Practical Techniques for Ray Tracing in Games

Gareth Morgan (Imagination Technologies)

Aras Pranckevičius (Unity Technologies)

March, 2014

What Ray Tracing is not!

- **Myth: Ray Tracing is only for photorealistic / physically accurate rendering**
- **Myth: Ray Tracing is incompatible with rasterized graphics**
- **Myth: Ray Tracing is a less efficient way to render a given number of pixels**

What Ray Tracing is!

- ~~Myth: Ray Tracing is only for photorealistic / physically accurate rendering~~

Truth: It is just a tool. It can be used for a range of purposes

- ~~Myth: Ray Tracing is incompatible with rasterized graphics~~

Truth: It can be used in a rasterized game engine for certain effects

- ~~Myth: Ray Tracing is a less efficient way to render a given number of pixels~~

Truth: For some effects, it is computationally cheaper to ray trace

What is Ray Tracing?

Ray tracing is the ability for the shading for one object to be aware of the geometry of other objects.

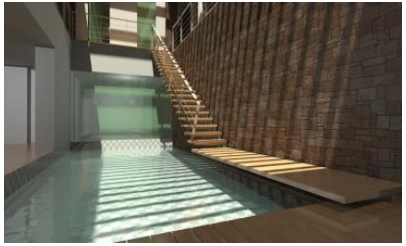
So what can you do with it?

Shadows

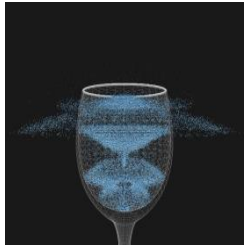
Reflections

Refractions

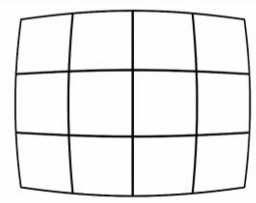
Ambient Occlusion

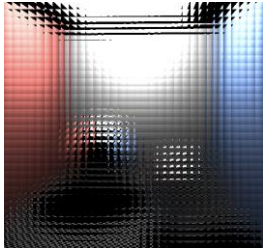


Global Illumination

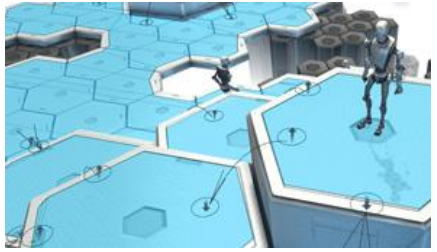


Physics & Collision Detection





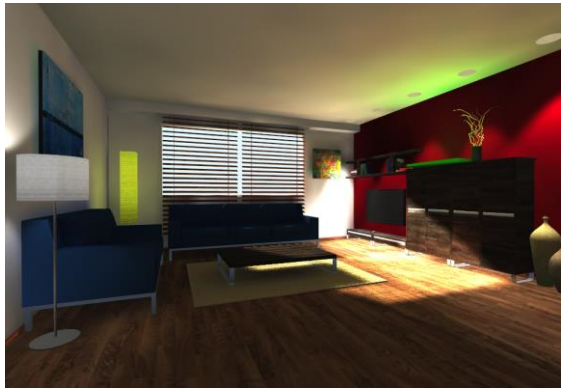
Virtual Reality
Lens correction, Ultra-low latency rendering, Lenticular Displays



A.I. / Line of Sight

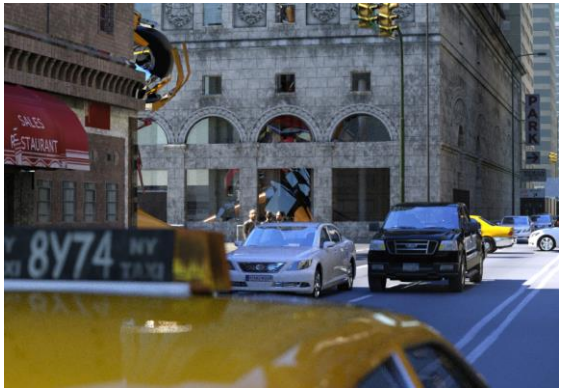
How do you add ray tracing to your game?

Many options!



In Pre-Baking,
eg. Unity 5 Editor

Hybrid Game Engine



Ray Trace Everything,
eg. Brigade & Arauna 2

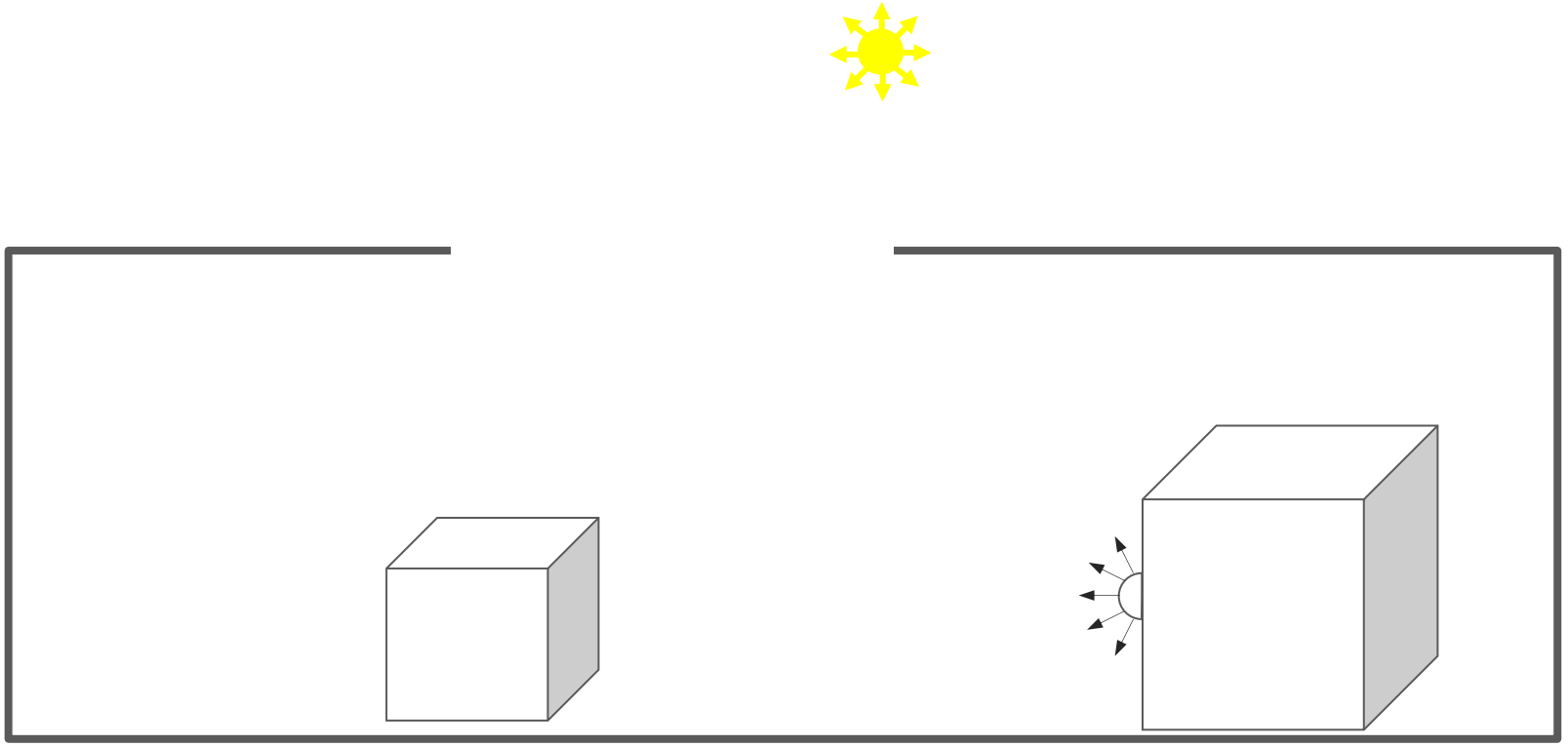
How do you add ray tracing to your game?

