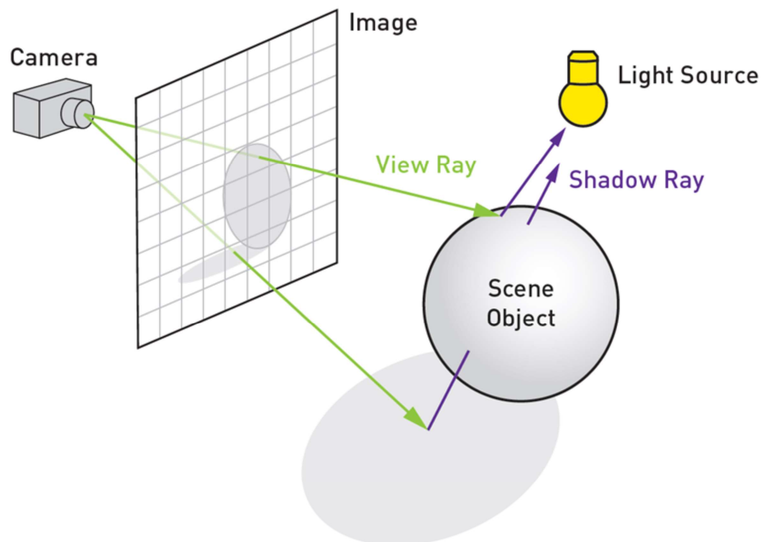
Ray Tracing in October 2022

Eric Haines
NVIDIA

From Jensen’s keynote, https://youtu.be/PWcNlRI00jo

Rays from the eye



Rays for shadows date back to Appel in 1968

Source: Ray Tracing Gems

1980: Classical Ray Tracing

Graphics and Image Processing — J.D. Foley, Editor

An Improved Illumination Model for Shaded Display

Turner Whitted
Bell Laboratories
Holmdel, New Jersey

To accurately render a two-dimensional image of a three-dimensional scene, global illumination information that affects the intensity of each pixel of the image must be known at the time the intensity is calculated. In a simplified form, this information is stored in a tree of "rays" extending from the viewer to the first surface encountered and from there to other surfaces and to the light sources. A visible surface algorithm creates this tree for each pixel of the display and passes it to the shader. The shader then traverses the tree to determine the intensity of the light received by the viewer. Consideration of all of these factors allows the shader to accurately simulate true reflection, shadows, and refraction, as well as the effects simulated by conventional shaders. Anti-aliasing is included as an integral part of the visibility calculations. Surfaces displayed include curved as well as polygonal surfaces.

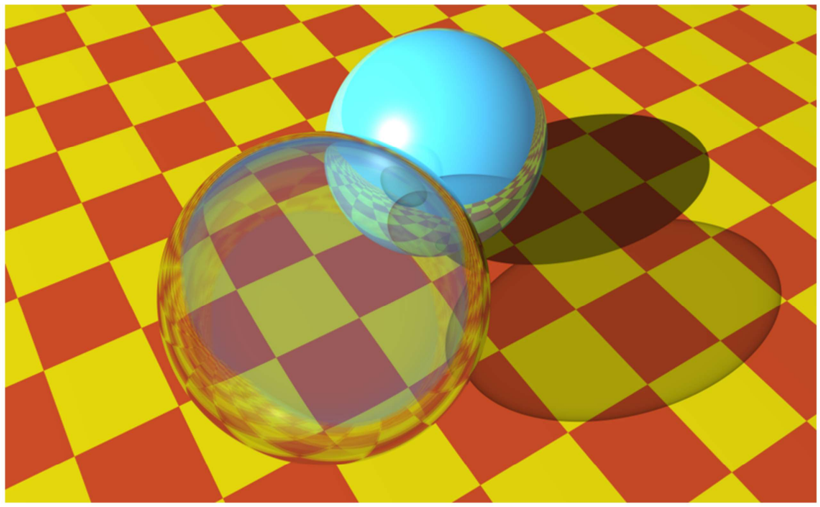


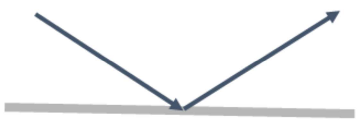
Image now generated in real time in NVIDIA OptiX™ (was 74 minutes per frame in 1980)

He even talks about previous ray tracing algorithms, such as MAGI and Arthur Appel 1968. Douglas Kay in 1979, "TRANSPARENCY FOR COMPUTER SYNTHESIZED IMAGES", almost did it.

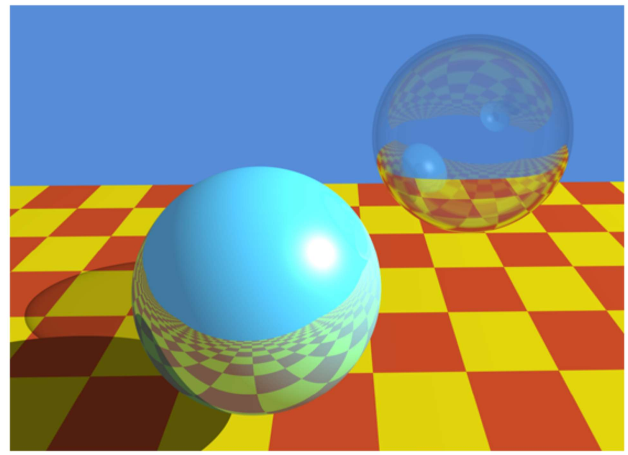
1980: Classical Ray Tracing

For each pixel

- Send ray from eye into scene
- Send a ray from the intersection to each light: shadows
- Spawn a new color ray for each reflection & refraction

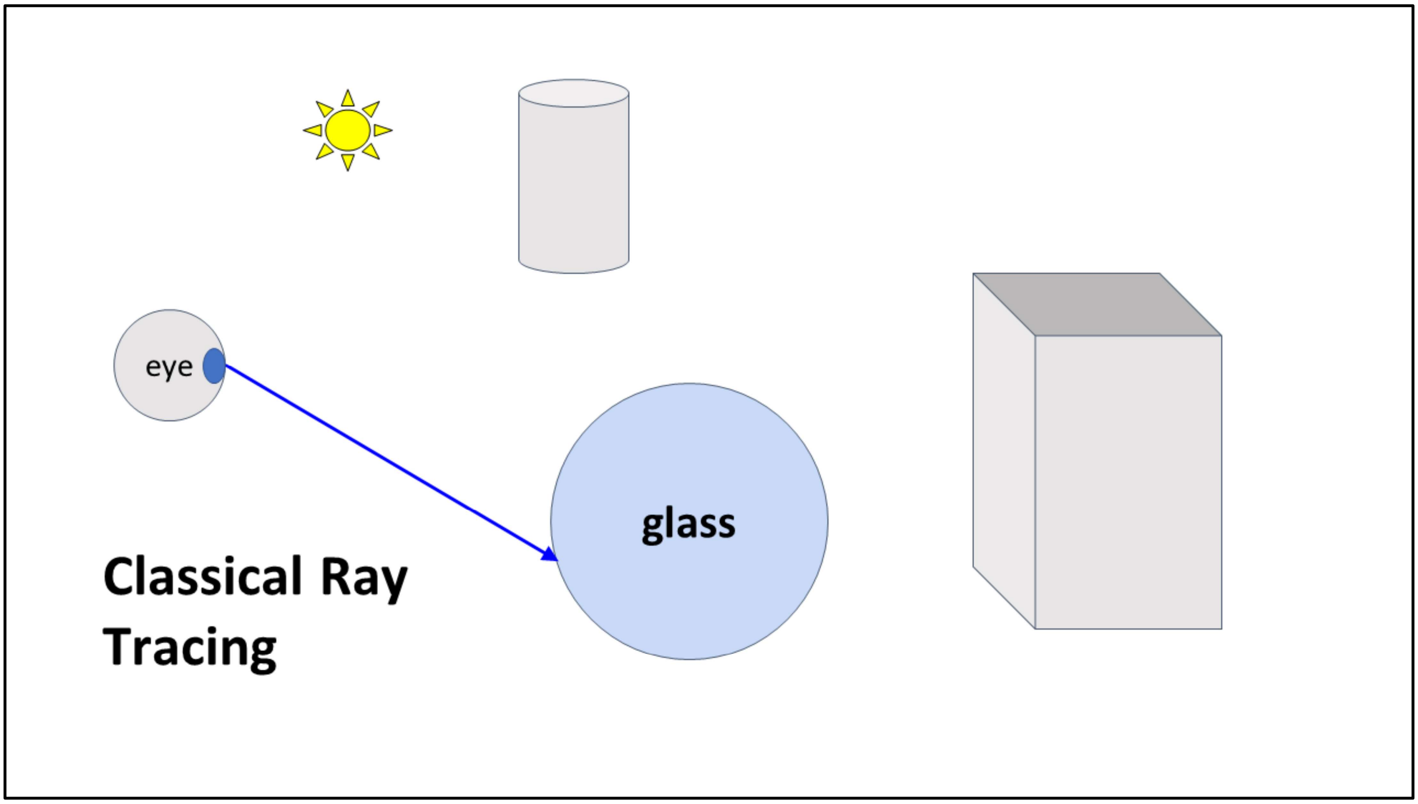


highly polished surface

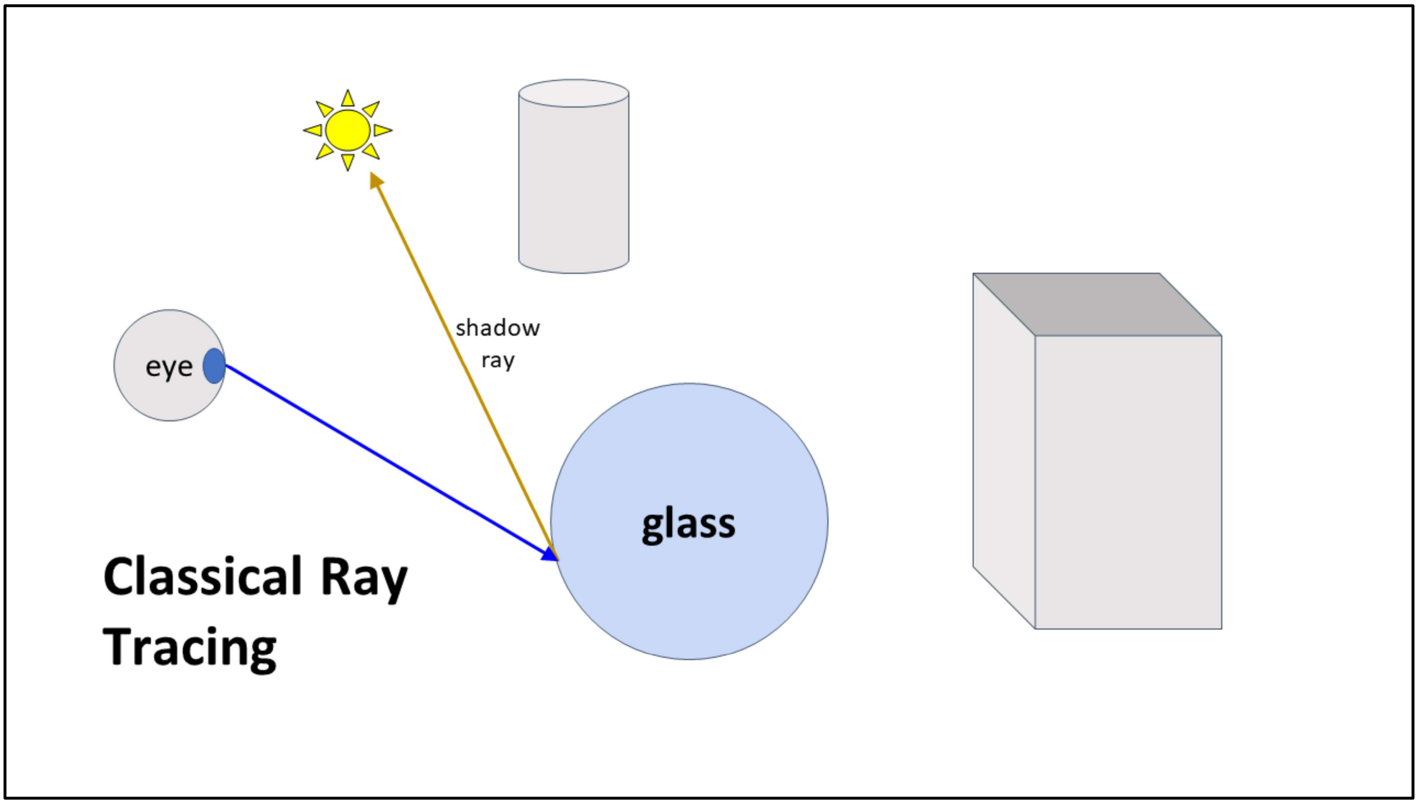


Generated using OptiX sample “optixWhitted”

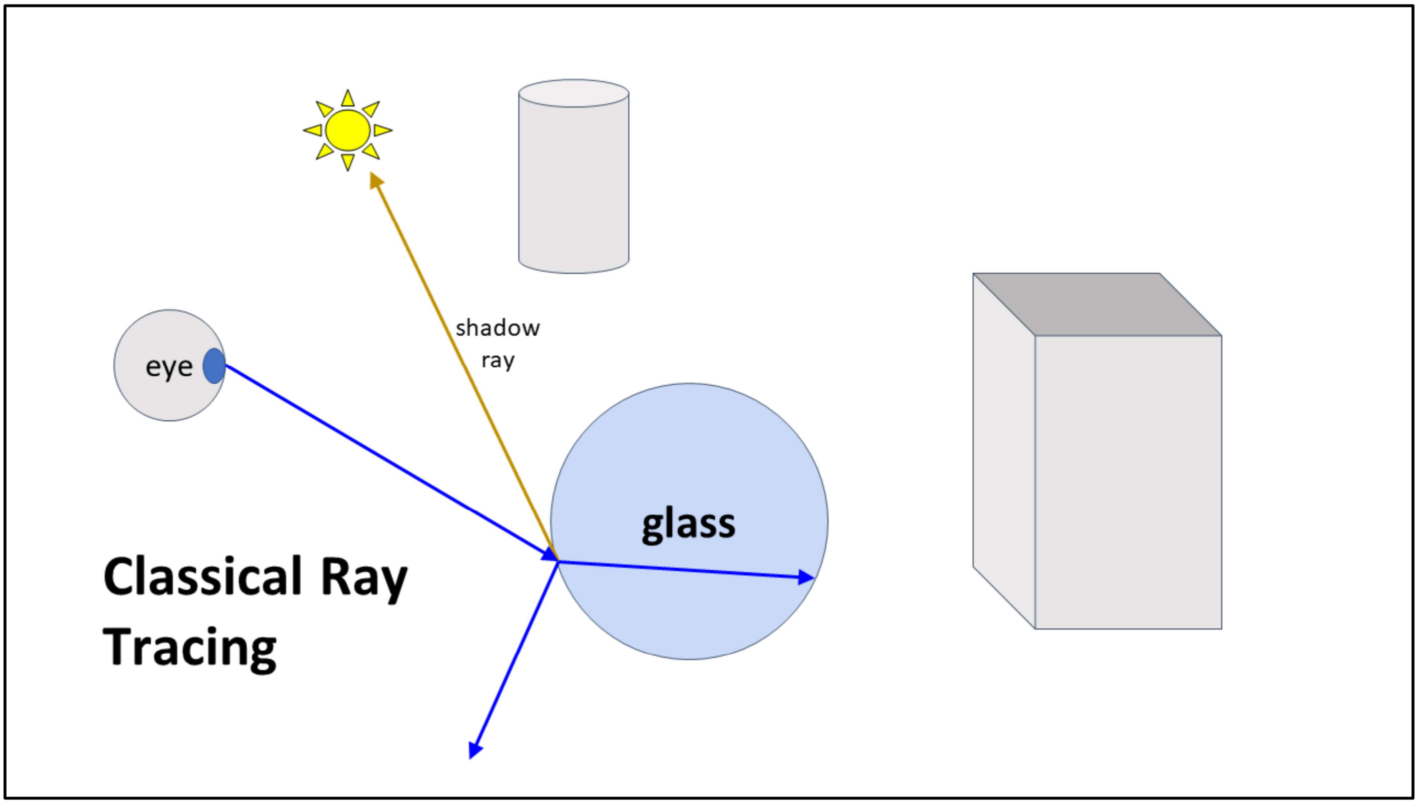
74 minutes on a VAX-11/780, 640 x 480



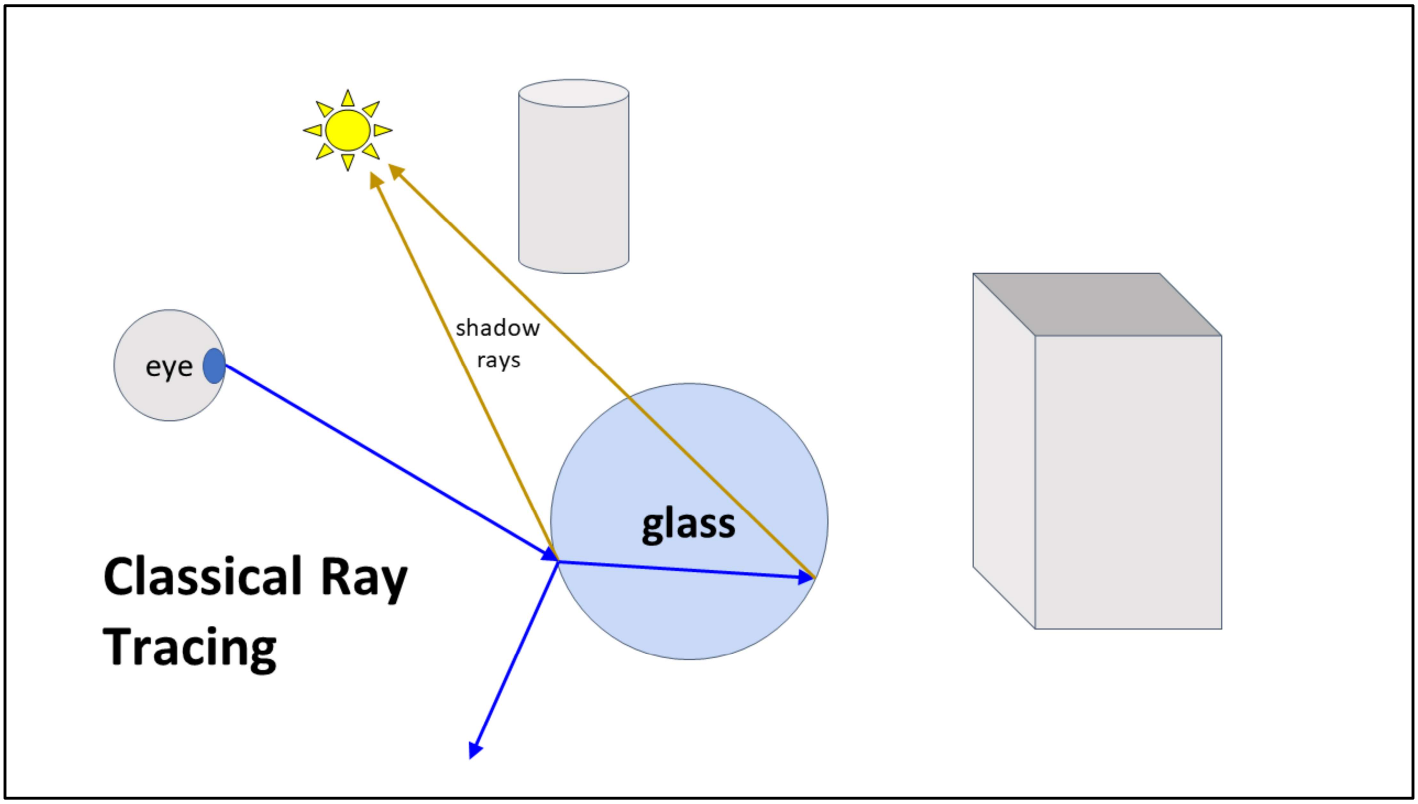
My own, started from Pete’s “1-Overview” intro to RT course slide



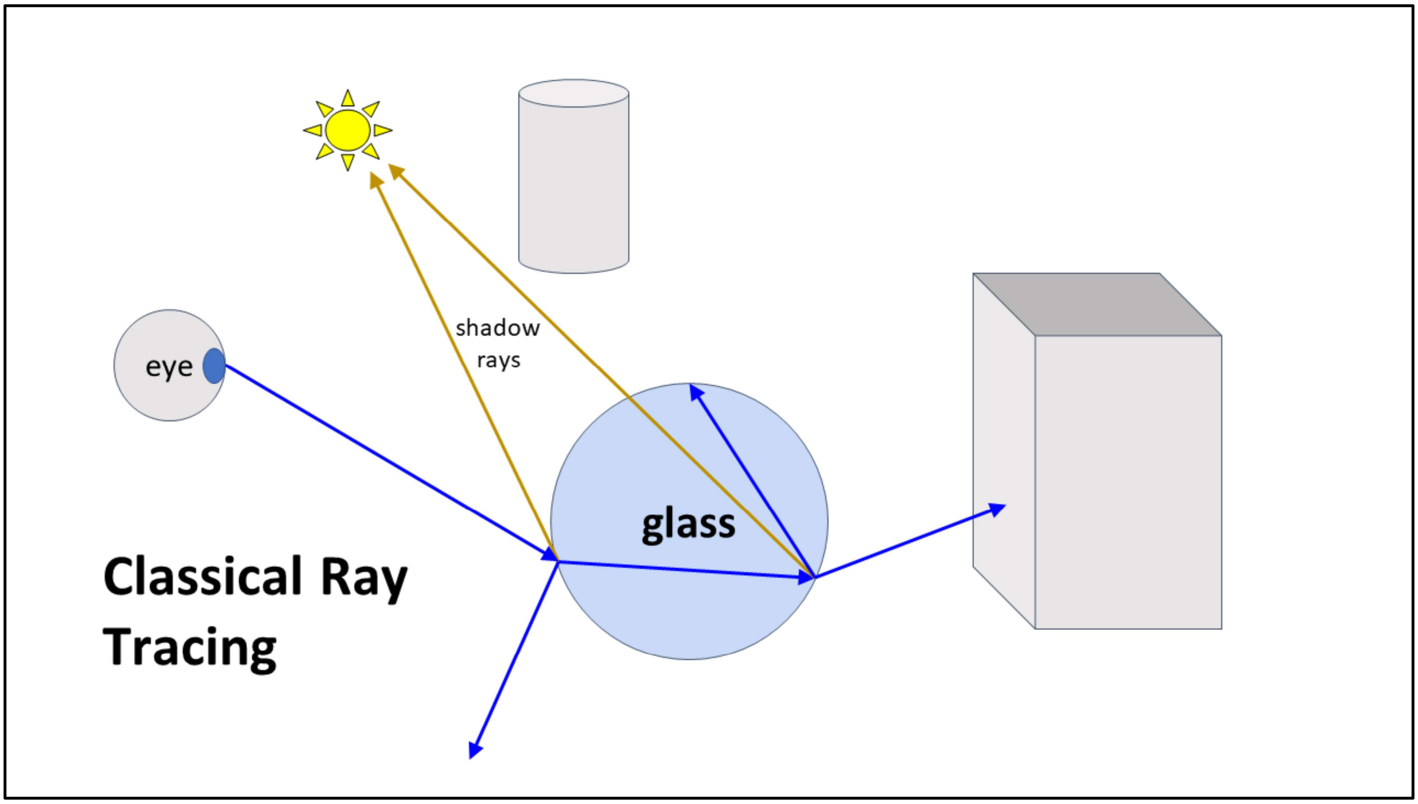
My own, started from Pete's "1-Overview" intro to RT course slide



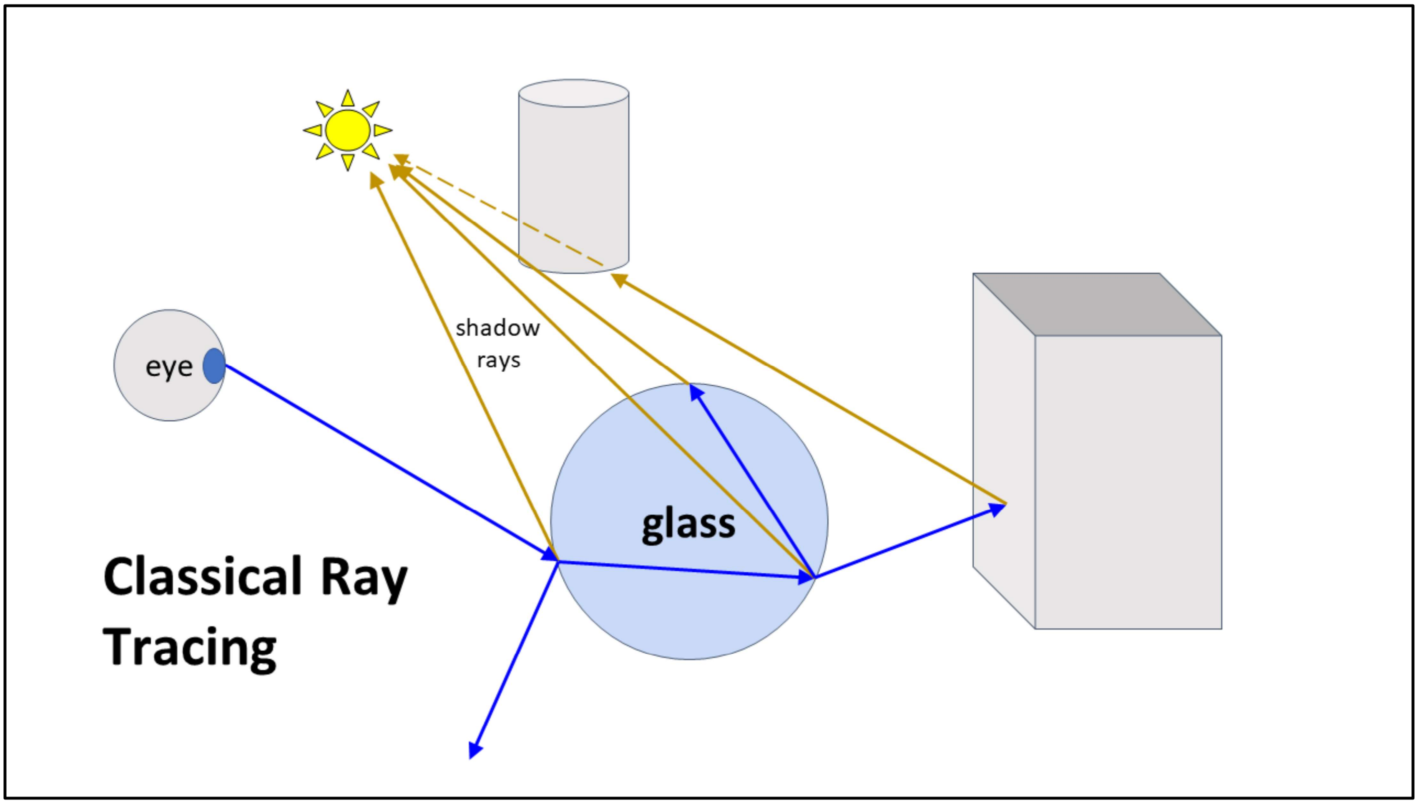
My own, started from Pete’s “1-Overview” intro to RT course slide



My own, started from Pete's "1-Overview" intro to RT course slide



My own, started from Pete's "1-Overview" intro to RT course slide



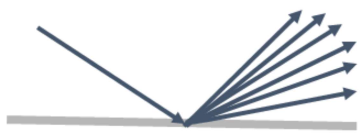
My own, started from Pete's "1-Overview" intro to RT course slide

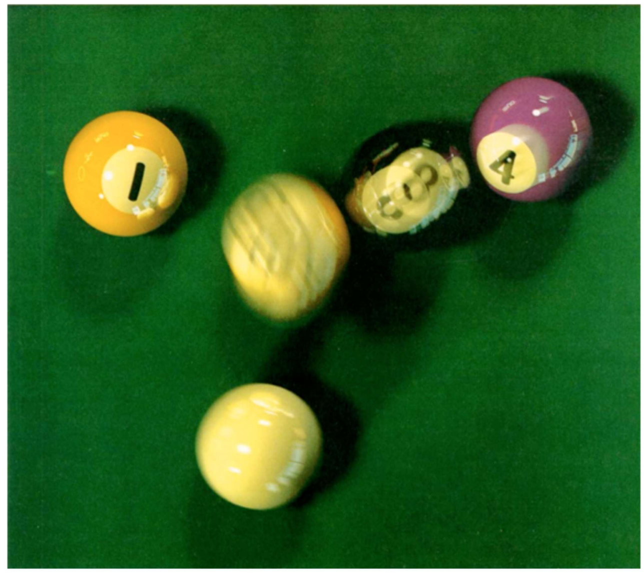
1984: Cook Stochastic (“Distribution”) Ray Tracing

Allow shadow rays to go to a random point on area light.

