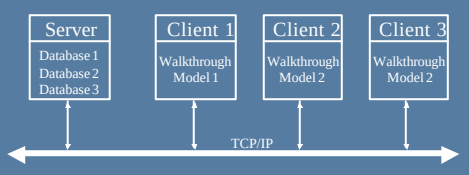
Transmission results



Downloading quality over navigation time. Using pre-fetching or not.

Objective

- **Real time walkthrough**
- **Client-server architecture**
- **Applied to 3D city models**



Bird's eye view of a city model with reconstructed procedural models

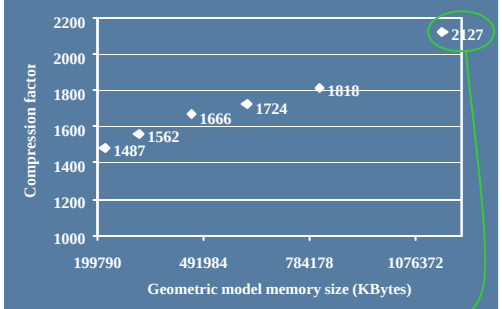
Results

Automatic frame rate adaptation during a walkthrough performed using a 56Kb/s network

Target fps set to 25fps, obtained 26.2, using 56194 polygons.

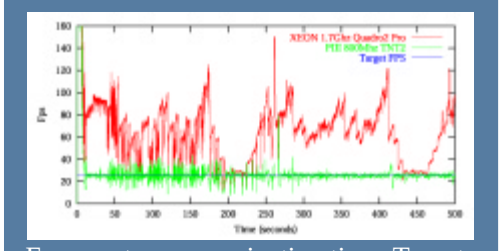
Target fps set to 40fps, obtained 41.7, using 5703 polygons.

Compression factors



Model size = 1,09 GB
Database size = 541.9 KB

Interactivity



Frame rate over navigation time. Target frame rate set to 25fps.

Real-time visualization

Average Coverage Hints (ACH)

A pre-computed selection metric for level of details

Off-line ACH computations

For each cell :

- height camera positions per cell
- six directions per camera position
- render the PVS using color Ids
  1- pixel count for each color (object)
  2- normalize values using total pixel number
  3- get a percentage of coverage for each object (the ACH)

4- store the ACHs values into the cell

On-line ACH computations

For the current cell :

1- perform frustum culling on potentially visible objects

2- renormalize ACH values of objects that are found to be visible

3- The obtained ACHs represent the visual importance of each object in the new frame

Automatic adaptation

Using LODs and ACHs to match a target frame rate

Share a polygon budget

- Analyze frame rate history
- Analyze polygon budget history

→ Deduce a polygon budget for the new frame

→ The polygon budget P is shared by the visible objects

$P_i = ACH_i \cdot P$

LOD automatic selection

- Each LOD node i selects its level of detail whose polygon count is nearest to P_i
- The portion of P_i that is not used for the level is given up to the next LOD node.