Light Maps in Unity Editor 5

Ray Trace Everything,
eg. Brigade & Arauna 2

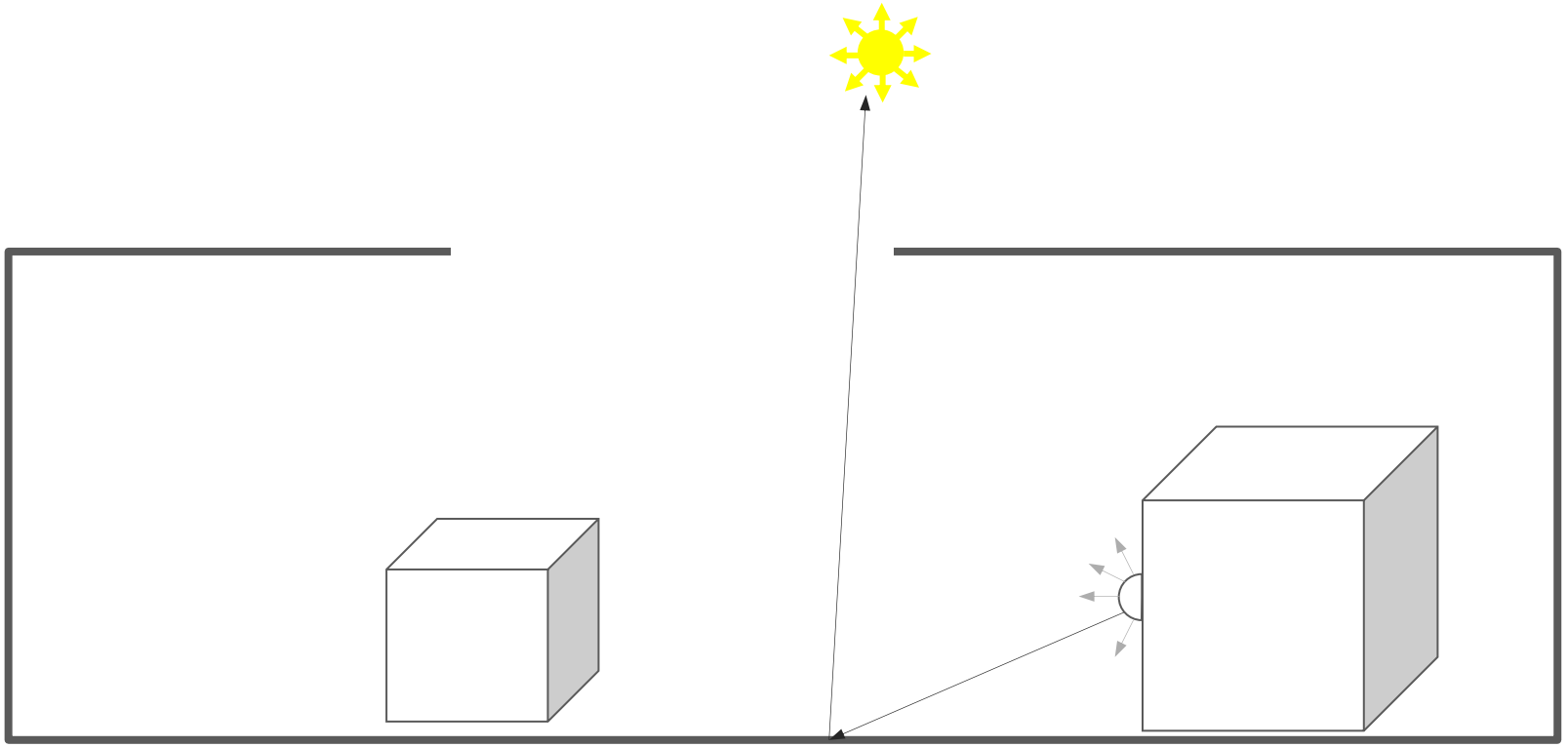
How it works

Ray tracing allows the behavior of light to be simulated



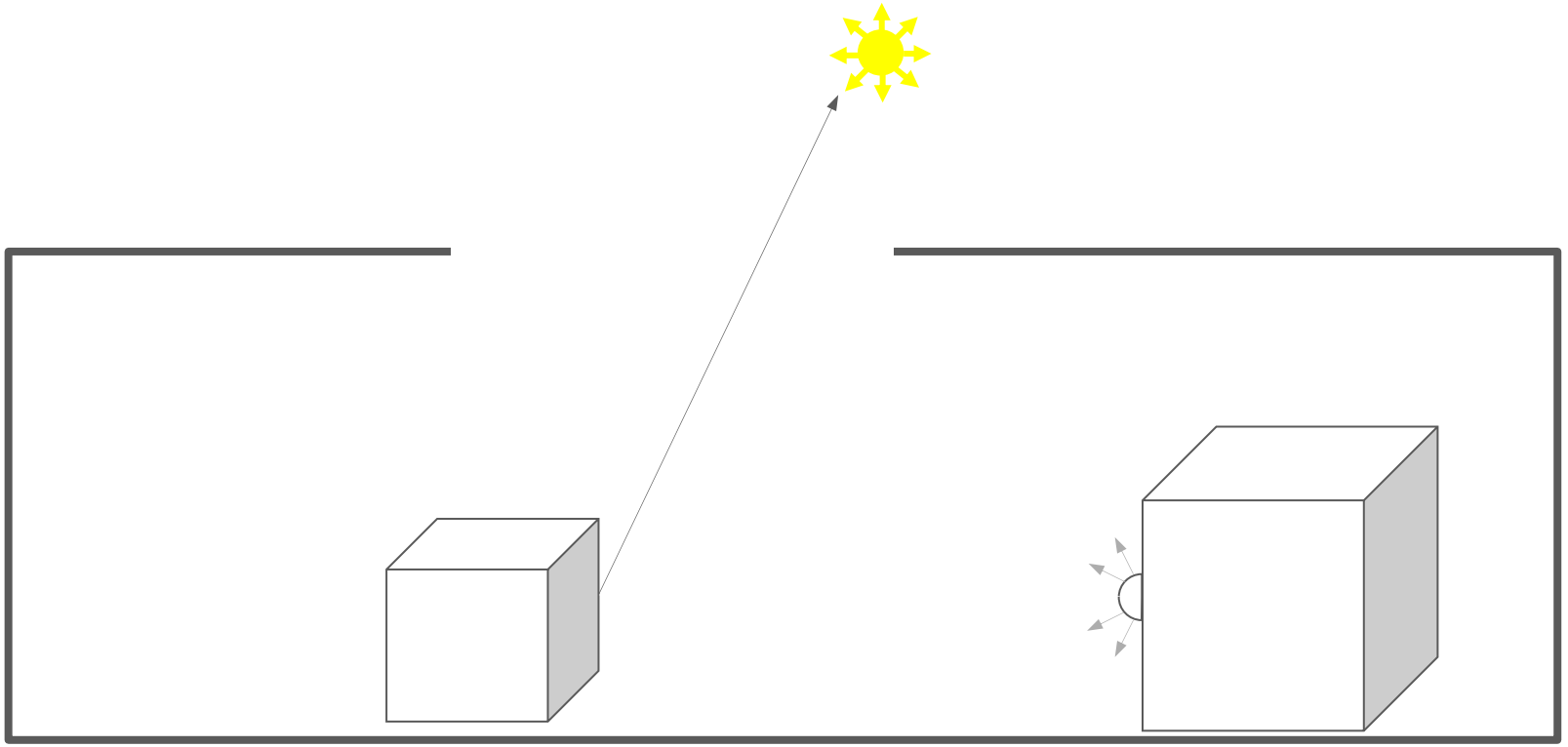
How it works

Ray tracing allows the behavior of light to be simulated