Allow specular rays to be perturbed specularly around the ideal reflection.

Shoot sometime during the frame for motion blur.



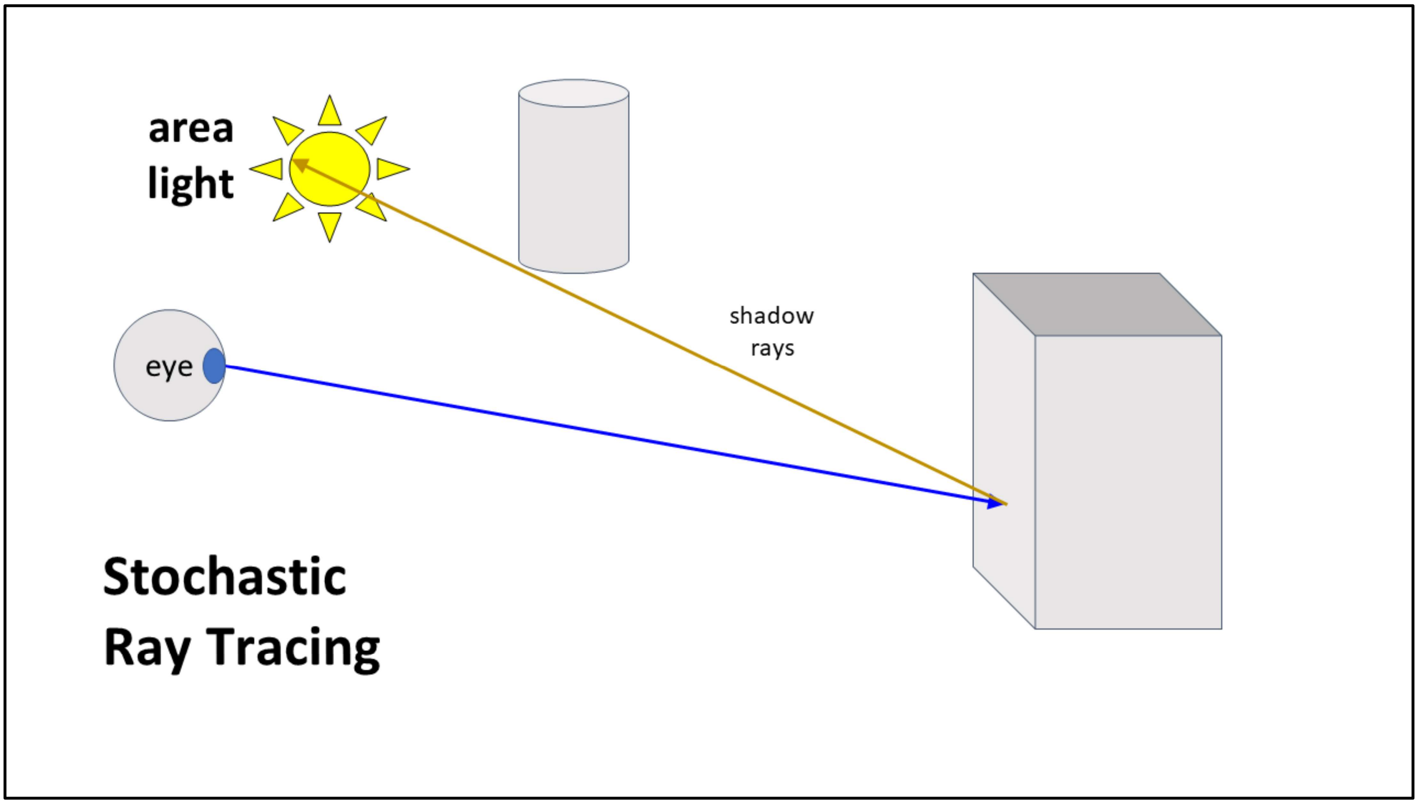


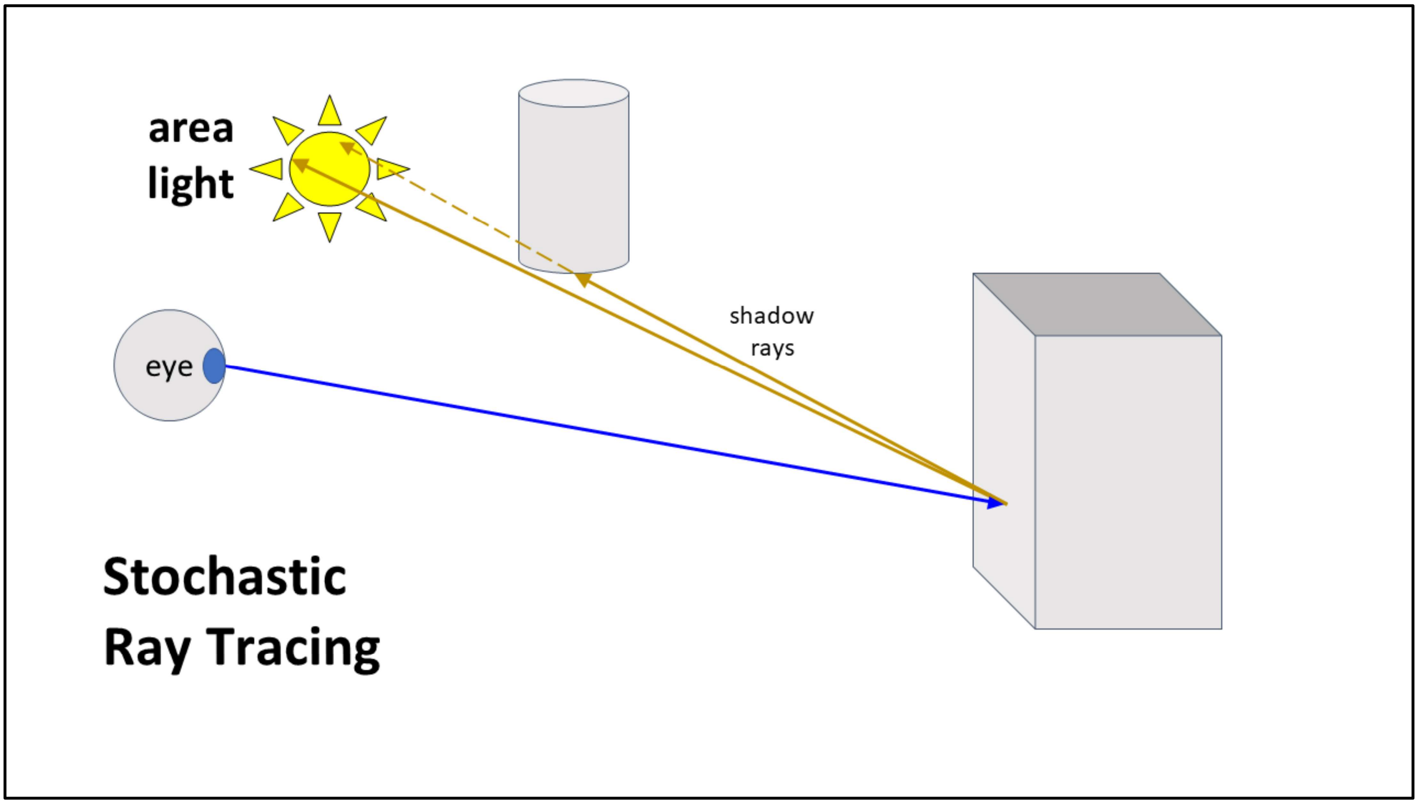
By Robert L. Cook, Tom Porter, and Loren Carpenter, Pixar

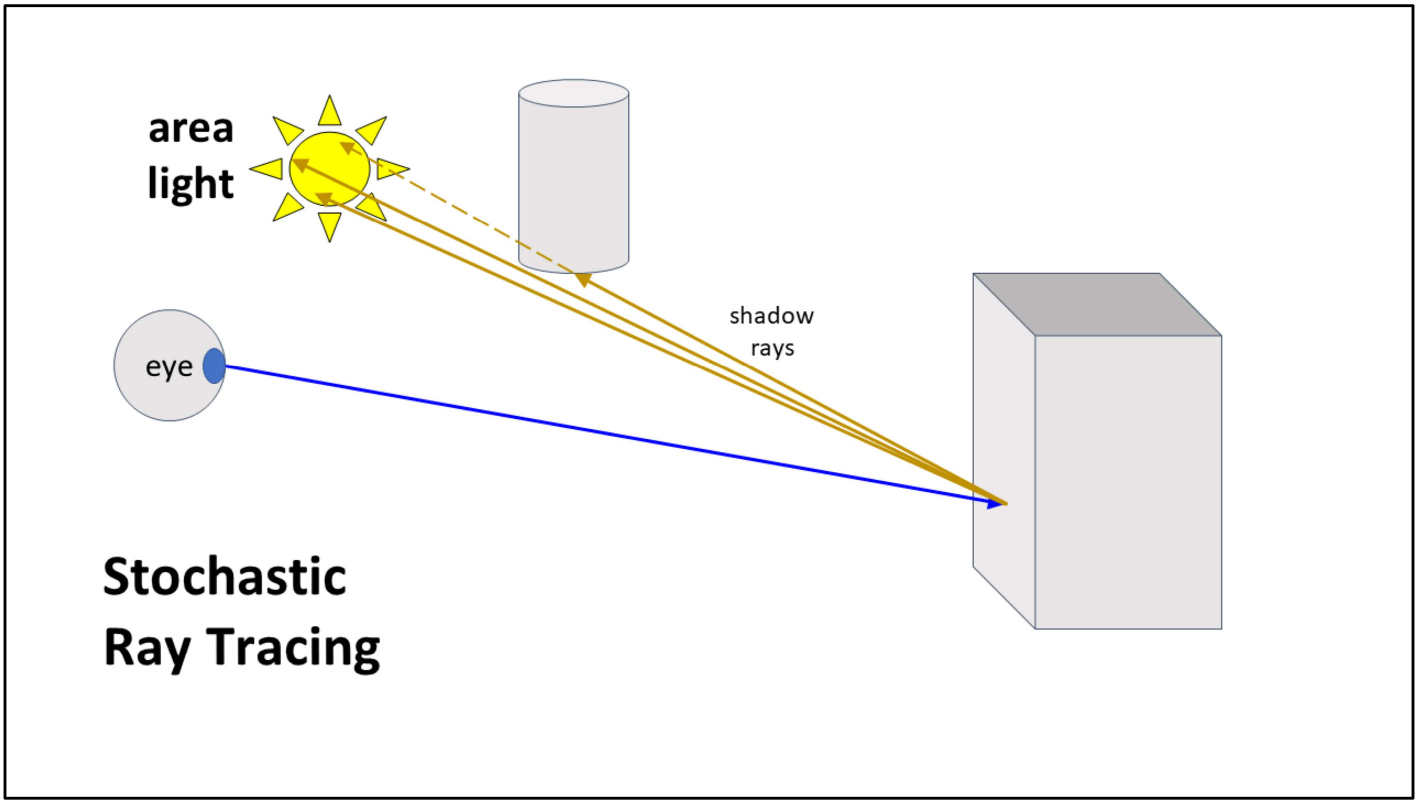
https://graphics.pixar.com/library/indexAuthorRobert_L_Cook.html

https://graphics.pixar.com/library/DistributedRayTracing/

Source: ACM, though better to credit Pixar (rights assignment has changed over the decades), SIGGRAPH 2019 OptiX course.pptx uses this image.


area
light
eye
shadow
rays
Stochastic
Ray Tracing


area
light
shadow
rays
eye
Stochastic
Ray Tracing

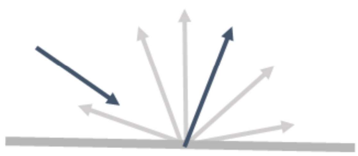

area
light
shadow
rays
eye
Stochastic
Ray Tracing

1986: Kajiya-Style Diffuse Interreflection

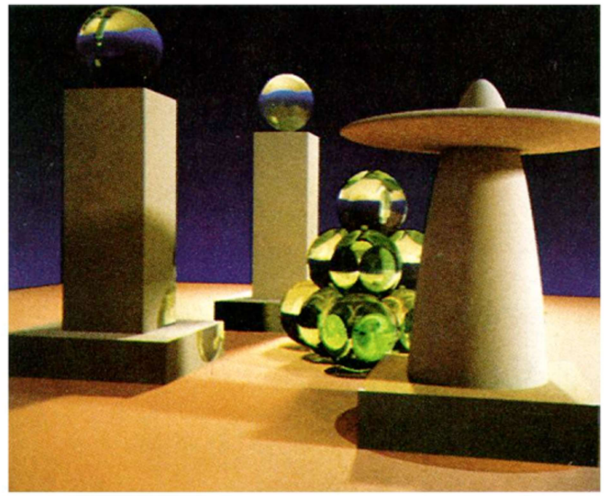
Path tracing: shoot each ray and follow it along a series of interreflections.

“The Rendering Equation”

Guaranteed to give the right answer *at the limit*.



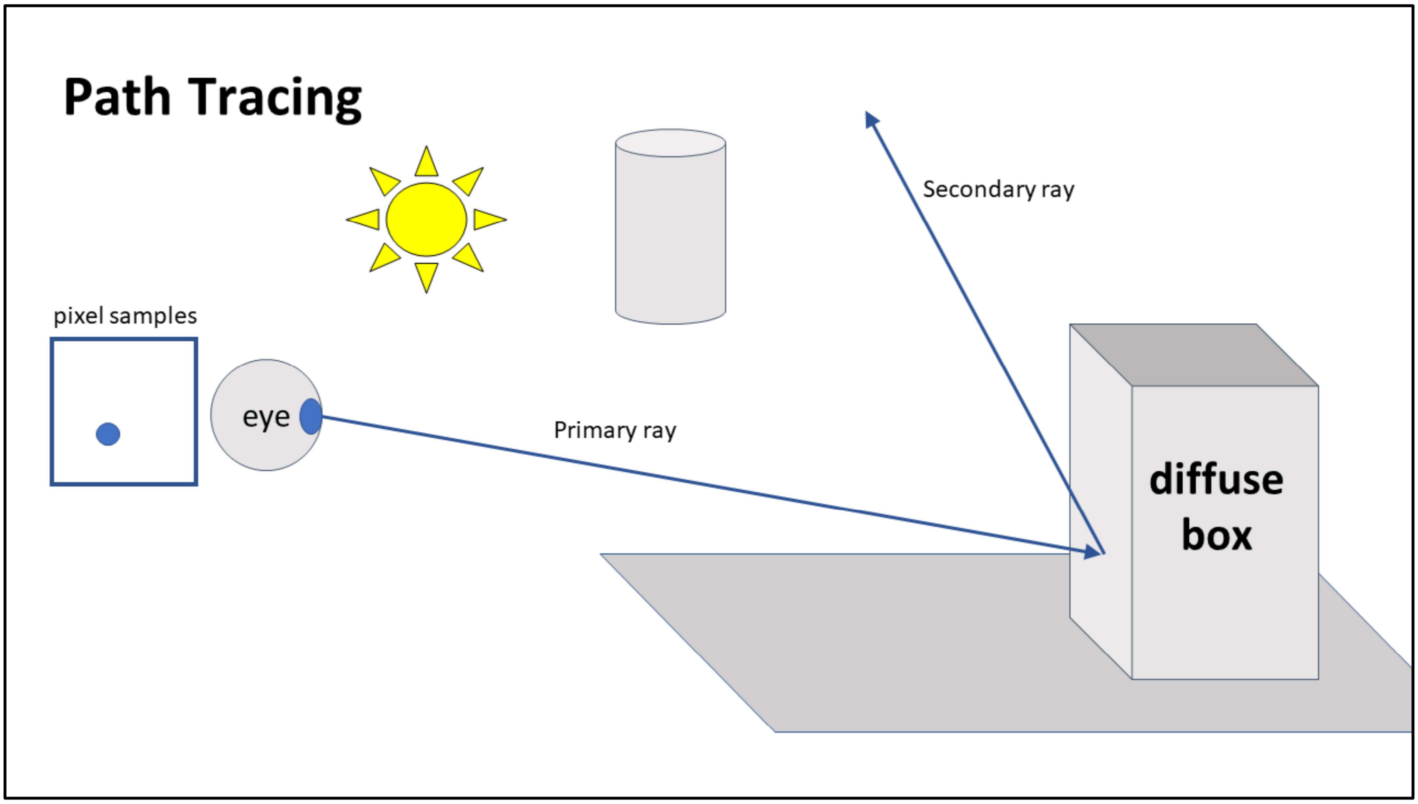
diffuse surface reflection

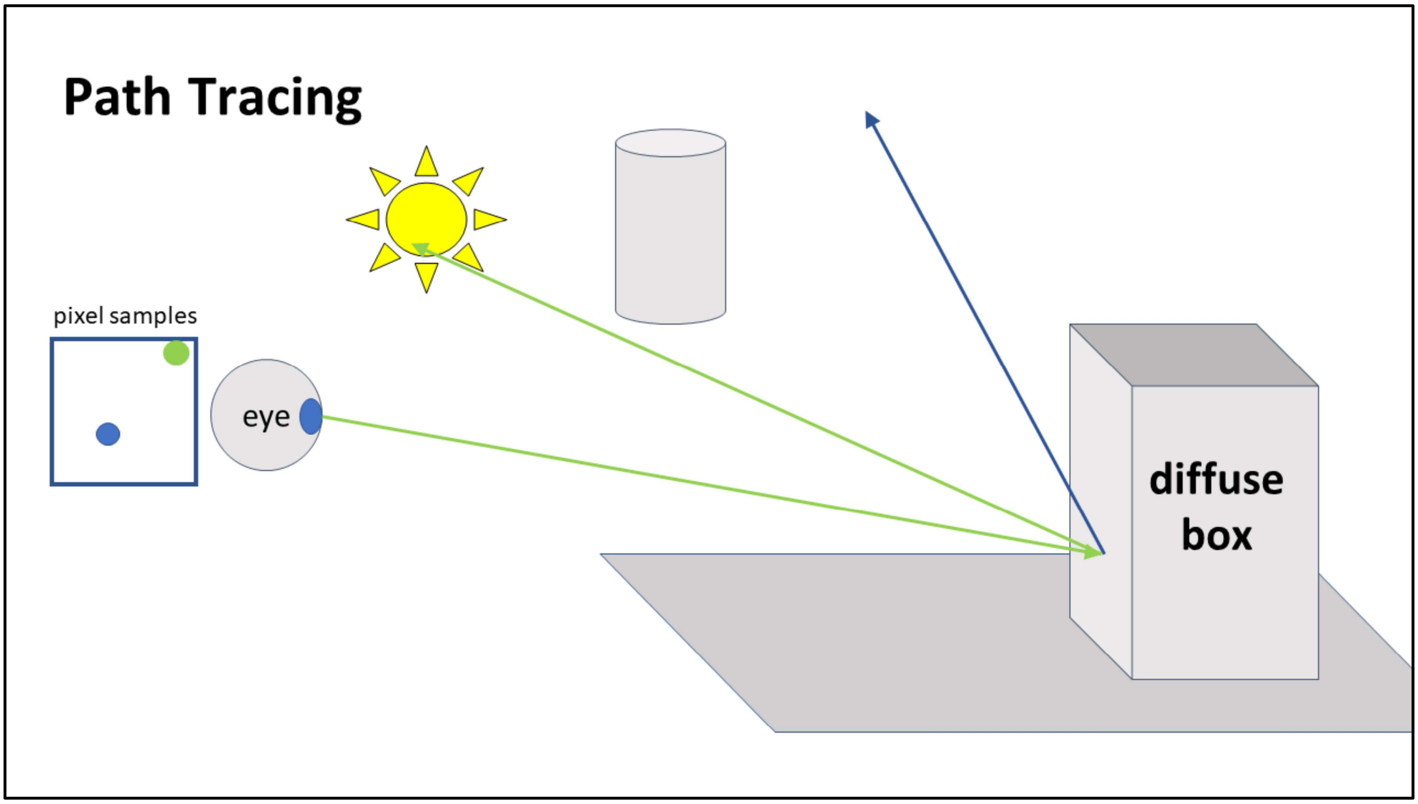


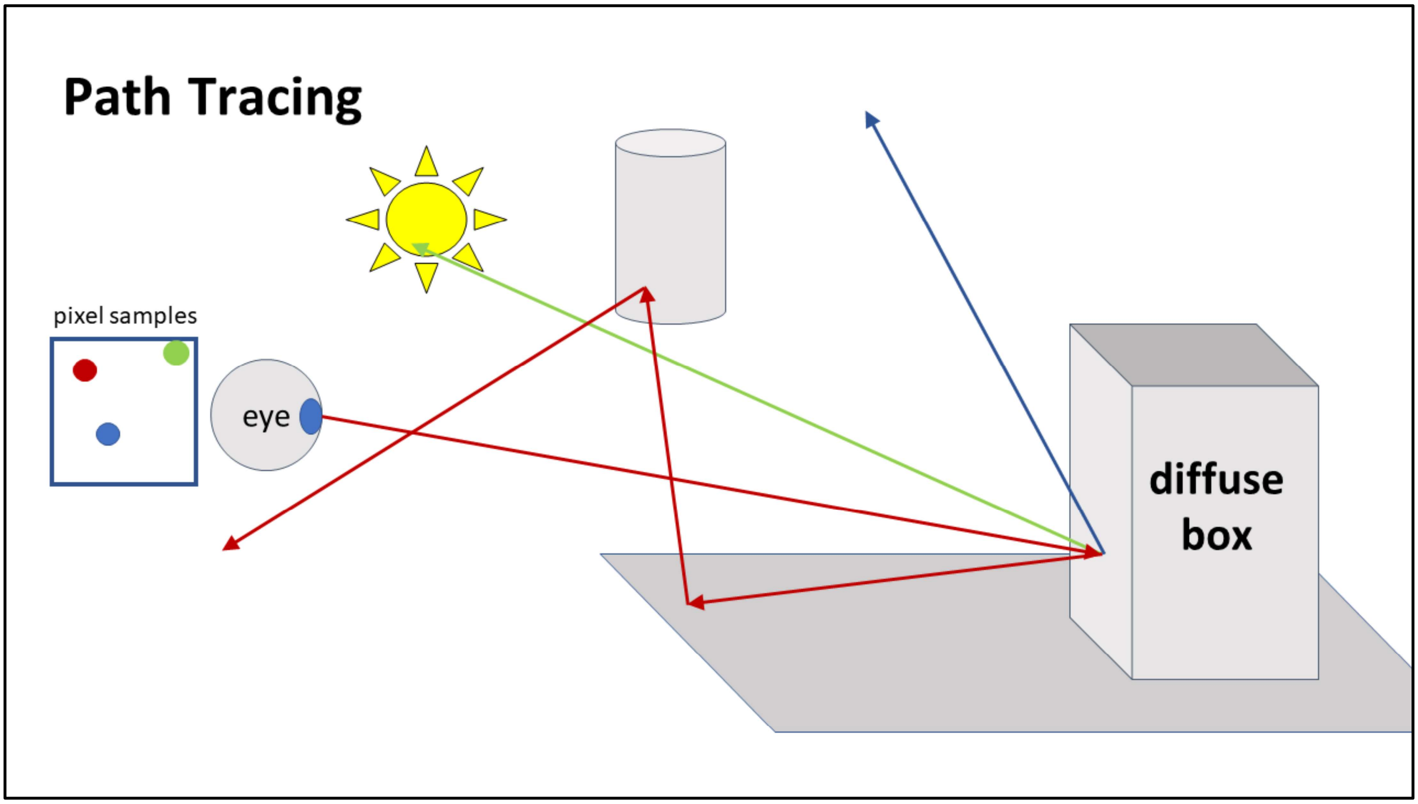
By James Kajiya, California Institute of Technology

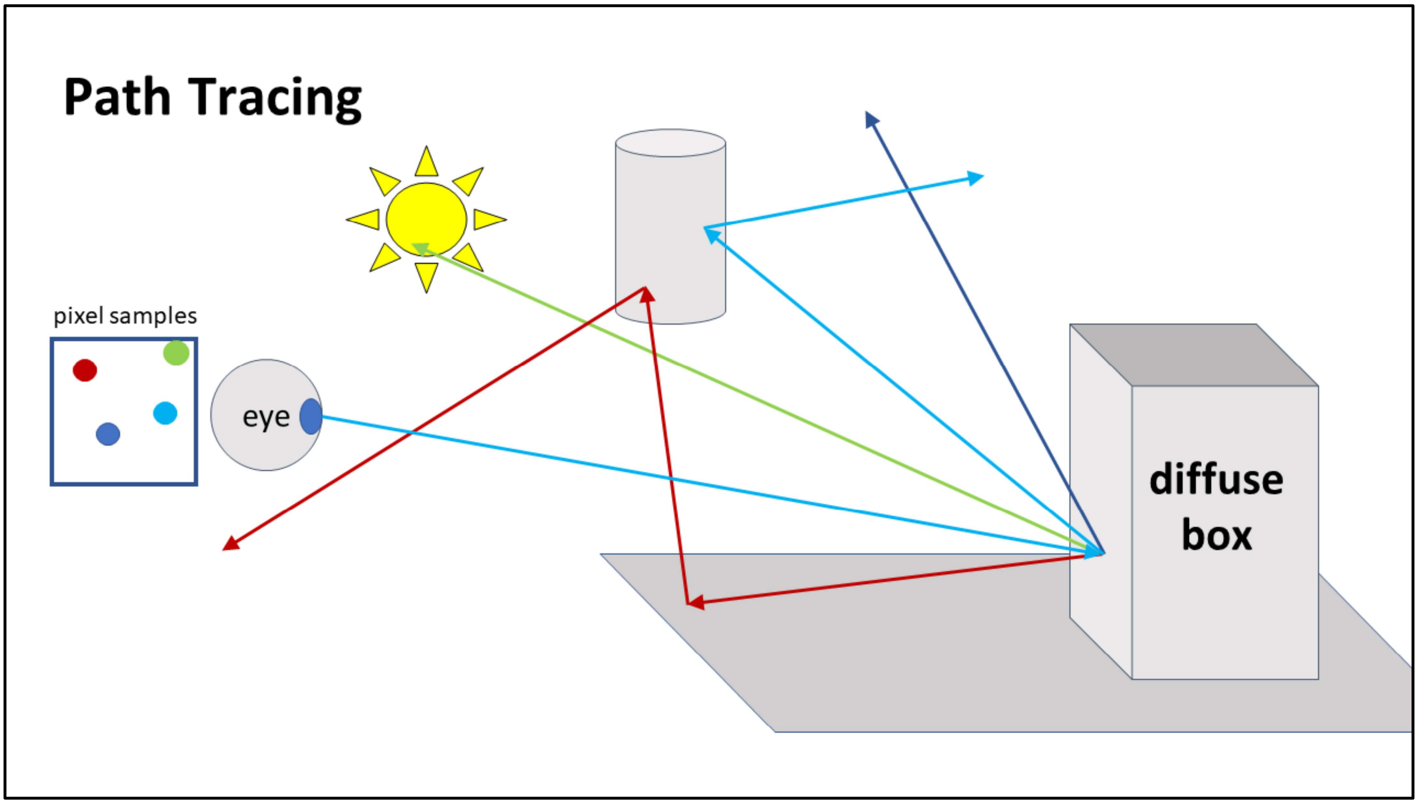
Note recursion: ray continues along a path until a light is hit (or something entirely black or considered “unchangeable,” such as an environment map.

