Ray Tracing for Games

Dr. Jacco Bikker - IGAD/BUAS, Breda, January 30

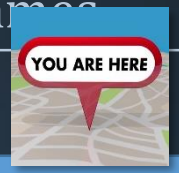
Welcome!

2

Agenda:

- Current State
- Advanced Whitted
- Generic Ray Queries
- Faster Whitted

Ray Tracing for Ga



Wednesday
13:00 – 17:00

course intro
LH2
template
Whitted
refactoring
RT-centric games

LAB 1

Thursday
09:00 – 14:00

advanced Whitted
audio, AI & physics
faster Whitted
Heaven7

LAB 2

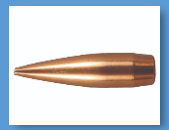
work @ home

End result day 2:

A solid Whitted-style ray tracer, as a basis for subsequent work.

Friday
09:00 – 17:00

optimization
profiling, rules of engagement
threading



LAB 3

SIMD
applied SIMD
SIMD triangle
SIMD AABB

LAB 4

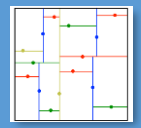
End result day 3:

A 5x faster tracer.

GLOBAL GAME JAM

Monday
09:00 – 17:00

acceleration
grid, BVH, kD-tree
SAH
binning



LAB 5

refitting
top-level BVH
threaded building

LAB 6

End result day 4:

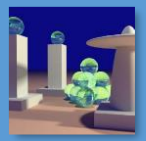
A real-time tracer.

Tuesday
09:00 – 17:00

Monte-Carlo
Cook-style
glossy, AA
area lights, DOF

LAB 7

path tracing



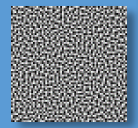
LAB 8

End result day 5:

Cook or Kajiya.

Thursday
09:00 – 17:00

random numbers
stratification
blue noise



LAB 9

importance
sampling
next event
estimation

LAB 10

End result day 6:

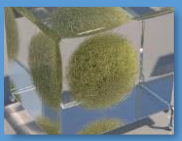
Efficiency.

Friday
09:00 – 17:00

future work

LAB 11

path guiding



LAB 10

End result day 6:

Great product.

End product:

1. Path tracer that produces pretty images in a few minutes.

   You have something on the screen / you dusted off your old ray tracer.

2. Real-time ray tracer on the CPU or GPU.

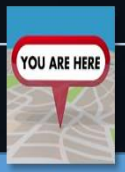
   You explored the template and/or managed to build LH2.
   You have something on the screen.

3. Raytraced game or demo.

   You formed a team and assigned roles.
   You have decided on a game concept.

4. RTX port of an existing game.

   You formed a team.
   You picked a game you want to port.



Wednesday
13:00 – 17:00

course intro
LH2
template
Whitted
refactoring
RT-centric games

LAB 1

Thursday
09:00 – 14:00

advanced Whitted
audio, AI & physics
faster Whitted
Heaven7

LAB 2

work @ home

End result day 2:

A solid Whitted-style ray tracer, as a basis for subsequent work.

Friday
09:00 – 17:00

optimization
profiling, rules of engagement
threading

LAB 3

SIMD
applied SIMD
SIMD triangle
SIMD AABB

LAB 4

End result day 3:

A 5x faster tracer.

GLOBAL GAME JAM

Today: Consolidation

Ensure that you have something that renders something today, beat the 'First Image' obstacle.

Take some time to brush up your code so that it is convenient to work with and feels comfortable.

Take in some information about speeding things up, for a concrete short term goal.

When you go to sleep tonight you reached ALPHA.
(i.e., something that works – we'll improve it beyond recognition)

Going for the Game Jam? You will need some tricks to have a working engine before Friday evening.

Wednesday
13:00 – 17:00

course intro
LH2
template
Whitted
refactoring
RT-centric games

LAB 1

Thursday
09:00 – 14:00

advanced Whitted
audio, AI & physics
faster Whitted
Heaven7

LAB 2

work @ home

End result day 2:

A solid Whitted-style ray tracer, as a basis for subsequent work.