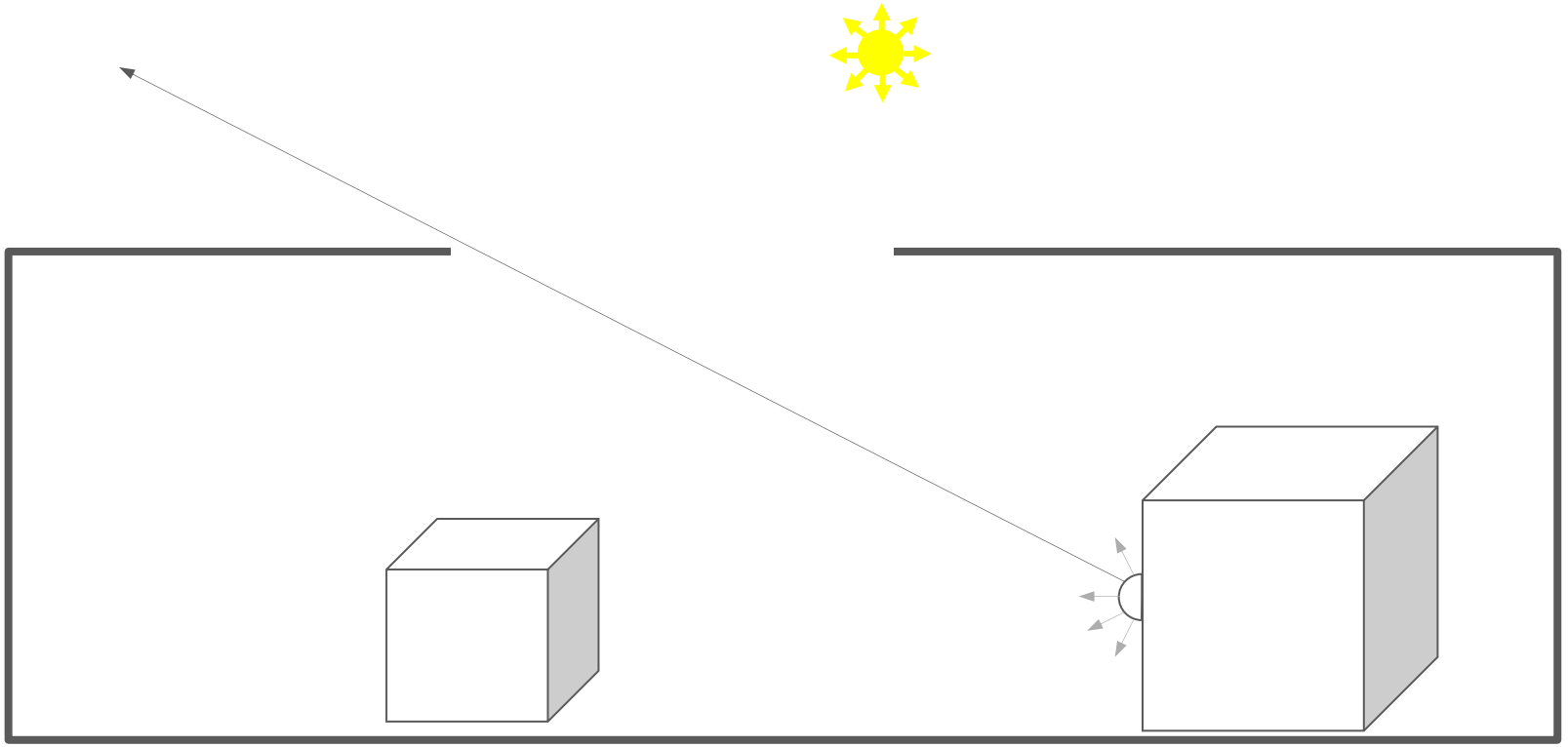
How it works

Ray tracing allows the behavior of light to be simulated



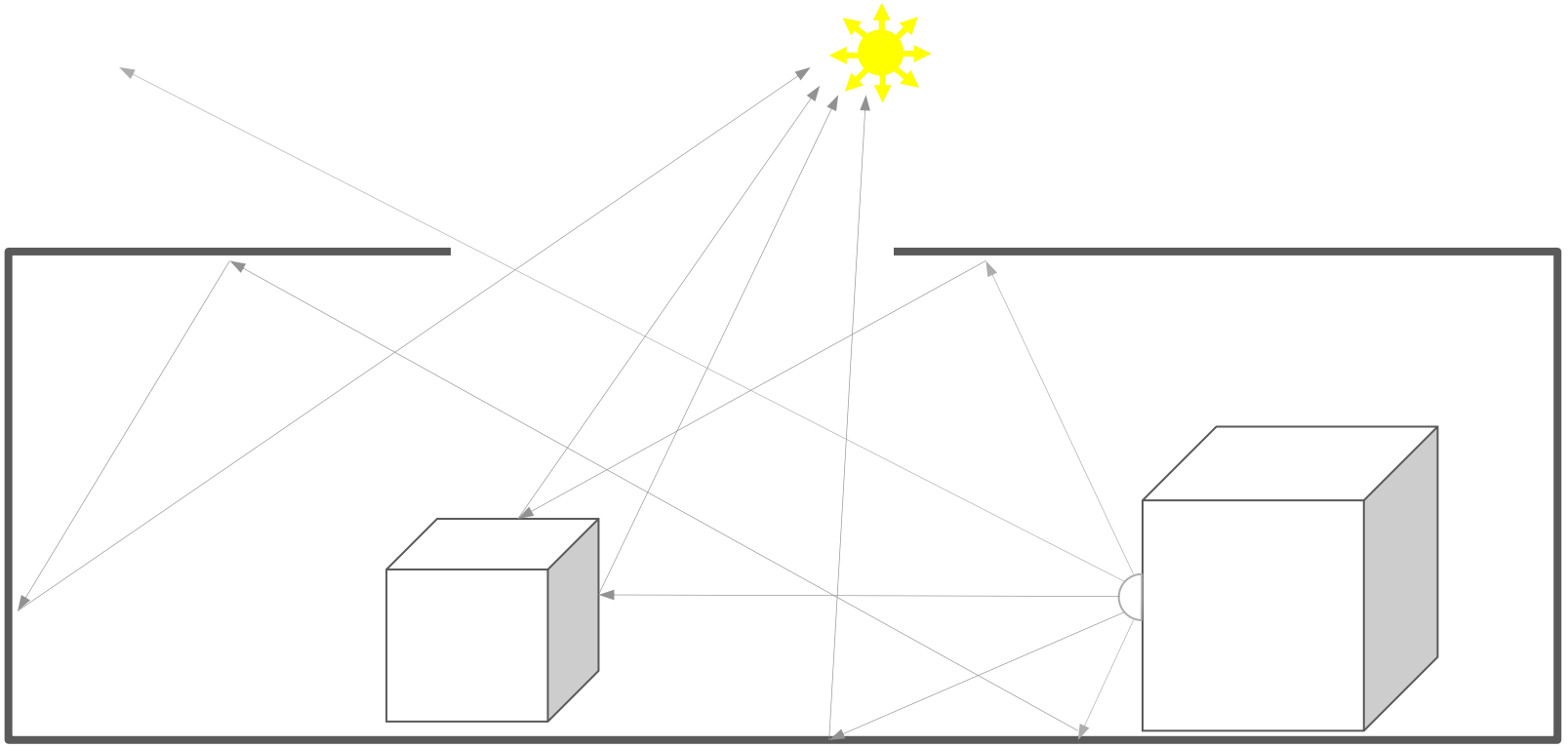
How it works

Ray tracing allows the behavior of light to be simulated

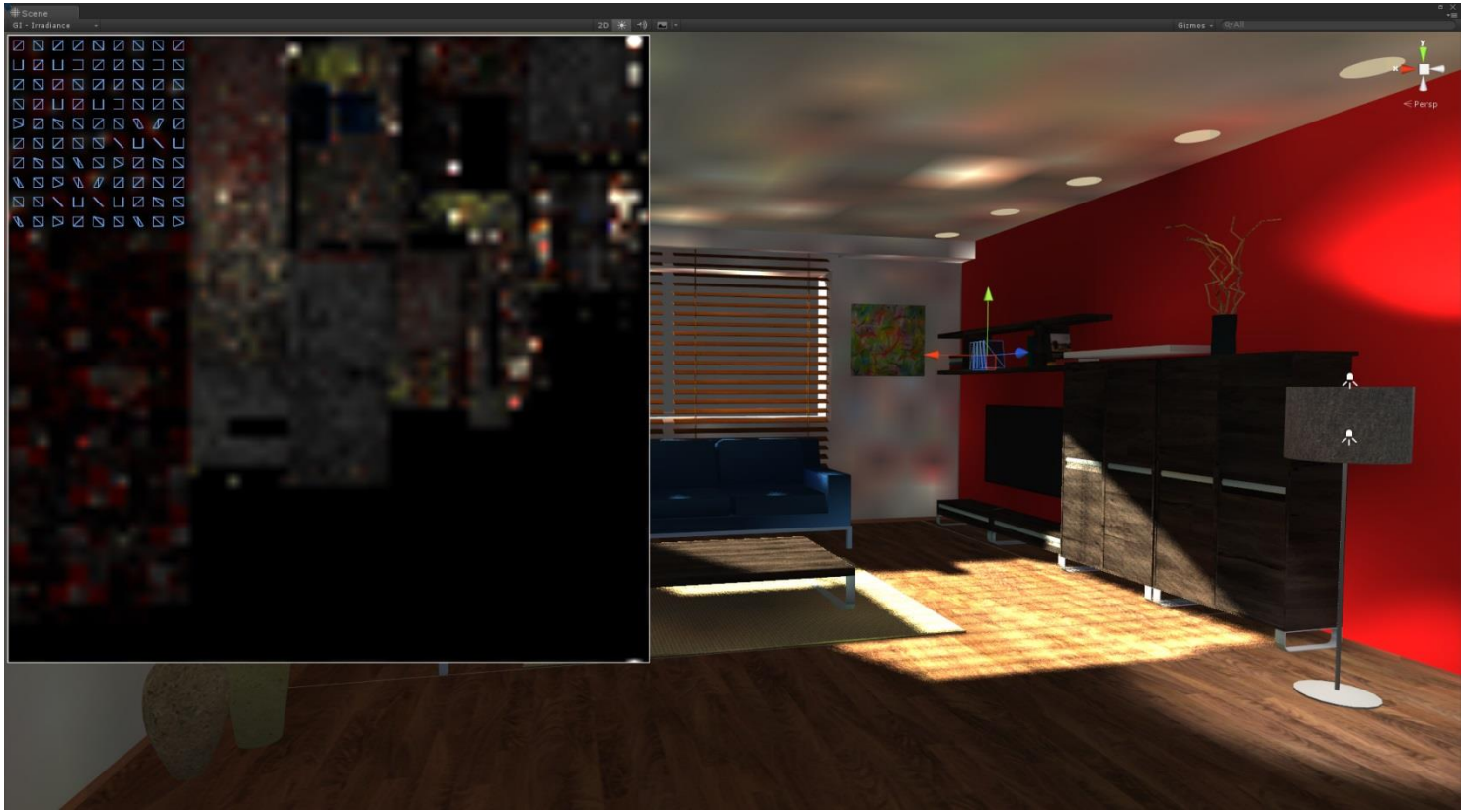


