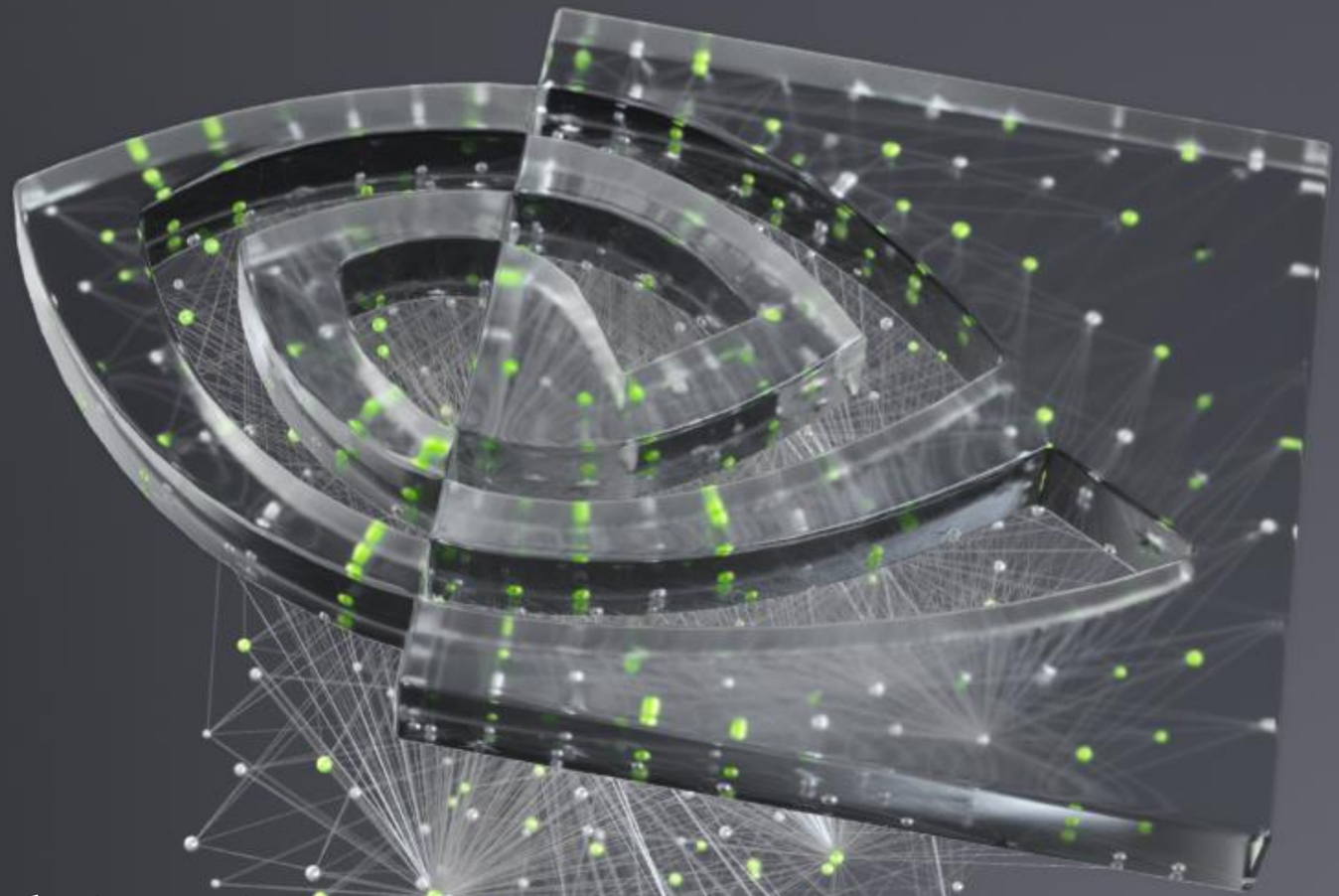


NVIDIA | GAME STACK LIVE

RTXDI: Details on Achieving Real-time Performance

Chris Wyman

Principal Research Scientist, NVIDIA

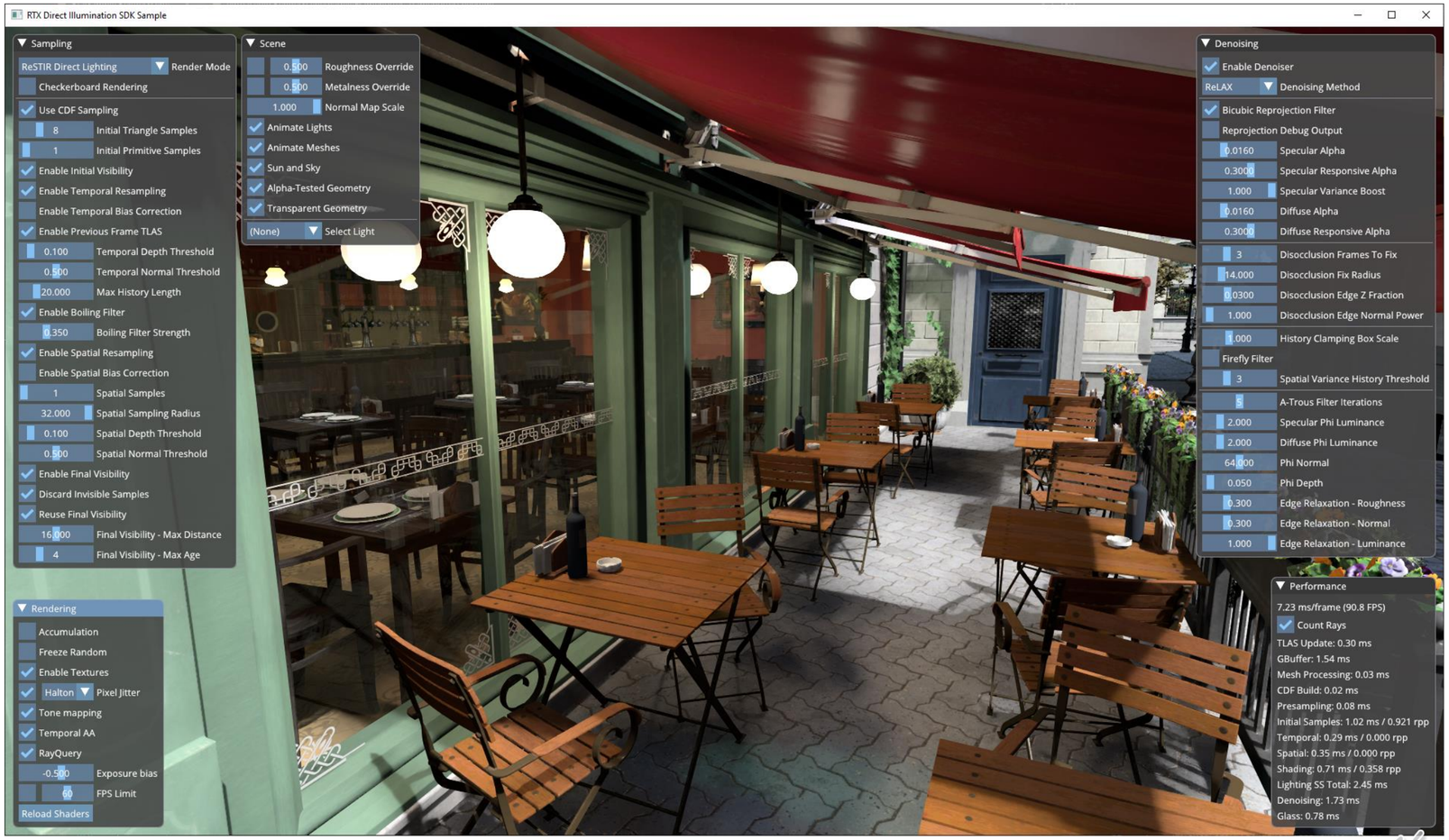
RECENTLY LAUNCHED RTX DIRECT ILLUMINATION (RTXDI)

https://developer.nvidia.com/rtxdi

- Render with millions of dynamic lights
  - Primitives of all sorts (triangles, spheres, cylinders, etc.), with or without IES profiles
  - Also textured and environment map emissives

SDK INCLUDES WHITE-BOX SOURCE

- Sample app demonstrates integration
- Led by Alexey Panteleev
- Go take a look!



TODAY: EXPLORE ALGORITHM DETAILS

A technical deep dive into how we made RTXDI fast

- Currently, RTXDI largely based on “ReSTIR”
  - Reservoir spatiotemporal importance resampling
- Presented basics a few times last year:
  - At SIGGRAPH 2020, “Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting”
  - At GTC 2020, “Rendering games with millions of ray-traced lights”
  - At HPG 2020, “Reframing light transport for real-time”
- Will do a quick high-level review
- **Today’s focus: improvements to achieve real-time performance**

AGENDA

Quick Overview

What is RIS, ReSTIR, resampling? How does it help rendering?

