Peer-Assisted Rendering for Networked Virtual Environments on Mobile Devices

黃瑋璨
新加坡國立大學
Peer-Assisted Rendering for Networked Virtual Environments on Mobile Devices
NetGames / MMVE 2010

Data Acquisition

- Getting network traces is hard
  - Players are (rightly) paranoid
  - Even online affiliation provided little incentive
- Interpreting network traces is hard
  - WoW protocol is pithy and partly secured
  - Mitigation: internet is a treasure trove of information
- End result: simulation input traces which:
  - Avatar position, movement, and some activity
  - ‘Attribution’ of most message bytes which
    successfully parsed
  - Non-parsed, non-attributed bytes discarded
Recommended Requirements for Second Life Client

System Requirements

Your computer must meet these REQUIREMENTS, or you may not be able to participate in Second Life.

<table>
<thead>
<tr>
<th>Windows</th>
<th>Minimum Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Connection*</td>
<td>Cable or DSL connection</td>
</tr>
<tr>
<td>Operating System:</td>
<td>XP, Vista, or Windows 7</td>
</tr>
<tr>
<td>Computer Processor:</td>
<td>CPU with SSE2 support, including Intel Pentium 4, Pentium M, Core or Atom, AMD Athlon 64 or later.</td>
</tr>
<tr>
<td>Computer Memory:</td>
<td>512 MB or more</td>
</tr>
<tr>
<td>Screen Resolution:</td>
<td>1024x768 pixels</td>
</tr>
</tbody>
</table>

- NVIDIA GeForce 6600 or better
- OR ATI Radeon 8500, 9250 or better
- OR Intel 945 chipset

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics Bus</td>
<td>PCI Express</td>
</tr>
<tr>
<td>Memory Interface</td>
<td>128-bit</td>
</tr>
<tr>
<td>Memory Bandwidth</td>
<td>16.0 GB/sec.</td>
</tr>
<tr>
<td>Fill Rate (texels/sec.)</td>
<td>4.0 billion</td>
</tr>
<tr>
<td>Vertices per Second</td>
<td>375 million</td>
</tr>
<tr>
<td>Memory Data Rate</td>
<td>1000 MHz</td>
</tr>
<tr>
<td>Pixels per Clock (peak)</td>
<td>8</td>
</tr>
<tr>
<td>RAMDACs</td>
<td>400 MHz</td>
</tr>
<tr>
<td></td>
<td>400 MHz</td>
</tr>
</tbody>
</table>
Peer-Assisted Rendering for Networked Virtual Environments on Mobile Devices
iPhone 4S: A5 Chip with 7x faster graphics
(Based on benchmark of PowerVR SGX 543MP2 in the A5 in an iPad 2)
Features

- **Instant Messaging**
  Initiate or respond to individual or group IM's.

- **Chat**
  Converse with nearby avatars.

- **World Map**
  View sims, teleport, search or use landmarks.

- **MiniMap & Who's Nearby**
  Zoom in, move & turn, see who's nearby.

- **Inventory**
  View notecards & pictures, give & accept items.

- **Profiles**
  View profiles, make payments, teleport, befriend.

- **Groups**
  View groups, join & leave, send invites.

- **Search**
  Find people, groups, places, regions, and more.

- **Photos**
  Snap pictures and upload as textures.

- **Much More!**
  The most full-featured mobile app for SL.
how to render HQ 3D NVE on mobile devices?
Peer-Assisted Rendering for Networked Virtual Environments on Mobile Devices
3D information

NVE data server

rendering server or cloud

action and parameters

render

video

action and parameters
Interactive 3D Streaming by Alexander Sterkin
posted by Guest Blogger on June 11, 2008

Second Life® and World of Warcraft® are among the most prominent MMOGs. They demand lots of computing power – both from the CPU and Graphics. These demands overload any mobile device of today or near future, even including MIDs. By the time the mobile clients have caught up, the performance requirements for MMOGs will grow higher yet.