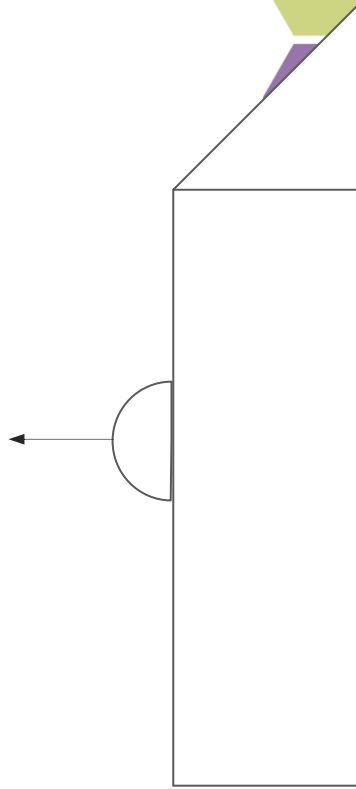
How it works

Ray tracing allows the behavior of light to be simulated

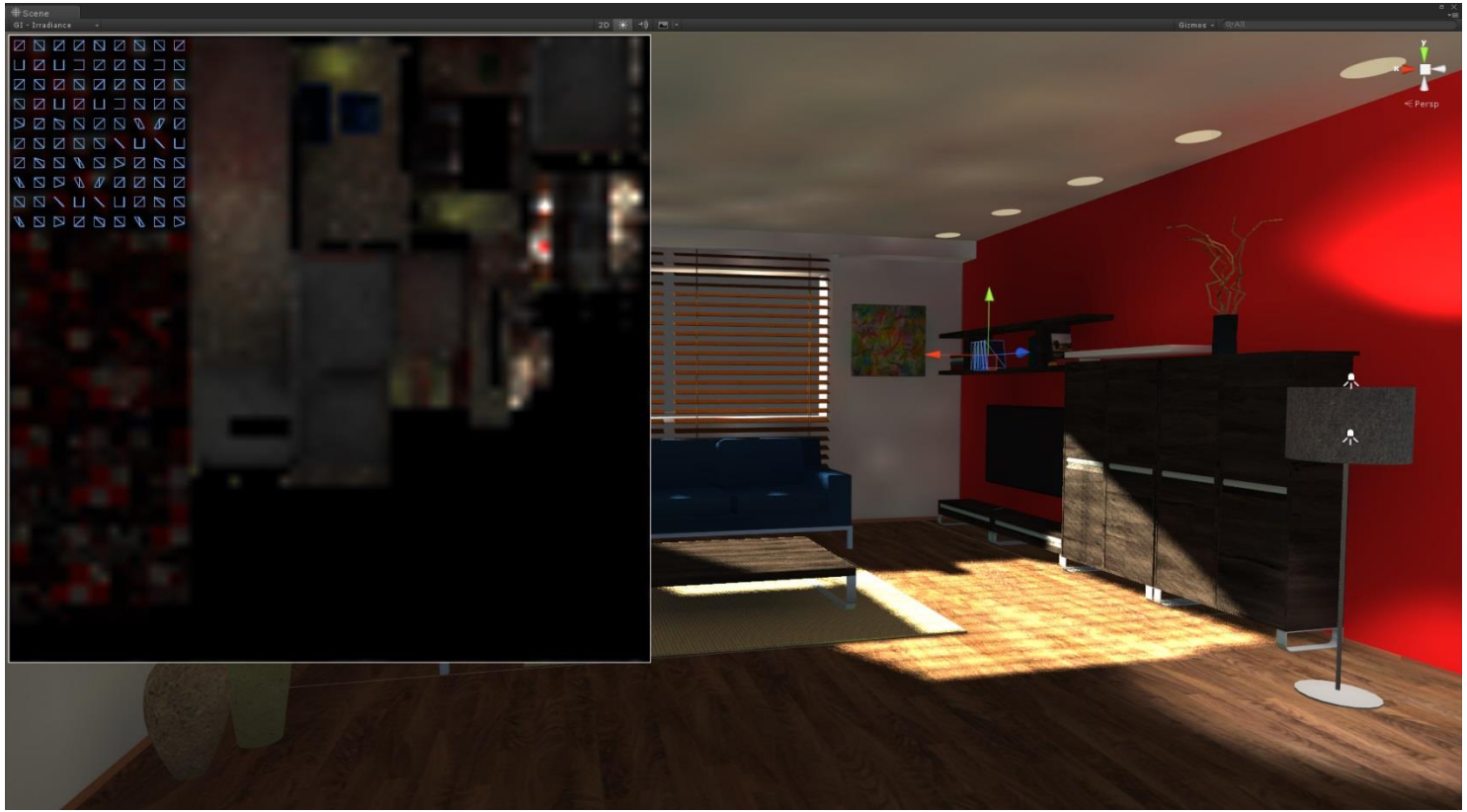


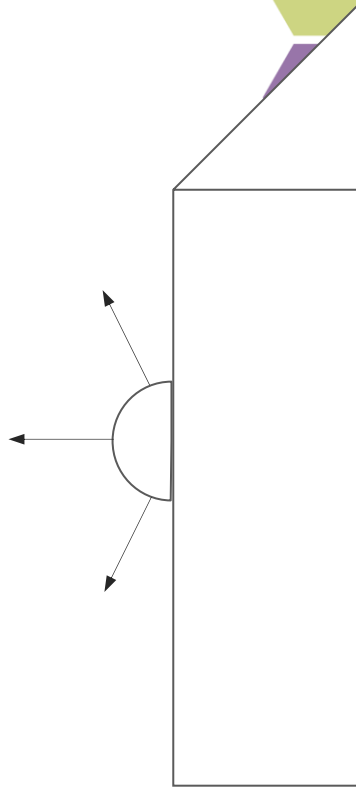
Progressive Refinement



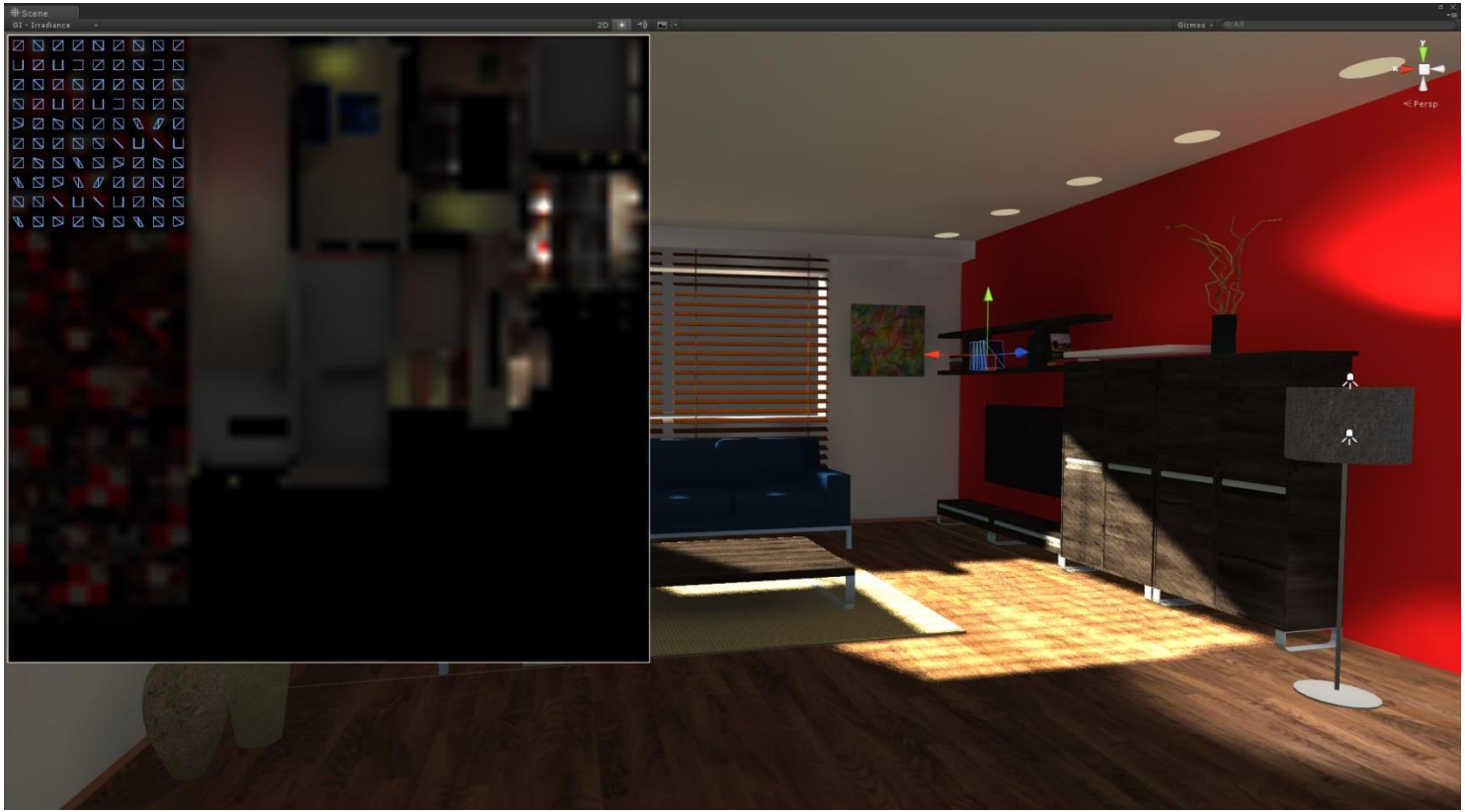


