A Vrml97-X3D Extension for Massive Scenery Management in Virtual Worlds

Jean-Eudes Marvie
INSA of Rennes

Pr Kadi Bouatouch
University of Rennes

INRIA / IRISA French laboratory
Objective

- Real time navigation
- Client-server architecture
- Applied to complex virtual worlds
Virtual Worlds

• Three different parts
  – Globally dynamic: cars, pedestrians, …
  – Locally dynamic: doors, windows, …
  – Static: walls, roofs, …

• The scenery is made of
  – Locally dynamic and static parts
Sceneries: occlusion
VRML/X3D sufficient?

- Inline, ProximitySensor, VisibilitySensor, Switch, ...
- Insufficient
  - Only apply to parallelepipedic volumes,
  - Cannot tune downloading easily,
  - No control on memory usage,
  - Etc.
Used solutions

• **Database pre-processing**
  – Spatial subdivision of the navigation space
    • Produces a set of cells (the navigation space),
  – Visibility preprocessing,
    • Cell-to-geometry, cell-to-cell or hybrid relationships
  – Database described using our VRML extension

• **Streaming & visualization**
  – Data streaming using partial visibility graphs
  – Data pre-fetching using motion prediction
  – Globally dynamic parts defined in root file
Overview

I – Basis concepts
II – Visibility relationships
III – Database optimisations
IV – Client side managements
V – Some videos
I – Basis concepts

Convex cell
Viewpoint tracking
Navigation space
I - Basis concepts

• Convex Cell

ConvexCell {
  field MFInt32 cadjIndex []
  field MFString cellUrl []
  field SFNode coord NULL
  field MFInt32 coordIndex []
  field MFInt32 cpvsIndex []
  field MFNode lpvs []
  field MFNode opvs []
}
I - Basis concepts

• Convex Cell
  – Cell’s volume described through fields:
    
    | field          | coord      | NULL |
    | field          | MFInt32    | coordIndex |
    
  – Similar to the IndexedFaceSet node
  – Volume has to be **convex** and **solid**
  – Used to find quickly
    • if a point is inside or outside the cell
    • If the cell has to be frustum culled
I - Basis concepts

• Linked viewpoint

```plaintext
# Viewpoint fields plus
eventOut SFString cellUrl ""
```

– New field contains an **extended URL**
  • pointing to the **current cell**
– If cell exists and its convex hull contains the viewpoint’s position
  • both are said to be **linked**
I - Basis concepts

• Extended URL
  – Refer to a node located in another file
  – Similar to EXTERNPROTO
  – Used to distribute the cells descriptions into ≠ files
  – Usage
    • "cells_0.wrl#c2"
      – refer to the cell c2 defined in the file cells_0.wrl
    • "#c2"
      – refer to the cell c2 defined in the current file
I - Basis concepts

• Adjacent cells
  – Refered to through the cell’s fields
    
    ```
    field MFInt32 cadjIndex []
    field MFString cellUrl []
    ```
  
  – Can share some extended URLs with
    • The potentially visible cells (seen later)
  
  – The cells adjacent to the current cell
    • Are downloaded at once
I - Basis concepts

• Navigation space & current cell tracking
  – If viewpoint moves and is not in current cell
    • The adjacent cell that contains the new position becomes the new **current cell**
    • Otherwise it is handled as a collision with the cell’s volume
  – Navigation space
    • The set of cells that can be accessed through this mechanism
I - Basis concepts

• Navigation space
  – Can be the rooms of an indoor scenery
  – Or the streets of an urban scenery
II – Visibility relationships

Cell-to-geometry
Cell-to-cell
Hybrid
II - Visibility relationships

• Cell-to-geometry

Hidden objects

Potentially visible Objects (OPVS)

Current cell
II - Visibility relationships

• Cell-to-geometry
  – Objects can be defined into separated files
  – And referred to in the cell’s field
    
    field MFNode opvs []

  – Using either Inline nodes (bad idea)
  – Or SharedInline nodes (seen later)
  – Thus, an object can be referred to by several cells
II - Visibility relationships

• Cell-to-geometry
  – During the navigation,
    • the OPVS of the current cell
      – Is first downloaded if not present
    • At each new frame rendering
      – The nodes of the OPVS are then frustum culled
      – And only visible ones are rendered