Memory Coherence Issues

Coherence → big impact in ReSTIR; tackle with unusual approach

Intelligent Compute Refactoring

Leveraging or removing redundant compute in original research

PHYSICALLY-BASED RENDERING

Let’s start with basics: What is it?

PHYSICALLY-BASED RENDERING

Let's start with basics: What is it?

- Solve mathematical equations describing light interactions
- Different renderers may focus on *different aspects*

- Typically, the *rendering equation* is involved

$$L(x, \omega_o) = \int_{\Omega} \rho(x, \omega_i, \omega_o) L(x, \omega_i) \langle w_i \cdot N \rangle d\omega_i$$

PHYSICALLY-BASED RENDERING

Let's start with basics: What is it?

- Solve mathematical equations describing light interactions
- Different renderers may focus on *different aspects*

- Typically, the *rendering equation* is involved

$$L(x, \omega_o) = \int_{\Omega} \rho(x, \omega_i, \omega_o) L(x, \omega_i) \langle w_i \cdot N \rangle d\omega_i$$

The two hard parts in raster

PHYSICALLY-BASED RENDERING

Let's start with basics: What is it?

- Solve mathematical equations describing light interactions
- Different renderers may focus on *different aspects*

- Typically, the *rendering equation* is involved

$$L(x, \omega_o) = \int_{\Omega} \rho(x, \omega_i, \omega_o) L(x, \omega_i) \langle w_i \cdot N \rangle d\omega_i$$

The two hard parts in raster

Not easier for ray tracing; incoming light still comes from everywhere

In DXR, just call TraceRay()

HANDLING HIGH-COMPLEXITY

How do you solve a tough integral?

HANDLING HIGH-COMPLEXITY

How do you solve a tough integral?

$$F = \int f(x)dx \approx \frac{1}{N} \sum_{1 \le i \le N} \frac{f(x_i)}{p(x_i)}$$

- Use Monte Carlo integration with *importance sampling*
  - Monte Carlo = numerically sampling the integral
  - Importance sampling = minimizing N with good choice of distribution $p(x)$

HANDLING HIGH-COMPLEXITY

How do you solve a tough integral?

$$F = \int f(x)dx \approx \frac{1}{N} \sum_{1 \le i \le N} \frac{f(x_i)}{p(x_i)}$$

- Use Monte Carlo integration with *importance sampling*
  - Monte Carlo = numerically sampling the integral
  - Importance sampling = minimizing N with good choice of distribution $p(x)$

- Can we do *perfect* importance sampling? I.e., get N=1

(APPROXIMATELY) PERFECT IMPORTANCE SAMPLING

- Perfect sampling means $p(x) \propto f(x)$, specifically:

$$p(x) = \frac{f(x)}{\int f(x)dx}$$

To perfectly sample $\int f(x)dx$, first need solution to $\int f(x)dx$…

(APPROXIMATELY) PERFECT IMPORTANCE SAMPLING

- Perfect sampling means $p(x) \propto f(x)$, specifically:

$$p(x) = \frac{f(x)}{\int f(x)dx}$$

To perfectly sample $\int f(x)dx$, first need solution to $\int f(x)dx$…

- Can *approximate* it, using more Monte Carlo integration:

$$F = \int f(x)\,dx \approx \frac{1}{N}\sum\left[\frac{f(x_i)}{\hat{p}(x_i)}\,\frac{1}{M}\sum\frac{\hat{p}(x_j)}{p(x_j)}\right]$$

resampled importance sampling; see Talbot et al.