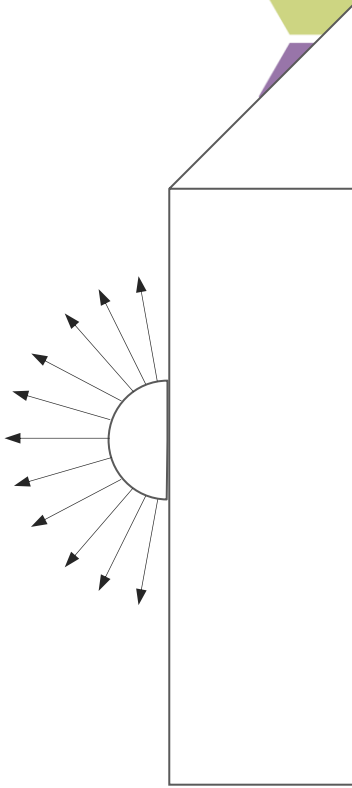
Progressive Refinement (a moment later…)





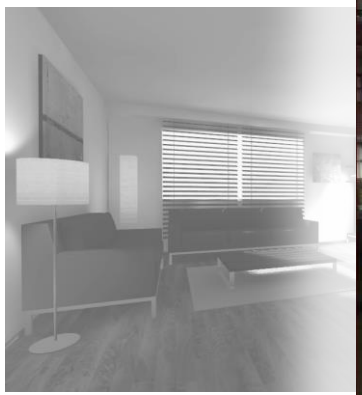
Progressive Refinement (another moment later…)





Demo: Interactive Lightmap Preview In Unity Editor 5

How do you add ray tracing to your game?



