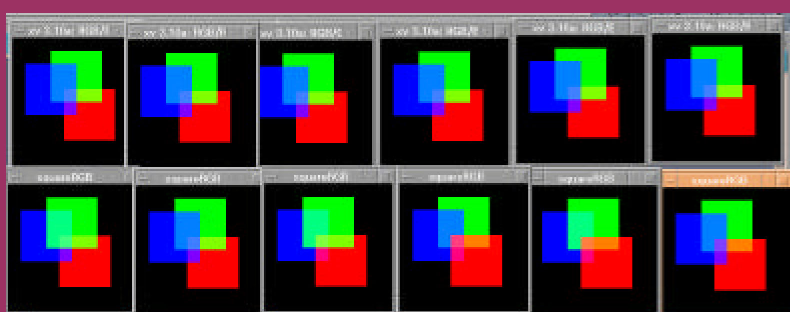
invent

r-buffer: a pointerless a-buffer hardware architecture

craig m. wittenbrink
hewlett-packard laboratories
now at nvidia
cwittenbrink@nvidia.com

outline

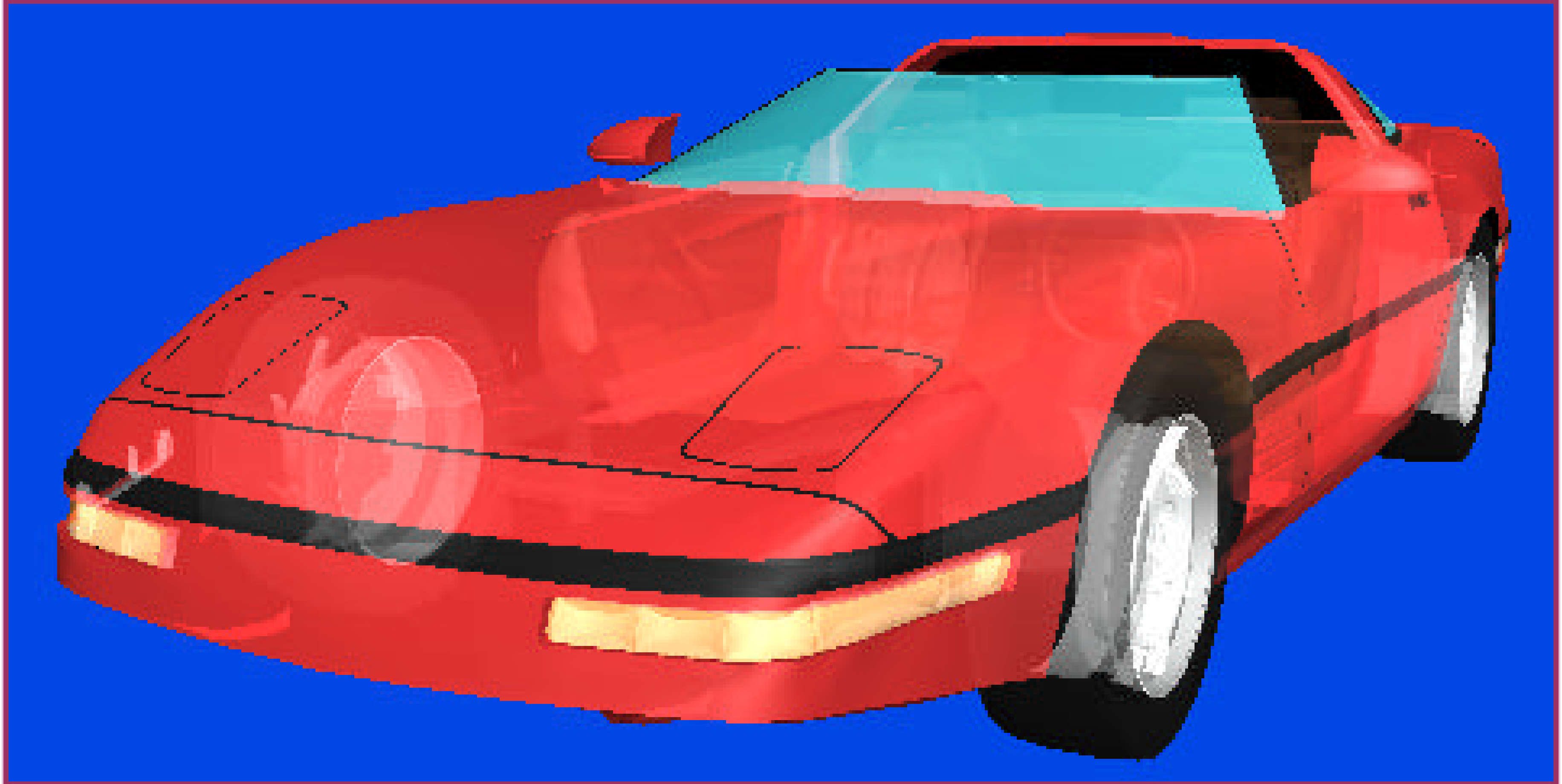


- order independent transparency
- r-buffer
- related work
- results on scenes
- architecture
- example
- performance implications
- conclusions

problem

objects composited in drawing order

solution

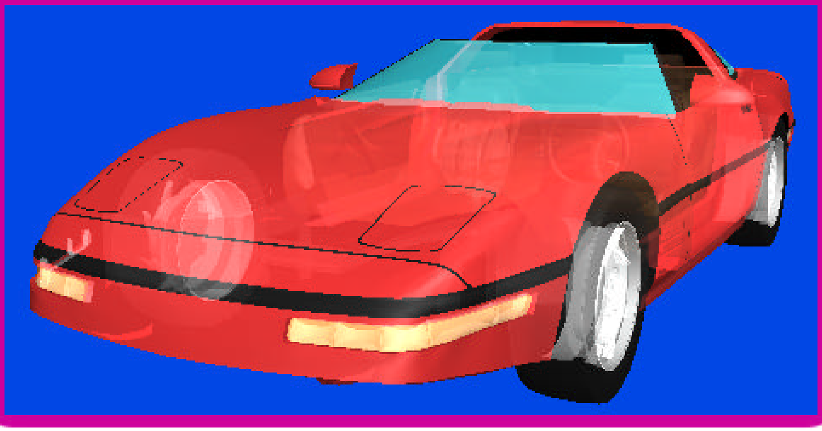


objects composited in depth order

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

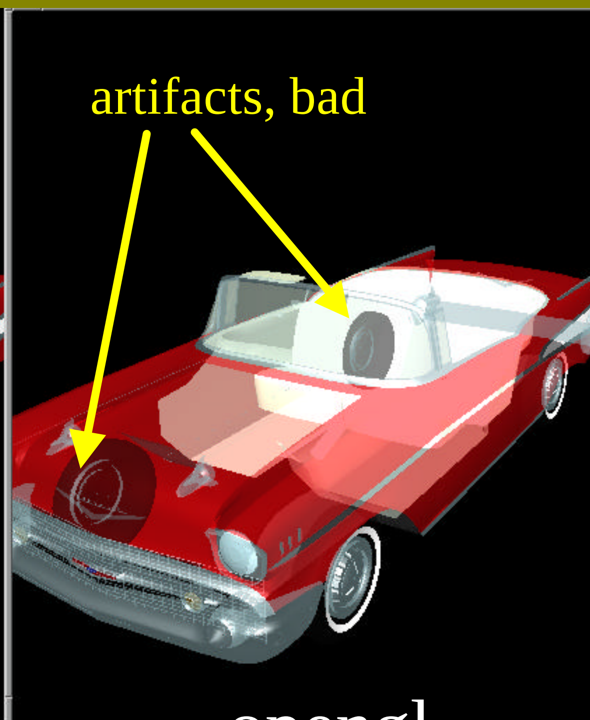
  torborg kajiya96, snyder lengyel98, kreeger kaufman99