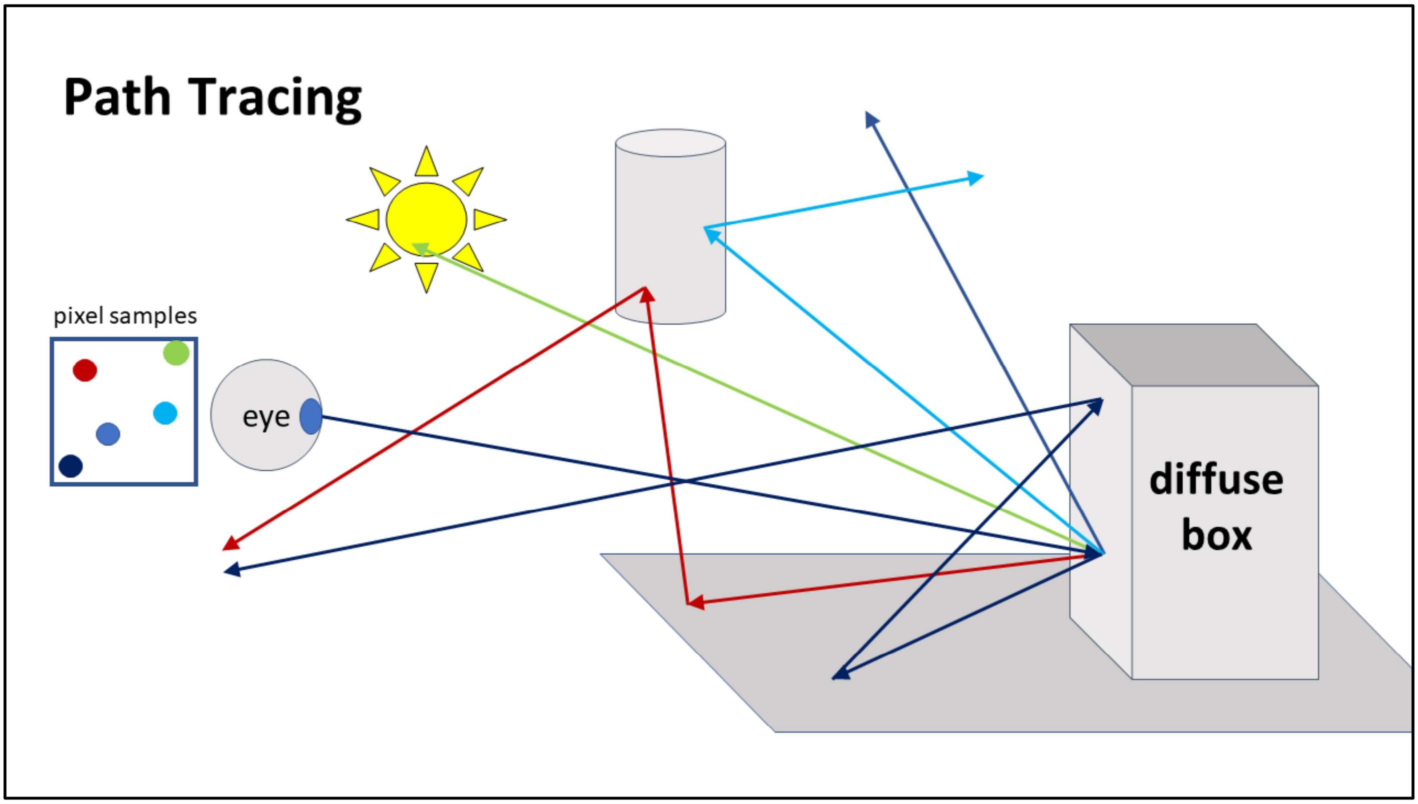
Source: ACM, or Caltech.

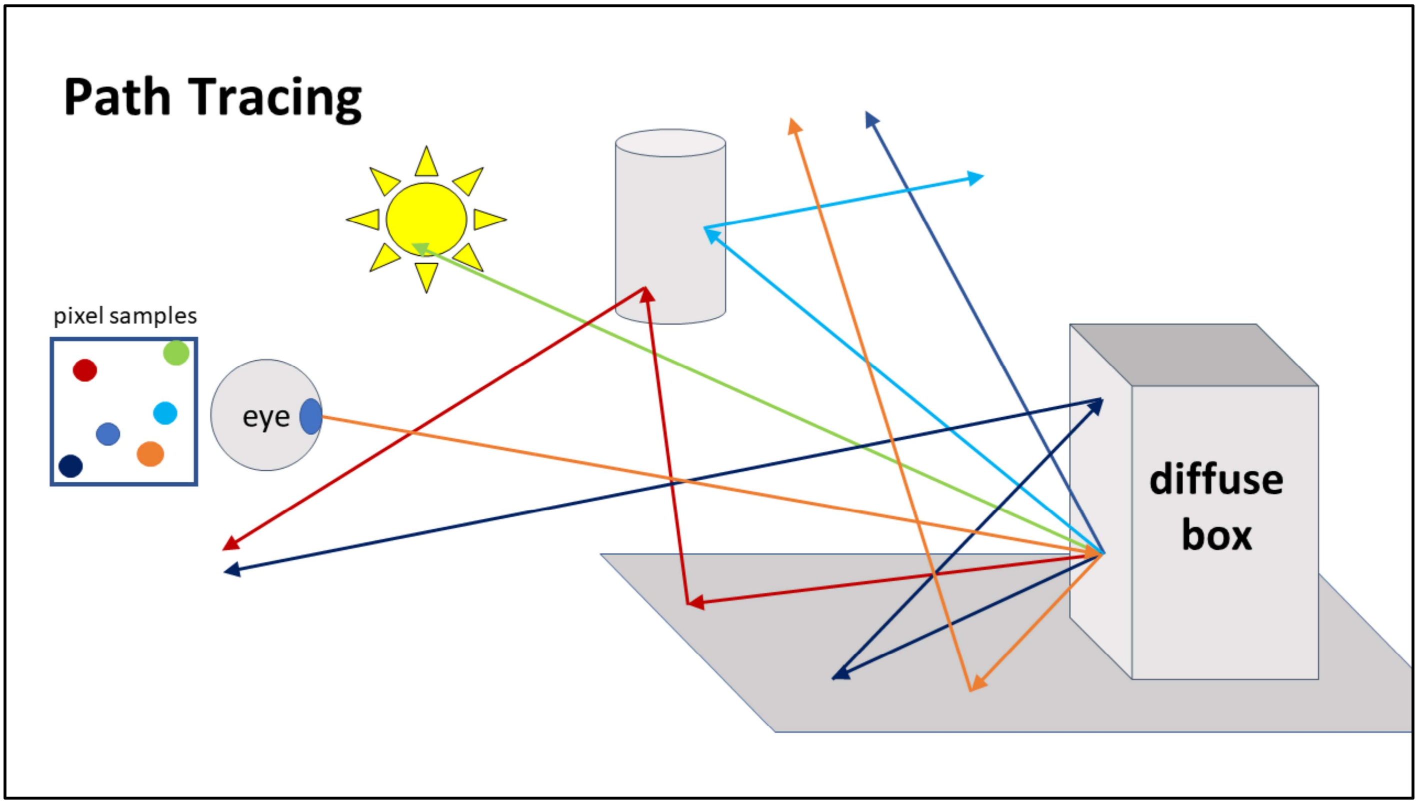

Path Tracing
pixel samples
eye
Primary ray
Secondary ray
diffuse
box

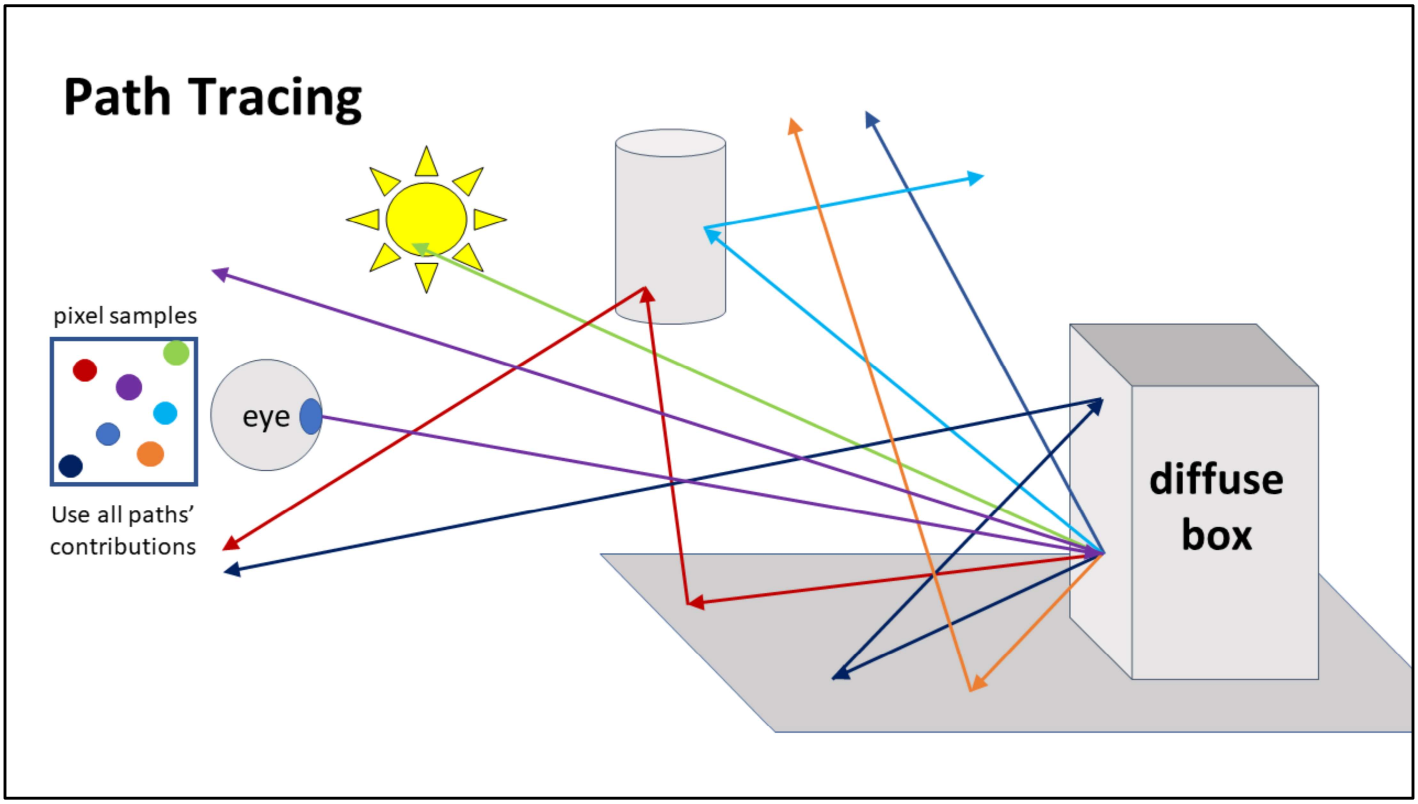

Path Tracing
pixel samples
eye
diffuse
box


Path Tracing
pixel samples
eye
diffuse box


Path Tracing
pixel samples
eye
diffuse
box

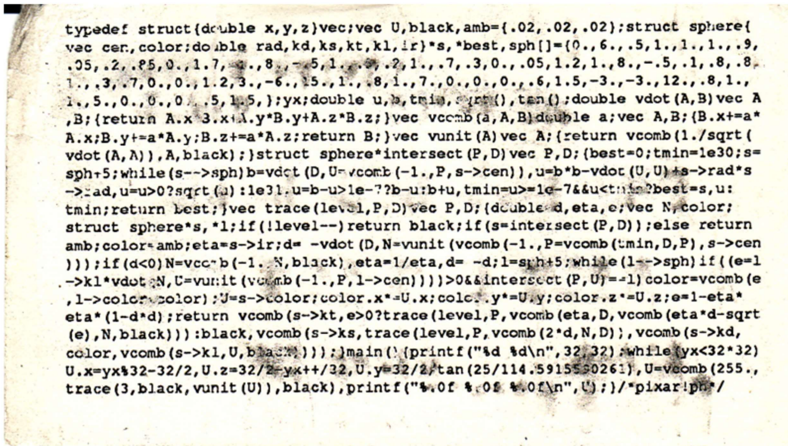

Path Tracing
pixel samples
eye
diffuse
box


Path Tracing
pixel samples
eye
diffuse
box



My own, started from

Why Ray Tracing is Great

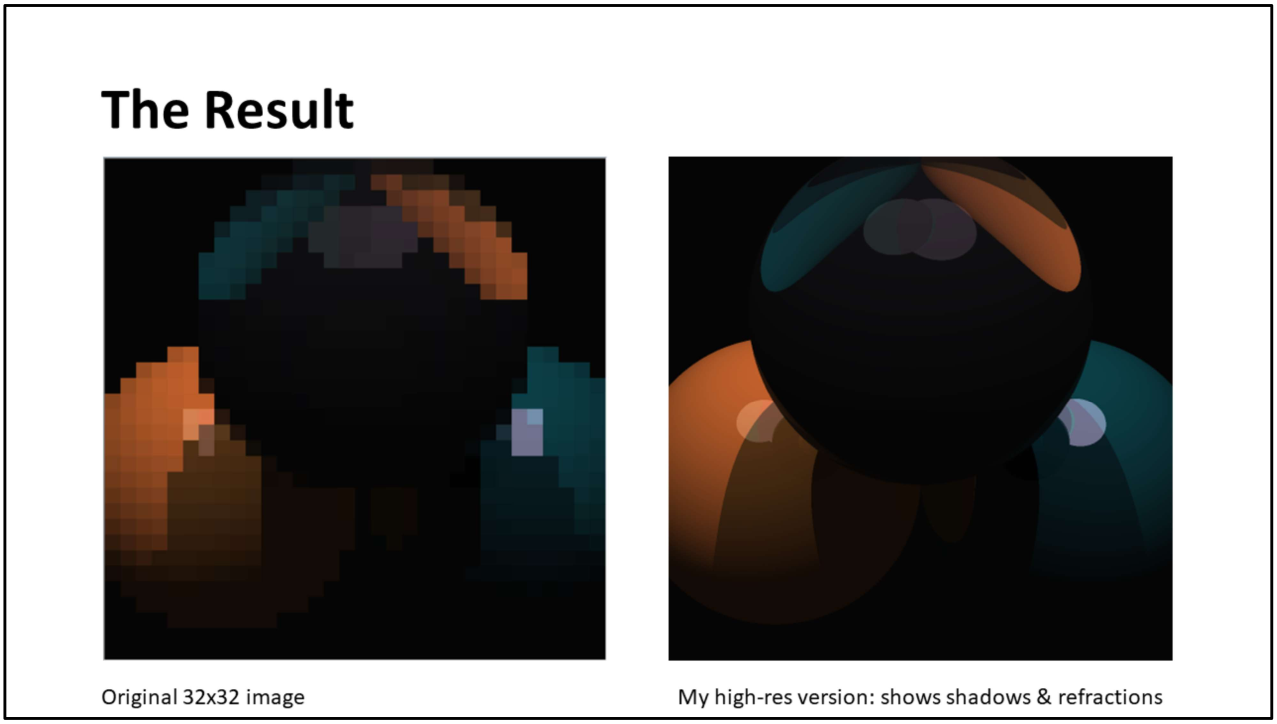


```
typedef struct{double x,y,z}vec;vec U,black,amb={.02,.02,.02};struct sphere{
vec cen,color;double rad,kd,ks,kt,kl,ir}*s,*best,sph[]={0.,6.,.5,1.,1.,1.,.9,
.05,.2,.85,0.,1.7,-1.,8.,-.5,1.,.5,.2,1.,.7,.3,0.,.05,1.2,1.,8.,-.5,.1,.8,.8,
1.,.3,.7,0.,0.,1.2,3.,-6.,15.,1.,.8,1.,7.,0.,0.,0.,.6,1.5,-3.,-3.,12.,.8,1.,
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*/
```

The back of Paul Heckbert's business card, 1607 bytes. Includes tricks from Darwyn Peachey and Joe Cychosz

From 1994, Ray Tracing Gems IV, but the seminal paper being *[Ray Tracing Jell-O Brand Gelatin](#)*

http://www.cs.cmu.edu/~ph/ for code, etc.



From 1994, Ray Tracing Gems IV

http://www.cs.cmu.edu/~ph/ for code, etc.

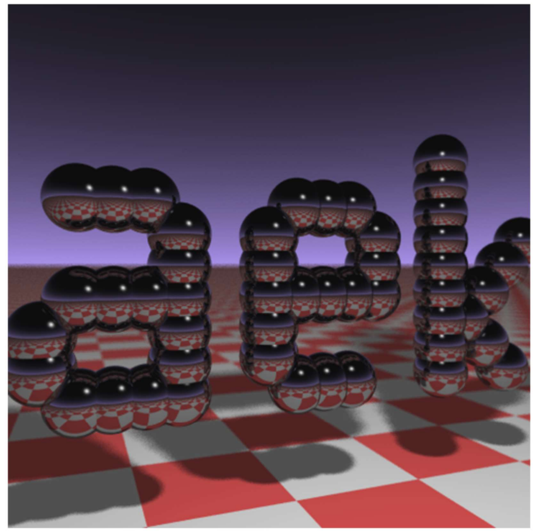
Highlights, shadows, and refraction. 1024x1024 took 10 seconds to run on my CPU.

Another Business Card

```
#include <stdlib.h>   // card > aek.ppm
#include <stdio.h>
#include <math.h>
typedef int i;typedef float f;struct v{
f x,y,z;v operator+(v r){return v(x+r.x
,y+r.y,z+r.z);}v operator*(f r){return
v(x*r,y*r,z*r);}f operator%(v r){return
x*r.x+y*r.y+z*r.z;}v(){}v operator^(v r
){return v(y*r.z-z*r.y,z*r.x-x*r.z,x*r.
y-y*r.x);}v(f a,f b,f c){x=a;y=b;z=c;}v
operator!(){return*this*(1/sqrt(*this%*
this));}};i G[]={247570,280596,280600,
249748,18578,18577,231184,16,16};f R(){
return(f)rand()/RAND_MAX;}i T(v o,v d,f
&t,v&n){t=1e9;i m=0;f p=-o.z/d.z;if(.01
<p)t=p,n=v(0,0,1),m=1;for(i k=19;k--;)
for(i j=9;j--;)if(G[j]&1<<k){v p=o+v(-k
,0,-j-4);f b=p%d,c=p%p-1,q=b*b-c;if(q>0
){f s=-b-sqrt(q);if(s<t&&s>.01)t=s,n=!(
p+d*t),m=2;}}return m;}v S(v o,v d){f t
;v n;i m=T(o,d,t,n);if(!m)return v(.7,
.6,1)*pow(1-d.z,4);v h=o+d*t,l=!(v(9+R(
),9+R(),16)+h*-1),r=d+n*(n%d*-2);f b=l%
n;if(b<0||T(h,l,t,n))b=0;f p=pow(l%r*(b
>0),99);if(m&1){h=h*.2;return((i)(ceil(
h.x)+ceil(h.y))&1?v(3,1,1):v(3,3,3))*(b
*.2+.1);}return v(p,p,p)+S(h,r)*.5;}i
main(){printf("P6 512 512 255 ");v g=!v
(-6,-16,0),a=!(v(0,0,1)^g)*.002,b=!(g^a
)*.002,c=(a+b)*-256+g;for(i y=512;y--;)
for(i x=512;x--;){v p(13,13,13);for(i r
=64;r--;){v t=a*(R()-.5)*99+b*(R()-.5)*
99;p=S(v(17,16,8)+t,!(t*-1+(a*(R()+x)+b
*(y+R())+c)*16))*3.5+p;}printf("%c%c%c"
,(i)p.x,(i)p.y,(i)p.z);}}
```

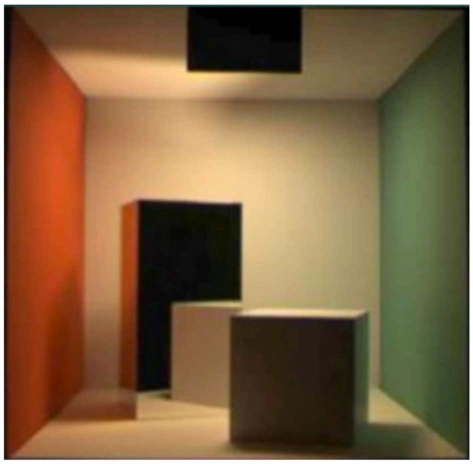
Andrew Kensler's 1337 byte program.

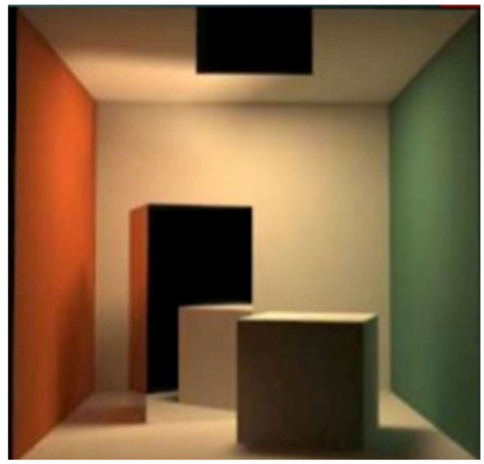
For example, spheres are stored here:
G[]={247570,280596,280600,249748,18578,18577,231184,16,16};



http://eastfarthing.com/blog/2016-01-12-card/
https://fabiensanglard.net/rayTracing_back_of_business_card/
and https://gist.github.com/sungiant/9524044

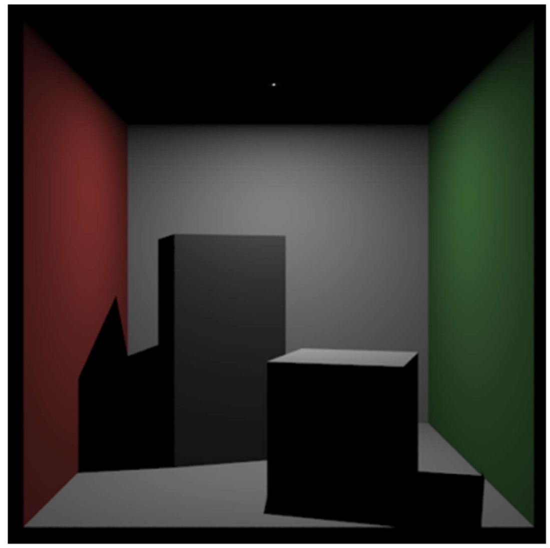
Which One Is Real?





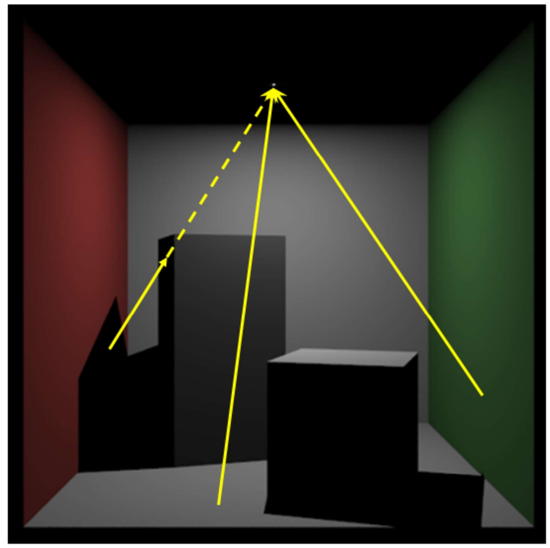
Left image is a photograph, right image is rendered by path tracing. This famous ground-truth test of a renderer is the origin of the "Cornell box" 3D models—there's a real box at Cornell.

Hard Shadows



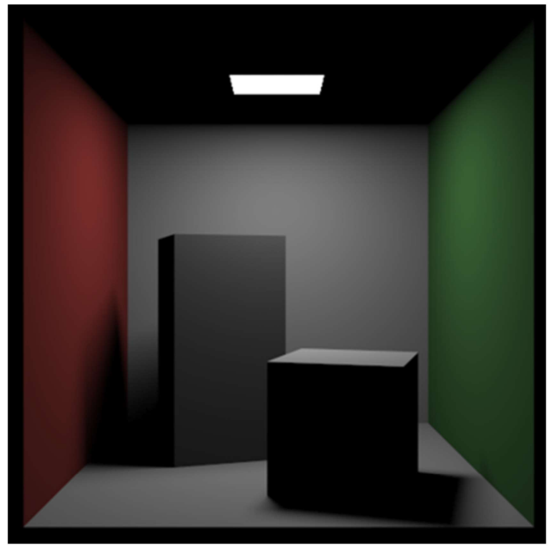
So, how are these shadows generated?

Hard Shadows



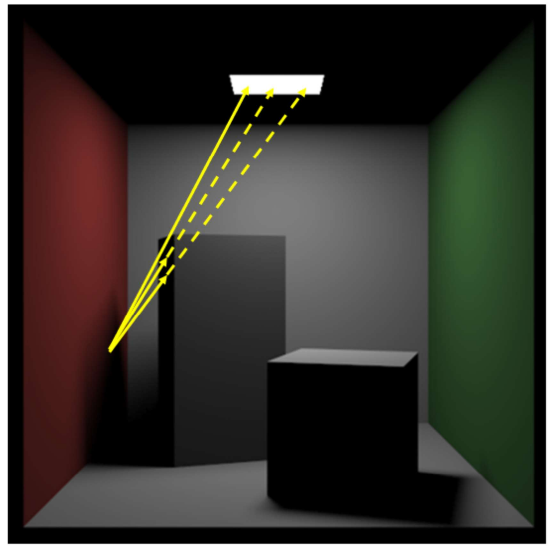
If you can't see the light, you're in shadow. Another way to think of it is, if you look from the light's location, whatever you see is lit, everything else is in shadow.

Soft Shadows



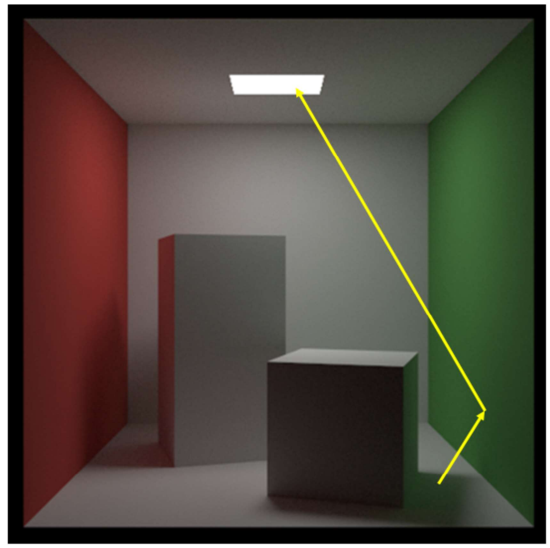
What about these shadows?

Soft Shadows



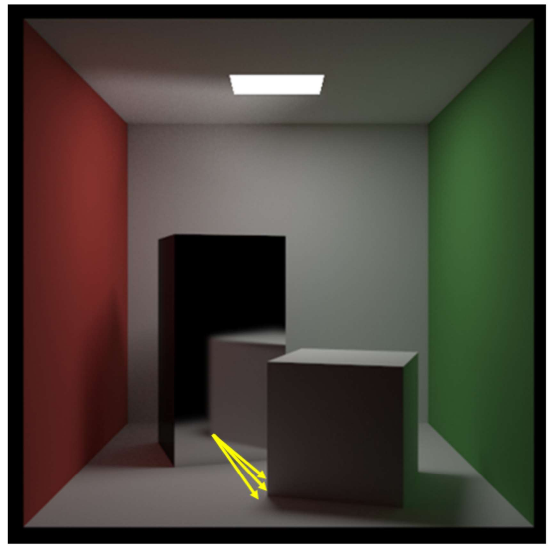
Interreflection
a.k.a.
indirect lighting
a.k.a.
color bleeding
a.k.a.
global
illumination

Interreflection
a.k.a.
indirect lighting
a.k.a.
color bleeding
a.k.a.
global
illumination



Glossy Reflections

Glossy Reflections



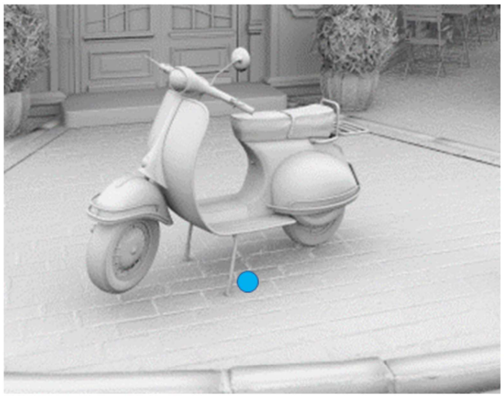
Glossiness can vary, even using textures to control glossiness on different parts of the surface.

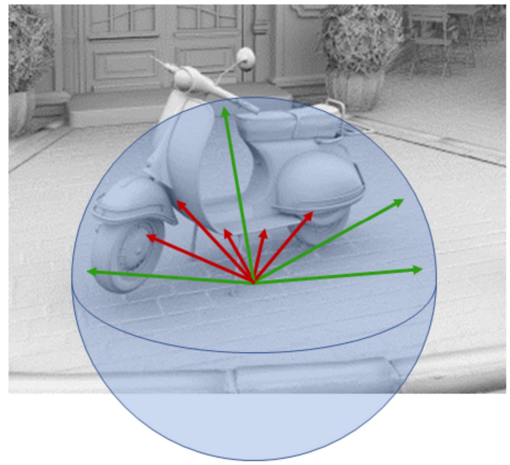
Here’s your quiz question: which of these effects can be seen in this image?


Interreflection Throughout
Glossy Reflections
Soft Shadows

From Chris Wyman, of a scene free to reuse (Bistro outdoors, from ORCA collection).