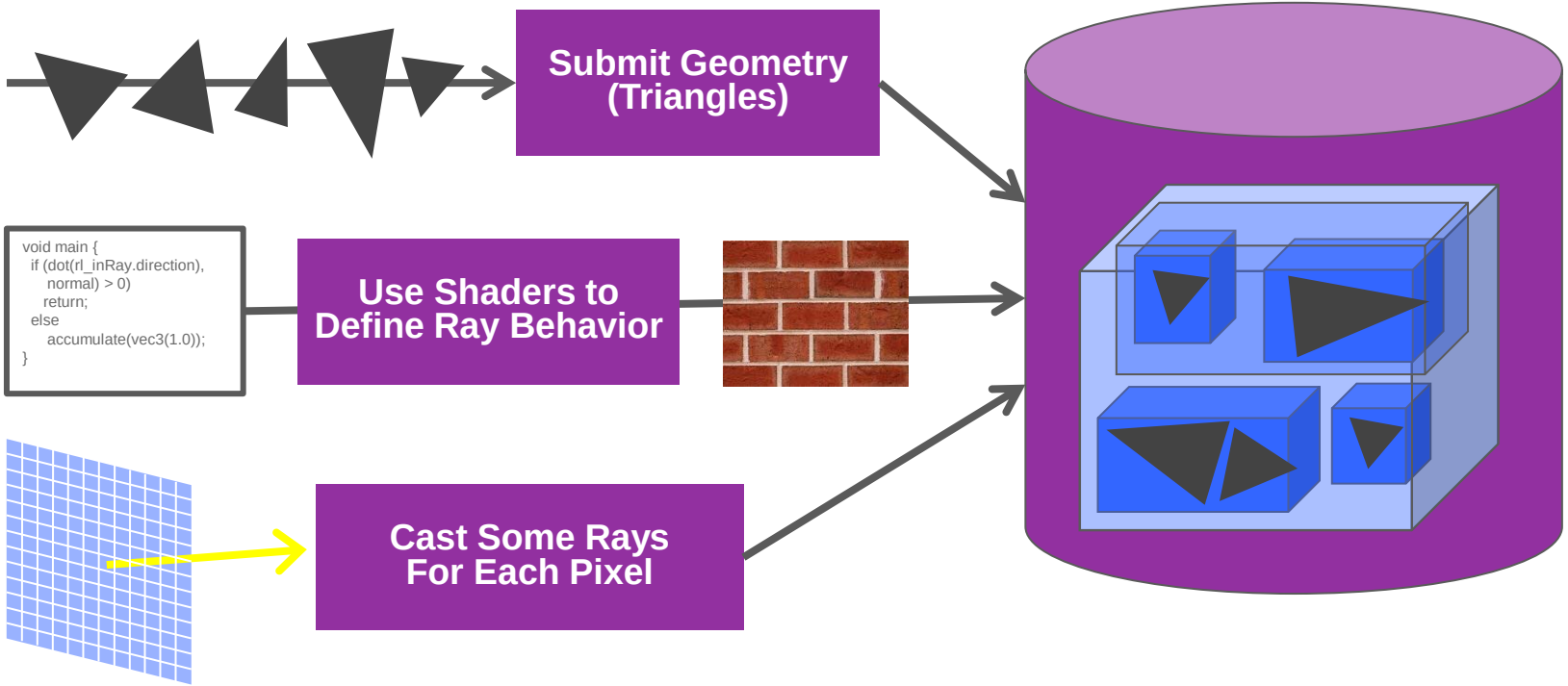
In Pre-Baki
eg. Unity 5 E

Trace Everything,
igade & Arauna 2

Hybrid Game Engine

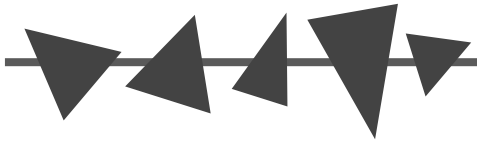
Ray Tracing details

World space scene intersection



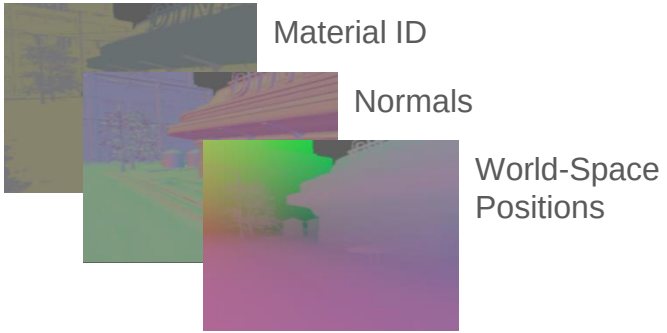
Modern raster-based game engine

Deferred shading used in most modern games engines



Render the scene into a deep frame buffer (G-Buffer)

G-Buffer



Material ID

Normals

World-Space Positions

Compute Lighting on each G-Buffer pixel (No Overdraw)

Lighting Buffer(s)

Lighting calculations executed in screen space

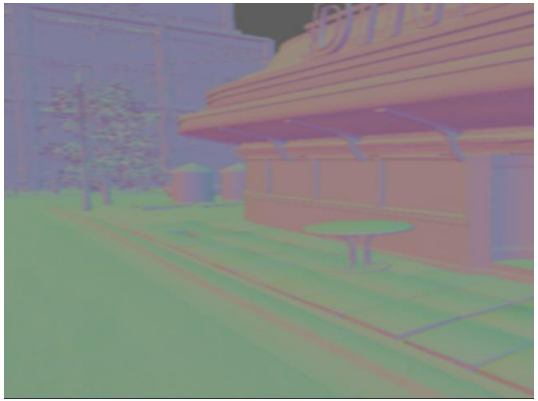
Assemble the final frame and perform screen-space effects

Final Result Frame

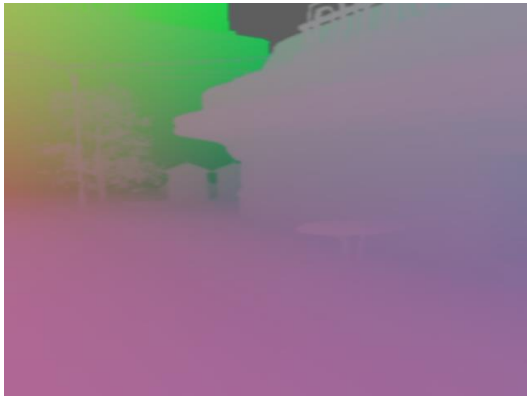
G-Buffer has everything you need to ray trace

G-buffer contents are in world space!

G-Buffer



Normals



World-Space Positions
or Depth Buffer

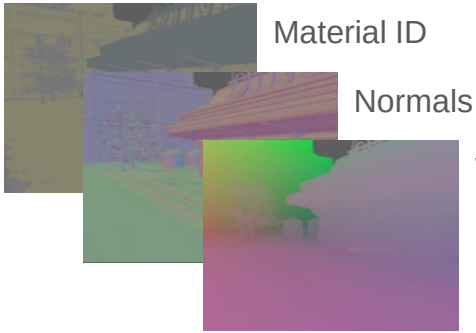
Material IDs

Hybrid Rendering

Use the G-Buffer to set up your rays

Render the scene into a deep frame buffer (G-Buffer)

G-Buffer



Material ID

Normals

World-Space Positions

Trace rays to determine the lighting

Lighting Buffer

Assemble the final frame and perform screen-space effects

Final Result Frame

Ray Traced Effects, Side-By-Side

Ray Traced Effects, Side-By-Side

Shadows

We can ray trace, now what?