related work comparison

	# of layers	correct	pointerless	app burden
proposed	any	y	y	low
lee	any	n	n	med
z^3	4-20	n	y	med
talisman	fixed	y	n	high
baker	any	y	n	na
mammen	any	y	y	high

r-buffer simulator output
chevy

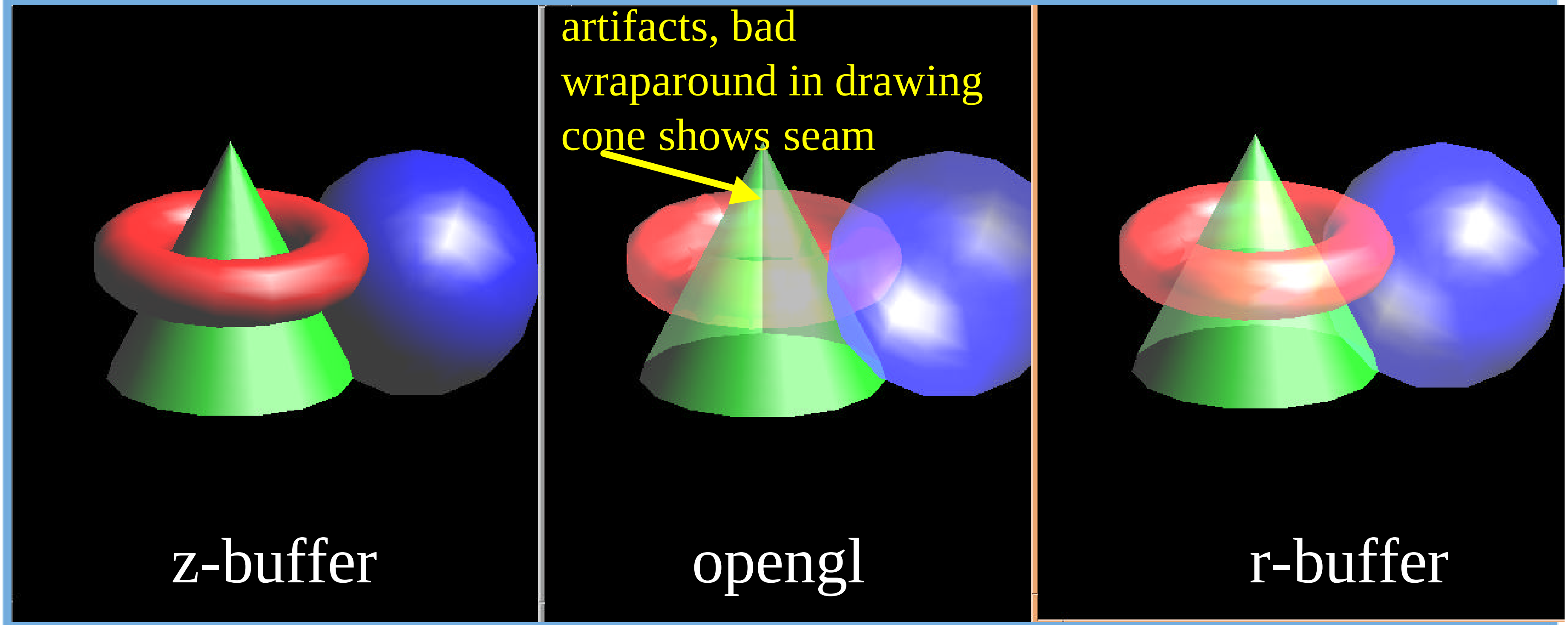
z-buffer



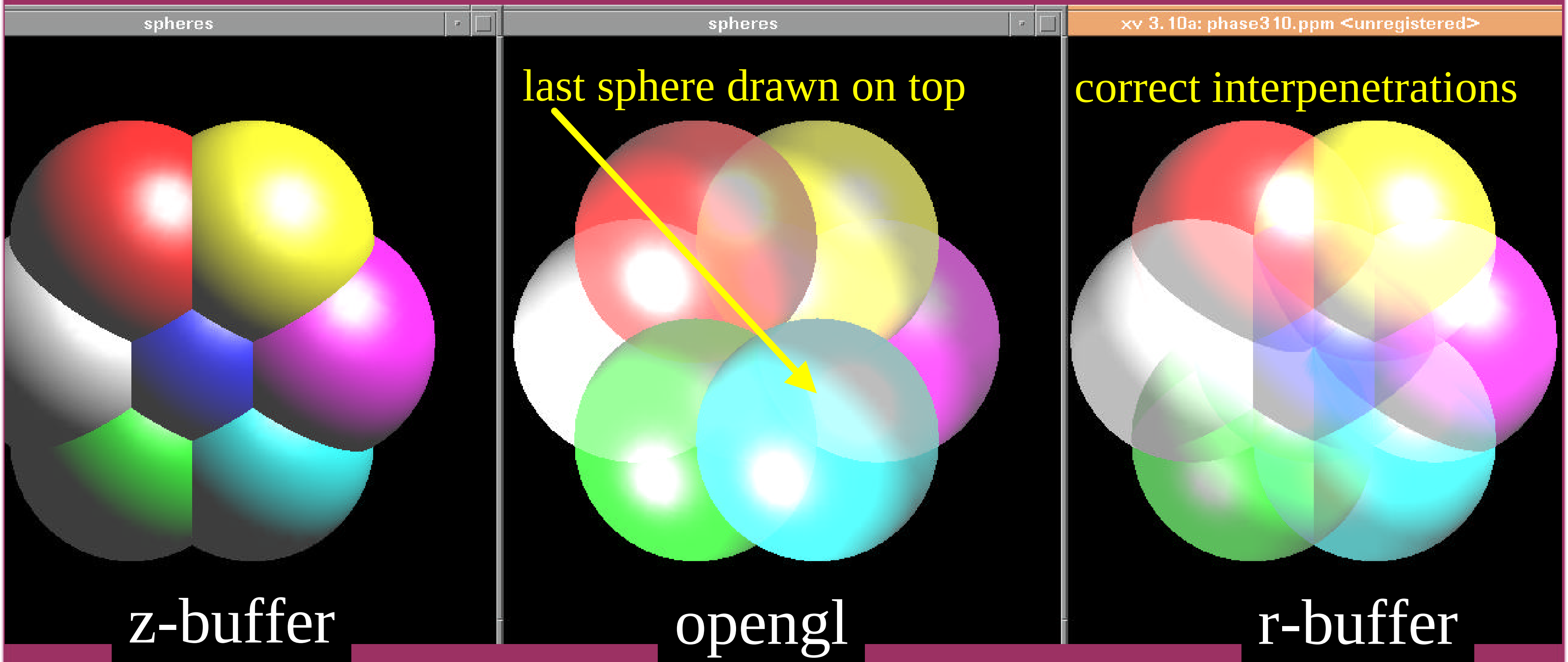
opengl