WHY USE RESAMPLED IMPORTANCE SAMPLING

And what is ReSTIR?

$$F = \int f(x)\,dx \approx \frac{1}{N}\sum\left[\frac{f(x_i)}{\hat{p}(x_i)}\,\frac{1}{M}\sum\frac{\hat{p}(x_j)}{p(x_j)}\right]$$

We need a color at our pixel

WHY USE RESAMPLED IMPORTANCE SAMPLING

And what is ReSTIR?

$$F = \int f(x)\,dx \approx \frac{1}{N}\sum\left[\frac{f(x_i)}{\hat{p}(x_i)}\,\frac{1}{M}\sum\frac{\hat{p}(x_j)}{p(x_j)}\right]$$

We need a color at our pixel

Desire minimal ray count N

WHY USE RESAMPLED IMPORTANCE SAMPLING

And what is ReSTIR?

$$F = \int f(x)\,dx \approx \frac{1}{N}\sum\left[\frac{f(x_i)}{\hat{p}(x_i)}\,\frac{1}{M}\sum\frac{\hat{p}(x_j)}{p(x_j)}\right]$$

We need a color at our pixel

Desire minimal ray count N

Reuse neighbor pixel samples to *improve* sampling at current pixel

WHY USE RESAMPLED IMPORTANCE SAMPLING

And what is ReSTIR?

$$F = \int f(x)\,dx \approx \frac{1}{M_0}\sum\left[\frac{f(x_{i_0})}{\widehat{p_0}(x_{i_0})}\,\frac{1}{M_1}\sum\left[\frac{\widehat{p_0}(x_{i_1})}{\widehat{p_1}(x_{i_1})}\frac{1}{M_2}\sum\left[\frac{\widehat{p_1}(x_{i_2})}{\widehat{p_2}(x_{i_2})}\cdots\right]\right]\right]$$

Resampling chain can continue indefinitely!

WHY USE RESAMPLED IMPORTANCE SAMPLING

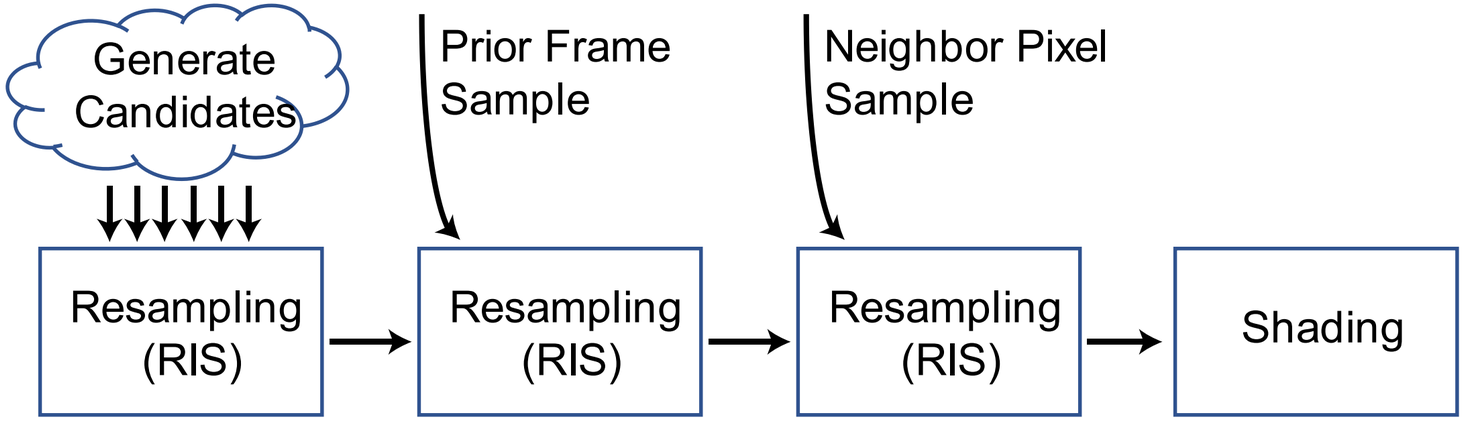
And what is ReSTIR?

$$F = \int f(x)\,dx \approx \frac{1}{M_0}\sum\left[\frac{f(x_{i_0})}{\widehat{p_0}(x_{i_0})}\,\frac{1}{M_1}\sum\left[\frac{\widehat{p_0}(x_{i_1})}{\widehat{p_1}(x_{i_1})}\frac{1}{M_2}\sum\left[\frac{\widehat{p_1}(x_{i_2})}{\widehat{p_2}(x_{i_2})}\dots\right]\right]\right]$$

Resampling chain can continue indefinitely!

Feed spatial samples, temporal samples, etc., to improve current pixel samples

ReSTIR Pipeline:



WHAT DOES THIS GET US?

WHAT DOES THIS GET US?

Paper shows:
3 million emissive triangles
~50 ms

THAT’S NOT QUITE GAME PERFORMANCE…

SDK TARGET: IMPROVE PERFORMANCE

3 million emissives
Lighting: 4.6 ms

AGENDA

Quick Overview

What is RIS, ReSTIR, resampling? How does it help rendering?

