Ray Tracing

Daniel Scherzer

Institute of Computer Graphics and Algorithms

Vienna University of Technology

Why Ray Tracing is Great

- Size

Paul Heckbert

Dessert Foods Division
Pixar
PO Box 13719
San Rafael CA, 94913
415-499-3600

network address: ucbvax!pixar!ph

Why Ray Tracing is Great

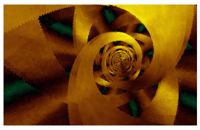
- Size

```
typedef struct{double x,y,z}vec;vec U,black,amb={.02,.02,.02};struct sphere{
vec cen,color;double rad,kd,ks,kt,kl,ir}*s,*best,sph[]={0.,6.,.5,1.,1.,1.,.9,
.05,.2,.85,0.,1.7,-1.,8.,-.5,1.,.5,.2,1.,.7,.3,0.,.05,1.2,1.,8.,-.5,.1,.8,.8,
1.,.3,.7,0.,0.,1.2,3.,-6.,15.,1.,.8,1.,7.,0.,0.,0.,.6,1.5,-3.,-3.,12.,.8,1.,
1.,5.,0.,0.,0.,.5,1.5,};yx;double u,b,tmin,sqrt(),tan();double vdot(A,B)vec A
,B;{return A.x*B.x+A.y*B.y+A.z*B.z;}vec vcomb(a,A,B)double a;vec A,B;{B.x+=a*
A.x;B.y+=a*A.y;B.z+=a*A.z;return B;}vec vunit(A)vec A;{return vcomb(1./sqrt(
vdot(A,A)),A,black);}struct sphere*intersect(P,D)vec P,D;{best=0;tmin=1e30;s=
sph+5;while(s-->sph)b=vdot(D,U=vcomb(-1.,P,s->cen)),u=b*b-vdot(U,U)+s->rad*s
->rad,u=u>0?sqrt(u):1e31,u=b-u>1e-7?b-u:b+u,tmin=u>=1e-7&&u<tmin?best=s,u:
tmin;return best;}vec trace(level,P,D)vec P,D;{double d,eta,e;vec N,color;
struct sphere*s,*l;if(!level--)return black;if(s=intersect(P,D));else return
amb;color=amb;eta=s->ir;d= -vdot(D,N=vunit(vcomb(-1.,P=vcomb(tmin,D,P),s->cen
)));if(d<0)N=vcomb(-1.,N,black),eta=1/eta,d= -d;l=sph+5;while(l-->sph)if((e=l
->kl*vdot(N,U=vunit(vcomb(-1.,P,l->cen))))>0&&intersect(P,U)==l)color=vcomb(e
,l->color,color);U=s->color;color.x*=U.x;color.y*=U.y;color.z*=U.z;e=1-eta*
eta*(1-d*d);return vcomb(s->kt,e>0?trace(level,P,vcomb(eta,D,vcomb(eta*d-sqrt
(e),N,black))):black,vcomb(s->ks,trace(level,P,vcomb(2*d,N,D)),vcomb(s->kd,
color,vcomb(s->kl,U,black))));}main(){printf("%d %d\n",32,32);while(yx<32*32)
U.x=yx%32-32/2,U.z=32/2-yx++/32,U.y=32/2/tan(25/114.5915590261),U=vcomb(255.,
trace(3,black,vunit(U)),black),printf("%.0f %.0f %.0f\n",U);}/*pixar!ph*/
```

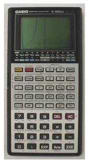
Why Ray Tracing is Great

- Size

Tube by Baze



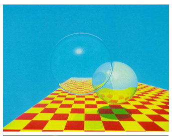
256 byte program



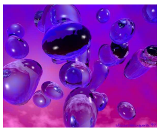
422 byte program for a Casio FX7000Ga, Stéphane Gourichon, 1991

Why Ray Tracing is Great

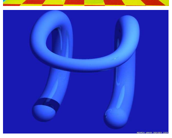
- Shapes: intersectable == renderable



Turner Whitted



William Hollingworth

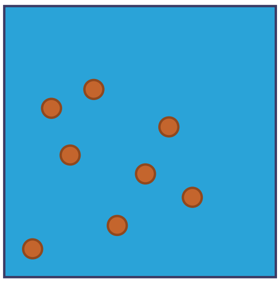


Henrik Wann Jensen

Ken Musgrave

Why Ray Tracing is Great

- Sampling: nonuniform, adaptive

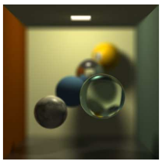


Why Ray Tracing is Great

- Stochastic Effects

by Tom Porter based on research by Rob Cook, Copyright 1984 Pixar

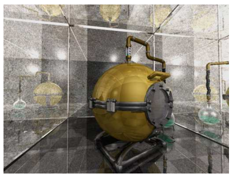
Matt Roberts



Jason Waltman

Why Ray Tracing is Great

- Reflections, Refractions

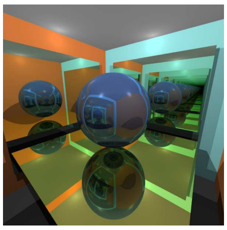


Användare:Mewlek, wikimedia

Gilles Tran, wikimedia

… is Great.

Which is ray traced, which is rasterized?