r-buffer

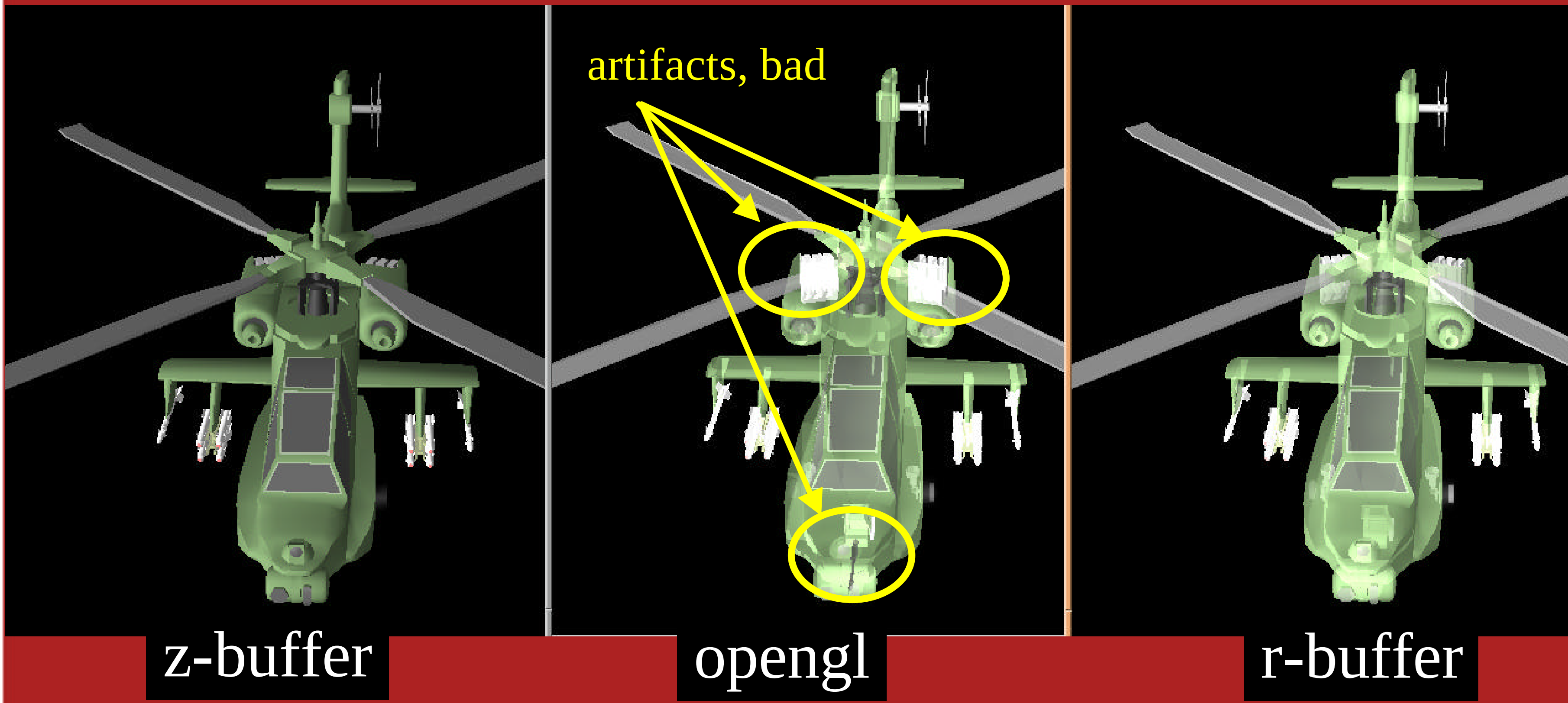
scene


artifacts, bad
wraparound in drawing
cone shows seam
z-buffer
opengl
r-buffer

spheres

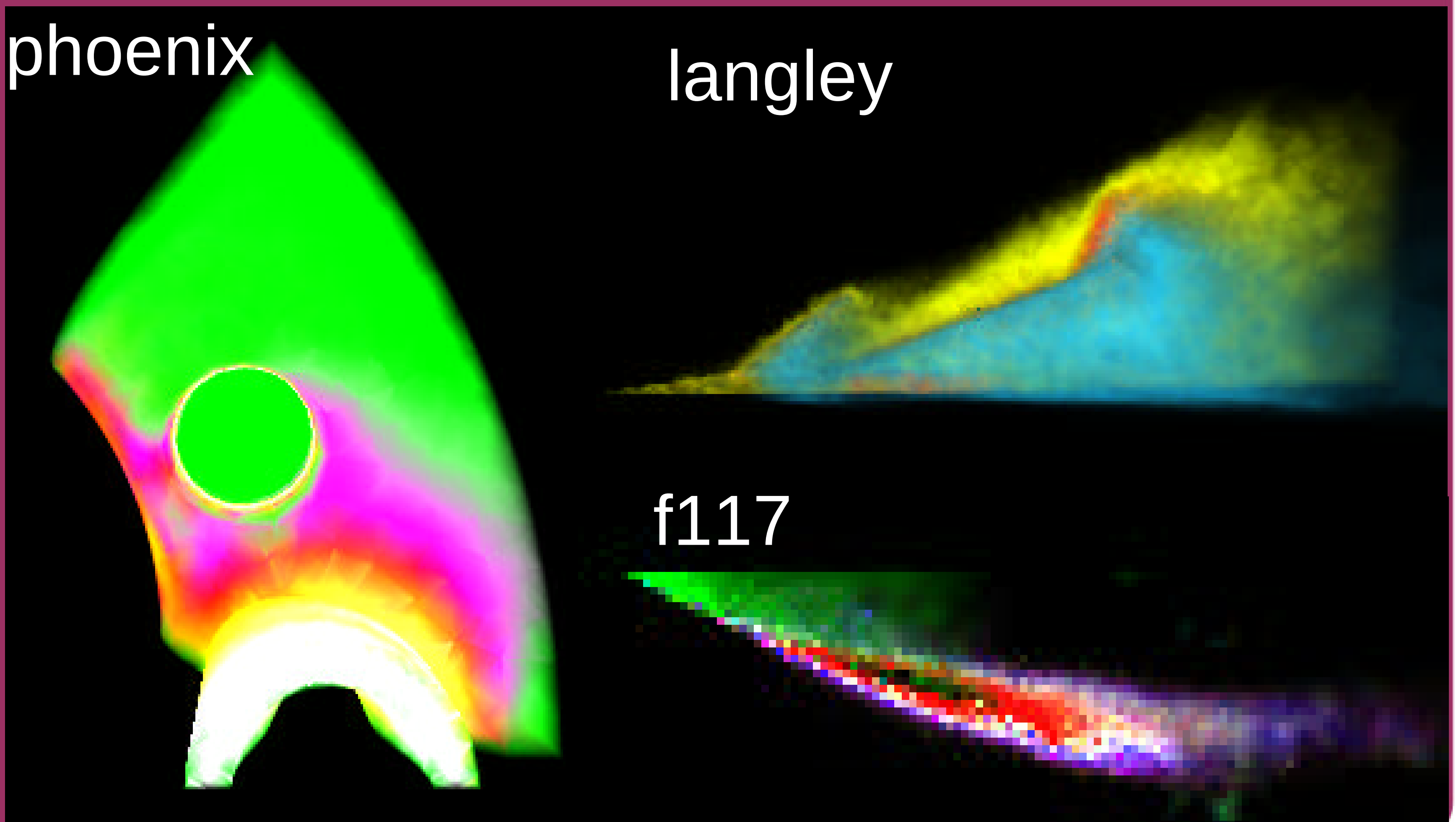

spheres
spheres
xv 3.10a: phase310.ppm <unregistered>
last sphere drawn on top
correct interpenetrations
z-buffer
opengl
r-buffer