Friday
09:00 – 17:00

optimization
profiling, rules of
engagement
threading

LAB 3

SIMD
applied SIMD
SIMD triangle
SIMD AABB

LAB 4

End result day 3:

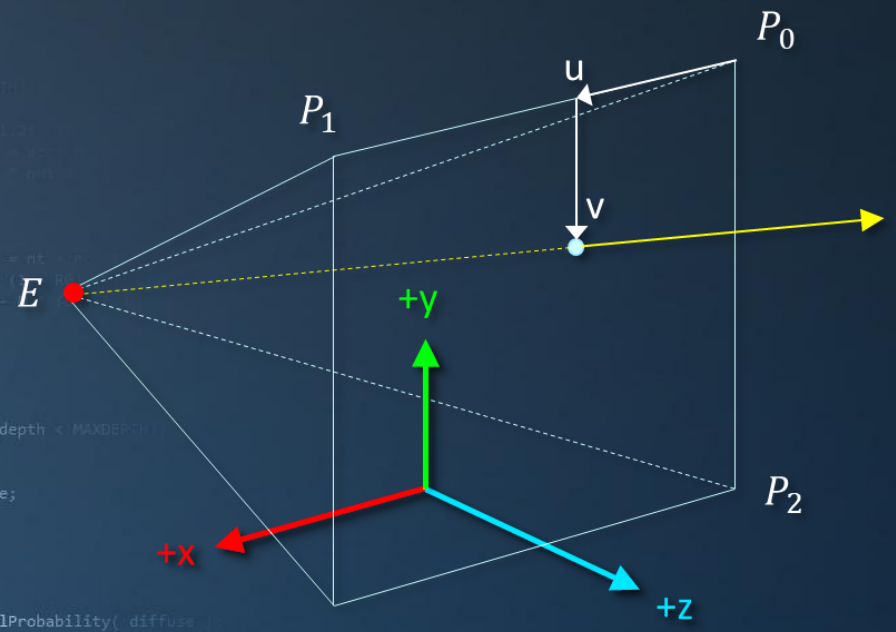
A 5x faster tracer.

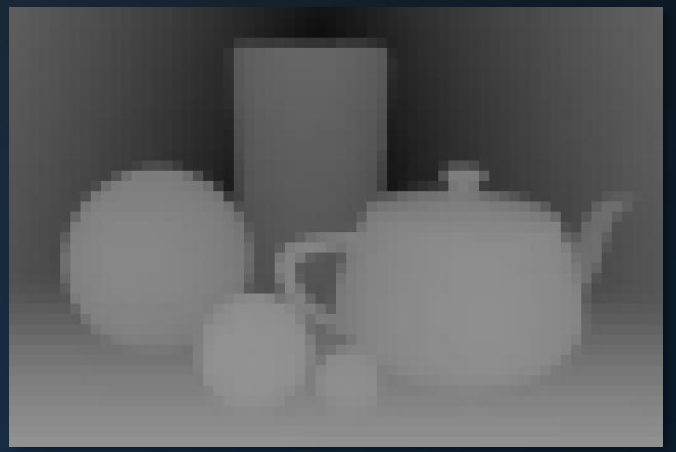
GLOBAL GAME JAM

Agenda:

- Current State
- Advanced Whitted
- Generic Ray Queries
- Faster Whitted

Whitted – Bare Necessities





float3* screen = … ;

void NearestIntersection(…);

bool IsOccluded(…);