Recall that objects described in the root file are also rendered classically
II - Visibility relationships

- Cell-to-cell

  Current cell &
  local objects (LPVS)

  Hidden cells

  Potentially visible
  Cells (CPVS) &
  local objects (LPVS)
II - Visibility relationships

• Cell-to-cell
  – Cells can be described into separated files
  – And referred to through the cell’s fields

    field MFString cellUrl []
    field MFInt32 cpvsIndex []

  – Objects of the LPVS are described or referred to in the field

    field MFNode lpvs []
II - Visibility relationships

• Cell-to-cell
  – During the navigation,
    • the LPVS and the CPVS of the current cell
      – Are first downloaded if not present
    • then, the LPVSs of the cells of the CPVS
      – Are also downloaded if not present
  • At each new frame rendering
    – The cells of the CPVS are frustum culled
    – The nodes of the LPVS of the visible cells are then frustum culled
    – The nodes of the LPVS of the current cell are also frustum culled
II - Visibility relationships

- Hybrid (HPVS = OPVS + CPVS)
II – Visibility relationships

• Hybrid
  – Uses all the fields depicted for previous relationships
  – Uses downloading and visualization mechanisms presented before
  – Is a first optimisation to reduce the database size
    • Especially the number of references used to describe each PVS
III – Database optimizations

SharedInline

SharedTransform
III – Database optimizations

• Shared inline: motivation
III – Database optimizations

• Shared inline
  – Same prototype as Inline node
    • but the name: SharedInline
  – Can be used anywhere
  – Must be used
    • to refer to the external objects in the opvs field
  – Especially interesting
    • to refer to some external objects in the lpvs field
    • reduces the memory used on client side
  – The URL can be an extended URL
III – Database optimizations

- Shared inline
  - Example:
    Sharing in \textit{lpvs} field
III – Database optimizations

- Shared transform: motivation
  - Objects of the LPVS have to be
    - Contained by the convex hull of the cell
  - If object is partially in two cells
  - Shared transform prevents from cutting the object
  - Object is referred to by the two cells but rendered only once
III – Database optimizations

• SharedTransform
  – Example:
    Sharing in \texttt{lpvs}
    Between two cells
IV – Client side managements

Pre-fetching
Memory management
IV – Client side managements

- Fetching and pre-fetching
  - Example with cell-to-object
  - Step 1: fetch the current cell and its OPVS.
IV – Client side managements

• Fetching and pre-fetching
  – Step 2: pre-fetch the adjacent cells and their OPVS.
IV – Client side managements

• Fetching and pre-fetching
  – Step 3: **pre-fetch** the cells adjacent to the ray-casted cell and their OPVS.
IV – Client side managements

• Memory management
  – Uses the partial adjacency graph to perform topological replacements.
V - Videos
V - Videos

- Cell-to-cell relationships
  - University of Kerlan scenery (370,000 polygons)
- Navigation space
  - Model subdivided using a constrained BSP
- Visibility pre-processing
  - Shooting, non-conservative but fast to compute
- Bandwidth
  - Test performed on a LAN (100Mbit/s)
  - Pre-fetching disabled
V - Videos

• Cell-to-geometry relationships
  – City scenery with procedural geometry
    • (1 million polygons)
  – 1GBytes model encoded using 540KBytes.

• Navigation space
  – Streets generated together with the model.

• Visibility pre-processing
  – Shooting, non-conservative but fast to compute.

• Bandwidth
  – Test performed on a modem (56Kbit/s).
V - Videos

- Cell-to-cell relationships
  - Museum scenery (17,000 polygons)
  - 90MBytes of progressive texture maps
- Navigation space
  - Model subdivided using a constrained BSP
- Visibility pre-processing
  - Shooting, non-conservative but fast to compute
- Bandwidth
  - Test performed at several bandwidths
    - 128Kbits/s, 256Kbits/s, 1Mbits/s and 100Mbits/s.
VI – Conclusion
Contribution

• **Streaming** in VRML97 thanks to visibility
  – Simple and efficient **pre-fetching** and **memory management** solutions.

• Support all types of relationships
  – **Cell-to-cell, cell-to-geometry, hybrid.**

• Complete specification in VRML97
  – With a **single node**,  
  – Plus two optimization nodes.
Questions ?

• More details at:

www.irisa.fr/siames/jean-eudes.marvie