helicopter

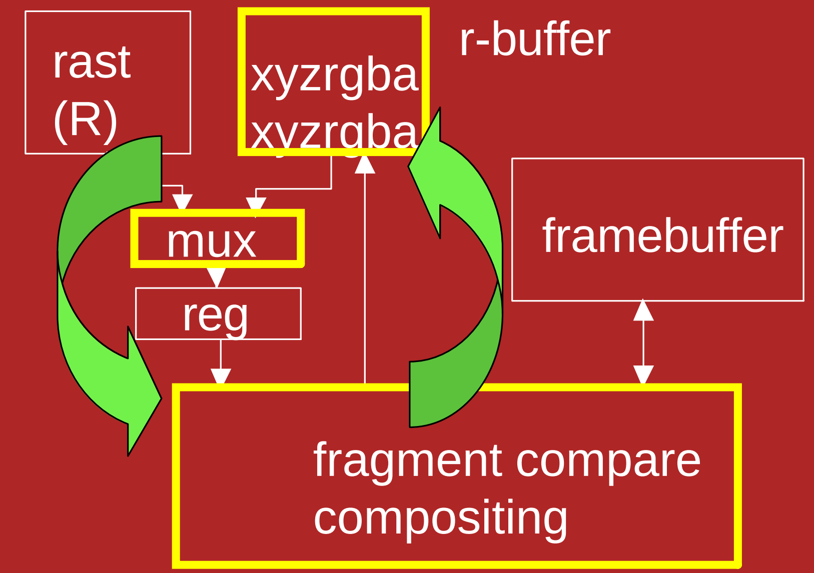


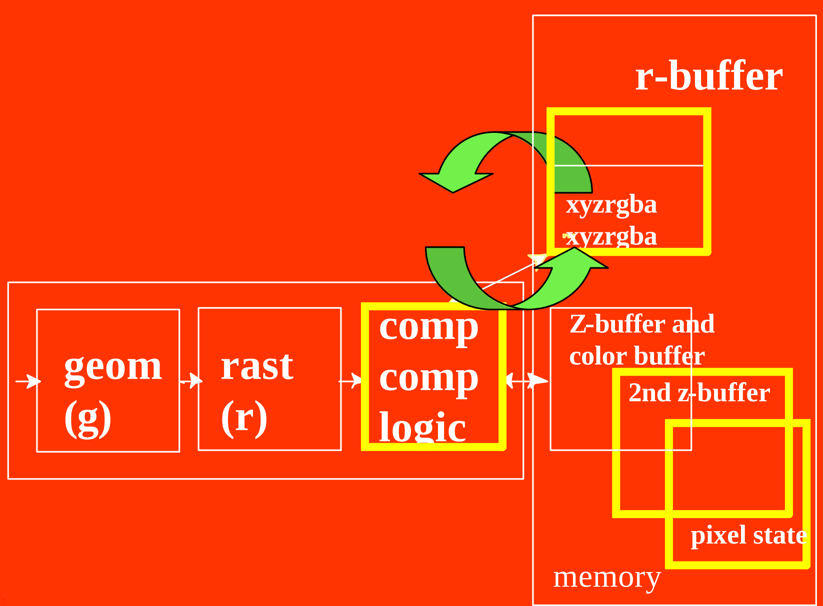
corvette

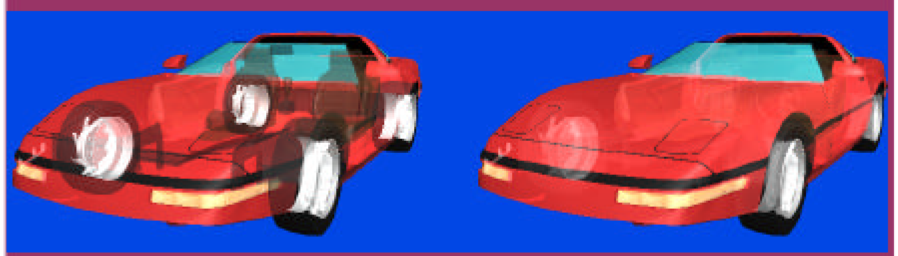
unstructured volumes
see king, wittenbrink, wolters
volume graphics 2001



r-buffer architecture recirculating fragments



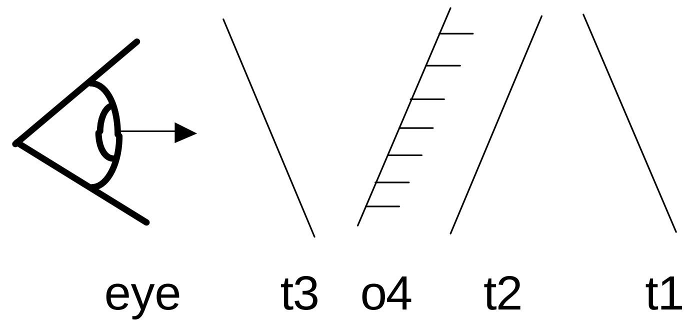




r-buffer high level algorithm

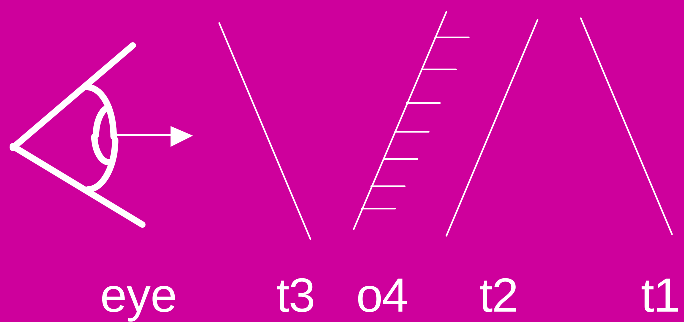
```
initialize frame buffer
phase1(geom,fbuf,rbufn)
while(!empty(rbufn)){
  swap(rbufn,rbufc)
  phase2/3x(rbufc,fbuf,rbufn)
  }
```

example

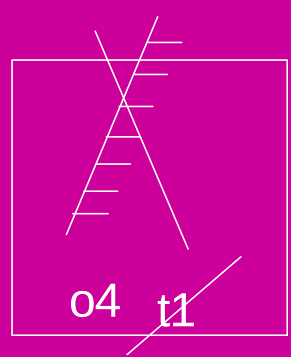


- rendered fragments
- viewed from left
- drawing order: t1, t2, t3, o4
- t-transparent
- o-opaque