Plane: $P \cdot \vec{N} + d = 0$

Ray: $P(t) = O + t\vec{D}$

Substituting for $P(t)$, we get: $t = -(O \cdot \vec{N} + d)/(\vec{D} \cdot \vec{N})$

```
void Sphere::IntersectSphere( Ray ray )
{
    vec3 C = this.pos - ray.O;
    float t = dot( C, ray.D );
    vec3 Q = C - t * ray.D;
    float p2 = dot( Q, Q );
    if (p2 > sphere.r2) return; // r2 = r * r
    t -= sqrt( sphere.r2 - p2 );
    if ((t < ray.t) && (t > 0)) ray.t = t;
}
```

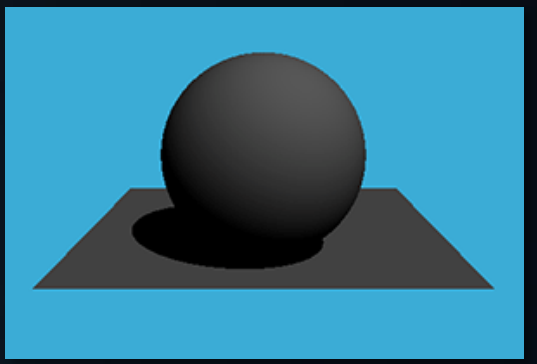
Whitted – Intermediate

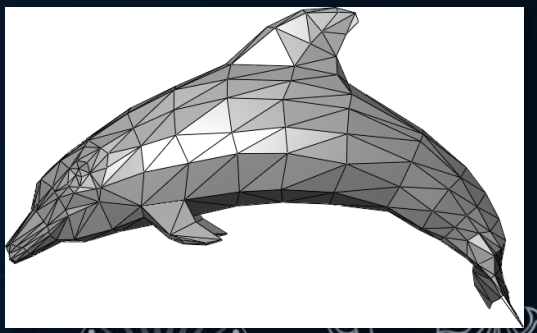
Shadows
Direct illumination: mind $\frac{1}{dist^2}$ and $N \cdot L$
Prevent shadow acne

Reflections
Normals, reflection vector, recursion, recursion cap
No shadow rays for reflection
Prevent $t = 0$ reflections

Materials
Blending reflection & ‘Lambert’
Textures, texture coordinates (detail textures, specular map, …)

Primitives
You may want to limit yourself to triangles –
Supporting multiple primitives is a pain.





Whitted – Advanced

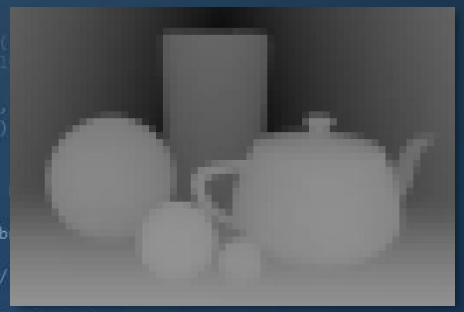
Transmission
Snell's law, Fresnel equations, Beer's law

Soft shadows
'Cook-style', stochastic, stratification, blue noise, …

Glossy, anti-aliasing

…

```
Color DirectIllumination( I, N )
   Color sum = BLACK
   for each light l
      L = light.pos – I
      dist2 = L.squareLength()
      if (!IsOccluded( I, L, sqrt(dist2) ))
         normalize(L)
         sum += (light.color * dot(N,L))/dist2)
   return sum
```

