r-buffer: a pointerless a-buffer hardware architecture

craig m. wittenbrink
hewlett-packard laboratories
now at nvidia
cwittenbrink@nvidia.com
• order independent transparency
• r-buffer
• related work
• results on scenes
• architecture
• example
• performance implications
• conclusions
problem

objects composited in drawing order
solution

objects composited in depth order
r-buffer
recirculating fragment buffer

- low cost
- small change to current hardware
- memory sufficient now:
  - 64 MBs support average depth complexity of 10 for 1280x1024 output from simulator:
hardware related work

• software only (many) carpenter84 . . .

• fifo buffer

  mark and proudfoot 2001

• fixed number of layers

  mulder et al.98, jouppi and chang99, kelley94/winner97

• linked lists a-buffer

  baker94, lee kim2000, torborg kajiya96,

• multigeometry passes

  mammen89, kelley94/winner97, kreeger kaufman99

• nonstandard hardware pipeline-app sorting

  torborg kajiya96, snyder lengyel98, kreeger kaufman99
<table>
<thead>
<tr>
<th></th>
<th># of layers</th>
<th>correct</th>
<th>pointerless</th>
<th>app burden</th>
</tr>
</thead>
<tbody>
<tr>
<td>proposed</td>
<td>any</td>
<td>y</td>
<td>y</td>
<td>low</td>
</tr>
<tr>
<td>lee</td>
<td>any</td>
<td>n</td>
<td>n</td>
<td>med</td>
</tr>
<tr>
<td>$z^3$</td>
<td>4-20</td>
<td>n</td>
<td>y</td>
<td>med</td>
</tr>
<tr>
<td>talisman</td>
<td>fixed</td>
<td>y</td>
<td>n</td>
<td>high</td>
</tr>
<tr>
<td>baker</td>
<td>any</td>
<td>y</td>
<td>n</td>
<td>na</td>
</tr>
<tr>
<td>mammen</td>
<td>any</td>
<td>y</td>
<td>y</td>
<td>high</td>
</tr>
</tbody>
</table>
r-buffer simulator output
chevy

z-buffer  artifacts, bad
opengl  keep the rubber on the road
r-buffer
Scene

- z-buffer
- OpenGL
- r-buffer

Artifacts, bad wraparound in drawing cone shows seam
spheres

z-buffer  
last sphere drawn on top  
correct interpenetrations

opengl
helicopter

z-buffer

artifacts, bad

opengl

r-buffer
unstructured volumes
see king, wittenbrink, wolters
volume graphics 2001

phoenix

langley

f117
r-buffer architecture
recirculating fragments
r-buffer high level algorithm

initialize frame buffer
phase1(geom, fbuf, rbufn)
while(!empty(rbufn)){
  swap(rbufn, rbufc)
  phase2/3x(rbufc, fbuf, rbufn)
}
example

- rendered fragments
- viewed from left
- drawing order: t1, t2, t3, o4
- t-transparent
- o-opaque
example cont
phase 1

eye  t3  o4  t2  t1

opaque invalidate

frame buffer

start=next
next
next
example cont.

phase 2

eye

frame buffer

t1

t2

t3

t4

cull

next

start

Requeue

next

start

start
example cont.
phase 31

eye t3 o4 t2 t1

t3 blend over o4
frame buffer

t3 blend over o4
start=next

next
start
intermix all fragments on r-buffer
12 passes frame buffer

back most

small changes
hardware state machines and thorough validation

168 test cases permute rgb drawing order and opaque layer
not a problem as texturing bandwidth dominates

2.35  3.93  2.17  2.66
average depth complexity for covered pixels
performance implications ratio of memory r-buffer/z-buffer systems
results of unstructured data through r-buffer
antialiasing and generalizing to deeper frame buffers

• convert recursive process
• add search mask
• recirculate fragments to compute a-buffer processing
• go front to back
• associative with transparency
• r-buffer with 2 frame stores

\[
C = C_{inA} \times A_{mA} + \]
\[
t_A \times C_{inB} \times A_{mA} + \]
\[
t_{AB} \times C_{inC} \times A_{mA} + \]
\[
t_{ABC} \times C_{inD} \times A_{mA} \]
antialiasing
example
2 frame stores

How they cover
summary r-buffer

advantages

1. no high per pixel dedicated storage
2. any depth complexity as long as average bounded
3. no on chip storage
4. no multiple geometry passes
5. no software sorting needed
6. no approximations
disadvantages

- \( O(Nd^2) \) sort cost
  - \( d \) – average depth complexity
  - \( N \) – number of pixels
- Deep pixel possible
  - \( O(Nd) \), small \( d \)
- Finite limit to on board memory
  - Paging possible with fifo
- Stencil and other modes not worked out
conclusions

- order independent transparency
- r-buffer
- related work
- results on scenes
- architecture
- example
- performance implications
- conclusions
acknowledgements
thank you

• anonymous reviewers
• mike goss
• kevin wu
• hans wolters
• john recker
• hp tcd esp. bruce blah