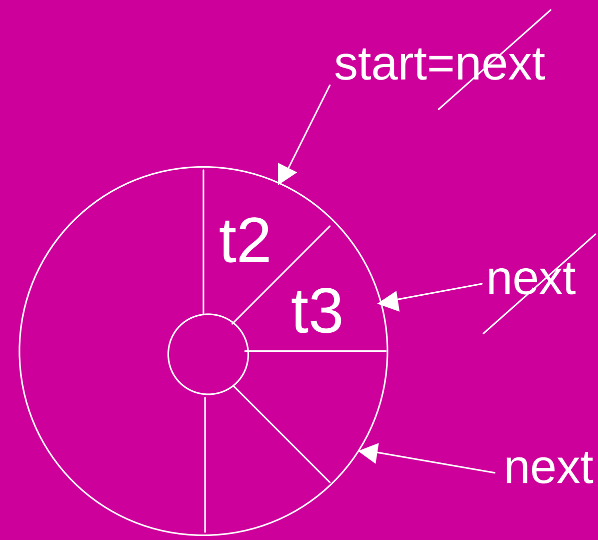
example cont
phase 1



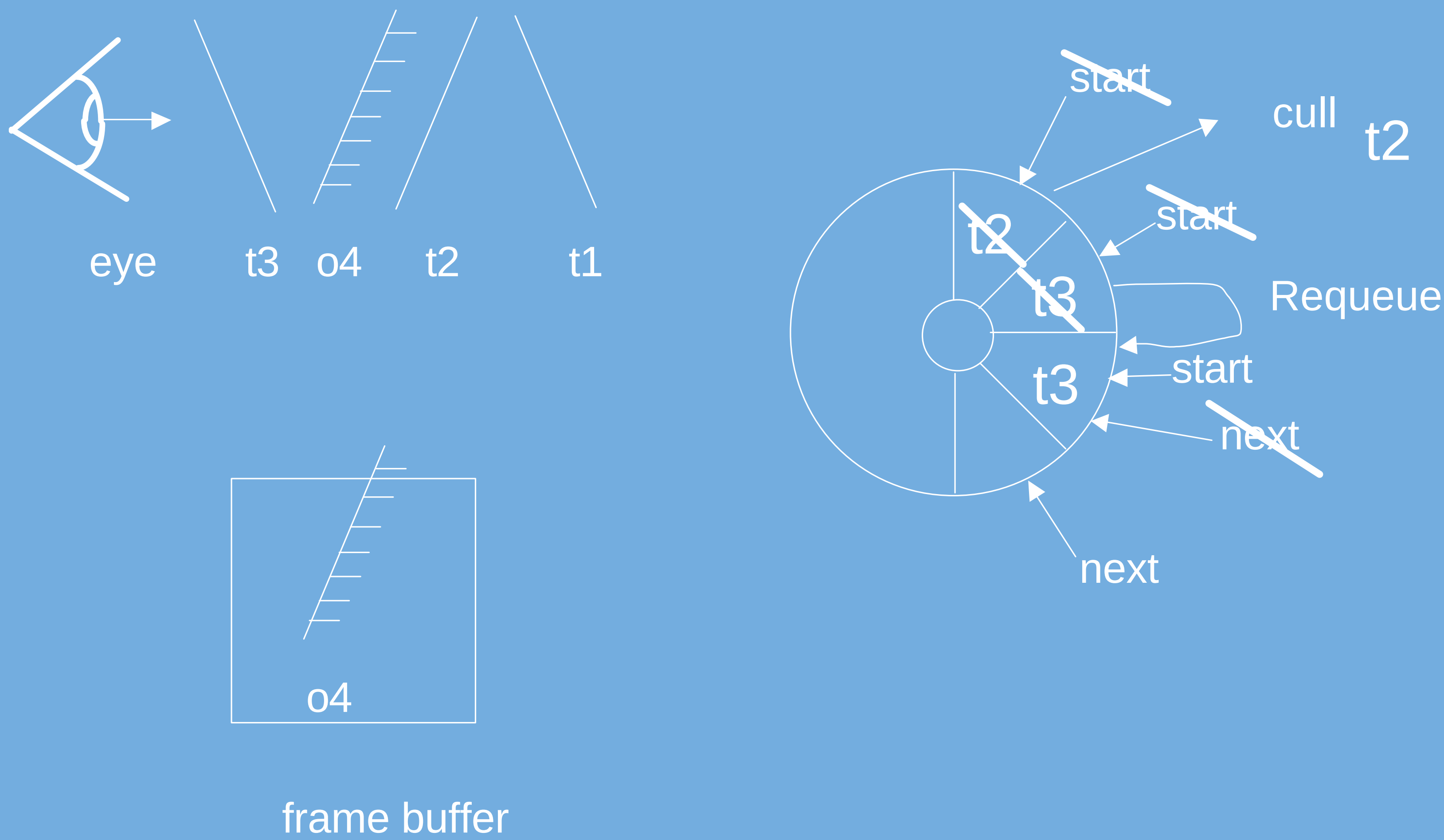
opaque invalidate



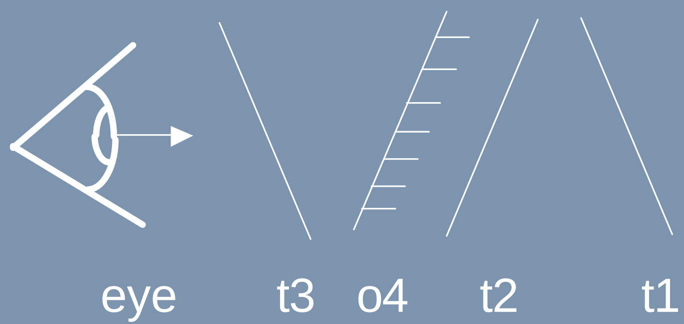
frame buffer

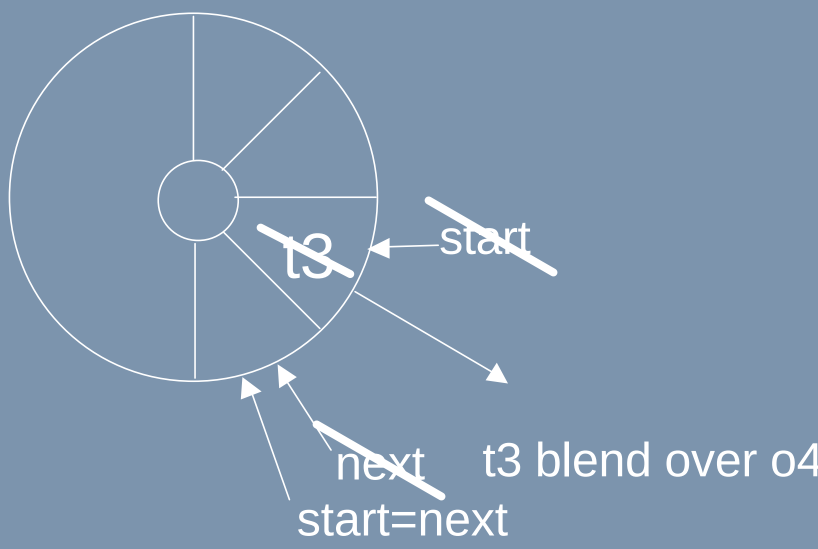


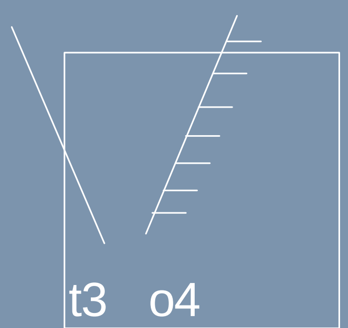