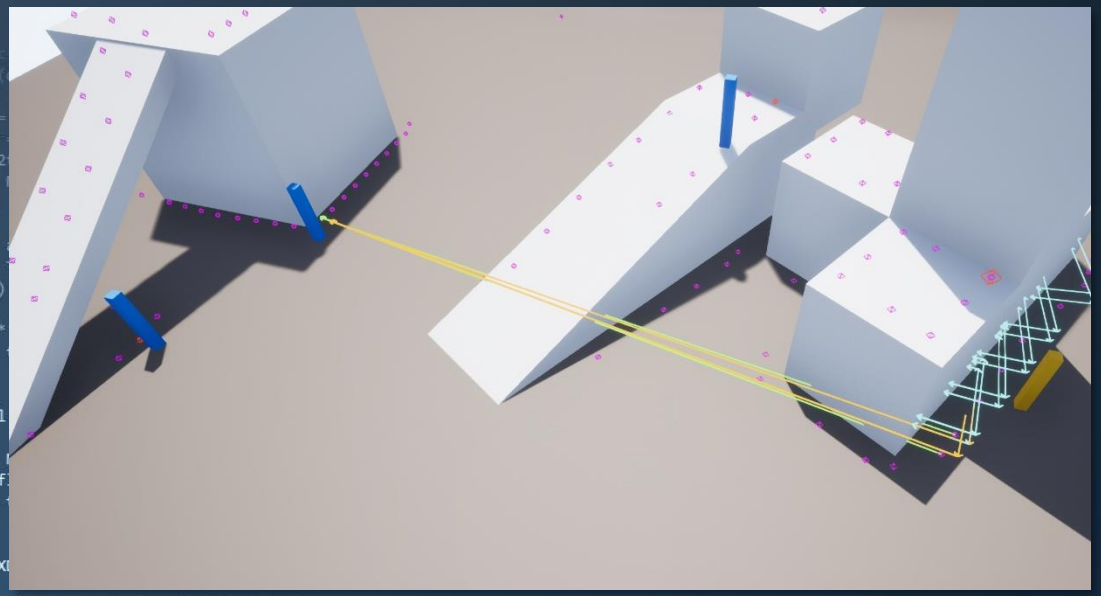


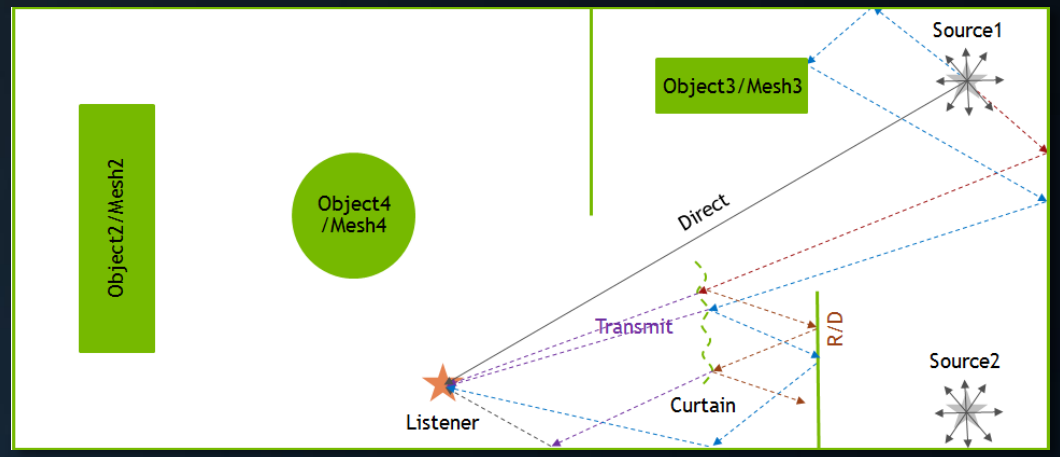
```
Color Trace( ray r )
   I, N, mat = NearestIntersection( scene, r )
   if (mat == DIFFUSE)
      return mat.color * DirectIllumination( I, N )
   if (mat == MIRROR)
      return mat.color * Trace( I, reflect( r.D, N ) )
```