- **Shadows are an extremely important part of accurate 3D rendering.**
- **Light from environment is occluded by other objects before it reaches surface**
- **Very well suited to ray tracing**

Shadow maps

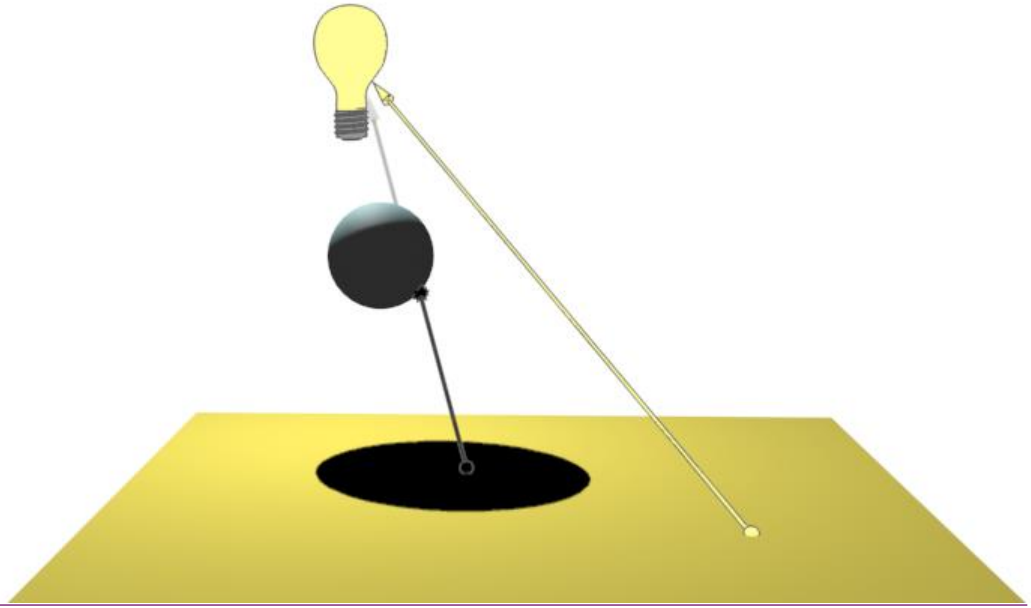
Shadows have been rasterized for a long time

- **Shadows used in games for a long time**
- **Most modern games use Shadow maps**
- **Many problems**
  - Resolution issues
  - Performance problems
  - Real life penumbra hard to simulate using filtering

Ray Traced Shadows

Shadows are easy when you can query world space scene database!

- **Shoot a ray between surface and the light:**
  - If the ray hits *anything* then do nothing (region is shadowed and unlit)
  - If the ray reaches the light without hitting anything then illuminate that pixel
- **Then you're done!**



Demo: Accurate Shadows

Soft shadows

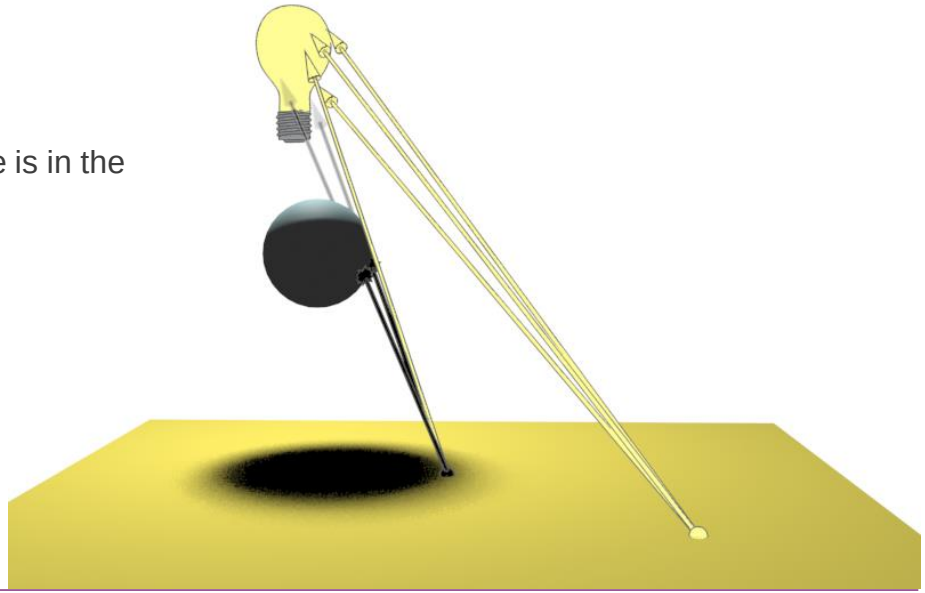
Shadows with soft edges

- **Shadows have sharp edges at noon on a clear day…**
- **But many situations cause soft shadows**
- **The region where a shadow transitions between fully and partially lit is called the penumbra.**
- **In the real world**
  - Light sources are not infinitely small points
  - Scattering occurs between the light source and surface

Ray Traced Soft Shadows

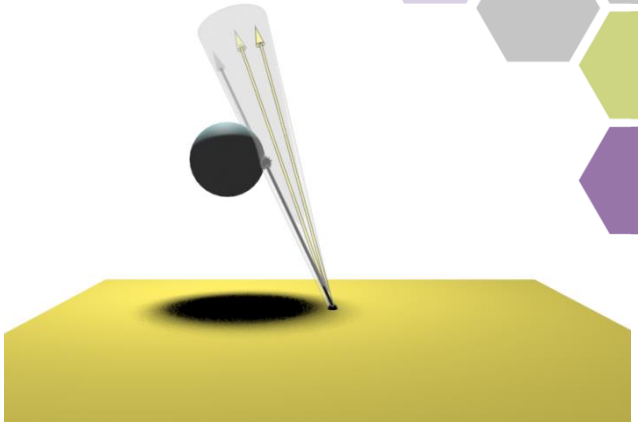
Penumbra rendering requires multiple rays per pixel

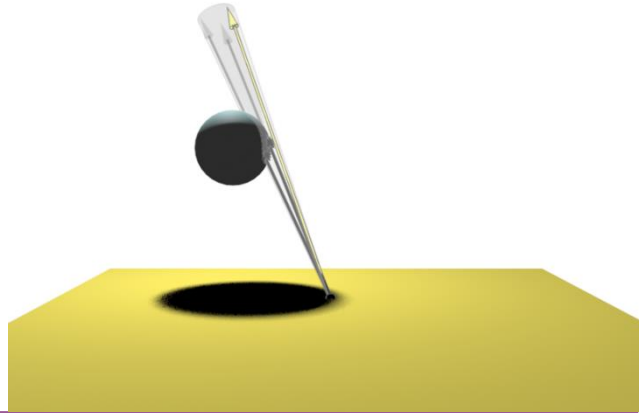
- **Instead of shooting a single ray for every surface point shoot several rays.**
- **Behaviour of each ray identical to hard shadow case,**
- **Average the results of all the rays for each pixel:**
  - If all rays are occluded surface is fully shadowed
  - If all rays reach the light source surface is fully lit
  - If some rays are occluded and some reach light the surface is in the penumbra region.



Choosing ray direction

- **If the light source in an area light, distribute the rays over the cross section of the light source visible from the surface**
- **To approximate daylight using an infinitely distant, directional light choose a cone of rays from the surface:**
  - To represent a perfectly clear day the solid angle of the cone is zero
  - To represent cloudier daylight solid angle becomes larger
- **We are estimating incoming light reaching the surface point**
- **To get good estimate, samples should evenly cover domain**

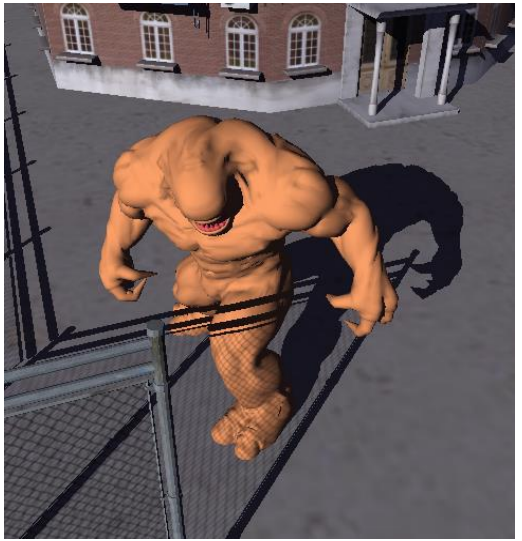




G-buffer continuity

Stop wasting rays!