example cont.
phase 2



example cont.
phase 31

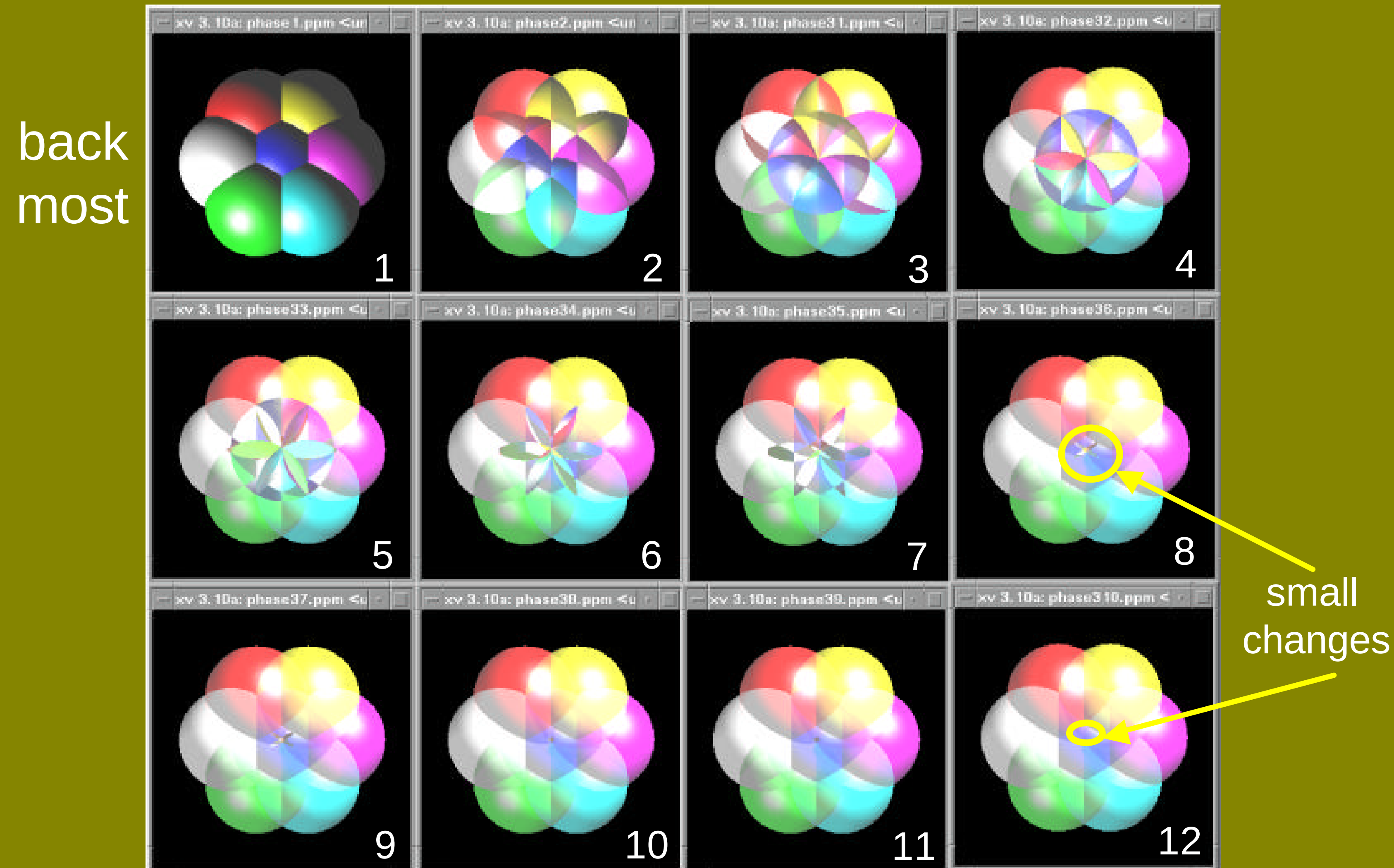




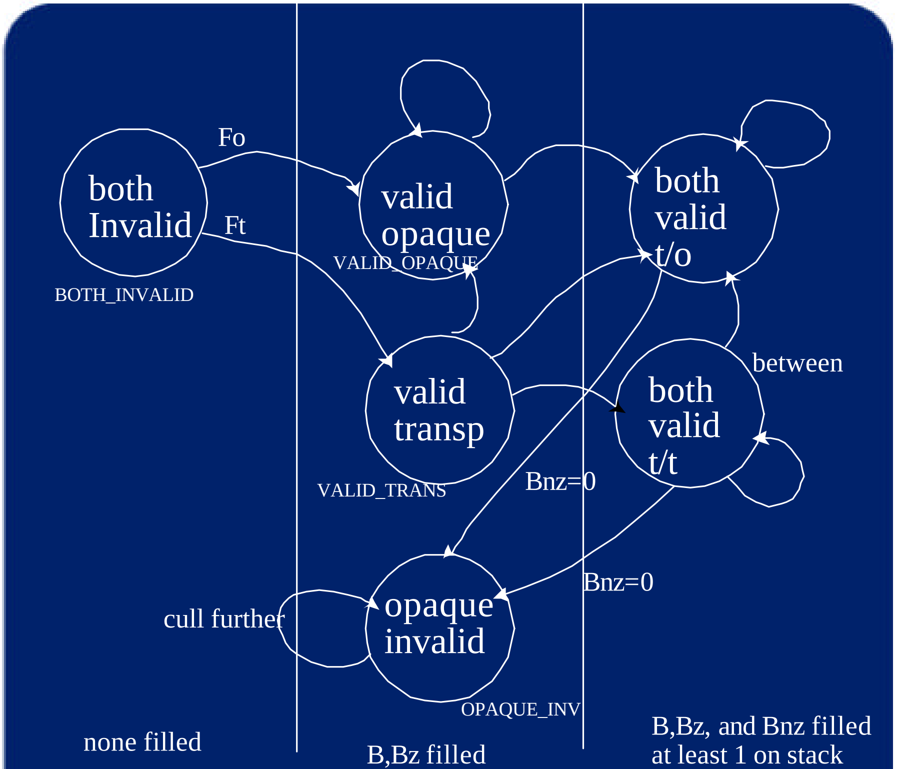


t3 blend over o4
frame buffer

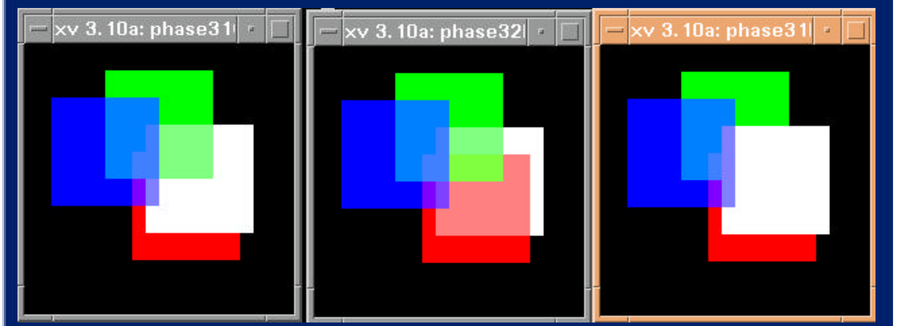
intermix all fragments on r-buffer
12 passes frame buffer