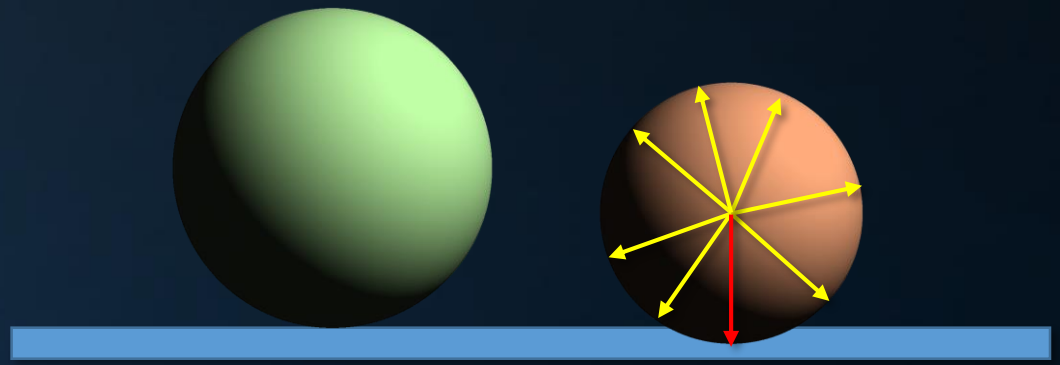
Agenda:

- Current State
- Advanced Whitted
- Generic Ray Queries
- Faster Whitted

Ray Queries







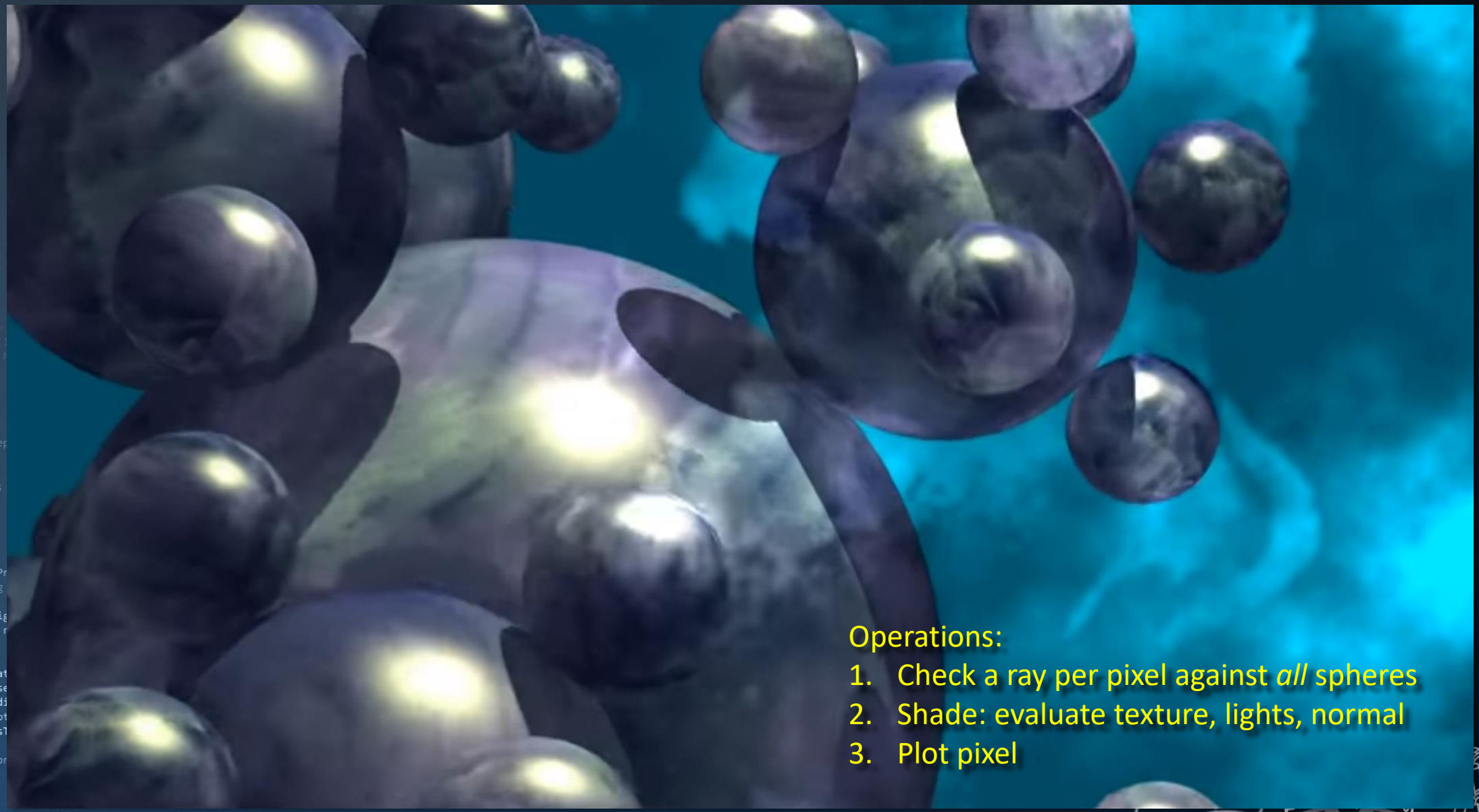
Make your ray queries accessible outside your engine:

```
NearestIntersection(…)
IsOccluded(…)
```