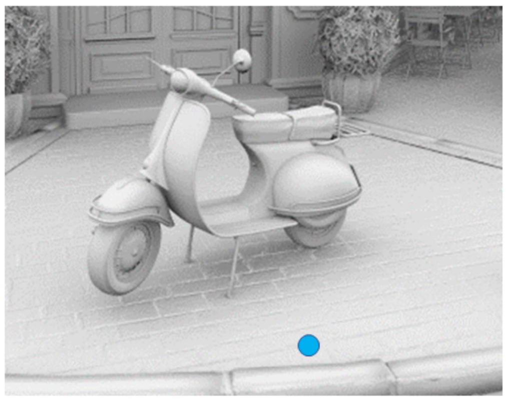
Ambient Occlusion

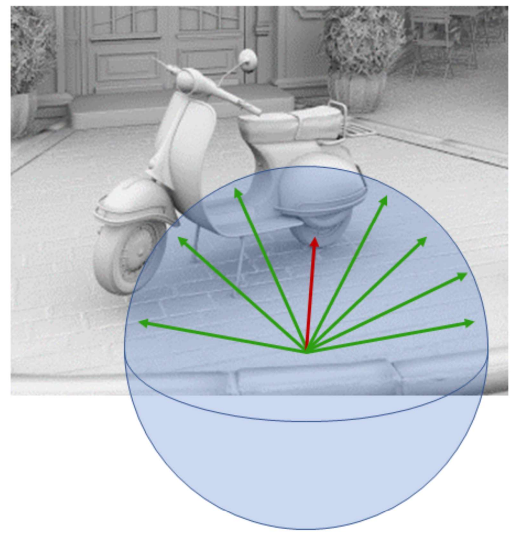




From Chris Wyman, of a scene free to reuse (Bistro outdoors, from ORCA collection).

Ambient Occlusion



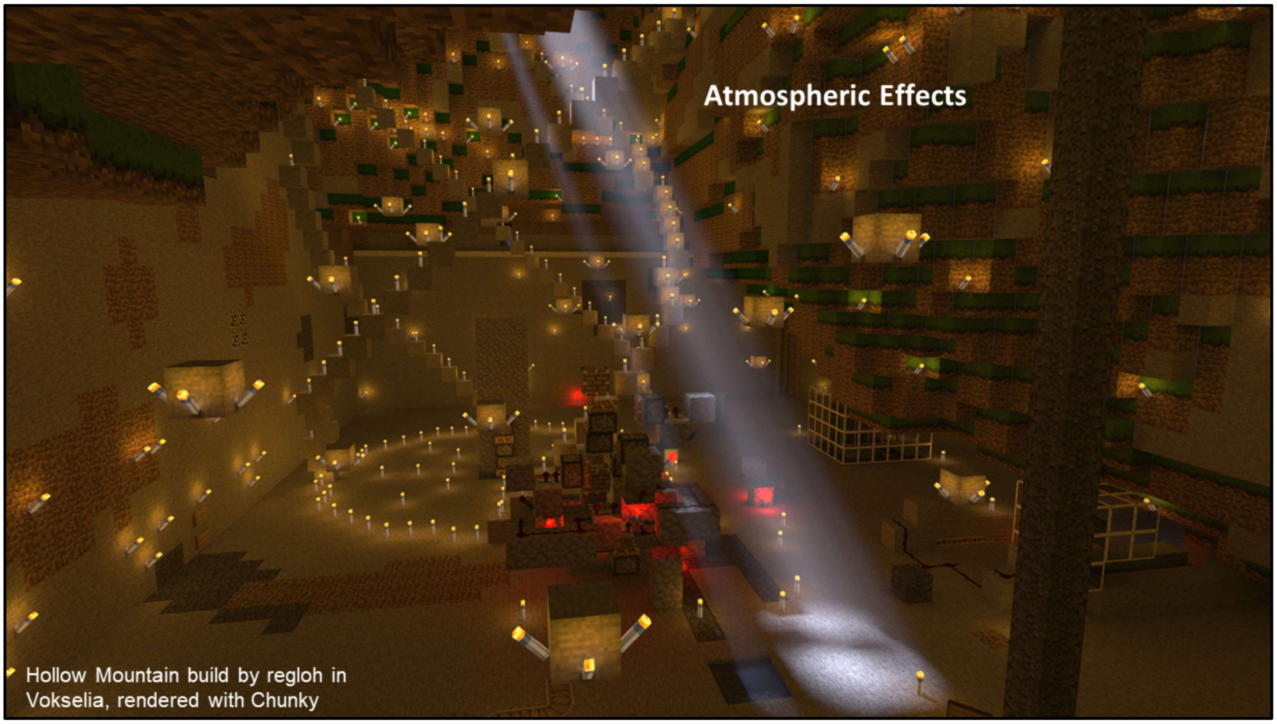


From Chris Wyman, of a scene free to reuse (Bistro outdoors, from ORCA collection).

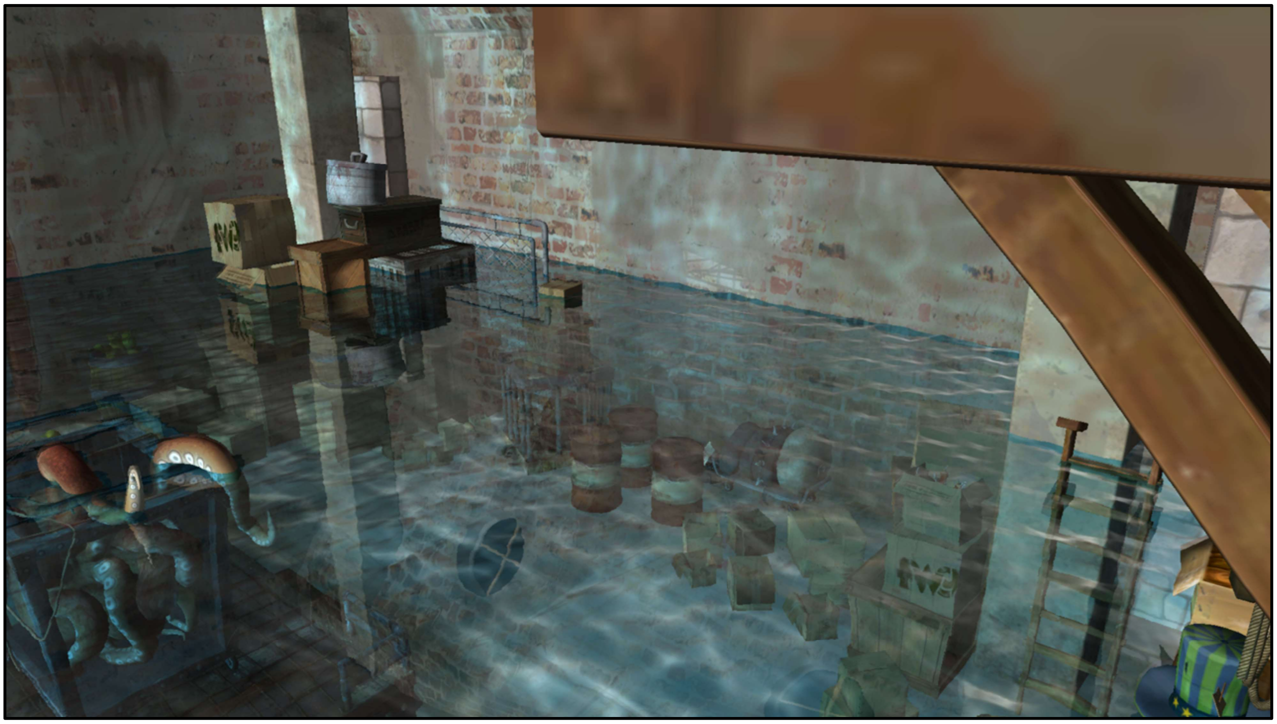
From Gavriil Klimov at NVIDIA

From Gavriil Klimov at NVIDIA

From Gavriil Klimov at NVIDIA



From vokselia.com, built by regloh of the Voxelians, rendered with Chunky –path trace

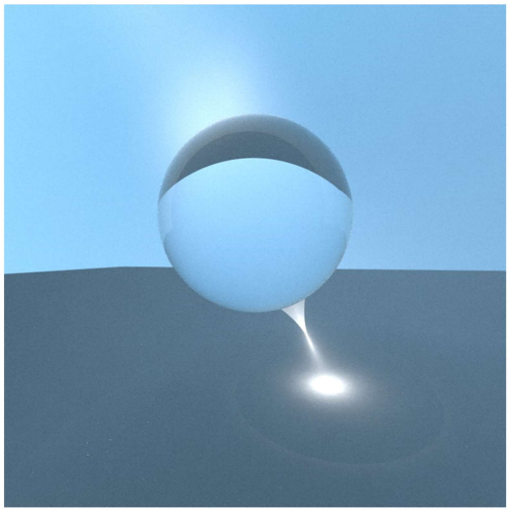


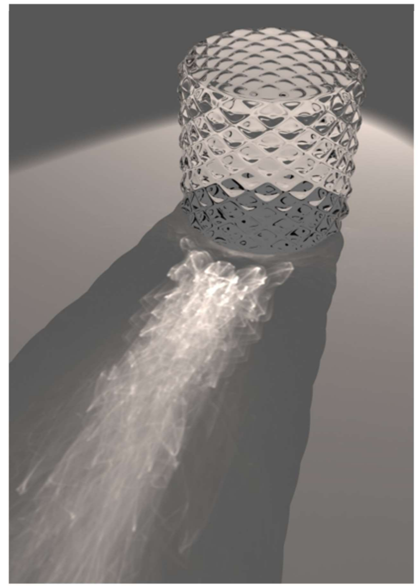
From Ray-Guided Volumetric Water Caustics in Single Scattering Media with DXR, NVIDIA. Ray Tracing Gems, http://raytracinggems.com

Most dangerous effect for last, and not because of the octopus here.

From Ray-Guided Volumetric Water Caustics in Single Scattering Media with DXR, NVIDIA.
Ray Tracing Gems, http://raytracinggems.com

Glass Caustics





Images courtesy Matt Pharr, Wenzel Jakob, and Greg Humphreys. Glass model by Simon Wendsche.

The Dangers of Ray Tracing

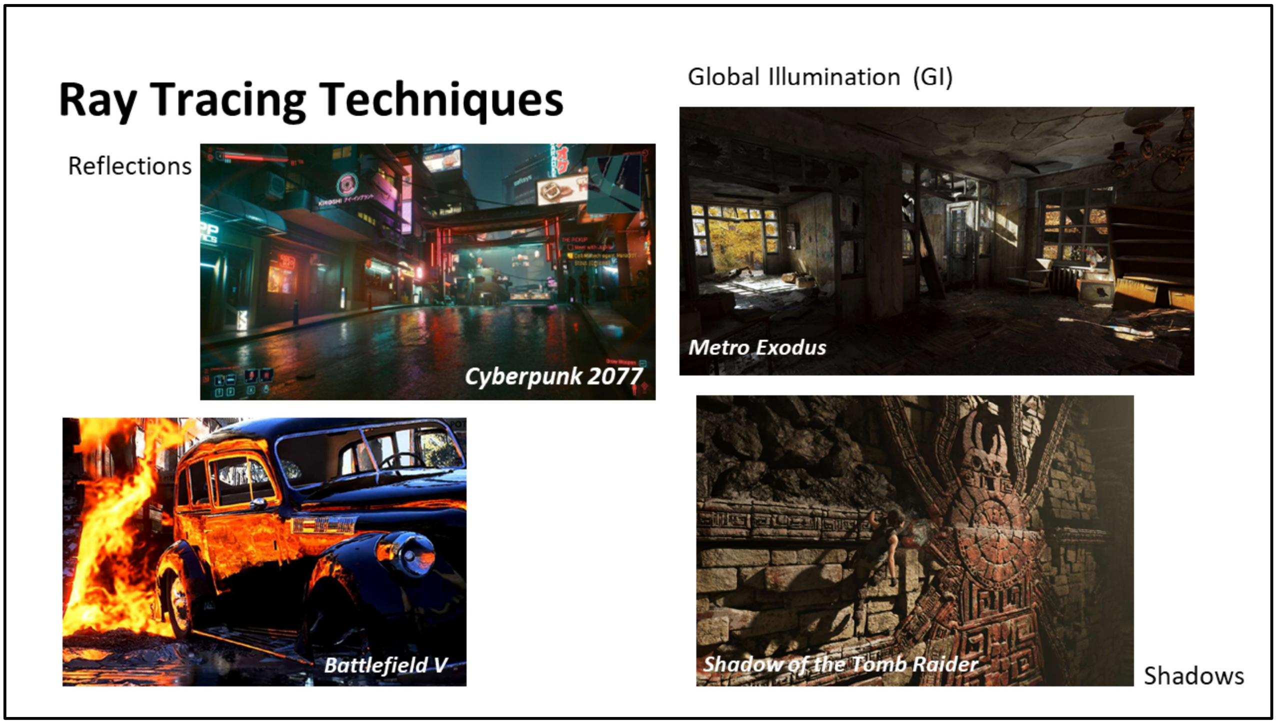
Not a render with a bad composited outside image, but reality. Let’s look closer…

The Dangers of Ray Tracing

Oh, that can't be good. Beware! Reality can burn you, literally.

The Dangers of Ray Tracing

Briefly exposed to the sun



Updated from Steve Parker (HPG 2019) and Chris Wyman (SIGGRAPH 2019 "THE PATH TO PERFORMANCE: SCALING GAME PATH TRACING"). Partners where we worked on the tech.

Embarrassingly Parallel

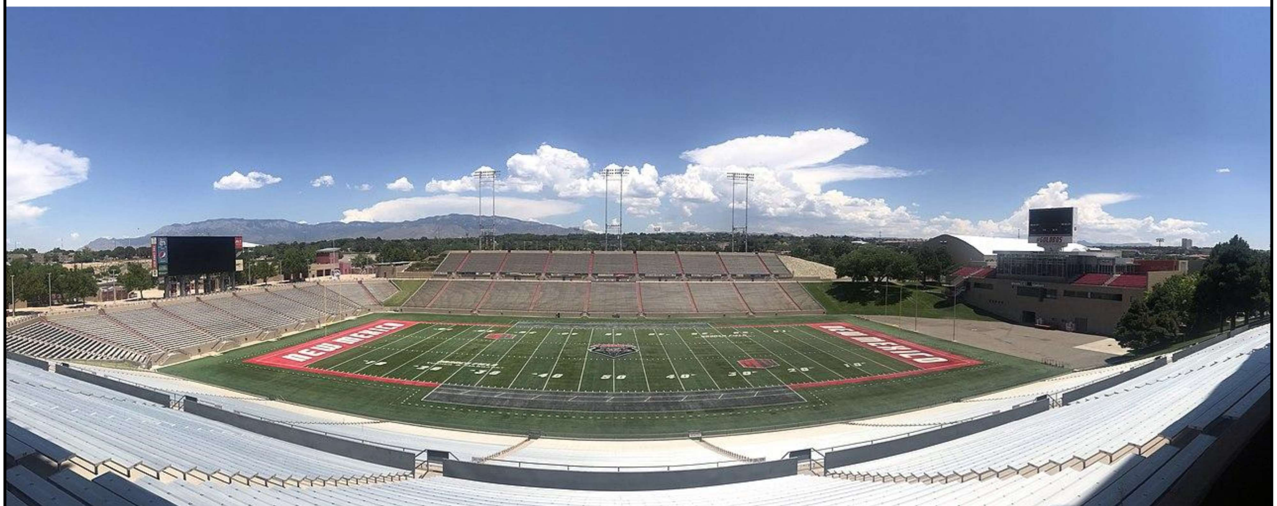


Image from Wikimedia Commons, File:UNM - Dreamstyle Stadium panorama.jpg

https://blogs.nvidia.com/blog/2018/08/01/ray-tracing-global-illumination-turner-whitted/
- includes the Compleat Angler film

https://commons.wikimedia.org/wiki/File:UNM_-_Dreamstyle_Stadium_panorama.jpg

Moore’s Law Is Ending (Really!)

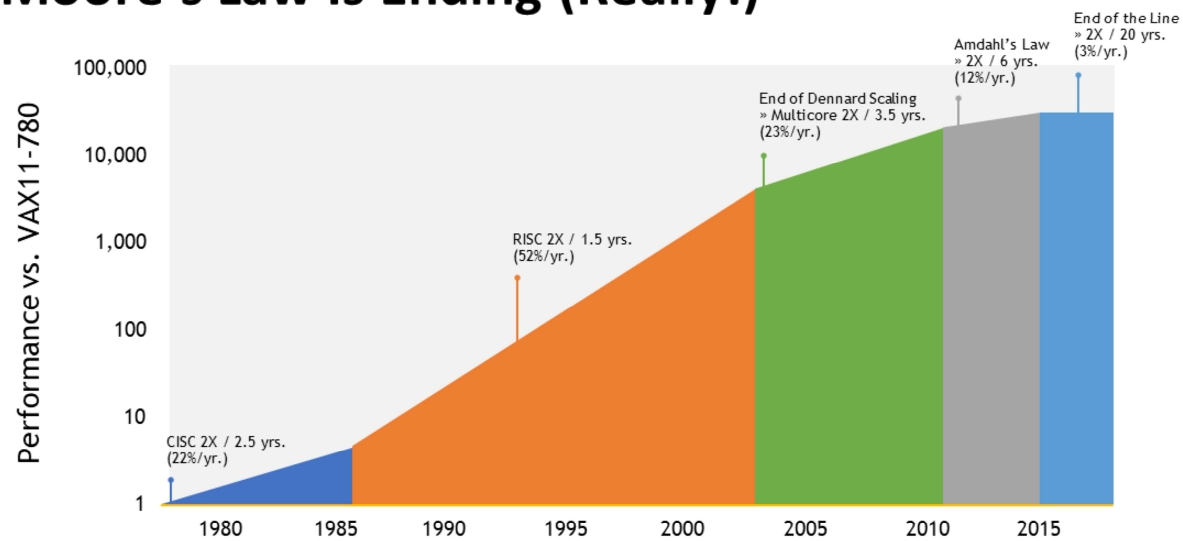
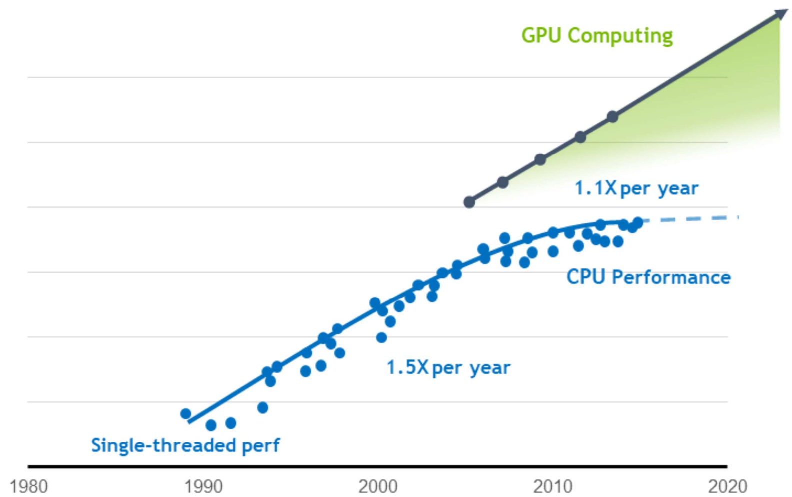


Figure after Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, “A Domain-Specific Architecture for Deep Neural Networks,” Communications of the ACM, September 2018, Vol. 61 No. 9, Pages 50-59.

“performance on standard processor benchmarks will not double before 2038”

Moore’s Law is Ending (Really!)

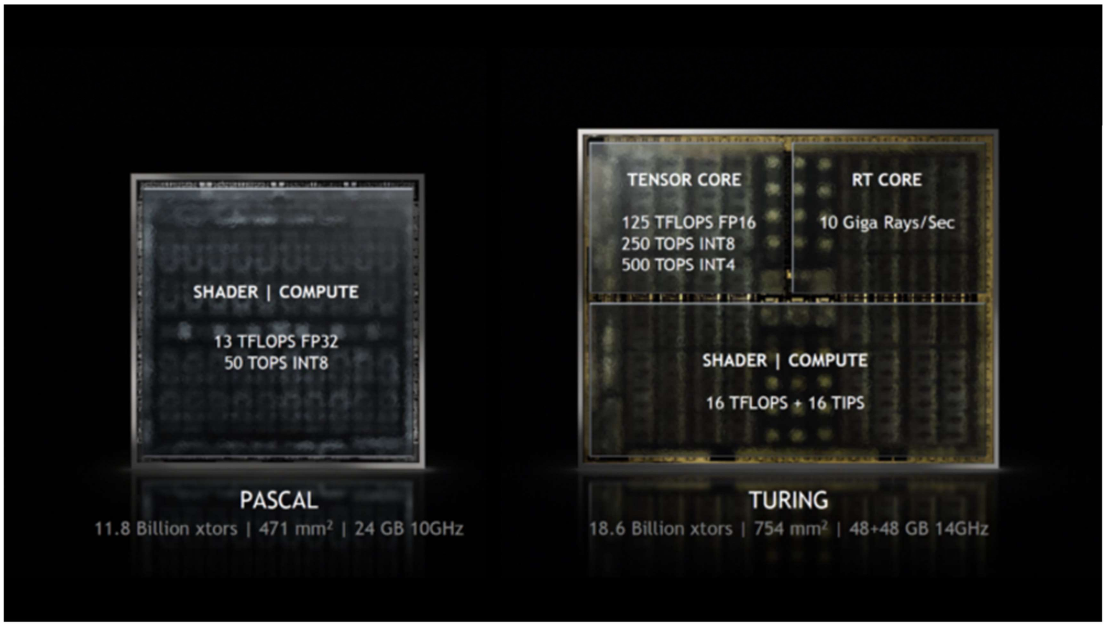


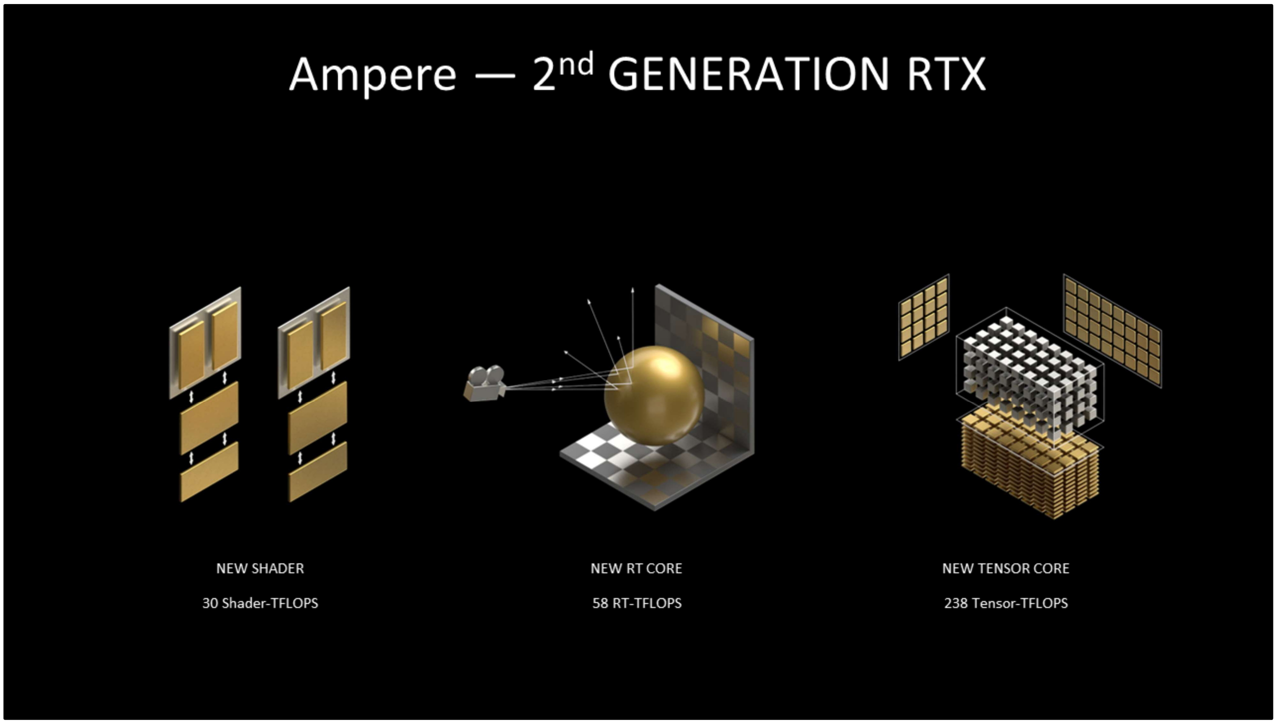
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

Jensen’s version from Kevin Acocella.

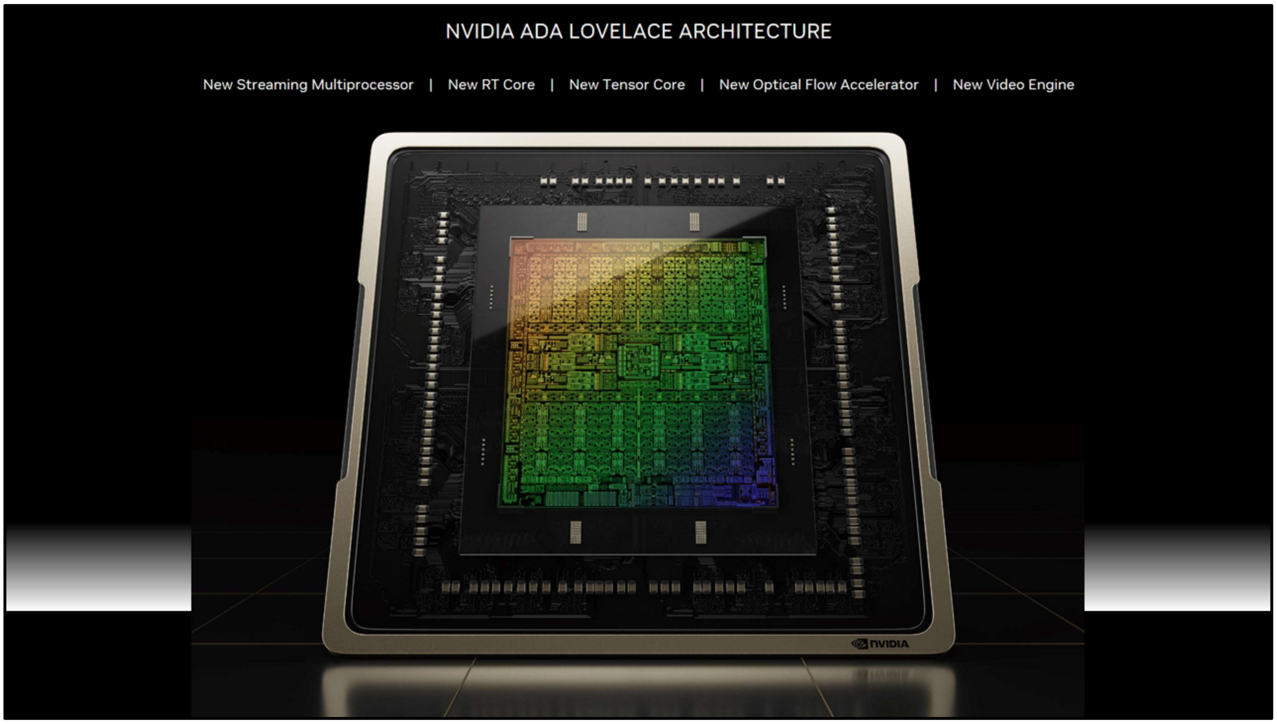
Moore's Law delivers twice the performance at the same cost, or at the same performance at half the cost, every year and a half

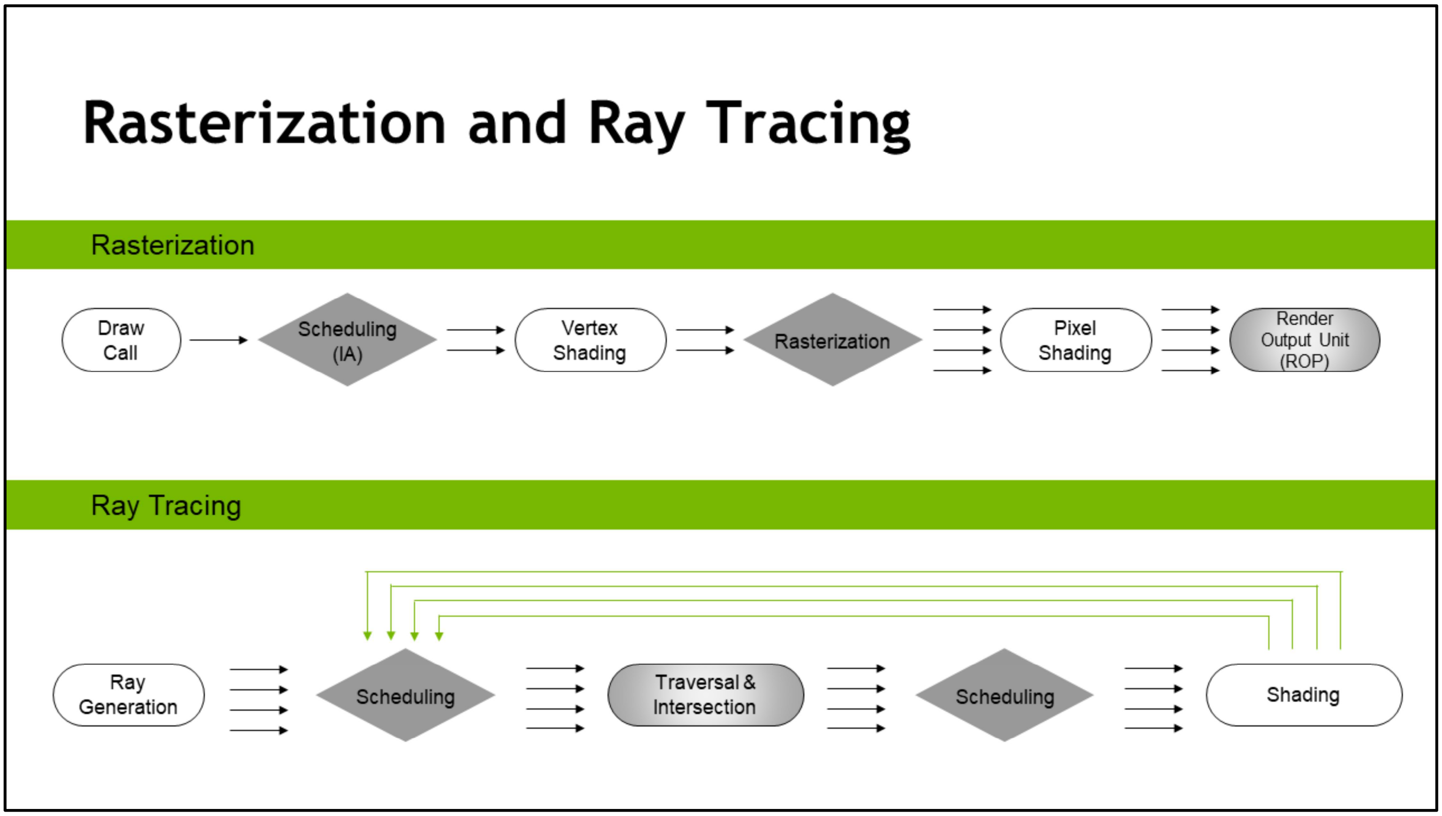
More Special-Purpose Hardware





https://t.co/fOpYWBh7K6 - a more in-depth briefing of the various new things in Ada.