hardware state machines and thorough validation



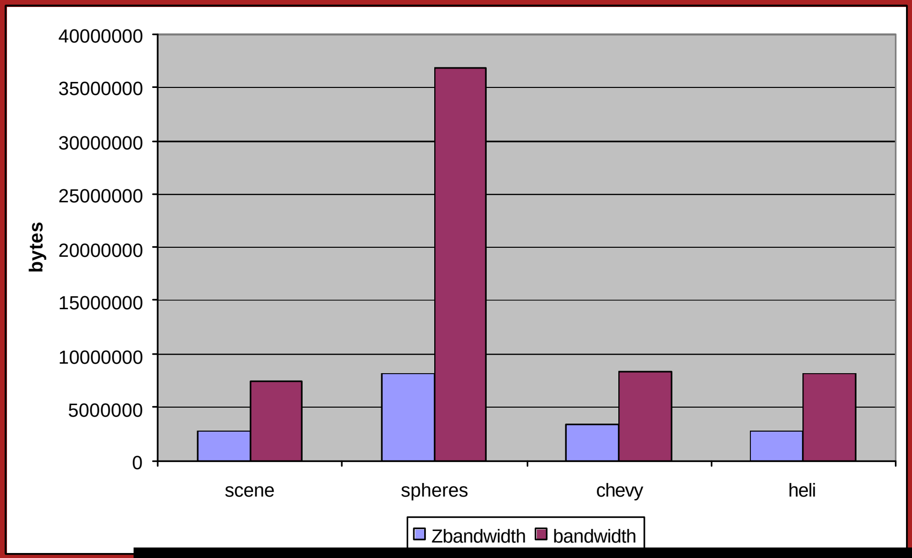
phase 1 pixel state machine



168 test cases
permute rgb drawing order and opaque layer

performance implications-bandwidth

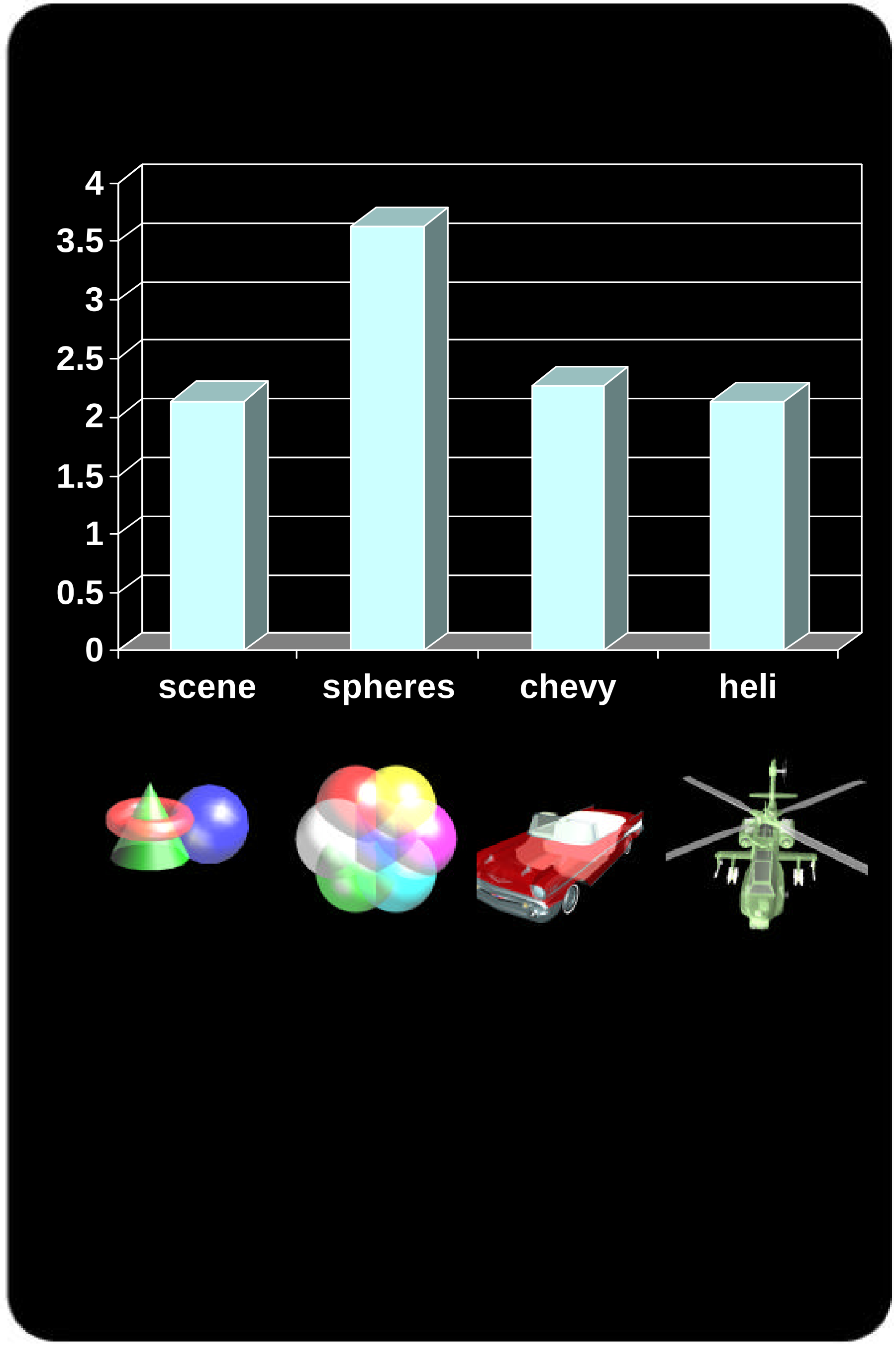
not a problem
as
texturing
bandwidth
dominates



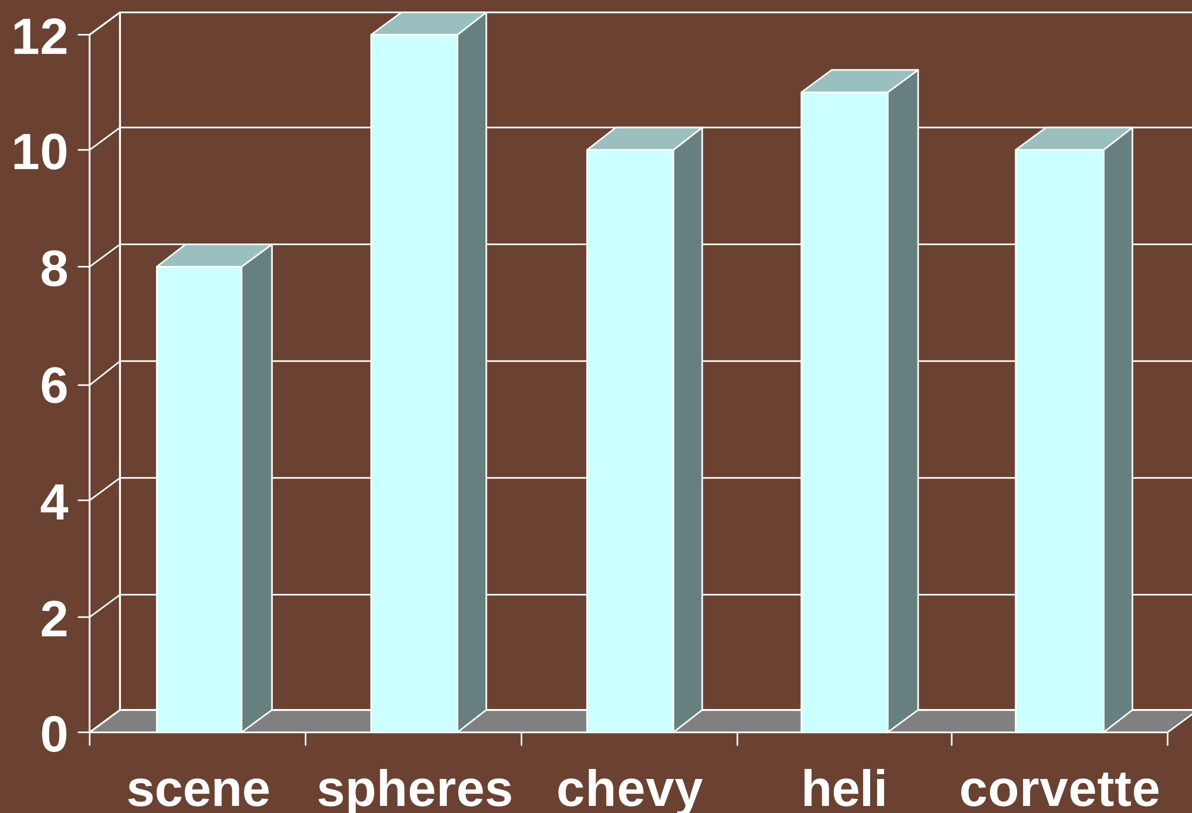
2.35 3.93 2.17 2.66

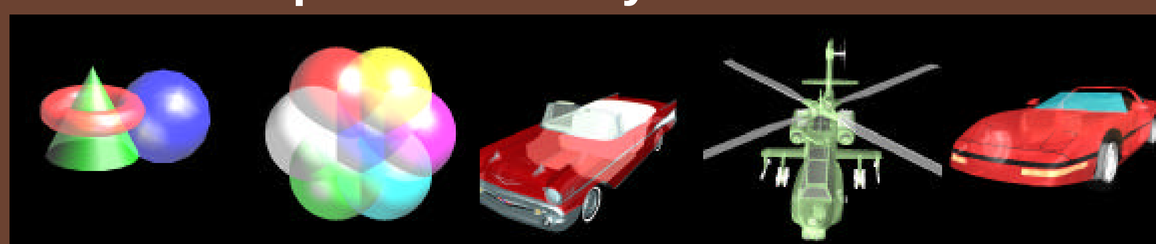
average depth complexity
for covered pixels