The 3D Streaming technology developed by Comverse® and Intel computes and renders the MMOG content on a powerful backend server, then smartly compresses and streams the graphics onto a client. A network gateway designed by Comverse allows streaming over both WiMAX and 3G cellular networks. With advanced software optimizations including SSE usage, a single Xeon 5400 backend system can serve simultaneously up to 14 clients.

What does this mean for users of Intel platforms? In fact, the Comverse 3D Streaming capability offers a great user experience across all Intel platforms. On the backend, it’s the opportunity to offer the power of visual computing on high-end IA multicore platforms. On the client, it’s a chance to drive the demand for MIDs over non-IA smartphones by offering content better suited for larger screens and more sophisticated UI offered by MIDs. Overall, it’s a chance for telecom operators and content providers to offer a completely new service – running on the infrastructure that’s up to 14 clients.
OnLive is cloud gaming, but what does that mean for you? It means instant access to the games you love on nearly any PC or Mac®, TV and many tablets and smartphones. Just like when you watch videos on demand, OnLive delivers the games you want—instantly—right over the Internet. And since our servers are always state-of-the-art, your gaming experience is, too.
6.5 Mbps

average video data rate measured
There are network issues, the upload is too slow to read the controller inputs properly, and a tightened bandwidth mushes the 720p video into an abstract, YouTube-on-a-56k-modem, Jackson Pollock-style blur.

--- WIRED’s review of OnLive
Measuring the Latency of Cloud Gaming Systems

Kuan-Ta Chen¹, Yu-Chun Chang¹², Po-Han Tseng¹, Chun-Ying Huang³, and Chin-Laung Lei²

¹ Institute of Information Science, Academia Sinica
² Department of Electrical Engineering, National Taiwan University
³ Department of Computer Science, National Taiwan Ocean University

ABSTRACT
Cloud gaming, i.e., real-time game playing via thin clients, relieves players from the need to constantly upgrade their hardware, which otherwise may not be able to support the required rendering performance. With the increasing popularity of cloud gaming, the need for accurate latency measurement has become increasingly important. In this paper, we experimentally measure the latency of two popular cloud gaming services, OnLive and StreamMyGame. Our results show that the latency for OnLive is typically in the range of 135-240 ms, while for StreamMyGame, it is in the range of 400-500 ms.
How to reduce bandwidth requirement and interaction delay?
key frames + depth
Image-based Warping Example
(SIGGRAPH Newsletter 33(4), by L. McMillan and S. Gortler)
Warped

Combined

Concealed

Monday, May 7, 12
key frames + depth

3D information

NVE data server

rendering server or cloud

action and parameters

render

action and parameters
Bandwidth reduced by **not** sending every frame

Latency reduced by **computing** new scenes locally
works fine for static scene (e.g., single user building walkthrough)

but NVE is highly dynamic (other avatars are moving)
either send more frequent key frames
or
have more holes in the rendered scene
relying on server or cloud is not scalable and could be expensive
Can we do better?
1. more scalable solution
2. less data sent
3. fewer holes
4. support dynamic scenes
Peer-Assisted Rendering for Networked Virtual Environments on Mobile Devices
Observation 1

NVE is accessed by multi-users simultaneously
Observation 2

Many clients use desktop with powerful GPU to access NVE
Observation 3

Many avatars move very little within the NVE
Observation 4

3D information is available
Observation 5

Many objects are static
Key Idea

Reuse rendering of objects at desktop clients for mobile clients
Can we do better?
1. more scalable solution
2. less data sent
3. fewer holes
4. support dynamic scenes
more scalable:
exploit peer resources instead of relying on servers
exploit multiple peers
less data sent:
rendering of objects are not sent if they change very little
fewer holes:
objects rendered separately without occlusion and composed at client
support dynamic scene:
dynamic objects can be rendered locally
how it works
3 types of rendering elements:

- far static objects (background)
- near static objects
- dynamic objects (avatar)
2 types of peers

assistant (desktop)
assistee (mobile)
repeat:
  determine best set of assistants
  check current list of static objects in view
  if does not have the impostor for the object or view of object has changed significantly
  then
    request for object impostors from assistants
  render the impostors and dynamic objects
repeat:

determine best set of assistants
check current list of static objects in view
if does not have the impostor for the object or view of object has changed significantly
then

request for object impostors from assistants
render the impostors and dynamic objects
repeat:

determine best set of assistants
check current list of static objects in view
if does not have the impostor for the object or
view of object has changed significantly
then
    request for object impostors from assistants
render the impostors and dynamic objects
questions