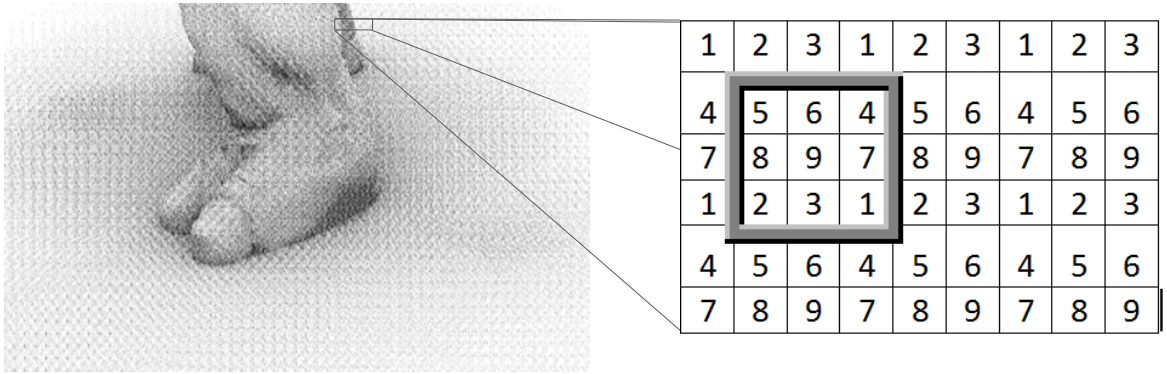
- A large number of rays are required to accurately sample a soft shadow.
- For most of the image the surface properties vary little from one pixel to its neighbors
- So, a ray sent from one pixel of the G-buffer will likely hit the same object as the same ray sent from a neighbouring pixel.
- Surely there is a way to use this fact to reduce the ray count but maintain visual accuracy?



Interleaved Sampling

Leverage the shadow ray data from neighboring pixels

- Tile a square 2D array of N^2 ray directions over the frame buffer
- Emit shadow rays based on grid
- The resulting image has the critical property that for any NxN region of the image the entire N^2 array of ray directions is represented.
- So use a box filter to remove noise from the image. Each output pixel is the average of N^2 neighbouring input pixels.
- Must handle discontinuity in the image.



Demo: Soft Shadows, Multiple Lights

Shadows recap

Raytraced shadows are better

- **Raytraced shadows….**
  - Easy to implement
  - Avoid the many artefacts of shadow maps
  - Are more efficient
  - Scale better to multiple lights

Reflections

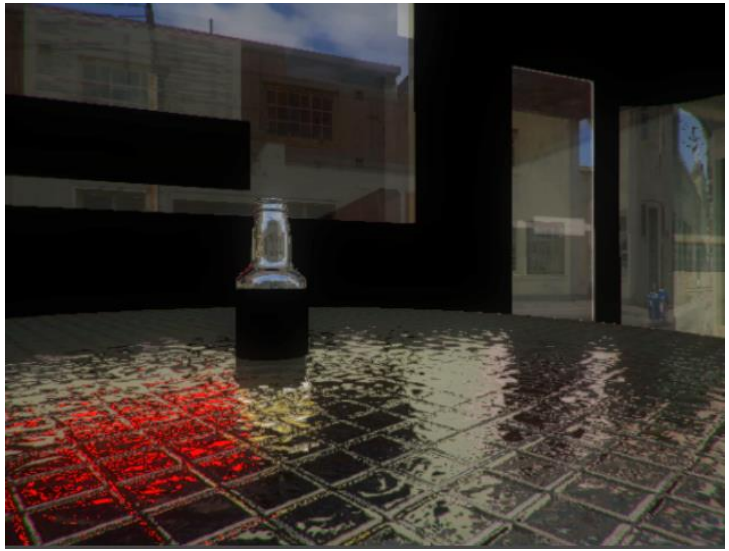
Reflections are more common than you think!

- When light hits a perfectly specular surface is it reflected at an angle the same as the incident angle.
- Basic physical law first codified by Euclid in 3rd century BC.
- In the real world reflective objects are common, not just chrome balls!

Raytraced reflections

Physically accurate

- Emit one extra ray from reflective surfaces, the direction of reflection ray is computed from incoming ray direction using law of reflection
- When ray hits an object in the scene, shade that surface using the same illumination calculation used for the directly visible surfaces



Demo: Reflections

Alpha blending

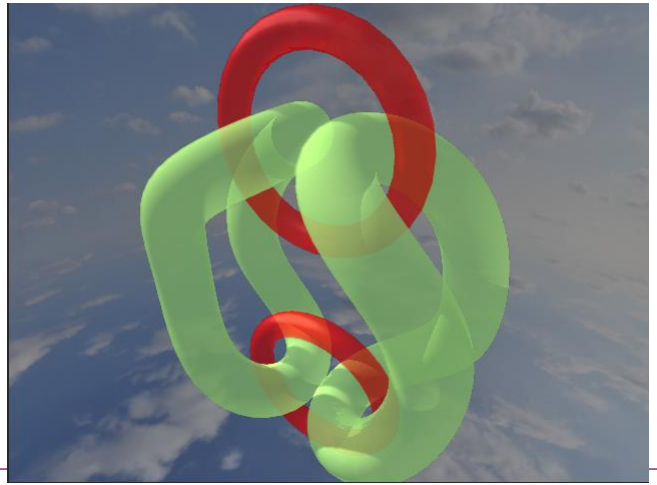
Approximation of transparency

- Not how transparency really works in real life
- Some scenes do not sort well!
- Not very compatible with deferred rendering

Raytraced transparency

“Real” transparency not alpha blending!

- In real life optics when light passes through a semi-transparent object some light is absorbed, some is not.
- Emit a ray from the back side of the surface.
- Shade like a reflection ray
- Order independent



Transparency and shadows

Shadow rays interact with transparent object!

- When shadow ray hits a transparent object it continues on towards the light.
- A shadow rays hitting a transparent object should be shaded and re-emitted as if it was a non-shadow ray.
- Shadow rays will pass through perfectly transparent regions of the surface.
- Shadow rays acquire color from translucent objects.

Demo: Transparency

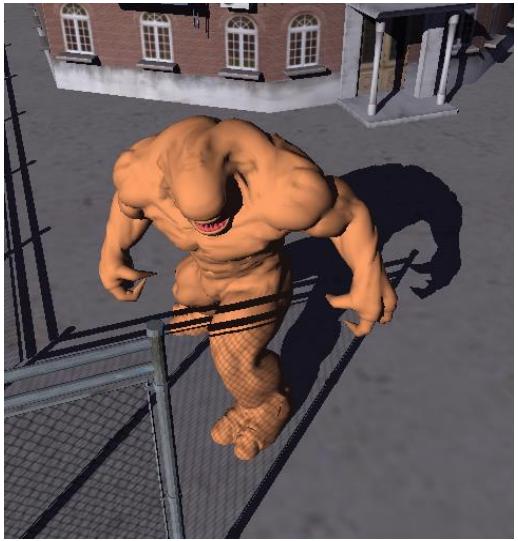
Conclusion

Raytracing is easy!

- **Makes realistic light simulation easy**
- **Easily combined with existing raster-based engines**

Try it yourself

- Talk to us at Booth 402, South Hall
- Download the OpenRL SDK
- Download the hybrid rendering example



http://www.imgtec.com/powervr/powervr_openrl_sdk.asp

Thank You!

Thanks to everyone who helped put this demo together. Including:

- **Denis Renshaw**
- **Mads Drøschler**
- **Stefan Morrell**
- **Jon Frisby**