NVIDIA ADA LOVELACE ARCHITECTURE
New Streaming Multiprocessor | New RT Core | New Tensor Core | New Optical Flow Accelerator | New Video Engine
NVIDIA



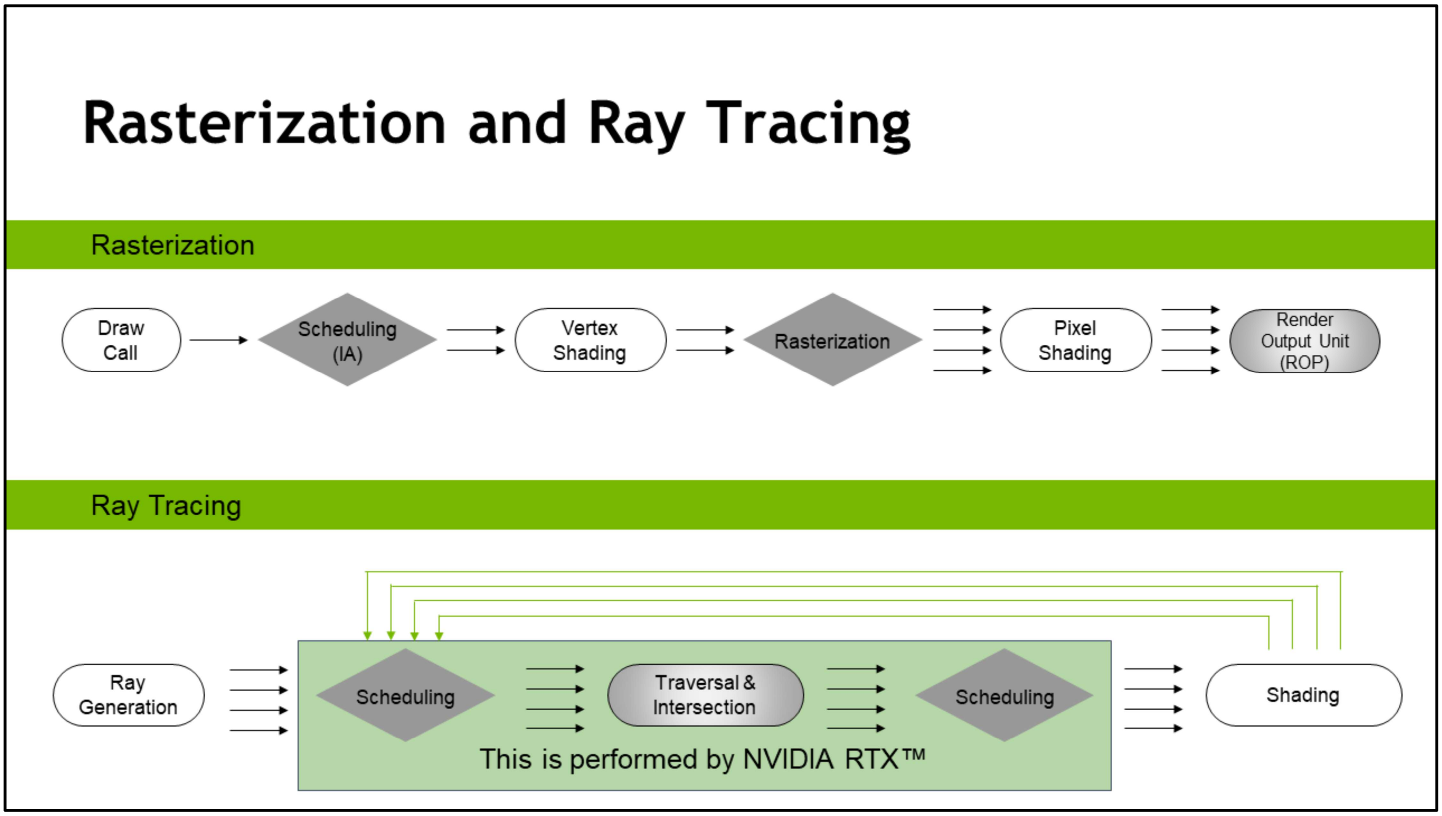
Comparing graphics and ray tracing pipeline

Gray = fixed-function / hardware. Improves over time.
Diamond = some kind of scheduling happening
White = programmable

<click>

Optix (and DXR and soon Vulkan) does Scheduling & Traversal, Intersection.
Ray generation and shading is the developers responsibility.
Workflow is often recursive, shaders can trace rays.

IA = input assembler



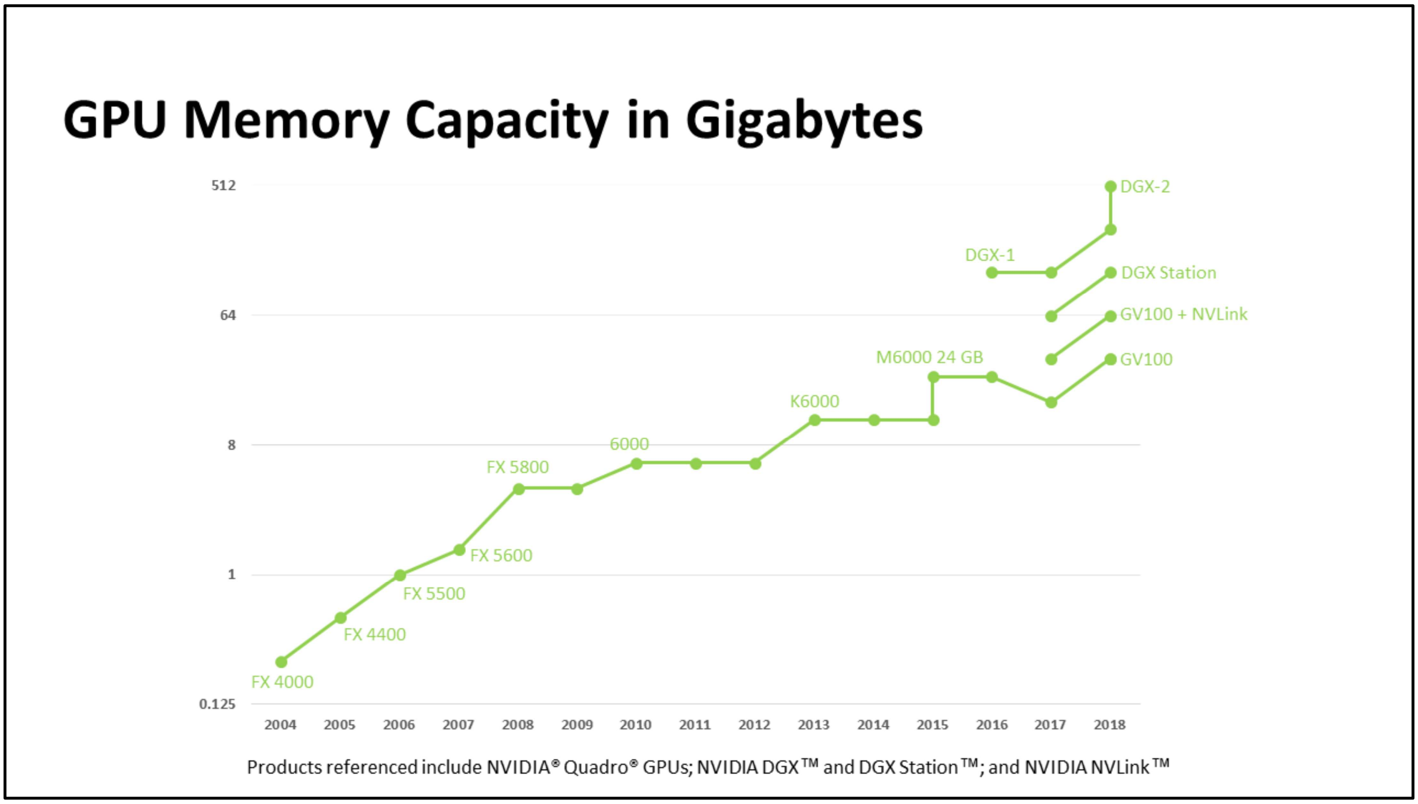
Comparing graphics and ray tracing pipeline

Gray = fixed-function / hardware. Improves over time.
Diamond = some kind of scheduling happening
White = programmable

<click>

Optix (and DXR and soon Vulkan) does Scheduling & Traversal, Intersection.
Ray generation and shading is the developers responsibility.
Workflow is often recursive, shaders can trace rays.

IA = input assembler



4K: 3840 x 2160 pixels takes 33 MB (including alpha) – means 30 images is 1 GB

Unreal Engine’s Nanite – The Matrix Awakens

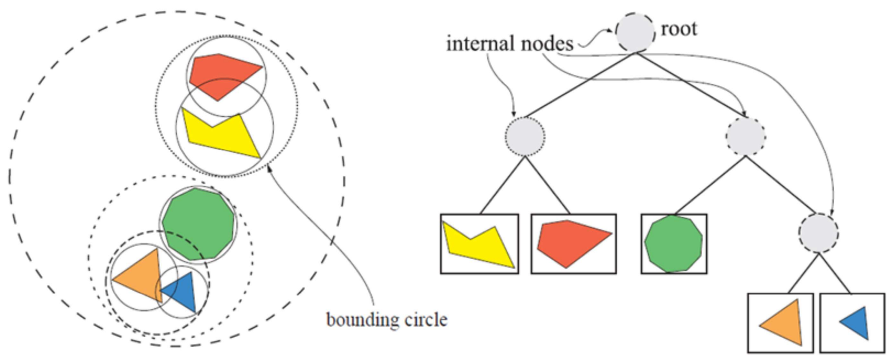
The city comprises seven million instanced assets, made up of millions of polygons each.

There are seven thousand buildings made of thousands of modular pieces, 45,073 parked cars (of which 38,146 are drivable), over 260 km of roads, 512 km of sidewalk, 1,248 intersections, 27,848 lamp posts, and 12,422 manholes.

Unreal Engine's Nanite

The Bounding Volume Hierarchy (BVH)

This scheme mostly won the efficiency data structure wars.

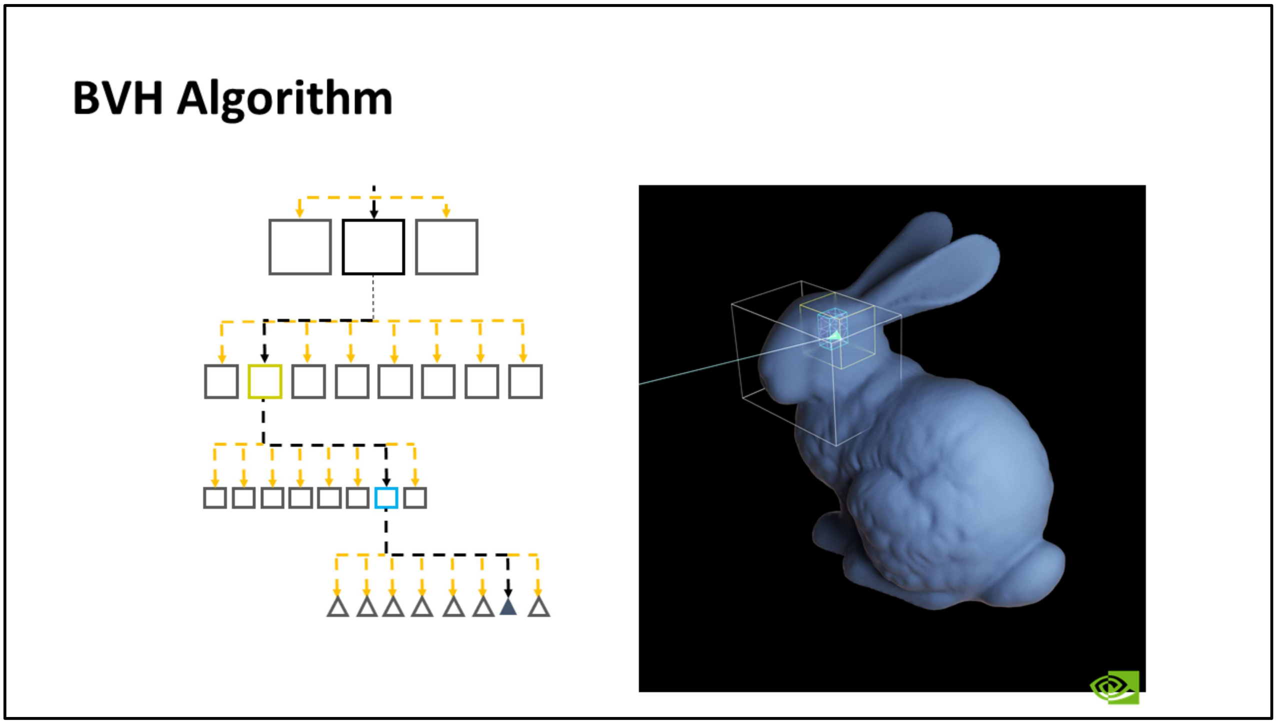


Traversing a BVH (i.e., tracing a ray) is typically **O(log N)**.

Image courtesy of
Real-Time Rendering

Nested grids do see use for voxel/volume rendering, and k-d trees for point clouds

Source: Real-Time Rendering (Eric coauthored, made figure)



Quick refresher on how RT tackles the scene representation problem.

10 box tests + 10 triangle tests vs 1000 triangle tests.

BVH not a new idea. Been around for decades. But devil is in the details if you want it really fast. Lots of research around that, both construction and traversal, from NV and many others.

Source: Steve Parker’s HPG 2019 talk

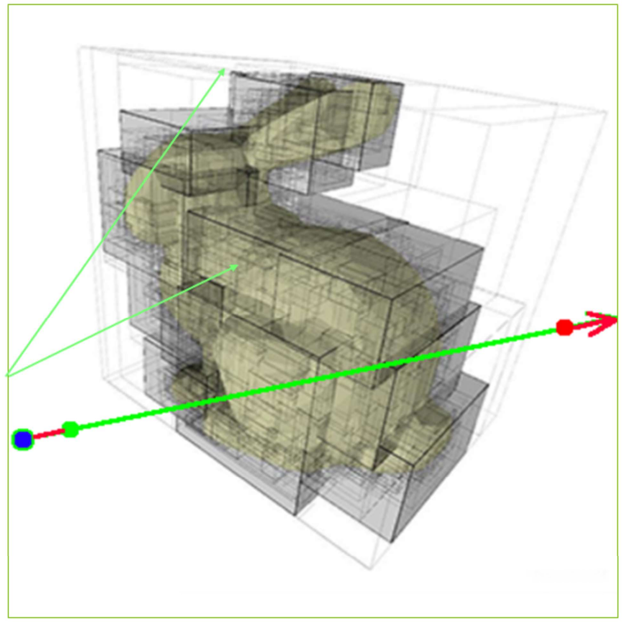
RT Cores

RT Cores perform:

- Ray-bounding volume hierarchy (BVH) traversal
- Ray-triangle intersection
- Instancing: 1 level

Return to shaders for:

- Multi-level instancing
- Custom intersection
- Shading



SM – streaming multiprocessor

Five Types of Ray Tracing Shaders

Ray-tracing pipeline split into ***five*** shaders:

- ***Ray generation shader*** — *define how to start tracing rays*
- ***Intersection shader(s)*** — *define how rays intersect geometry*
- ***Miss shader(s)*** — *shading for when rays miss geometry*
- ***Closest-hit shader(s)*** — *shading at the intersection point*
- ***Any-hit shader(s)*** — *run once per hit (e.g., for transparency)*

From Chris Wyman’s introduction to ray tracing SIGGRAPH 2019 notes

Five Types of Ray Tracing Shaders

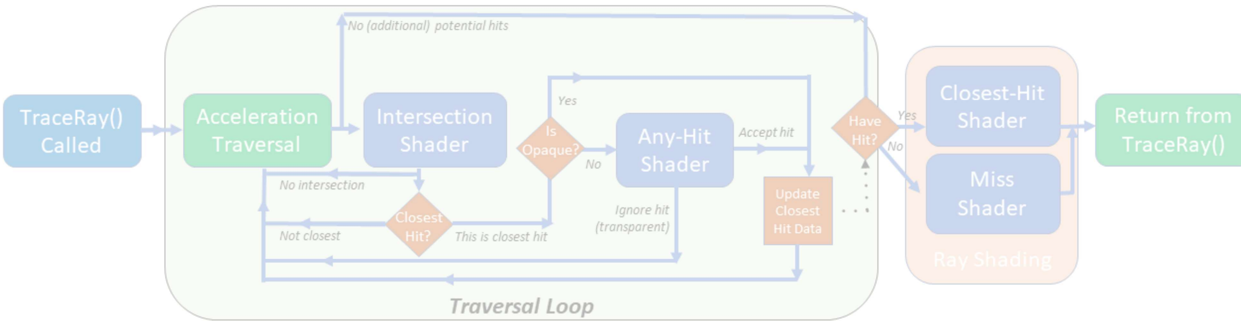
Ray-tracing pipeline split into ***five*** shaders:

- ***Ray generation shader*** ← Controls other shaders
- ***Intersection shader(s)*** ← Defines object shapes (one shader per type)
- ***Miss shader(s)***
- ***Closest-hit shader(s)***
- ***Any-hit shader(s)***

← Controls per-ray behavior (often many types)

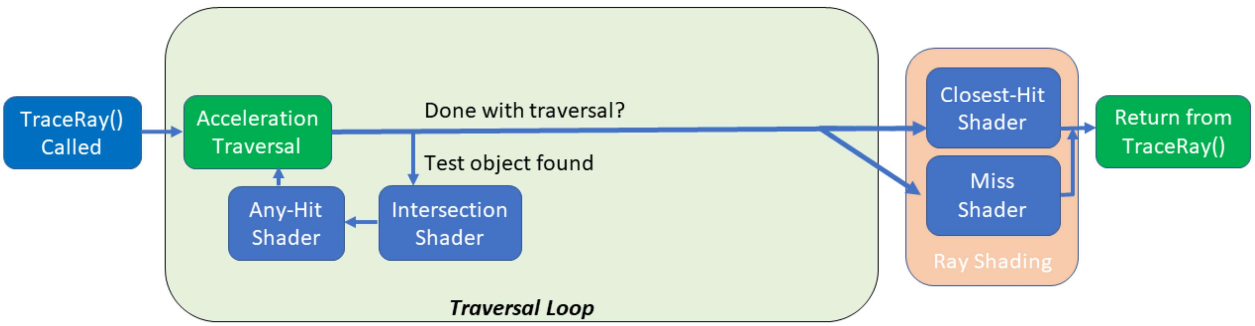
From Chris Wyman’s introduction to ray tracing SIGGRAPH 2019 notes

How Do These Fit Together? [Eye Fry Version]



How Do These Fit Together? [LOGICAL Version]

- Loop during ray tracing, test hits until there are no more; then shade.



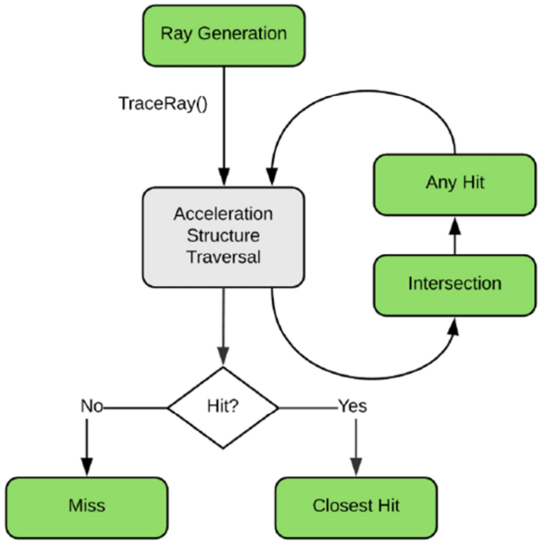
- Closest-hit shader can generate new rays: reflection, shadow, etc.

And the closest-hit (or even miss) shader can generate new rays, starting the process again.

Really, the any-hit shader should be shown as optional here, but the general idea is about right.

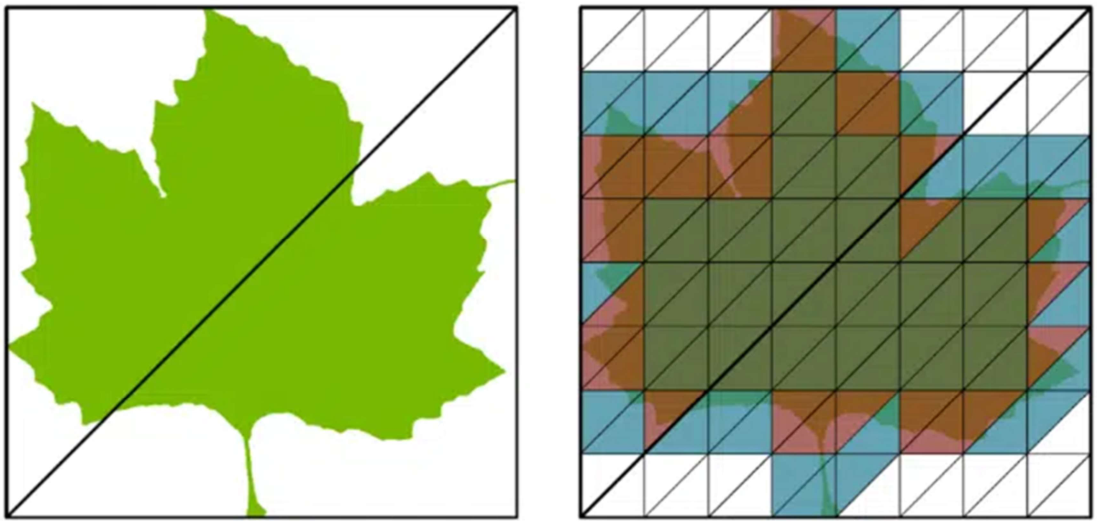
Accelerating Cutouts

Any Hit



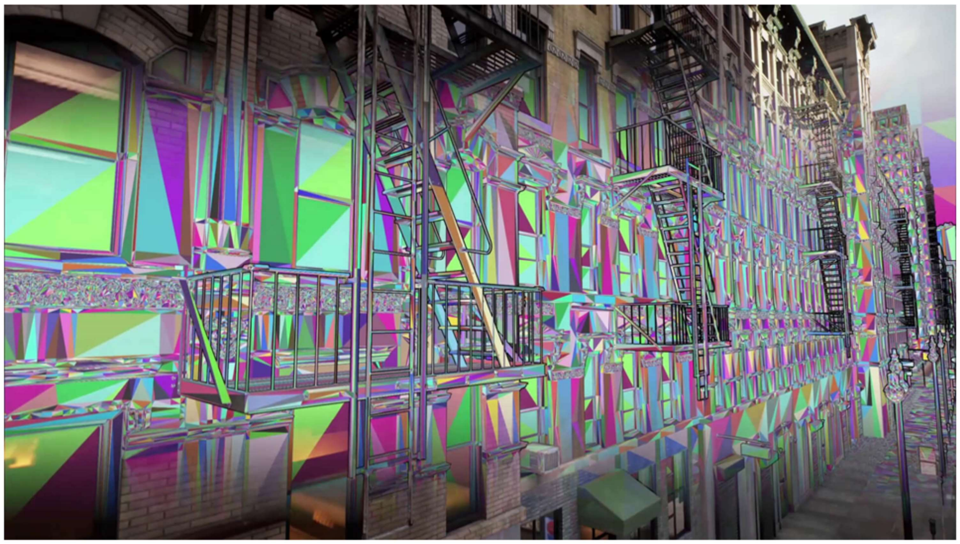
http://www.realtimerendering.com/Real-Time_Rendering_4th-Real-Time_Ray_Tracing.pdf

Opacity Masks



Grabbed from https://www.hardwaretimes.com/nvidia-ada-lovelace-architecture-what-makes-the-rtx-4090-4x-faster-than-the-rtx-3090-ti/

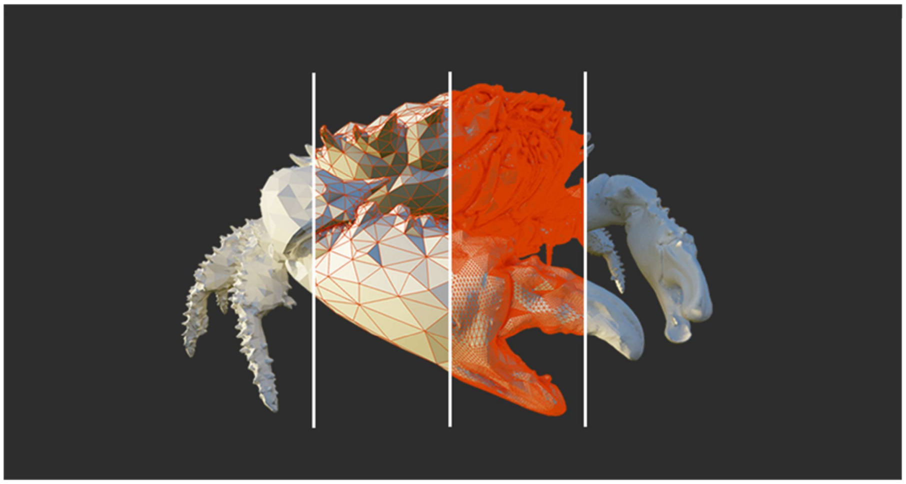
Unreal Engine’s Nanite – level of detail



https://developer.nvidia.com/rtx/ray-tracing/micro-mesh?nvid=nv-int-tblg-158743-vt10

Sample micro-mesh composed of 16K individual micro-meshes (left half), expanding to 2 million micro-triangles (right half), consuming ~1 byte per micro-triangle;
from: **threedscans.com**

Micro-Mesh in NVIDIA's Ada

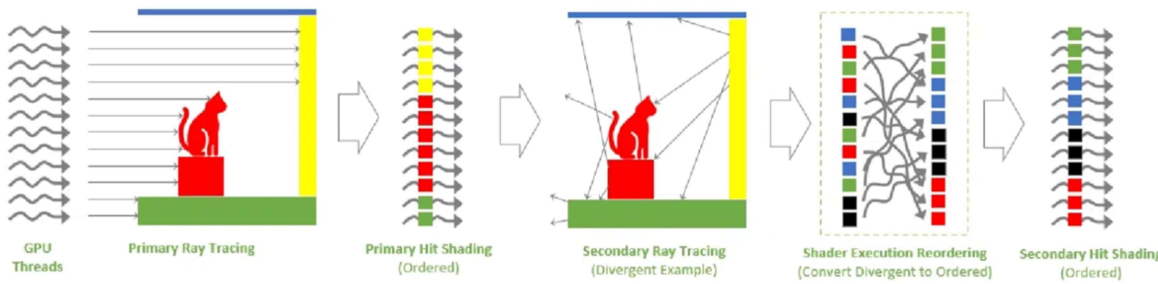


16k micro-meshes expands to 2 million micro-triangles, each consuming ~1 byte

https://developer.nvidia.com/rtx/ray-tracing/micro-mesh?nvid=nv-int-tblg-158743-vt10

Sample micro-mesh composed of 16K individual micro-meshes (left half), expanding to 2 million micro-triangles (right half), consuming ~1 byte per micro-triangle;
from: **threedscans.com**

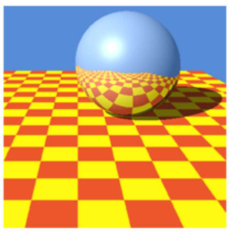
Shader Execution Reordering



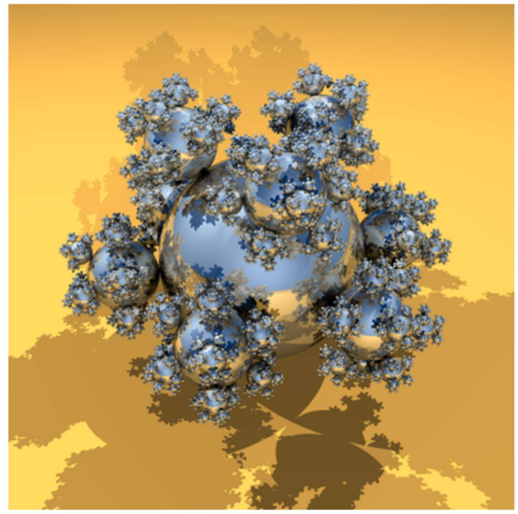
https://www.hardwaretimes.com/nvidia-ada-lovelace-architecture-what-makes-the-rtx-4090-4x-faster-than-the-rtx-3090-ti/

1987: AT&T Pixel Machine

AT&T Pixel Machine: Interactive, low-res, ray-traced mirror sphere atop a mandrill texture.