Memory Coherence Issues

Coherence → big impact in ReSTIR; tackle with unusual approach

Intelligent Compute Refactoring

Leveraging or removing redundant compute in original research

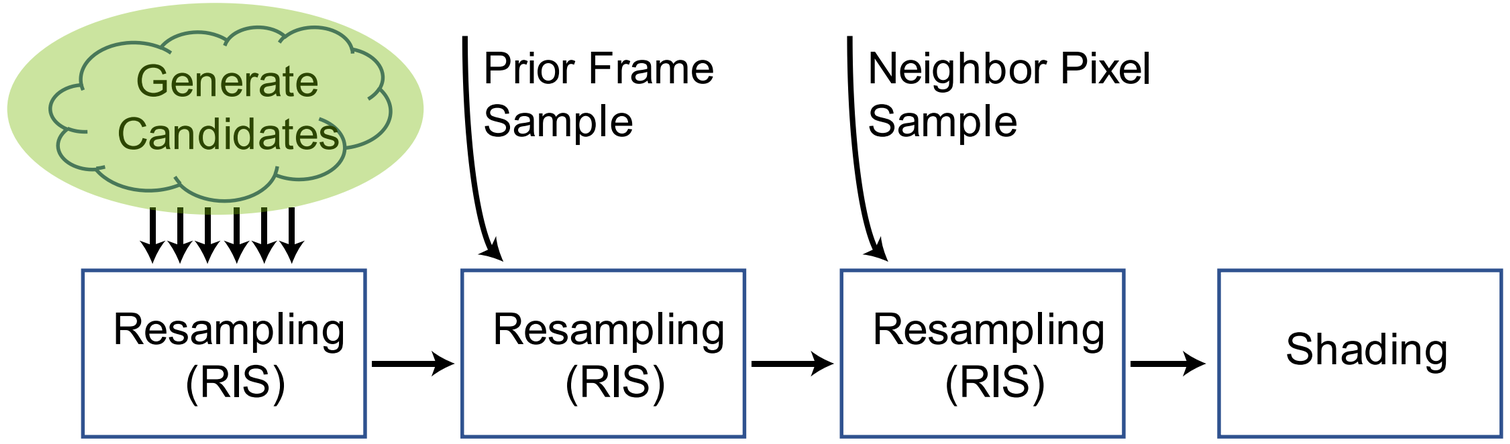
MEMORY COHERENCE AND BANDWIDTH

Initial performance analysis showed this as the key bottleneck

MEMORY COHERENCE AND BANDWIDTH

Initial performance analysis showed this as the key bottleneck

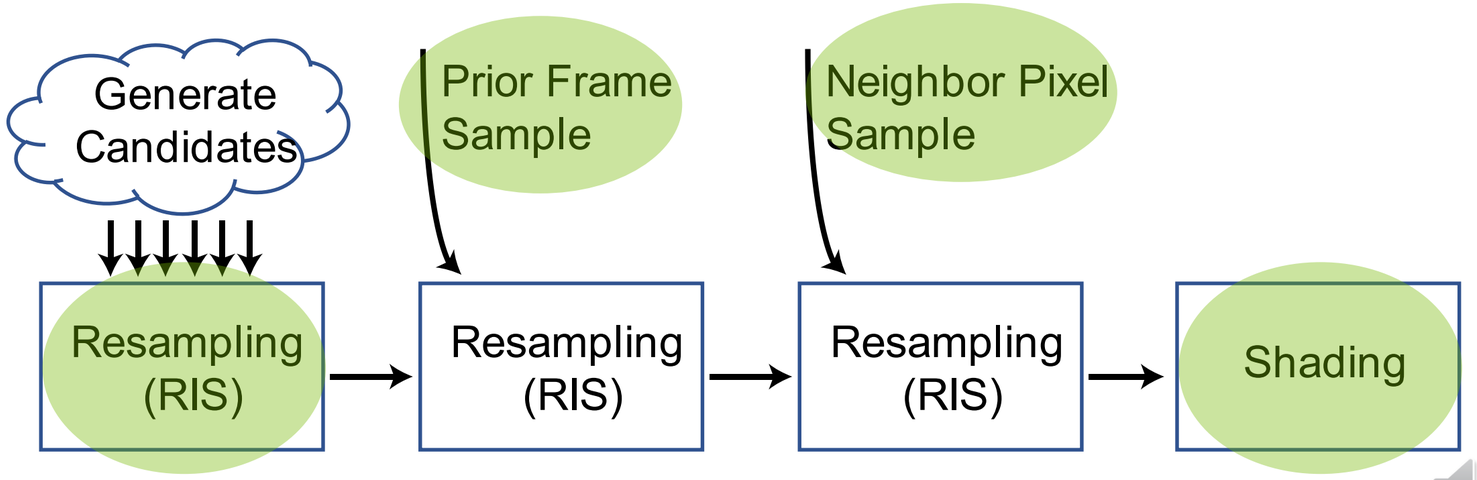
- Three main ways ReSTIR consumes memory
  - Randomly selecting lights