1. how to find assistants?
2. will there be enough of them in practice?
what makes a good assistant?
good assistants

1. have similar objects in view

(so no need to download extra objects from server)
good assistants

2. have similar view of objects
   (so less errors when warping to assistee’s view)
good assistants

3. have extra computational resources
good assistants

4. good network connectivity
   (low latency, loss rate, high throughput)
good assistants

5. trustworthy
(won’t send random stuff to assistee)
first step:
consider view similarity
View similarity = ratio of overlapped area in the 2D viewing frustum of two avatars
High overlapped area
but viewing angle cannot be too different
for multiple assistants, take the union of the overlapped regions.
VS(a, S): view similarity between assistee a and set of assistants S.
questions

1. how to find assistants?
2. will there be enough of them in practice?
Collected real traces from Second Life

1. insert bot into SL

2. log avatars seen (position, viewing directions) every 10s
<table>
<thead>
<tr>
<th>Region</th>
<th># Records</th>
<th># Avatars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freebies</td>
<td>5786</td>
<td>71</td>
</tr>
<tr>
<td>Japan Resort</td>
<td>5912</td>
<td>61</td>
</tr>
<tr>
<td>Sunland</td>
<td>2516</td>
<td>53</td>
</tr>
</tbody>
</table>

(one hour trace at noon, 23-01-11)
\( VS_{\text{max}}(a, k, t) : \)

maximum view similarity of
- assistee \( a \)
- over all subset of \( k \) assistants
- at time \( t \)
$VS_{avg}(a, k) :$

average $VS_{max}(a, k, t)$ over all $t$
where $a$ appears
Sunland CDF of $VS_{avg}(a, k)$, for $k = 1, 2, 3$
Freebies CDF of $V_{\text{avg}}(a, k)$, for $k = 1, 2, 3$
Japan Resort CDF of $V_{\text{avg}}(a, k)$, for $k = 1, 2, 3$
2-3 assistants are sufficient.

More than half of users can find 2 assistants with \( >0.75 \) similarity on average (in the best case).
questions

1. how to find assistants?
2. will there be enough of them in practice?
Given:

\( a \): assistee

\( k \): limit on # of assistants

\( A \): set of candidate assistants

Find a subset \( S \subseteq A \) such that \(|S| \leq k\) and \( VS(a, S) \) is maximum.
Greedy Heuristic

\( S = \{\} \)

repeat:

\[ x_0 = \arg \max_{x \in A} VS(a, S \cup \{x\}) \]

\( S = S \cup \{x_0\} \)

\( A = A - \{x_0\} \)

until \(|S| > k\) or \(VS(a,S) > \text{threshold}\)
Simulation Results
Rendering Results with Different Number of Assistants

k=1

k=3

k=5
Rendering Results with Different Similarity Threshold

0.6

0.8

1.0
Transmission Overhead

Cumulative Data Transmitted (KB) vs Frames

- Peer Assisted Rendering
- Video-Based Approach
On-going Research
Other considerations for choosing assistants

data completeness
CPU/GPU resources
network conditions
incentive
trust
Other work assistants can perform

<table>
<thead>
<tr>
<th>download extra 3D objects?</th>
<th>render extra 3D objects?</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Other work assistants can perform

download extra 3D objects? | render extra 3D objects?
---|---
X | X
X | ✓
Other work assistants can perform

<table>
<thead>
<tr>
<th>download extra 3D objects?</th>
<th>render extra 3D objects?</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Other ways of partitioning into rendering elements given a frame rate threshold, maximize quality by adjusting what is rendered locally and what is rendered remotely.
Conclusion
Two key ideas:

1. decompose into per-object impostors
2. exploit peers’ rendering capability
謝謝
歡迎發問及指教
Backup Slides