+



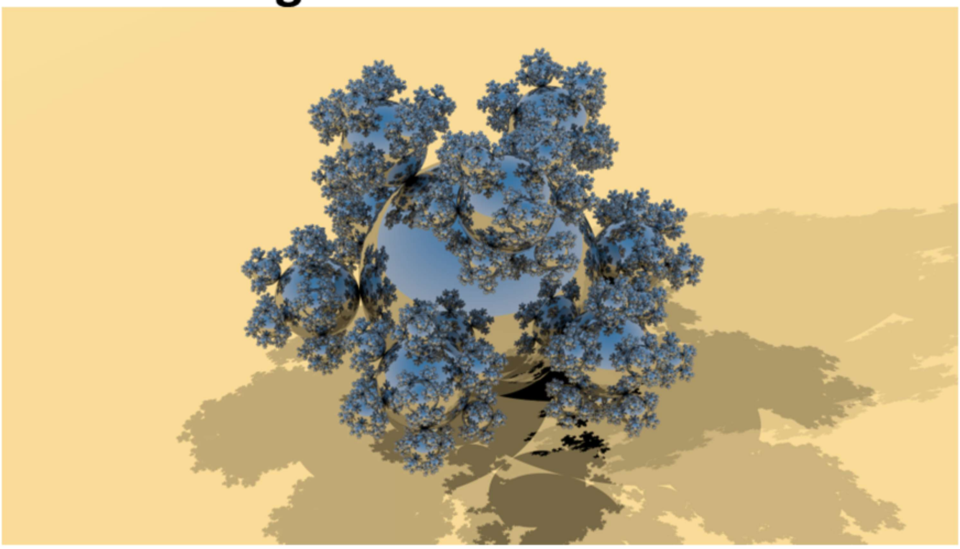
Sphereflake: 512 x 512 pixels of 7,381 spheres and plane rendered in just 30 seconds, later optimized to 16 seconds in 1988.

Sphereflake on pixel machine ran in 30 seconds, 16 seconds a year later due to software tuning. http://www.realtimerendering.com/resources/RTNews/html/rtnews4a.html#art4

Sphereflake is in the Standard Procedural Database program set.

Real-time browser demo here: https://www.shadertoy.com/view/wdtSWf

2018: Turing

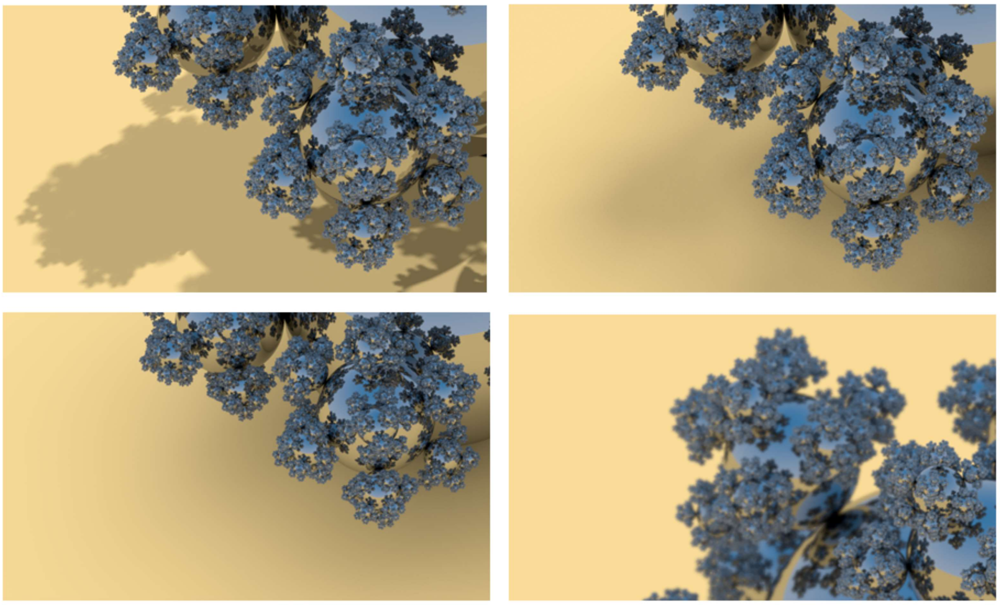


Sphereflake:
1920 x 1080 pixels of 48 million spheres and plane rendered at 60 FPS, running on an NVIDIA Titan V card.

Instancing could improve this further…

https://erich.realtimerendering.com/rtrt/index.html – 31 years later

Soft Shadows, Hemisphere lighting, Depth of Field



All of these were easy code changes.

Soft shadows, softer, hemispherical lighting, and depth of field.

https://erich.realtimerendering.com/rtrt/index.html – easier to see the depth of field there

The Rendering Equation

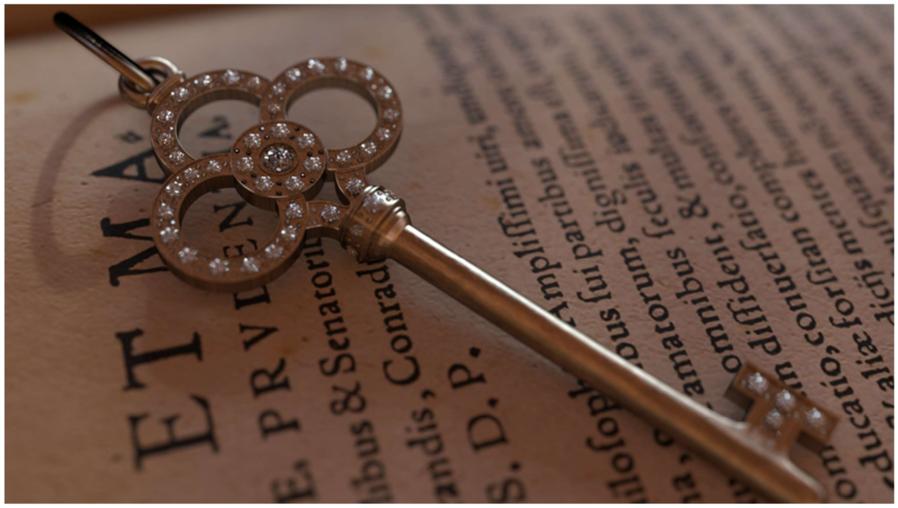


Image by Alexia Rubod

The Rendering Equation

$$L_{\mathrm{o}}(X, \hat{\omega}_{\mathrm{o}}) = L_{\mathrm{e}}(X, \hat{\omega}_{\mathrm{o}}) + \int_{\mathbf{S}^2} L_{\mathrm{i}}(X, \hat{\omega}_{\mathrm{i}}) \, f_X(\hat{\omega}_{\mathrm{i}}, \hat{\omega}_{\mathrm{o}}) \, |\hat{\omega}_{\mathrm{i}} \cdot \hat{n}| \, d\hat{\omega}_{\mathrm{i}}$$

Outgoing light   Emitted light   Incoming light   Material   Lambert

From Morgan McGuire’s “Path Tracing Review” – a pure path trace picks omega_i randomly in a uniform way.

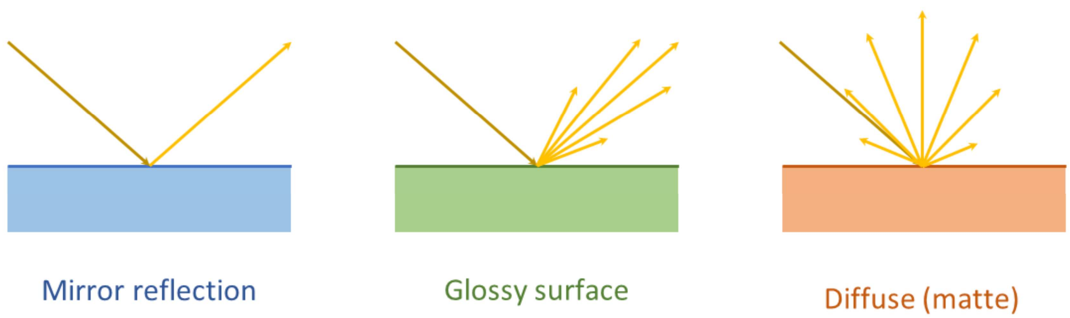
Path-Traced Game: *Quake II*

Simple assets and limit path types
Note: Initial implementation is open source, http://brechpunkt.de/q2vkpt/

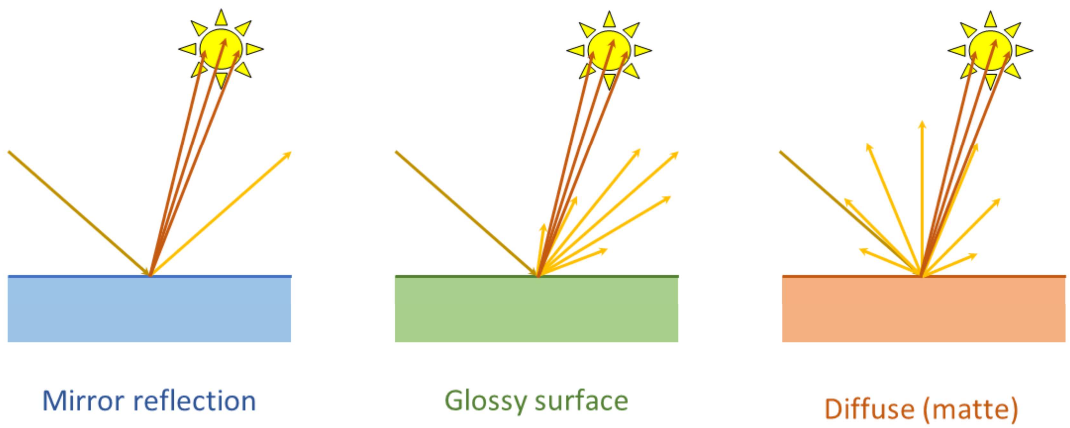
https://www.nvidia.com/en-us/geforce/campaigns/quake-II-rtx/

Original: http://brechpunkt.de/q2vkpt/

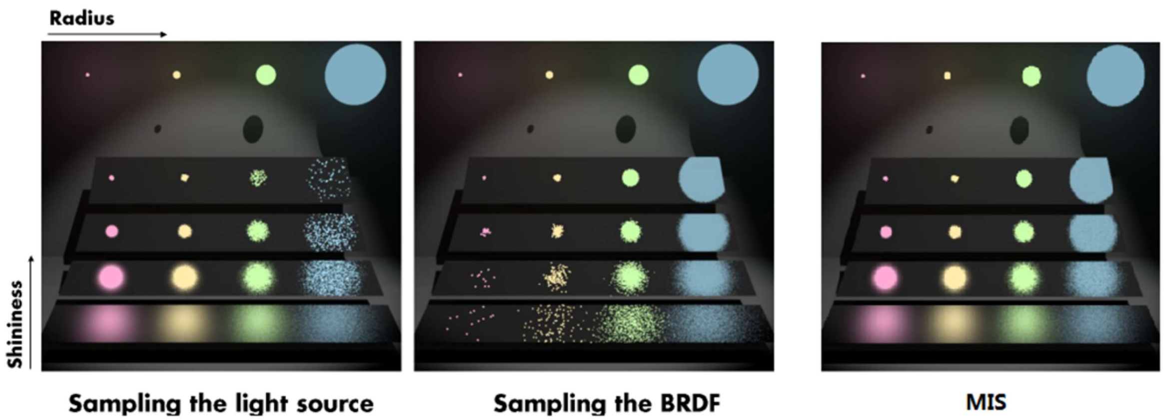
Mirror, Glossy, Diffuse



Mirror, Glossy, Diffuse with MIS



Multiple Importance Sampling



From Veach and Guibas, *Optimally Combining Sampling Techniques for Monte Carlo Rendering*, SIGGRAPH 1995

From **Multiple Importance Sampling** (MIS) demonstrated by Veach and Guibas in 1995.

Figure 2 permission purchased 12/16/2019 for use in this and derivative presentations.

http://graphics.stanford.edu/papers/combine/

Even with Turing, only have a budget of a few rays per pixel in real-time
10 GigaRays/sec: 20 rays/pixel at 4k@60Hz
Less in practice: game doesn't only do raytracing, scenes are complex, need shading, etc.

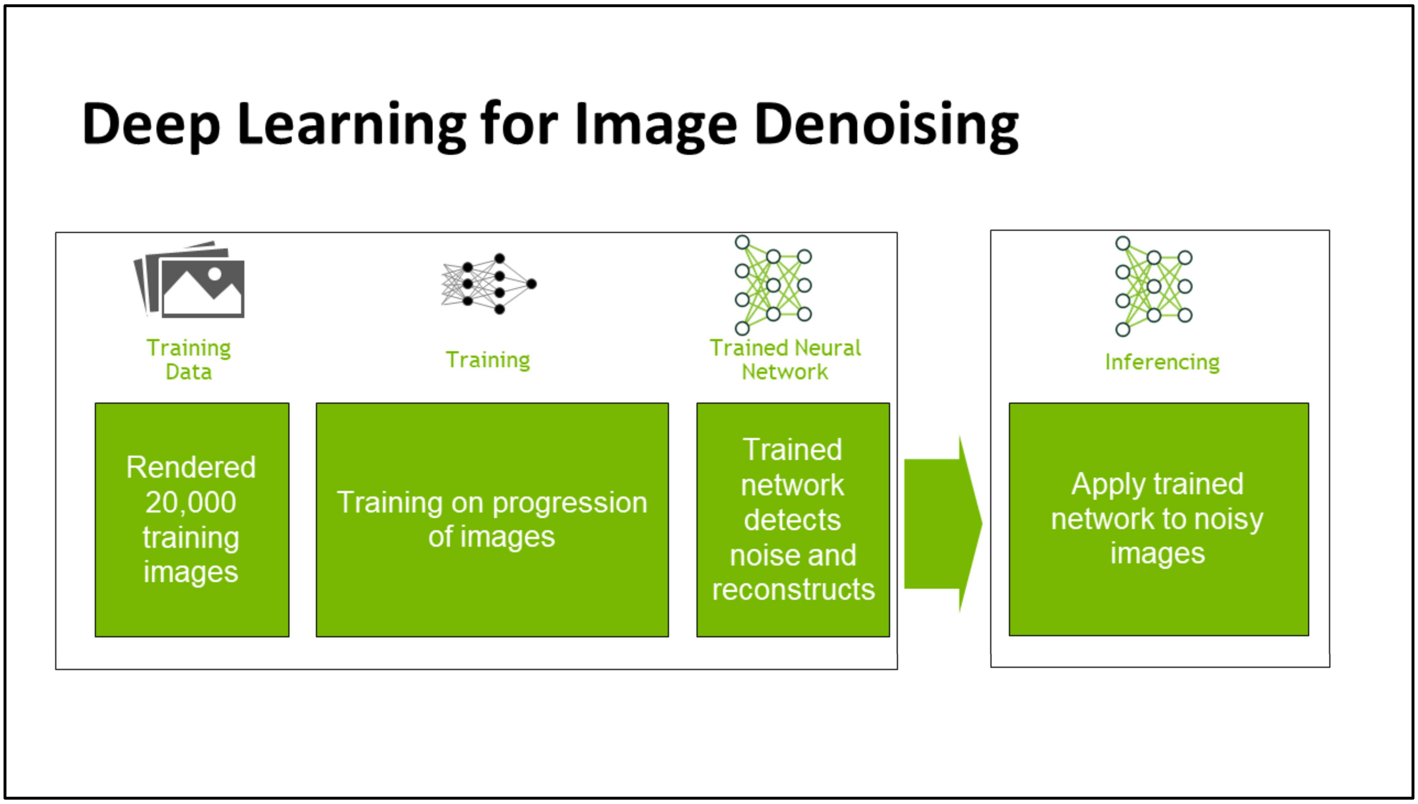
Important to use our rays wisely.
Use rays where they matter most
Hybrid Rendering of key visual effects (reflections, GI, shadows, AO, ..)
No point in ray tracing primary visibility, the rasterizer is still an efficient beast we've tuned for 25 years, no reason not to use it!

Start with a noisy result and reconstruct

Specialized non-graphical data for denoising, like tangents for hairs.

Even films use denoising

https://developer.nvidia.com/gameworks-ray-tracing



Developing an application that benefits from DL is different from traditional software development, where software engineers must carefully craft lots of source code to cover every possible input the application may receive.

From NVIDIA's "Deep Learning for Rendering" 2018

At the core of a DL application, much of that source code is replaced by a neural network.

To build a DL application, first a data scientist designs, trains and validates a neural network to perform a specific task.

The task could be anything, like identifying types of vehicles in an image, reading the speed limit sign as it goes whizzing past, translating English to Chinese, etc.

The trained neural network can then be integrated into a software application that feeds it new inputs to analyze, or "infer" based on its training.

The application may be deployed as a cloud service, on an embedded platforms, in an automobiles, or other platforms.

As you would expect, the amount of time and power it takes to complete inference tasks is one of the most important considerations for DL applications, since this determines both the quality/value of the user experience and the cost of deploying the application.

Test scene for raytraced shadows. Overcast sky so shadows are soft.
This what it looks like at 1spp without denoising.

And this is the results of applying our denoisers to 1spp ray traced shadows.

What this does is cleverly re-use and blend the samples from neighbor pixels as well as previous frames.

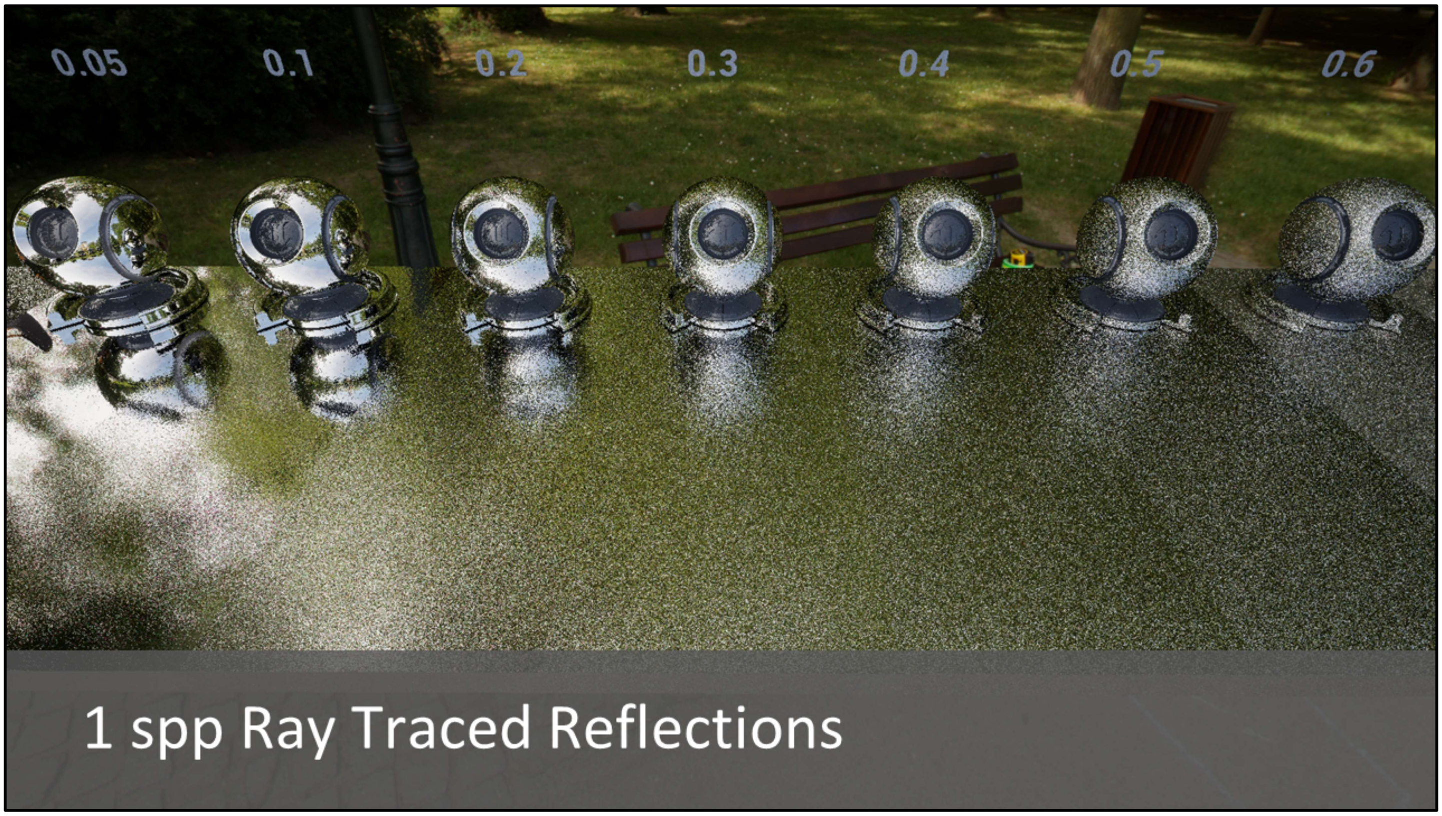
So this is a combination of spatial and temporal filtering.

And this is the ground truth, using hundreds of rays per pixel and no denoising.

We got really close with 1spp denoised!

Finally this is what you would get with shadow mapping. There is a bit of peter panning going on at the feet of the pedestrians, and we also lost the interesting contact hardening effect for the overcast sun soft shadows.

Not to mention the semi-transparent shadows of the trees that look completely different, because shadow maps can’t handle transparency well.



Let’s look at reflections.

1spp with different roughnesses.

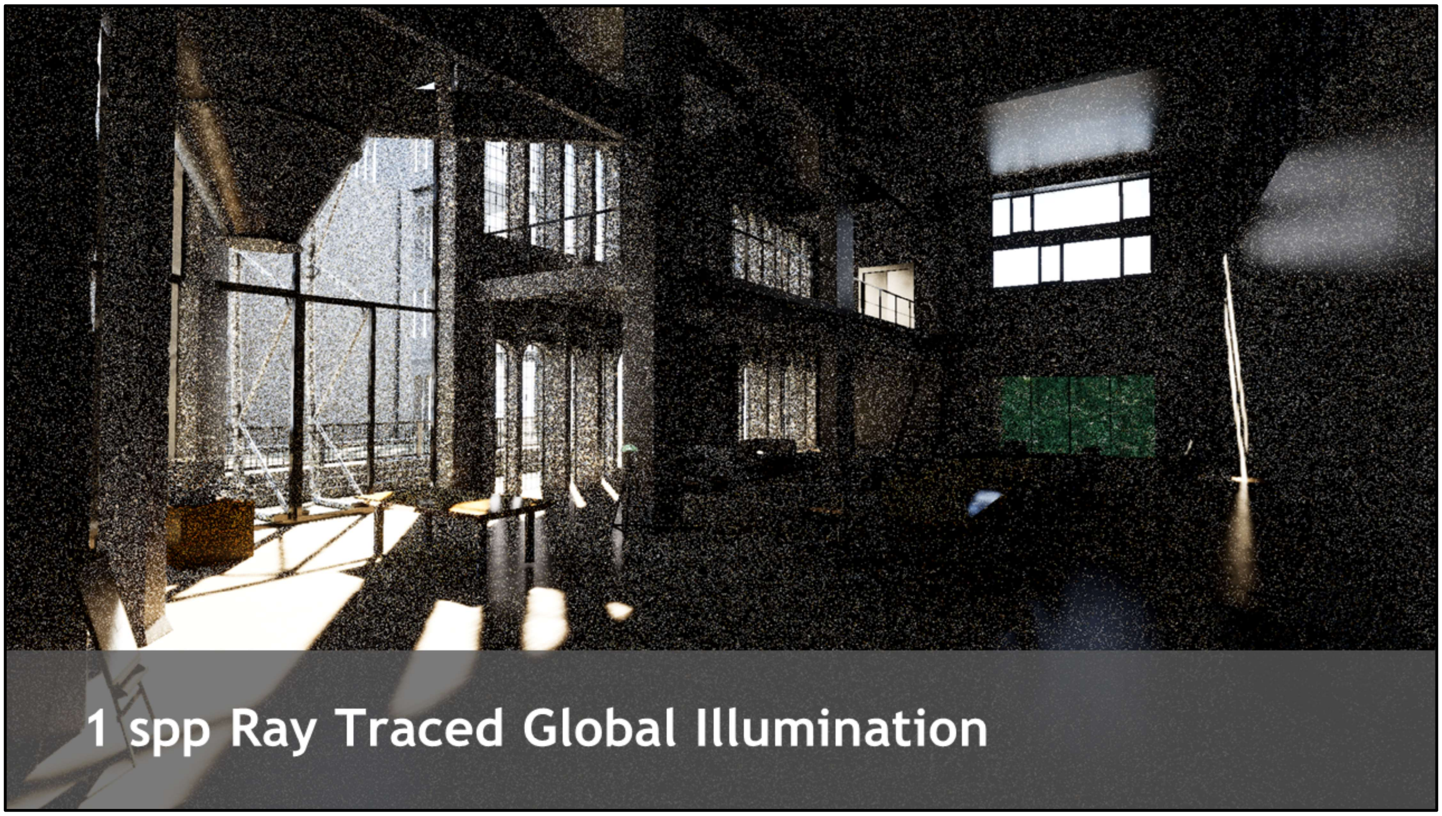
Denoising for reflections will take into account material parameters such as surface roughness.

And this is ground truth rendered with thousands of rays per pixel. Got quite close.