performance implications ratio of memory r-buffer/z-buffer systems

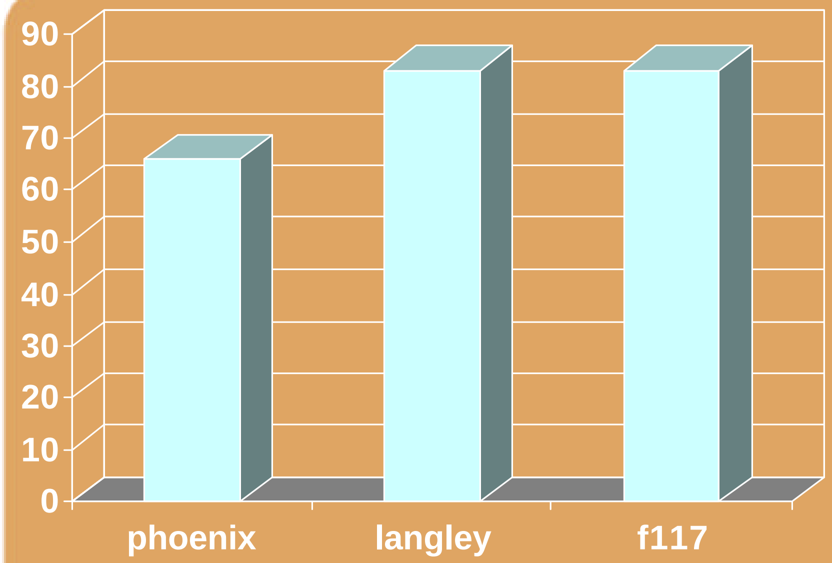


of passes

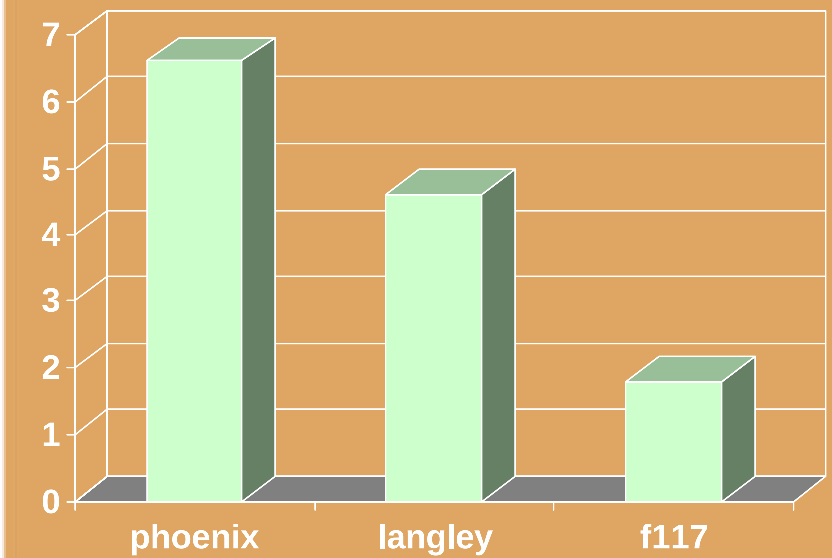




results of unstructured data through r-buffer



number of passes



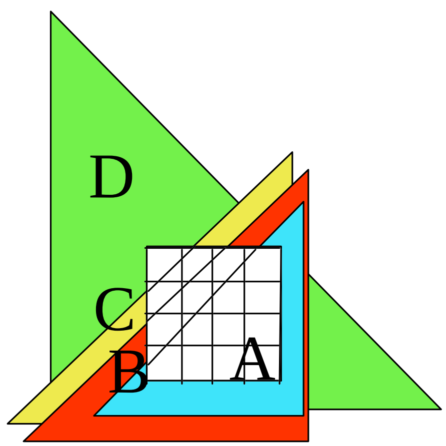
r-buffer memory traffic versus z traffic

antialiasing
and
generalizing to
deeper frame buffers

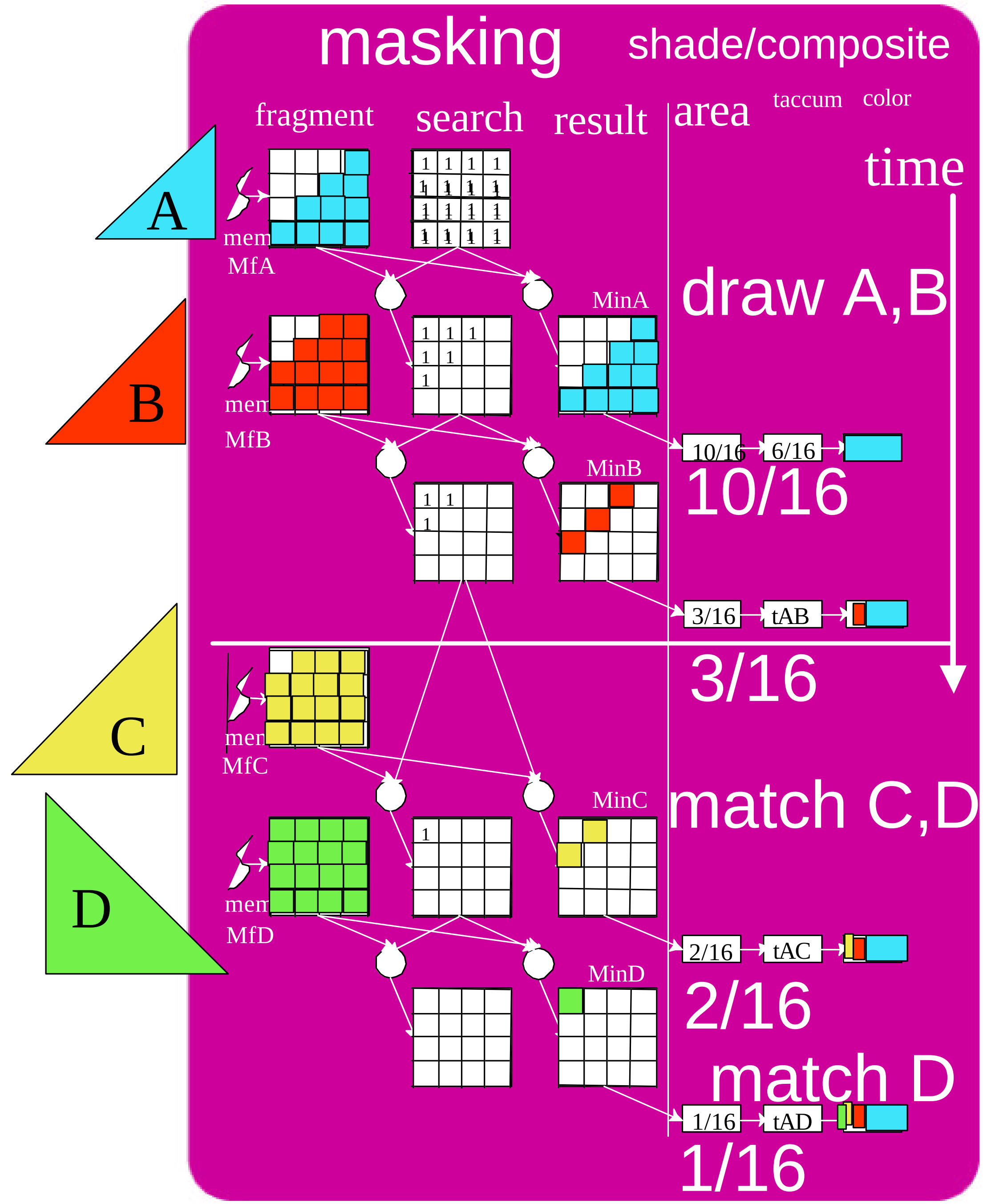
- convert recursive process
- add search mask
- recirculate fragments to compute a-buffer processing
- go front to back
- associative with transparency
- r-buffer with 2 frame stores

$$C = C_{inA} \times A_{inA} + t_A \times C_{inB} \times A_{inB} + t_{AB} \times C_{inC} \times A_{inC} + t_{ABC} \times C_{inD} \times A_{inD}$$

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 blaho