MEMORY COHERENCE AND BANDWIDTH

Initial performance analysis showed this as the key bottleneck

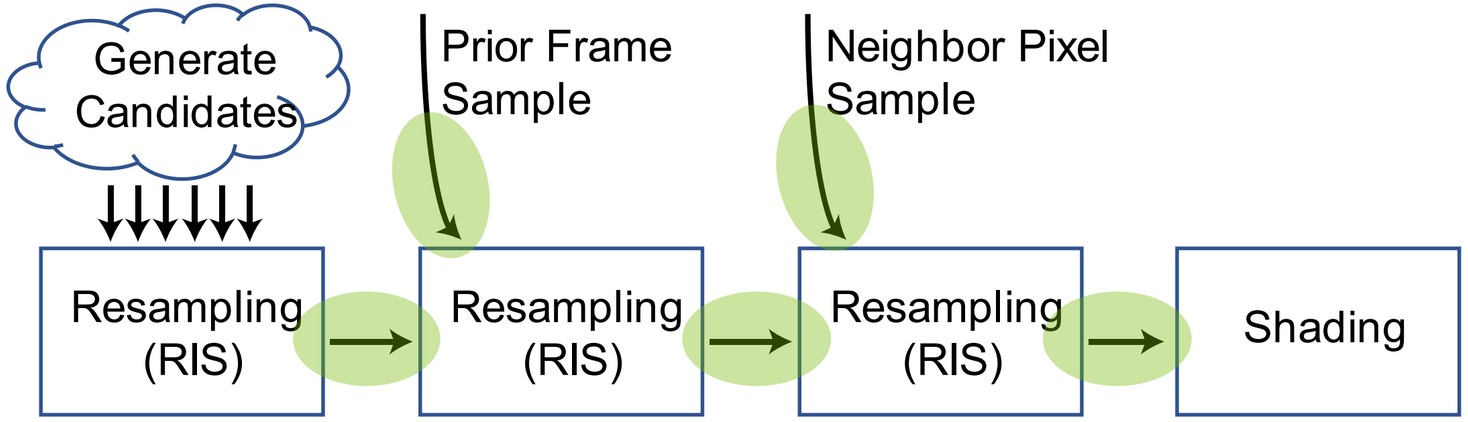
- Three main ways ReSTIR consumes memory
  - Randomly selecting lights
  - Reading G-buffer data



MEMORY COHERENCE AND BANDWIDTH

Initial performance analysis showed this as the key bottleneck

- Three main ways ReSTIR consumes memory
  - Randomly selecting lights
  - Reading G-buffer data
  - Reservoir I/O



MEMORY COHERENCE AND BANDWIDTH

Initial performance analysis showed this as the key bottleneck

- Three main ways ReSTIR consumes memory
  - Randomly selecting lights
  - Reading G-buffer data
  - Reservoir I/O

Improve in standard ways:

- Compress to minimize bandwidth
- Merge kernels to minimize intermediate read/writes
- Minimize size of reservoir (i.e., number samples)

MEMORY COHERENCE AND BANDWIDTH

Initial performance analysis showed this as the key bottleneck

- Three main ways ReSTIR consumes memory
  - Randomly selecting lights
  - Reading G-buffer data
  - Reservoir I/O

Some of our perf team:

Wondered why I'd design such an algorithm

So… Let's dive into this one

SAMPLING LIGHTS COHERENTLY

What’s the problem?

SAMPLING LIGHTS COHERENTLY

What's the problem?

- **Need** randomization for resampling (i.e., for correctness)
- Choosing small random subset from very large list ≡ incoherency
- Each pixel chooses a **different** random subset ≡ cache thrash
- Constant # lookups; costs vary >20x scene to scene

SAMPLING LIGHTS COHERENTLY

What's the problem?