Timothy Cooper

Kasper Høy Nielsen

And So Is Rasterization

- Reflections, Refractions, even Caustics

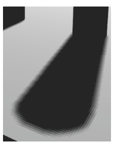
From *Crysis*, by Crytek

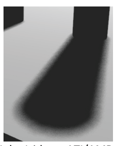
Musawir Ali, Univ. of Central Florida

And So Is Rasterization

- Stochastic Effects

Microsoft SDK



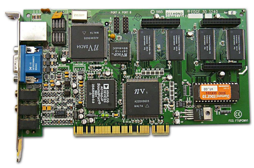


John Isidoro, ATI/AMD

NVIDIA "Toys" demo

And So Is Rasterization

- Shapes:



NVIDIA's first graphics card, the NV1 (circa 1995), supported ellipsoids. This design decision helped almost kill the company.

And So Is Rasterization

- Size: perhaps it cannot fit on a business card, but it can work on a cell phone or iPod.

3DMark Mobile ES 2.0

Z-Buffer

"*… the brute-force approach … is ridiculously expensive.*" - Sutherland, Sproull, and Schumacker, *A Characterization of Ten Hidden-Surface Algorithms,* 1974

Where Rasterization Is

From Battlefield: Bad Company, EA Digital Illusions CE AB

Rasterization Is Just That Simple…

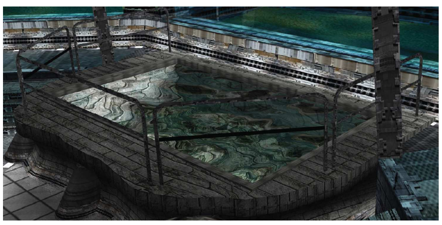
- Shadows
  - Cascading shadow maps, plus enhancements for objects that span the transition between two maps, plus separate buffer for animated objects, plus…
- Transparency
  - Sort objects: error-prone, expensive
  - Alpha to coverage: only good for cutouts
  - Depth peeling: too slow
  - Stencil routing: only on DirectX 10, uses lots of memory, and no AA
- Depth of Field, Motion Blur: …

Strength of Ray Tracing: Simplicity

- Ray tracing is generally easier to program and to think about.
  - Ray casting and ray spawning can do it all.
  - Core optimization pays off everywhere.
  - Maps well to the real world.
  - Easy to explain to artists.

Rasterization: Ray/Simple Object Intersection

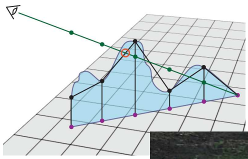
- Done in shader



2001, Alex Vlachos, ATI Technologies Inc.

Ray/More Complex Object Intersection

- Relief (or Parallax Occlusion) Mapping:



Crytek

Ray/Even More Complex Object Intersection

GPU sampled rays, Natalya Tatarchuk, AMD, Inc.

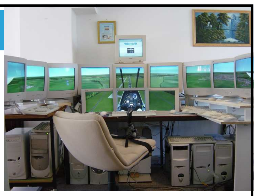
Is this rendered with rasterization or ray tracing? (And does it matter?)

NVIDIA® OptiX™ Ray Tracing Engine

GPUs are the only type of parallel processor that has ever seen widespread success... because developers generally don't know they are parallel! – Matt Pharr

Display Trends

- Rising resolution
- Higher sampling rates
- Larger filtering kernels

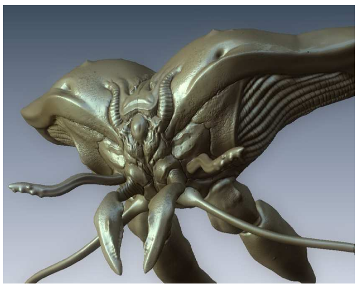


HIPerSpace, UC San Diego, 287 million pixels

You can always use up processing with higher res. Rasterization uses MSAA, CSAA, mipmaps, more.

Scenes are More Complex

- Rate is much faster than display increase.



25 million quads, interactive.

Autodesk Mudbox

Strength of RT: Scene Complexity

- Favors ray tracing, in that the efficiency structure gives effectively O(log n) search (but not build).
  - Look at "rasterization is O(n) vs. ray tracing is O(log n)" argument
  - Rasterization:
    - Hierarchical frustum culling is a given.
    - Level of detail is vital, for pipeline and for limiting memory use; could be useful for ray tracing shadows and reflections, etc. "Space is Speed."
    - Occlusion culling is getting better (still not great). GPU hierarchical occlusion culling is built-in (HyperZ).

Interactive Rendering

- What is the most important effect in interactive/real-time rendering of any sort?

Interactivity: 6+ FPS
Real-time: 30+ FPS

Challenge: Upper Cost Limit

- You have 33 ms for 30 FPS (or 16.7 ms for 60 FPS).
- Reflection maps and similar are constant-cost.
- Shadow volumes aren't, so are dying out.
- Ray tracing: zoom on refractive object and the ray tree explodes, killing frame rate
- Complex shaders and algorithms make this also a rasterization problem.

Strength of Ray Tracing: No API

- Rasterization performance
  - Minimize state changes
  - Avoid small batches
- CPU ray tracing works on any computer - no chip or driver dependencies.
- API
  - Not needed
  - Productivity aid, not as a limiter

Ray Tracing: Massively Parallel

- If each computer renders one pixel…

There is an old joke that goes, "Ray tracing is the technology of the future, and it always will be!"
– David Kirk

Rasterization/Ray Tracing Hybrids

Rinspeed's sQuba car

Rasterization/Ray Tracing Hybrids

Rinspeed's sQuba car

Strength of Ray Tracing: It's Right

- Monte Carlo ray tracing ultimately gives the right answer. It's the "ground truth" algorithm. [Well, ignoring polarization, diffraction, etc.]
- We can (and must) simplify any number of elements – BRDFs, light transport paths - for the sake of FPS. We simplify less each year.
- Long and short, the basic idea of ray tracing will be around a very long time.