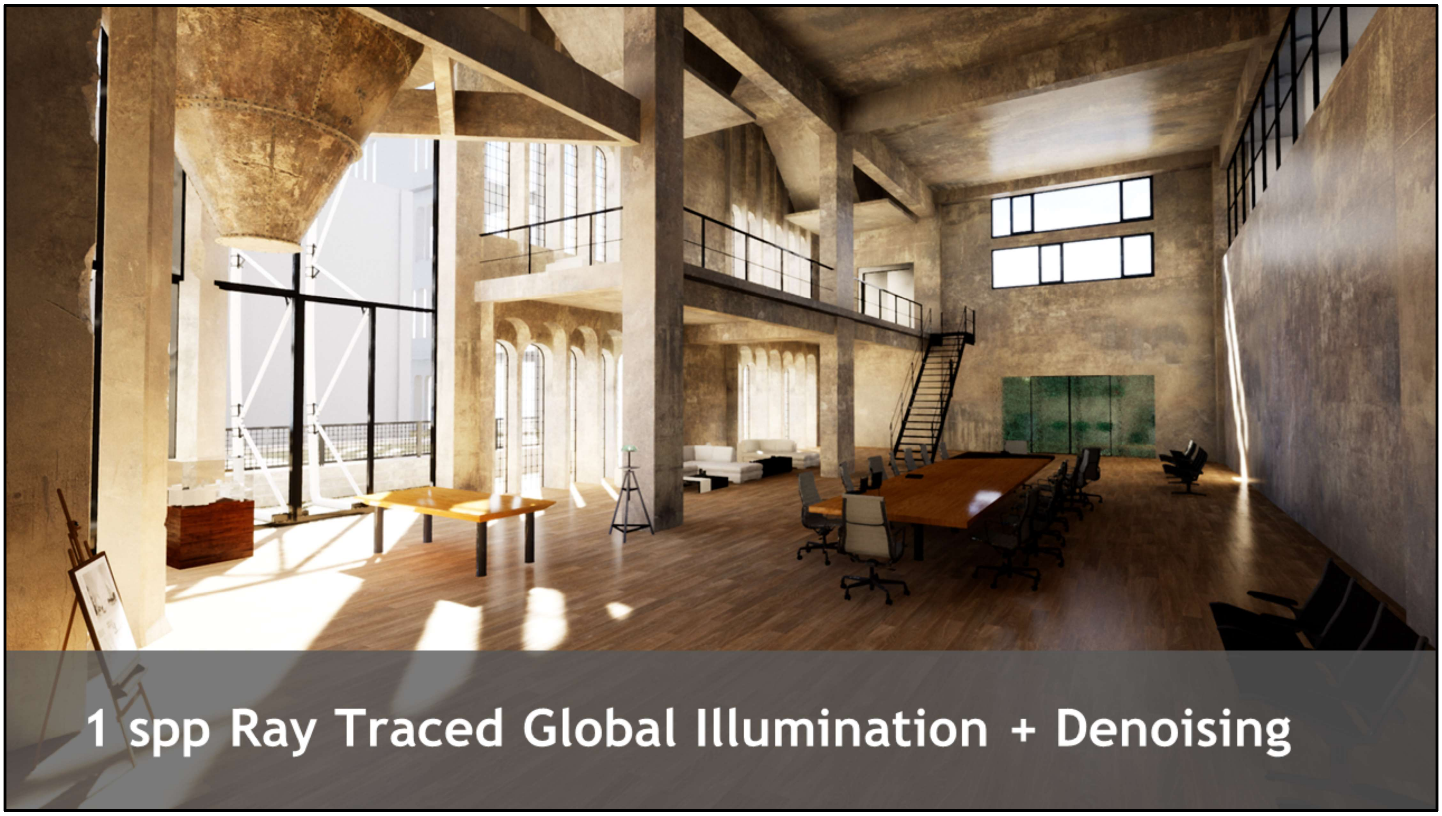
This is what we would get with traditional stochastic SSR combined with reflection probes.

I.e. this is what a traditional game would look like (I think this is actually stock UE4).

See all the typical SSR artifacts.



For example, this is what you would get with pure 1 sample per pixel path traced indirect diffuse global illumination. As I said before you can notice there is a lot of pixels that are just black, because they failed to sample a valid light path that connect the camera to the light source.



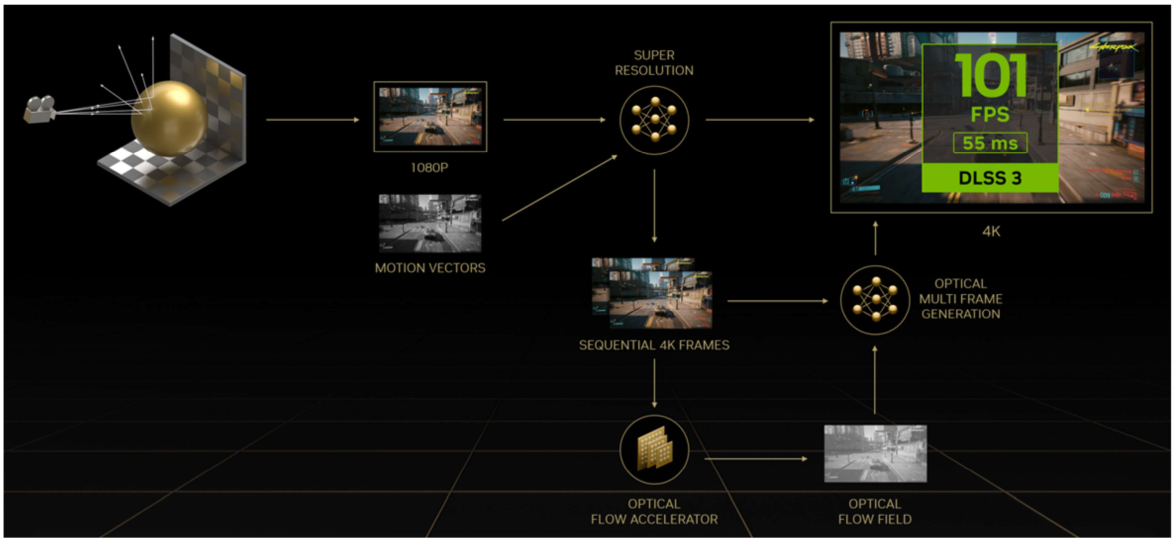
Now boom, this is the results of applying our denoisers for GI. Things are looking much cleaner now. And if you look closer, the indirect shadows under those pillars and tables are actually not washed out either.

This is the ground truth image. I think we have matched it pretty closely. It does still have a bit more details in contact shadow region.

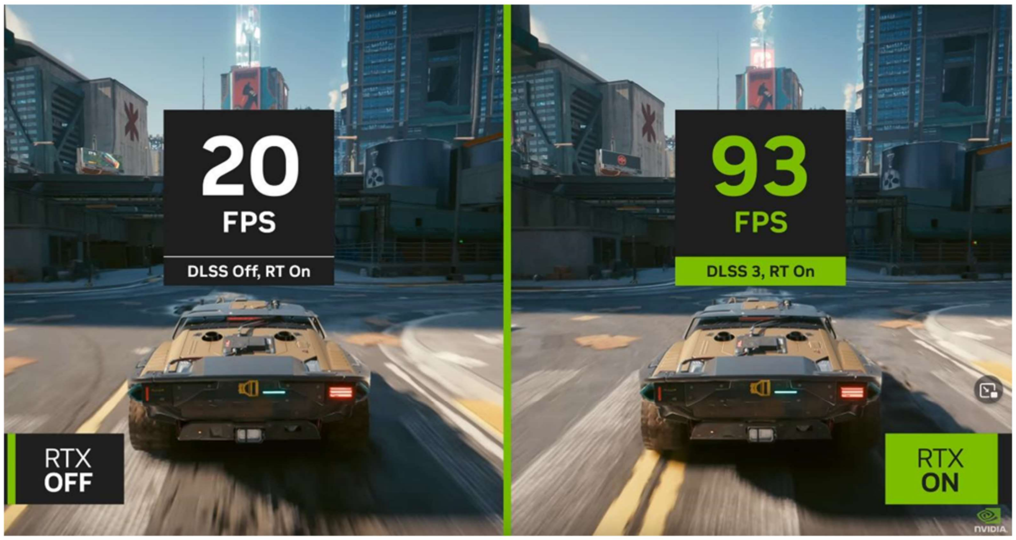
from *Death Stranding*

Checkerboard upscale using DL for figuring out what is best to “interpolate” (really, extrapolate)

Deep Learning Super Sampling 3 (DLSS3)

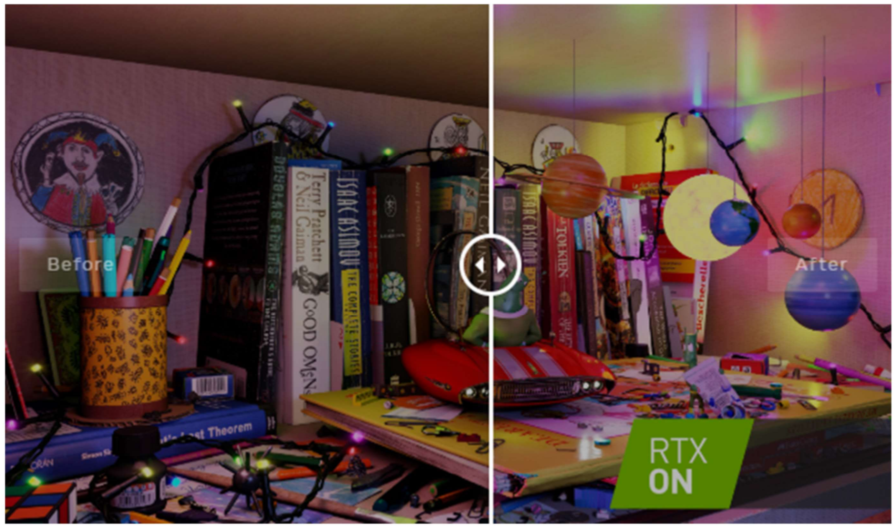


DLSS3 Performance



RTXDI – Direct Illumination, aka ReSTIR

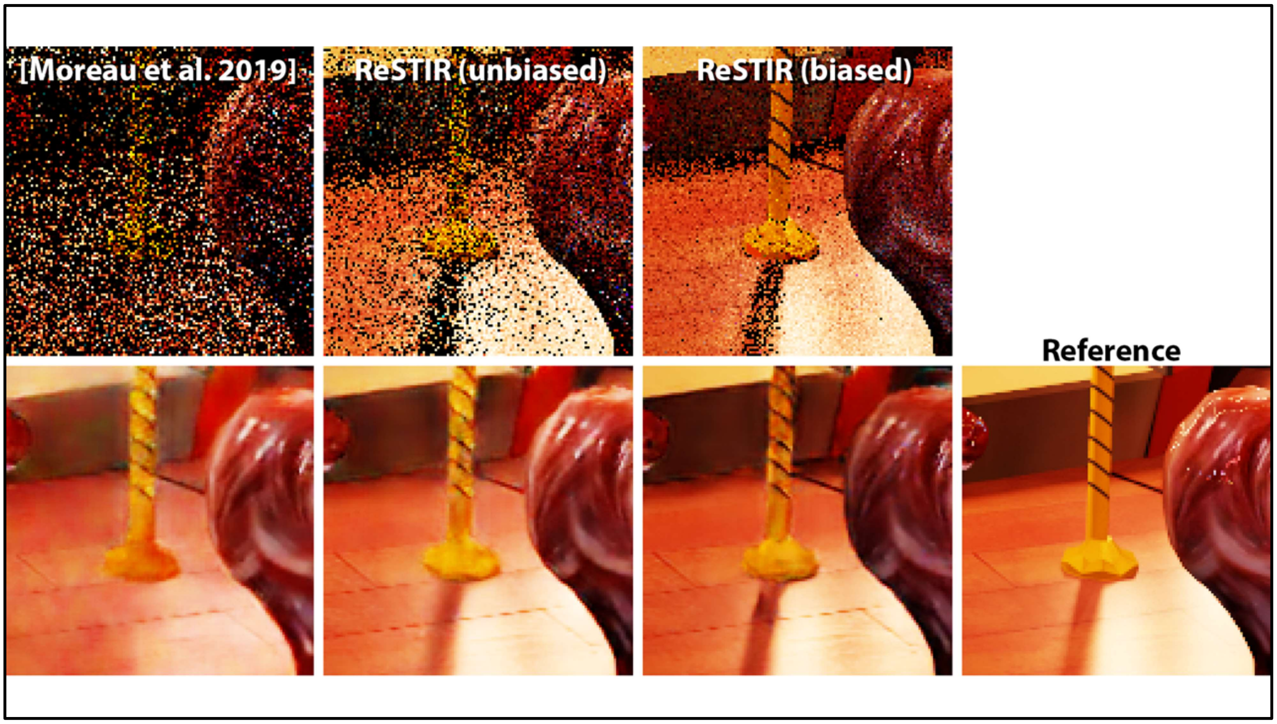
Find the important lights and use them to shade



https://developer.nvidia.com/rtx/ray-tracing/rtxdi

Research resources: http://lousodrome.net/blog/light/2022/05/14/reading-list-on-restir/

ReSTIR + Denoise



Can’t denoise data you don’t have, e.g., the striping around the pole.

RTXGI – Global Illumination, via probes

Use more elaborate probe types and update as needed

https://developer.nvidia.com/rtx/ray-tracing/rtxgi

Research: https://jcgt.org/published/0010/02/01/ and https://casual-effects.com/research/Majercik2021Resampling/index.html https://research.nvidia.com/publication/2021-06_real-time-neural-radiance-caching-path-tracing

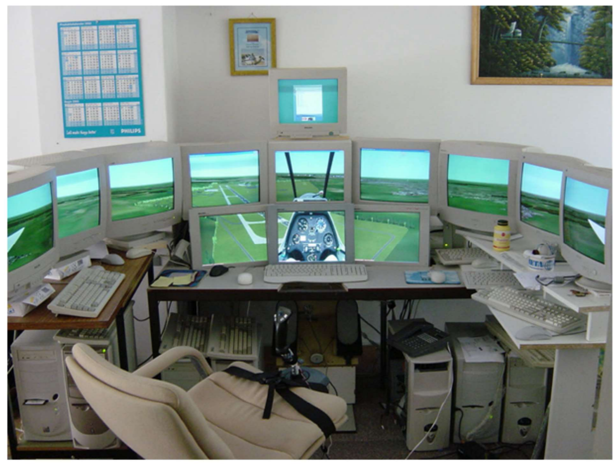
Fast MLP library for CUDA: https://github.com/NVlabs/tiny-cuda-nn https://github.com/NVlabs/instant-ngp

https://dreamfusionpaper.github.io/ and https://arxiv.org/abs/2209.14988

Imagen: https://imagen.research.google/

LeFohn's Law

The job of the renderer is not to make the picture; the job of the renderer is to collect enough samples that the AI can make the picture.



We like coherence for hardware – single instruction multiple data. But that's a waste.

Drunk loses his wallet in an alley, looks under a streetlight because it's easier to see. We need to look everywhere, but sensibly.

So sample N+1 should tell the AI as much as possible that it didn't already know from samples [1..N].

Enderton's Corollary

If your rays are coherent, you're shooting too many rays.

Or:

Cherish your samples

https://smile.amazon.com/Imperator-Scorpion-Gaming-Computer-Office/dp/B08HYRNJCH

Marbles at GTC
Marbles Now
720p @ 25 fps
1440p @ 30 fps
DLSS for AA (no scaling)
DLSS Upscaling
Recorded on RTX8000 (TU102)
Recorded on A6000 (GA102)
Indirect GI is on
Indirect GI is on
No DOF
DOF
1 dome light + 1 indirect light
85+ Lights

Ray Tracing: Media & Entertainment

Rasterization & ray tracing hybrid, Playstation 5

The Matrix Awakens
from Epic on Unreal Engine

It’s not just NVIDIA making ray tracing enhanced scenes.

https://www.fastcompany.com/90736343/how-epic-games-is-changing-gaming-and-maybe-the-metaverse

Ray Tracing: Architectural Design

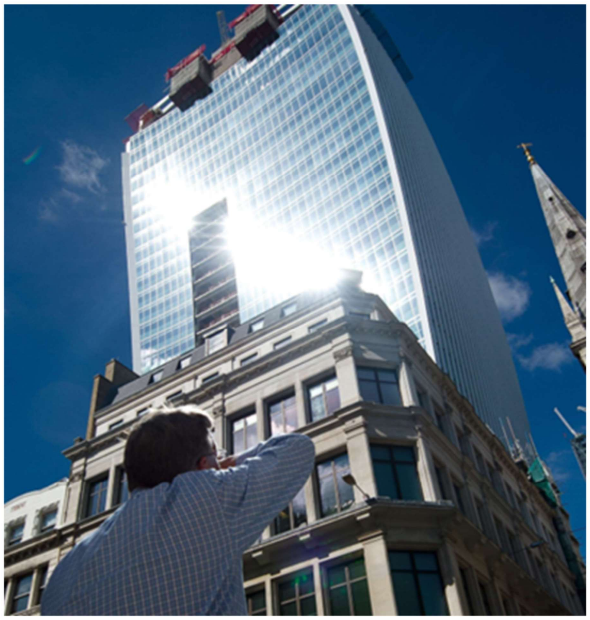
World Trade Center,
from "Ray Tracing: A Tool for All," by Jon Peddie

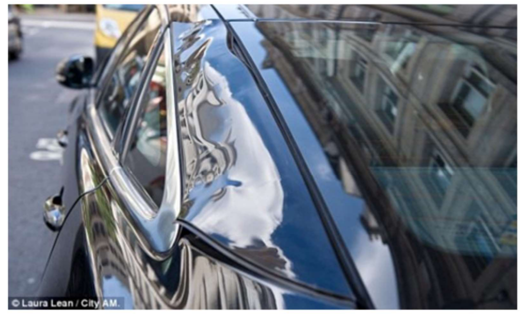
The one on the right is the real one.

Ray Tracing: Industrial Design

Bentley Motors virtual showroom, 110+ GB data, using the open source OSPRay Studio application

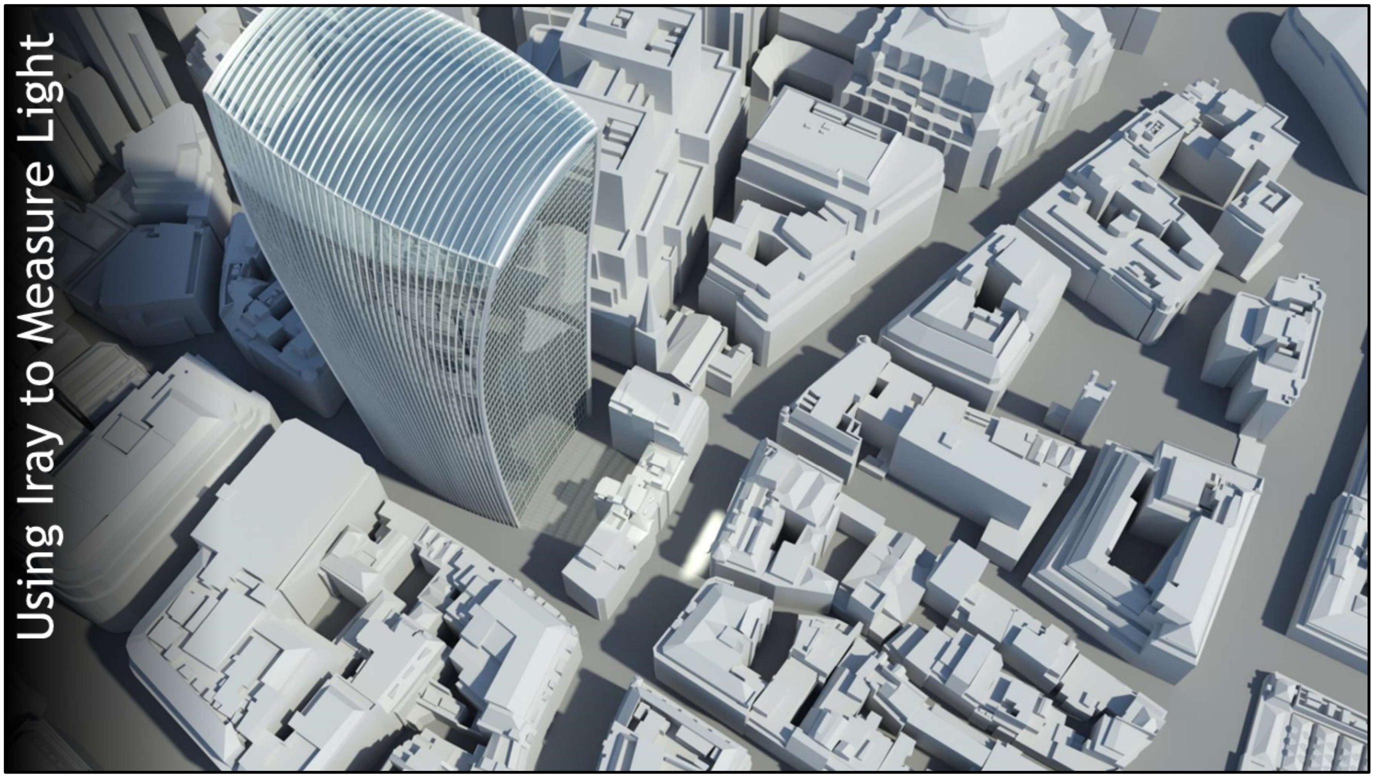
The London “Fryscraper”



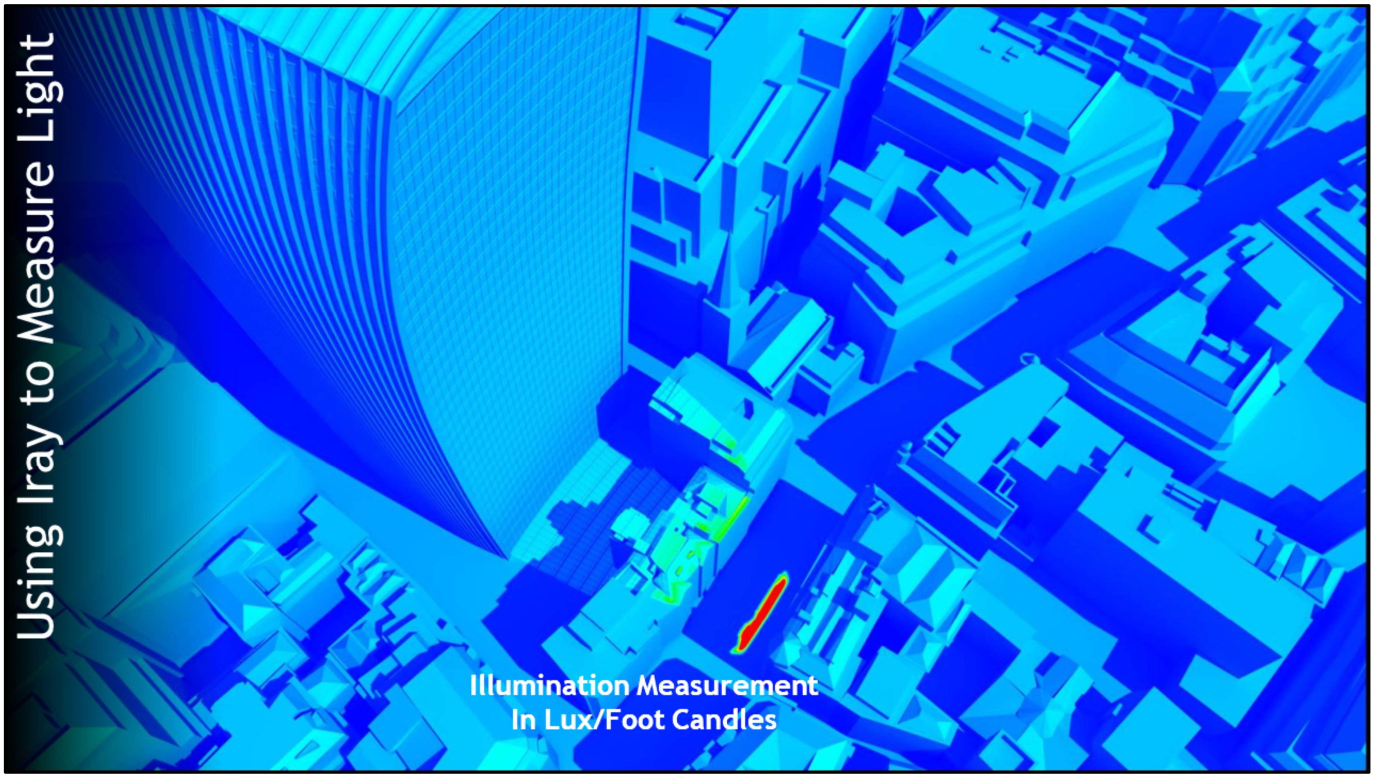


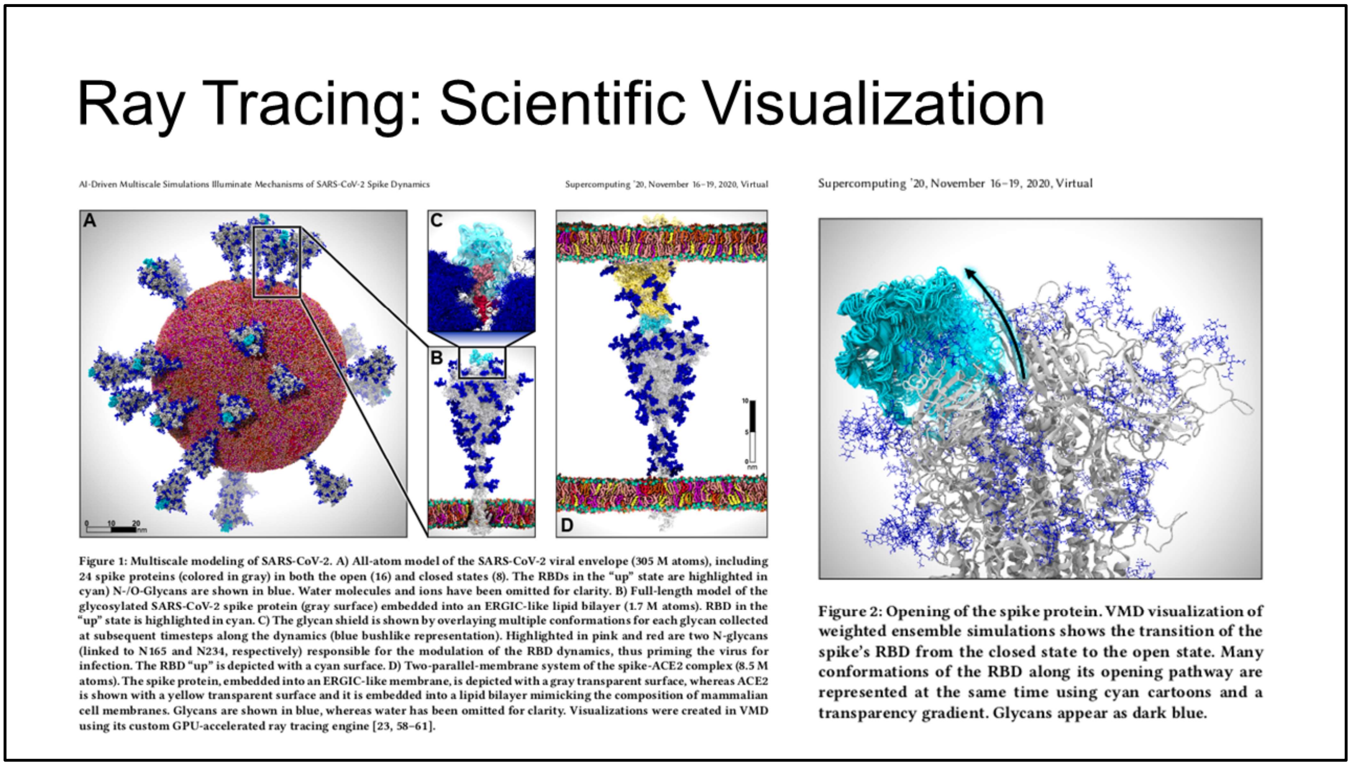
The “Walky Talky”. A Jaguar was melted, among others.

Typical story: https://www.independent.co.uk/news/uk/home-news/walkie-talkie-skyscraper-to-be-fitted-with-permanent-sunshade-after-it-melted-cars-9379037.html



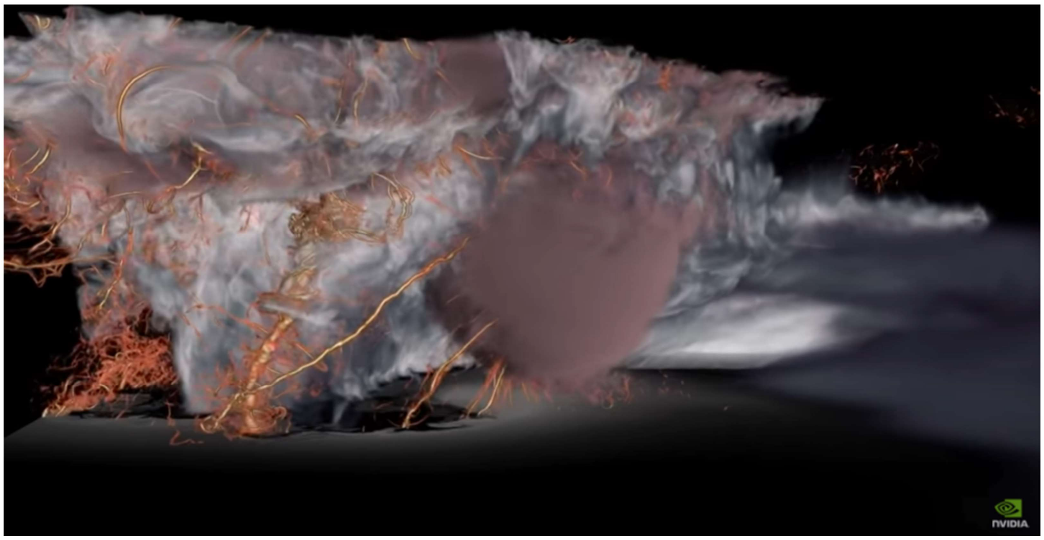
The curved building focuses the sun on a spot, making for a “burn zone” for cars, etc. – melted a Jaguar
https://www.solidsmack.com/cad/nvidia-melts-jaguar-cars-with-death-rays-to-show-off-rendering-power/


Using Iray to Measure Light
Illumination Measurement
In Lux/Foot Candles



1.7 million atoms, visible in real-time. Which, really, is nothing. ☺ Sphereflake was 48 million, recall.

