Data Management for Visualizing Large Virtual Environments

Tsai-Yen Li, and Chih-Wei Chiang
Computer Science Department
National Chengchi University
Taipei, Taiwan, R.O.C.
Email: {li, s8409}@cs.nccu.edu.tw

Outline of the Talk

- Introduction
- Problem descriptions and related work
- Proposed approach
  - visibility preprocessing
  - prioritized object prefetching
  - hybrid caching model
- System architecture and implementation
- Experimental results
- Conclusion and future work
Introduction

- **Motivation:** problems of interactive navigation in a large virtual environment
  - Physical memory is never enough for large virtual worlds.
  - Delay and discontinuity in scene changes is disturbing.
- **Intuitions: effective data management**
  - on-demand incremental data retrieval
  - distributing object loading smoothly in each frame
- **Proposed techniques:**
  - visibility precomputation
  - object prefetching
  - hybrid cache model

Problem Description

- **World description:** geometric models and sizes of the objects in the world are stored in a database, and a 2D layout of the virtual world is available.
- **Limited physical memory:** size of a 3D virtual world does not fit into available physical memory.
- **Time utilization for each frame update:**
  - perform object retrieval and graphics rendering in each frame cycle.
    - $t_c$: time for retrieving objects in the current frame
    - $t_g$: time for graphics rendering
    - $t_p$: time for retrieving predicted objects

\[ \begin{array}{c|c|c|c|c|c|c|c} \cdots & t_c & t_g & t_p & t_c & t_g & t_p & \cdots \\ \hline \hline \end{array} \]

\[ \overset{\text{frame n}}{\Rightarrow} \quad \overset{\text{frame n+1}}{\Leftarrow} \]
View Model and Notations

Current viewpoint configuration:
\[ q^c = (x^c, y^c, \theta^c) \]

Next predicted configuration:
\[ q^p = (x^p, y^p, \theta^p) \]

\( \nu \) : linear velocity  
\( \omega \) : angular velocity

Related Work

- **Interactive building walkthrough:**
  - Data management scheme: [Airey 90], [Funkhouser 92], [Teller 91], etc.

- **Scalability for a large number of clients:**
  - Reducing communication complexity: [Carlsson 93], [Li 99], [Macedoniam 95], etc.

- **Cache model:**
  - Least-Recently-Used(LRU) policy: [Franklin 92], etc.
  - Most Required Models(MRM): [Chim 98]

- **Prefetching strategy:**
  - One-step prefetching: [Chim 98], [Teller 91], etc.
Our Approach

- **Visibility pre-computation:**
  - For a given world description, precompute distinct visible sets for all possible discrete viewpoint configurations.
  - Use the visible set to quickly identify the set of objects that need to be retrieved at run time

- **Prioritized object prefetching:**
  - In each time frame, prefetch the most relevant objects of a viewpoint configuration for the near future

- **Hybrid cache model:**
  - Use both access time and object relevance to account for both temporal and spatial localities
  - Consistent with the above prefetching strategy.

Visibility Pre-computation

- Precompute distinct visible sets off-line for quickly identifying visible objects at run time

- Adopt a more conservative approach:
  - retrieving objects in the union of visible sets for all of the current configuration’s 1-neighbors
Object Prefetching:
Prioritized Spatial Prefetching

- Using a n-by-n relevance matrix to define the relevance values of neighboring configurations:
  - analytically defining relevance values according to distance from viewpoint, viewing direction, and velocity (linear and angular)

- Retrieving objects according to the order of their relevance to the predicted configuration:
  - visiting the configurations in the matrix in the order of their relevance values
  - prefetching objects in these configurations within available time

Parameters for Computing a Relevance Matrix

Relevance value \( w(a, d) \): \( w(a,d) = -ad^2 + c \), (c: constant)

\[
a = f(\phi) = a_{\min} + \left(\frac{(a_{\max} - a_{\min})}{\phi_{\max}}\right) \times \phi, \quad \text{if } \phi < \phi_{\max}
\]

\[
a_{\max} = \left(\frac{(a_{\max} - a_{\min})}{(\phi_{\max} - 2\pi)}\right) \times \phi, \quad \text{if } \phi < \phi_{\max}
\]

\[\phi_{\max} = \pi - 2 \times \tan^{-1}(\omega)\]

\[v \rightarrow \phi_{\max} \rightarrow a(\phi) \rightarrow w(a, d)\]
Examples of Computed Relevance Matrices

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>100</td>
<td>50</td>
<td>0</td>
<td>150</td>
<td>200</td>
</tr>
<tr>
<td>S2</td>
<td>0</td>
<td>50</td>
<td>100</td>
<td>200</td>
<td>250</td>
</tr>
<tr>
<td>S3</td>
<td>150</td>
<td>100</td>
<td>150</td>
<td>250</td>
<td>300</td>
</tr>
<tr>
<td>S4</td>
<td>200</td>
<td>200</td>
<td>200</td>
<td>300</td>
<td>300</td>
</tr>
</tbody>
</table>

υ = 0, ω = 0

υ = 0.5, ω = -0.5

Data Caching: Hybrid Cache Model

- Taking advantage of both temporal and spatial localities:
  - For objects retrieved at different frames: use traditional LRU.
  - For objects retrieved at the same frame: order the replacement queue by their relevance values.

- The model is consistent with the prefetching strategies we used while keeping the advantages of LRU.
Example of How Hybrid Caching Works

$q^c$: current configuration
$q^p$: predict configuration

Cache after $q^p$ is retrieved

Prefetching set ordered by object id

System Architecture: a Typical Data Flow Diagram
Implementation and Experiments

- All modules except for the VRML browser are implemented in Java.
- VRML browser $\leftrightarrow$ Control applet:
  - External Authoring Interface (EAI)
- Control applet $\leftrightarrow$ DBMS
  - JDBC
- Experimental Path
  - Collision-free navigation paths are generated by a path planner
- Experimental platform:
  - regular PC with a Pentium II 300 MHz processor running Windows NT

Examples of Experimental Environments

Example 1: 128 objects, 1068 steps

Example 2: 370 objects, 1780 steps
Graphical User Interface for Simulation

Experimental Results: Improvement of display smoothness

- Experiment data of the second example:

<table>
<thead>
<tr>
<th></th>
<th>case 1</th>
<th>case 2</th>
<th>case 3</th>
<th>case 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVRG (ms)</td>
<td>305.6</td>
<td>97.2</td>
<td>97.7</td>
<td>103.8</td>
</tr>
<tr>
<td>STD (ms)</td>
<td>133.6</td>
<td>52.6</td>
<td>51.8</td>
<td>46</td>
</tr>
<tr>
<td>Hit Ratio (%)</td>
<td>N/A</td>
<td>90.9</td>
<td>90.8</td>
<td>95.2</td>
</tr>
</tbody>
</table>

- Improvement:
  - LRU cache + no prefetching:
    - AVRG: 214%, STD: 154% (case 2/case 1)
  - LRU cache + one-step prefetching:
    - slightly improved (case 3/case 2)
  - Hybrid caching + relevance-matrix prefetching:
    - STD was further improved to 190% (case 4/case 1)
Conclusion and Future Work

- Proposed an effective data management scheme utilizing
  - discrete visibility precomputation
  - hybrid cache replacement policy
  - prioritized object prefetching
  to achieve interactive visualization of large virtual worlds.
- Future extensions:
  - completing system integration and doing more experiments
  - finding out the effects between available cache size and allowable prefetching time
  - adopting adaptive prefetching time

Q & A