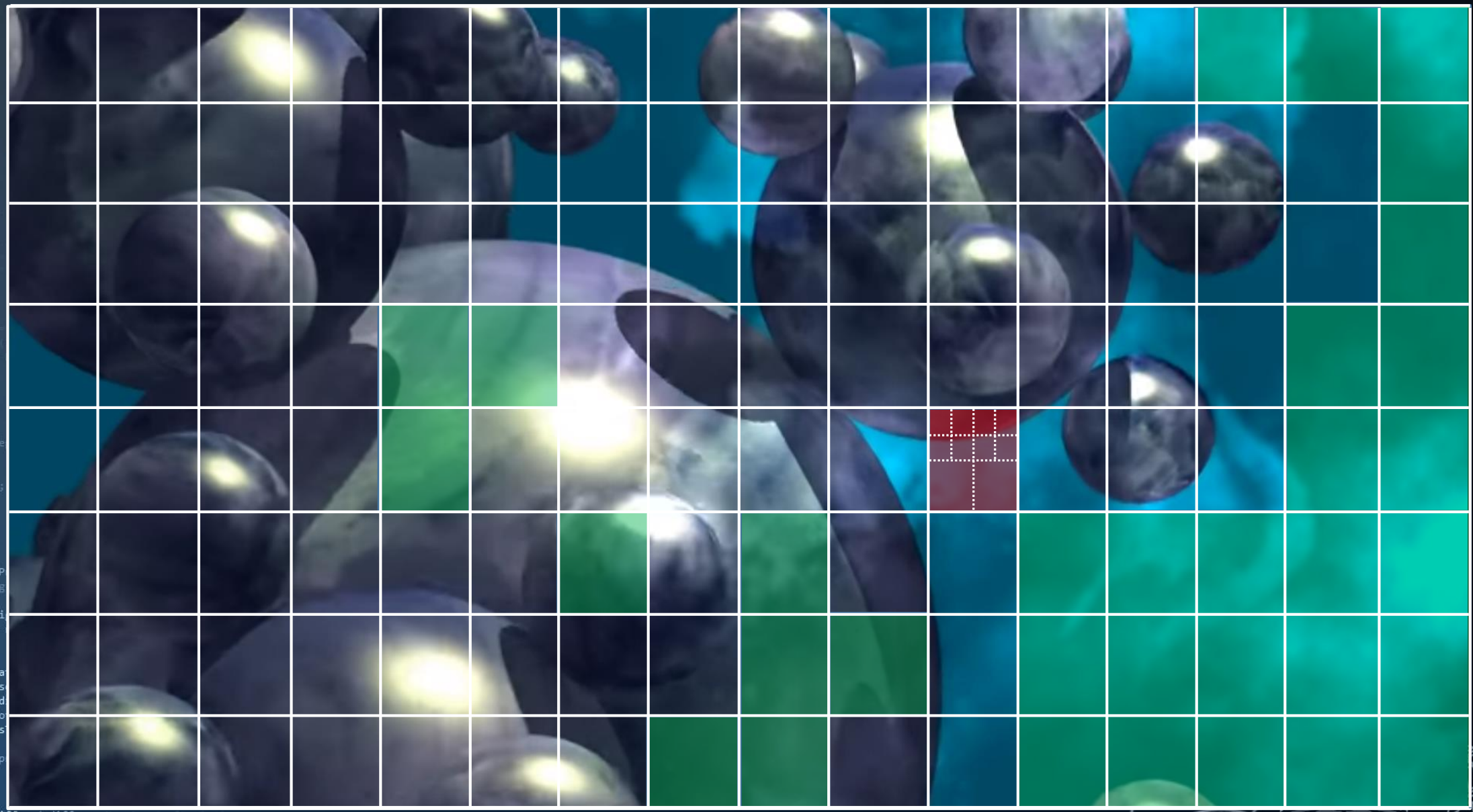
Agenda:

- Current State
- Advanced Whitted
- Generic Ray Queries
- Faster Whitted

Ray Tracing for Games



Ray Tracing for Games



Screen-space Subdivision

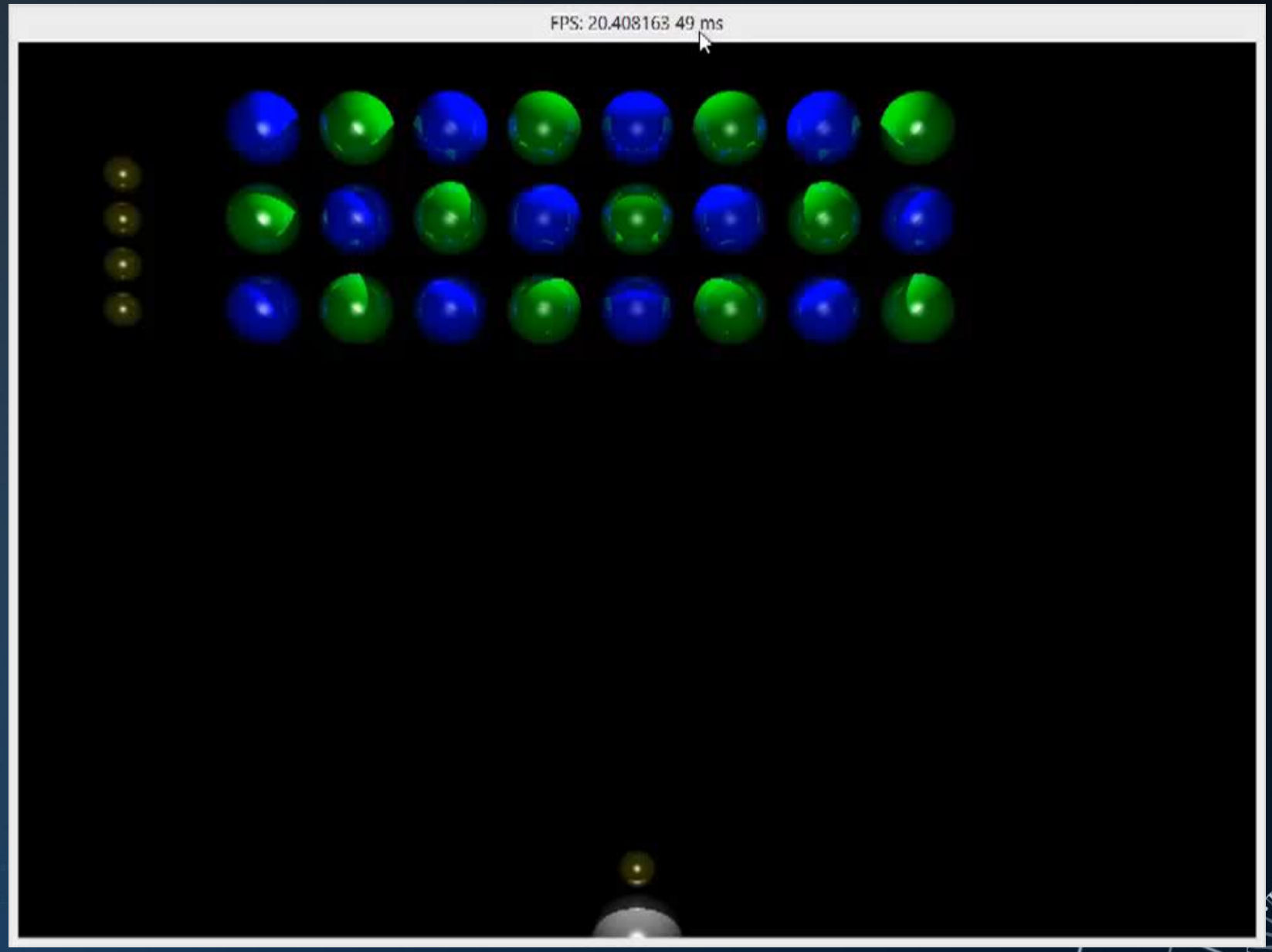
Quickly bring down the average amount of work per pixel.