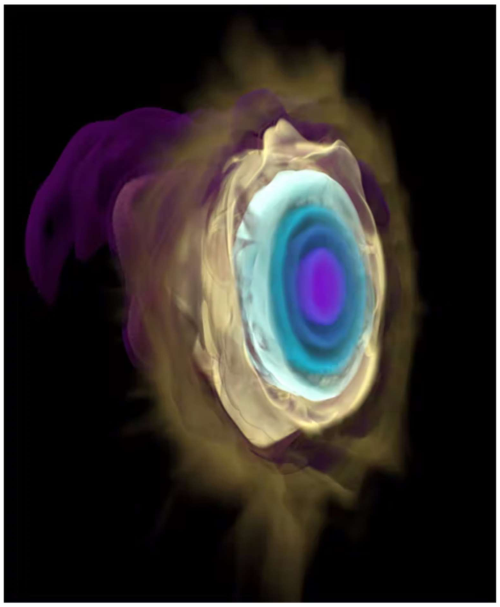
Ray Tracing: Scientific Visualization

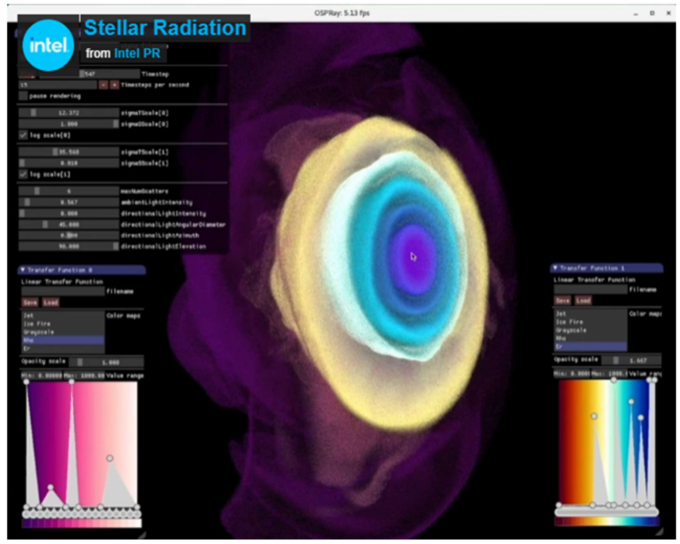


Supercell thunderstorm visualization, University of Wisconsin

Ray Marching
https://www.youtube.com/watch?v=SonfENaSesw&t=198s – done with Omniverse

Ray Tracing: Scientific Visualization

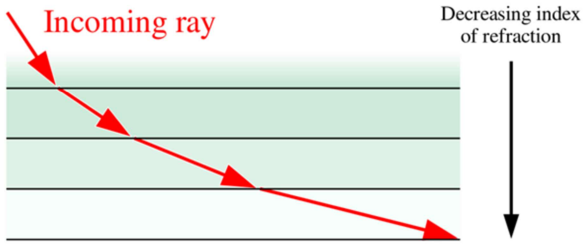




Stellar radiation vis. by G.P. Johnson & J. Insley, Argonne National Lab

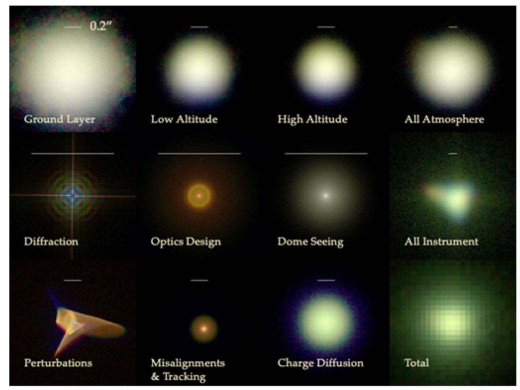
Ray Marching
https://www.electronicdesign.com/industrial-automation/article/21131013/ray-tracing-one-size-does-not-fit-all

Ray Tracing: Telescope Design



Light bends as it moves through the Earth's atmosphere

PhoSim simulation of how light interacts with an imaging system

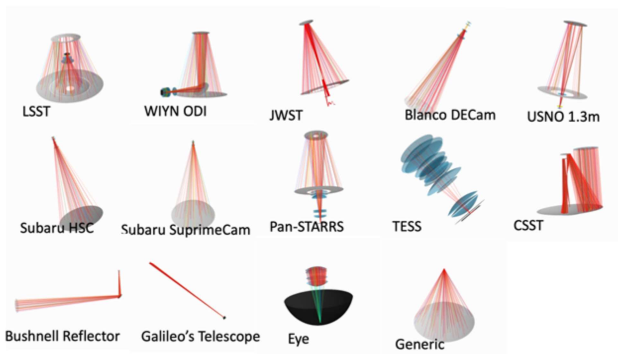


More Ray Marching

https://en.wikipedia.org/wiki/Ray_tracing_(physics)

https://bitbucket.org/phosim/phosim_release/wiki/Home

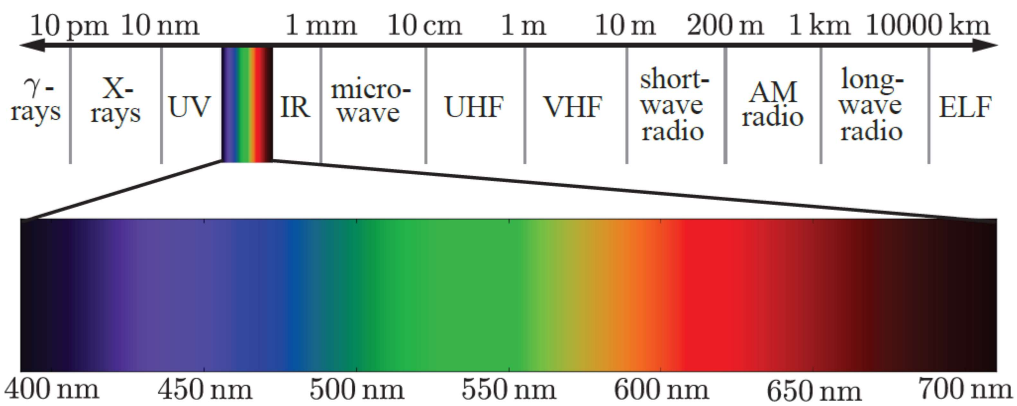
Ray Tracing: Telescope Design



Vera C. Rubin Observatory, currently under construction in Chile

https://en.wikipedia.org/wiki/Vera_C._Rubin_Observatory
https://bitbucket.org/phosim/phosim_release/wiki/Home

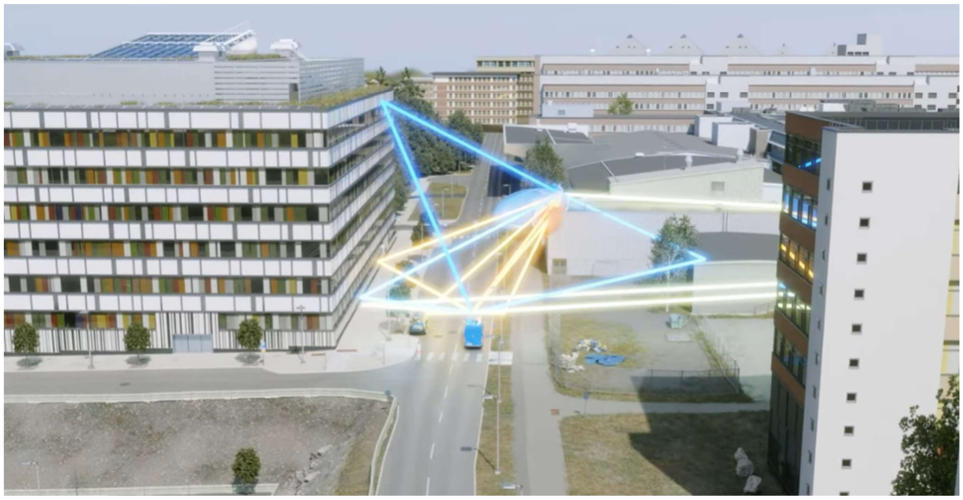
Photons



Leftmost: Gamma rays have the smallest wavelengths and the most energy of any wave in the electromagnetic spectrum. They are produced by the hottest and most energetic objects in the universe, such as neutron stars and pulsars, supernova explosions, and regions around black holes.

ELF – Extremely low frequency (ELF) fields includes alternating current (AC) fields and other electromagnetic, non-ionizing radiation from 1 Hz to 300 Hz. ELF fields at 60 Hz are produced by power lines, electrical wiring, and electrical equipment. Some epidemiological studies have suggested increased cancer risk associated with magnetic field exposures near electric power lines.

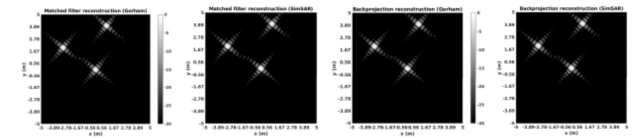
Ray Tracing: 5G Propagation Simulation



Digital Twin by Ericsson

https://www.youtube.com/watch?v=yTbUSXJ8M-8

Ray Tracing: Synthetic Aperture Radar



(a) MATLAB (b) OptiX simulator (c) MATLAB (d) OptiX simulator

Figure 6: (a,b) show MATLAB[27] and OptiX simulator SAR image reconstructions for 3 point targets using the matched filter algorithm. (c,d) show MATLAB and OptiX simulator SAR image reconstructions for 3 point targets using back projection algorithm.

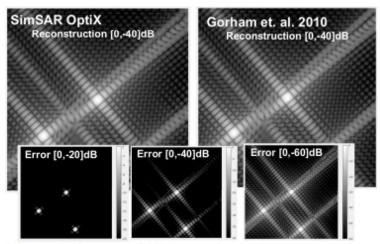


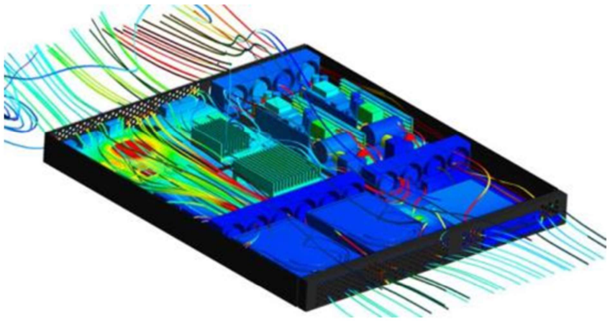
Figure 7: Numerical validation by forming a SAR image of a 3-point test target located having 3D locations (0,0,0), (-3,2,0), and (1,4,0). (left) shows OptiX accelerated simulation results. (right) shows results from Gorham et. a l.[27] Bottom figures show three images highlighting small (<0.1 percent) numerical differences in the reconstructed SAR image values.

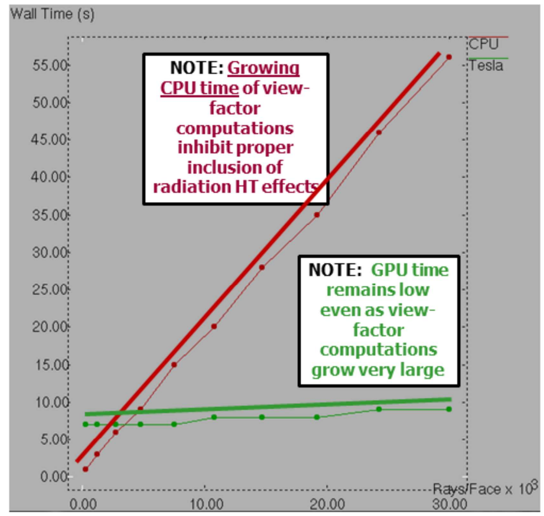
From Andrew R. Willis, Md Sajjad Hossain and Jamie Godwin, "Hardware-Accelerated SAR Simulation with NVIDIA-RTX Technology"

Used in target recognition, mapping, surveillance, oceanography, geology, forestry (biomass, deforestation), disaster monitoring (volcano eruptions, oil spills, flooding), and infrastructure tracking (urban growth, structure mapping).

https://arxiv.org/pdf/2005.09736.pdf

Ray Tracing: Simulation of Heat Transfer



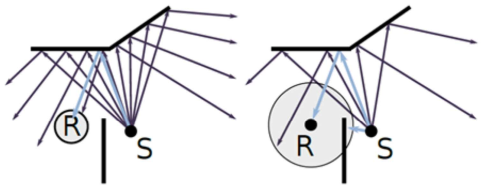


ANSYS FLUENT

OptiX in 2016, before ray tracing enhanced GPUs

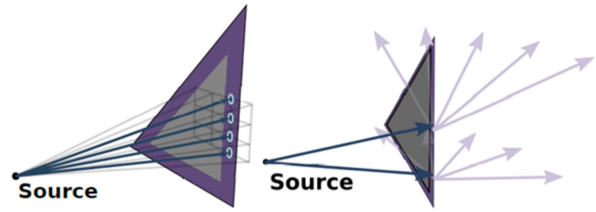
From 2016

Ray Tracing: Audio Simulation



(a) (b)

Fig. 3: Sample-based visibility: *Visibility rays are traced from source S into the scene. Paths that strike receiver R are then validated. (a) A small receiver requires dense visibility sampling to find the propagation path. (b) Using a larger receiver allows sparse sampling resulting in fewer visibility tests, however more validation tests are need to remove invalid path sequences.*



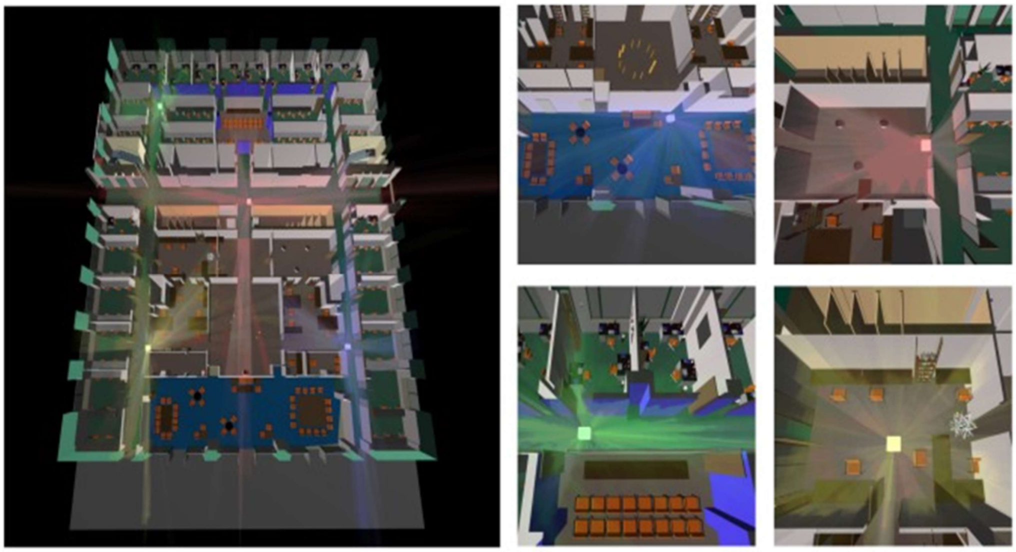
(a) (b)

Fig. 10: Edge Diffraction*: (a) Rays near the edge are detected for resampling. (b) Diffraction samples are cast through the shadow region, bounded by the adjacent triangle.*

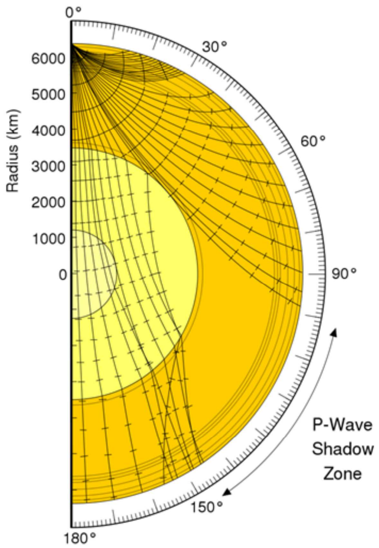
from "Guided Multiview Ray Tracing for Fast Auralization" by Micah Taylor, Anish Chandak, Qi Mo, Christian Lauterbach, Carl Schissler, and Dinesh Manocha, 2012

Ray Tracing: Audio Simulation

Dolby: Sound propagation



Ray Tracing: Seismology



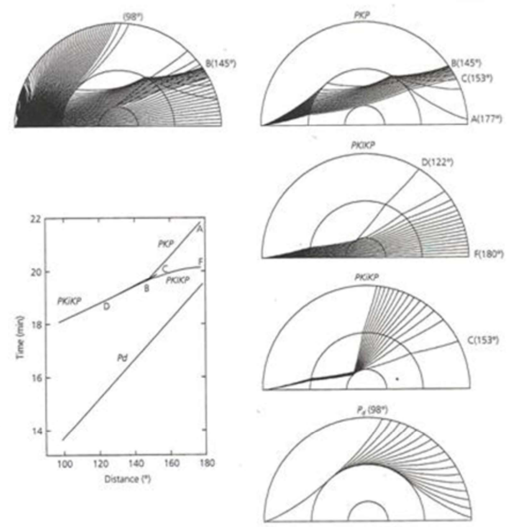
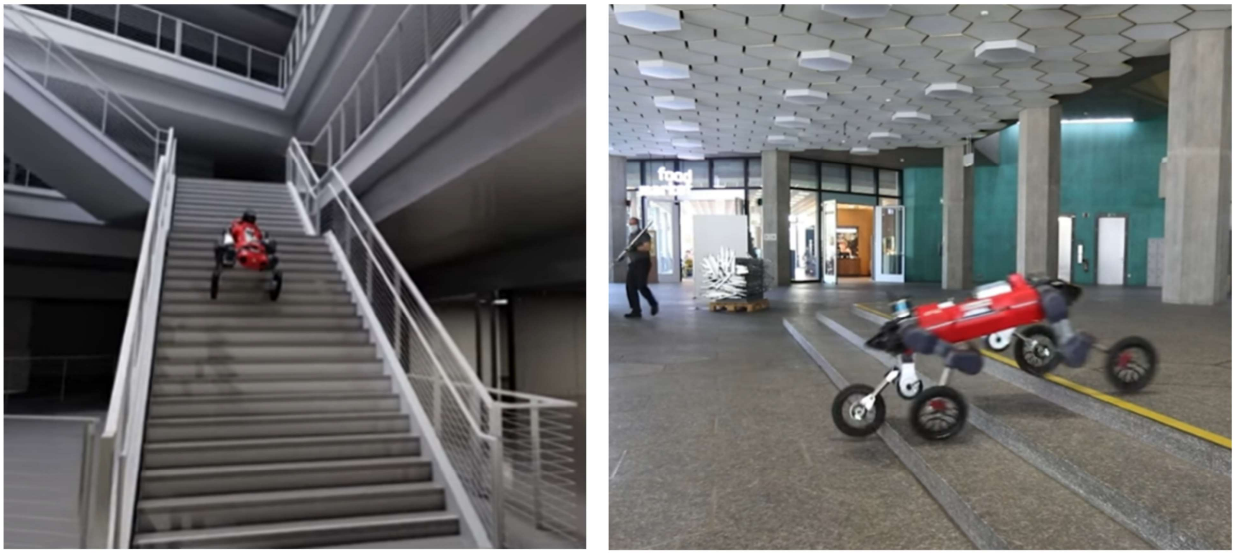


Fig. 3.5-7 Ray paths and travel times for major core phases, computed for earth model PREM. *Top left*: Paths for direct rays (i.e., excluding reflections and diffractions). *Right*: Ray paths for four other phases: *PKP* passes through the outer core, *PKIKP* also penetrates the inner core, *PKiKP* reflects from the boundary between the outer and inner cores, and P_d (also called P_{diff}) diffracts along the core–mantle boundary. *Lower left*: Travel time curves for these phases. Points on the earth's surface are labeled with their distances in degrees.

https://en.wikipedia.org/wiki/Ray_tracing_(physics)
https://web.ics.purdue.edu/~nowack/geos557/lecture14c-dir/lecture14c.htm

Ray Tracing: Training Robots



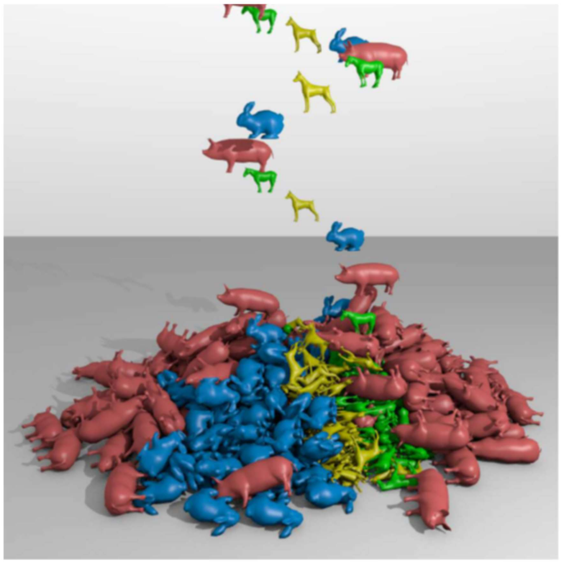
Anymal, a robot dog, trained by ETH Zurich and Swiss-Mile with Isaac Sim. Left: virtual world; right: real world

https://developer.nvidia.com/isaac-sim
https://www.youtube.com/watch?v=VW-dOMBFj7o

Ray Tracing: Simulation for Self-Driving Cars

https://www.nvidia.com/en-us/self-driving-cars/simulation/
https://youtu.be/UoPXzzK_g1Q

Ray Tracing: Collision Detection and Response



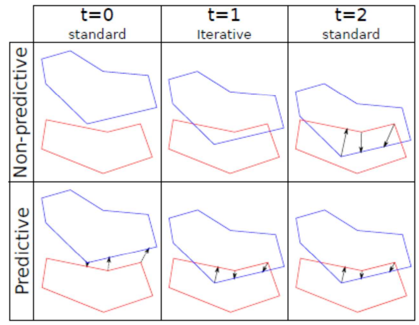


Figure 3: *Comparison of predictive and non-predictive collision detection, with predictive rays the collision is detected earlier resulting in less interpenetration.*

from *Ray-Traced Collision Detection: Interpenetration Control and Multi-GPU Performance*, by François Lehericey, Valérie Gouranton, Bruno Arnaldi

Source: Ray Tracing Gems

~~*“There is an old joke that goes, ‘Ray tracing is the technology of the future, and it always will be!’”*~~

– David Kirk, March 2008

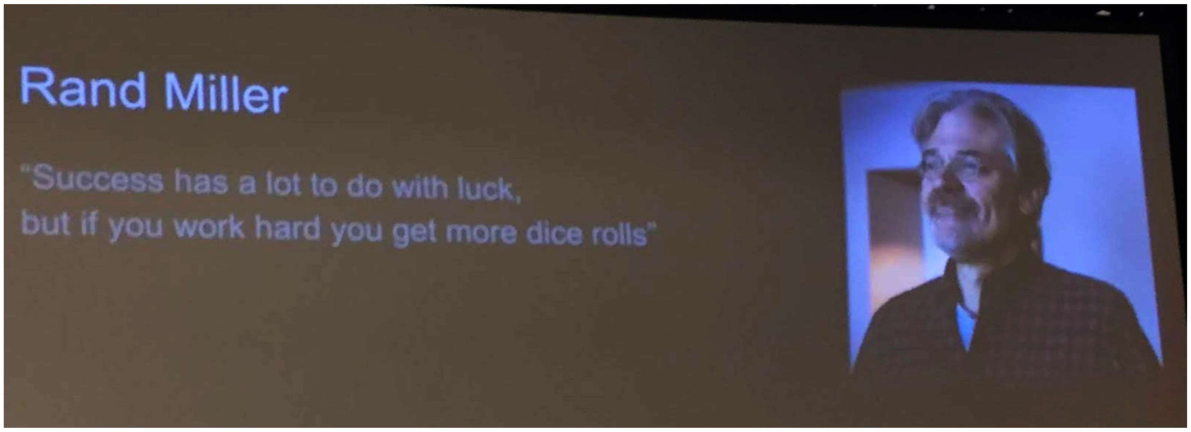
“Ray tracing is simple enough to fit on a business card, yet complicated enough to consume an entire career.”

– Steven Parker, May 2019

Can't use this joke any more. Ray tracing is a sea-change, it's like shadow mapping added to rasterization, times 3. At the same time David was saying this joke, he was starting to look into how to accelerate ray tracing by using dedicated hardware.

Circa 2008 - http://www.pcper.com/article.php?aid=530 – seems to be the first mention on the internet, actually

Seen at Pax East 2019



Rand Miller is the co-creator of the classic videogame "Myst"

Pro-ish Tips on Career

Do that extra thing, about something you enjoy and working on:

- **Make a website for yourself**; sites.google.com if nothing else, or github.com. Don't be stuck with your school's/work's URL.
- Volunteer at any conference, for any position – help is always needed, and you meet people. OBS Studio! Some conferences: HPG, I3D, EGSR, SIGGRAPH, CVPR.
- Sincerely ask questions of authors if you are interested.
- Blog or write articles on things you know or things you've tried. (And consider http://jcgt.org or at least arXiv.org)
- Help review papers in an area you know well. Say "yes."
- Work on some (usually public, open source) project you like, in a team or on your own. Get a different perspective. Make and share Shadertoys.
- Learning doesn't end once you've graduated. So many options… online classes are good.
- If you enjoy it: Write a book. Make a movie. Create a game. All quite doable!

Pet peeve: requesting to connect on LinkedIn without adding a note.

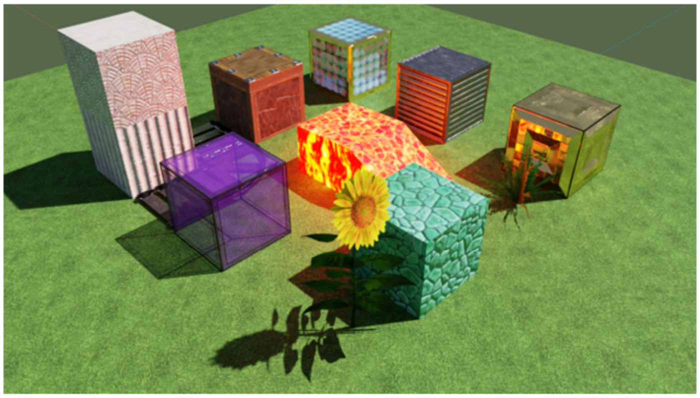
See my site about why you want a URL: http://www.realtimerendering.com/blog/moving-targets-and-why-theyre-bad/

People think you know something if you write a book. And, dozens of dollars to be made!

My gosh, never ever randomly ask for a connection to someone on LinkedIn without an introductory note.

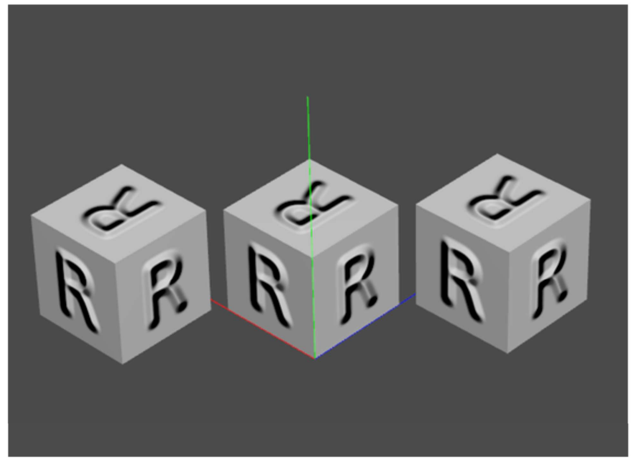
Side effect: writing about something makes you learn it well enough to write about it (and not look dumb).

Simple Example: McUsd



Simple scene with
cutouts, transparency,
reflection, emission

Test Normal Map bias and scaling



https://github.com/erich666/McUsd – just two models, basically. It sometimes doesn't take a huge amount of effort to make a meaningful contribution to the community.

At Work

- Ask questions when you don't know. Get over looking ignorant – everyone is ignorant about 99.99% of everything.
- Solo is fine, failing solo is fine, but failing with others is less likely – get help. A diverse set of views and skills is a win.
- Enjoy the ride!

"We are all experts in our own little niches." – Alex Trebek

"Let someone else praise you, and not your own mouth; an outsider, and not your own lips." – Proverbs 27:2
Pithier: "Self-praise is no recommendation."

I'm not religious, but the old testament nailed it there.

See http://www.realtimerendering.com/raytracing.html#books

Find these and more here:

http://bit.ly/rtrtinfo and http://raytracinggems.com

Source: Eric Haines, taken at NVIDIA booth

http://bit.ly/rtrtinfo and http://raytracinggems.com