Limitations:

- Smallest detail should be larger than a block
- Spheres could be flattened even if much larger than a block

Could we also use this for shadow rays?

Ray Tracing for Games



Making it Fast before we get to the Cool Stuff ™

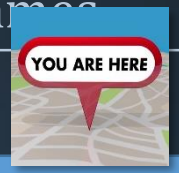
Rendering at interactive frame rates is important.

Methods:

- Render pixel blocks to speed up rendering; refine when the camera is stationary.
- Multithread. Use OpenMP, tiles of pixels and the dynamic scheduler for now.
- Keep your scene simple.

Ask yourself what you need at the bare minimum to test your game code;
build this functionality first – refine later.

Ray Tracing for Ga



Wednesday
13:00 – 17:00

course intro
LH2
template
Whitted
refactoring
RT-centric games

LAB 1

Thursday
09:00 – 14:00

advanced Whitted
audio, AI & physics
faster Whitted
Heaven7

LAB 2

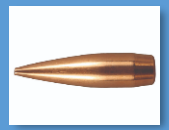
work @ home

End result day 2:

A solid Whitted-style ray tracer, as a basis for subsequent work.

Friday
09:00 – 17:00

optimization
profiling, rules of engagement
threading



LAB 3

SIMD
applied SIMD
SIMD triangle
SIMD AABB

LAB 4

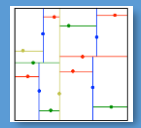
End result day 3:

A 5x faster tracer.

GLOBAL GAME JAM

Monday
09:00 – 17:00

acceleration
grid, BVH, kD-tree
SAH
binning



LAB 5

refitting
top-level BVH
threaded building

LAB 6

End result day 4:

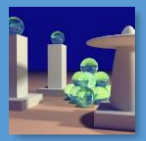
A real-time tracer.

Tuesday
09:00 – 17:00

Monte-Carlo
Cook-style
glossy, AA
area lights, DOF

LAB 7

path tracing



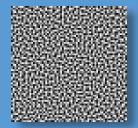
LAB 8

End result day 5:

Cook or Kajiya.

Thursday
09:00 – 17:00

random numbers
stratification
blue noise



LAB 9

importance
sampling
next event
estimation

LAB 10

End result day 6:

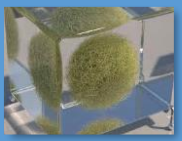
Efficiency.

Friday
09:00 – 17:00

future work

LAB 11

path guiding



LAB 10

End result day 6:

Great product.

Agenda:

- Current State
- Advanced Whitted
- Generic Ray Queries
- Faster Whitted

End of PART 2.