- **Need** randomization for resampling (i.e., for correctness)
- Choosing small random subset from very large list ≡ incoherency
- Each pixel chooses a **different** random subset ≡ cache thrash
- Constant # lookups; costs vary >20x scene to scene

- **Q: Can we "pre-randomize" to move incoherency out of inner loop?**

PRE-RANDOMIZING LIGHT SAMPLES

Key observation: Degenerate RIS steps allow ***reshaping*** computation

PRE-RANDOMIZING LIGHT SAMPLES

Key observation: Degenerate RIS steps allow ***reshaping*** computation

$$\int f(x)\,dx \approx \frac{1}{N}\sum\left[\frac{f(x_i)}{\hat{p}(x_i)}\,\frac{1}{M}\sum\frac{\hat{p}(x_j)}{p(x_j)}\right]$$

- What if we let $p(x) = \hat{p}(x)$?

PRE-RANDOMIZING LIGHT SAMPLES

Key observation: Degenerate RIS steps allow ***reshaping*** computation

$$\int f(x)\,dx \approx \frac{1}{N}\sum\left[\frac{f(x_i)}{\hat{p}(x_i)}\,\frac{1}{M}\sum\frac{\hat{p}(x_j)}{p(x_j)}\right]$$

- What if we let $p(x) = \hat{p}(x)$?

$$\int f(x)\,dx \approx \frac{1}{N}\sum\left[\frac{f(x_i)}{p(x_i)}\,\frac{1}{M}\sum\frac{p(x_j)}{p(x_j)}\right] = \frac{1}{N}\sum\left[\frac{f(x_i)}{p(x_i)}\,\frac{1}{M}\sum 1\right]$$

- This is the standard Monte Carlo estimator…

*But in **two** steps: first select M samples, then pick N of those!*

PRE-RANDOMIZING LIGHT SAMPLES

Key observation: Degenerate RIS steps allow ***reshaping*** computation

- Pre-process (once per frame):
  - Create S_i sets of lights, each containing S_M lights
  - Select S_M lights using source pdf $p(x)$ (previously) used per-pixel

PRE-RANDOMIZING LIGHT SAMPLES

Key observation: Degenerate RIS steps allow ***reshaping*** computation

- Pre-process (once per frame):
  - Create S_i sets of lights, each containing S_M lights
  - Select S_M lights using source pdf $p(x)$ (previously) used per-pixel
- Per-pixel:
  - Randomly select one of the S_i sets of lights
  - (Uniformly) sample *M* initial candidates from the S_M lights in the set
  - Continue ReSTIR as if you had selected *M* random candidates

PRE-RANDOMIZING LIGHT SAMPLES

Key observation: Degenerate RIS steps allow ***reshaping*** computation

- Pre-process (once per frame):
  - Create S_i sets of lights, each containing S_M lights
  - Select S_M lights using source pdf $p(x)$ (previously) used per-pixel
- Per-pixel:
  - Randomly select one of the S_i sets of lights
  - (Uniformly) sample *M* initial candidates from the S_M lights in the set
  - Continue ReSTIR as if you had selected *M* random candidates
- Further cache improvement:
  - Pixel blocks randomly sample the same set S_i of lights (i.e., pick random S_i per pixel block)
  - 8x8 blocks seem the sweet spot

PRE-RANDOMIZING LIGHT SAMPLES

Key observation: Degenerate RIS steps allow ***reshaping*** computation

Found 128 sets of 1024 lights
works across most or all scenes

- Pre-process (once per frame):
  - Create S_i sets of lights, each containing S_M lights
  - Select S_M lights using source pdf $p(x)$ (previously) used per-pixel
- Per-pixel:
  - Randomly select one of the S_i sets of lights
  - (Uniformly) sample *M* initial candidates from the S_M lights in the set
  - Continue ReSTIR as if you had selected *M* random candidates
- Further cache improvement:
  - Pixel blocks randomly sample the same set S_i of lights (i.e., pick random S_i per pixel block)
  - 8x8 blocks seem the sweet spot

ADDITIONAL ADVANTAGE OF PRE-RANDOMIZATION

Seamlessly handles multiple light types (e.g., light probes, triangles, spheres, points)

ADDITIONAL ADVANTAGE OF PRE-RANDOMIZATION

Seamlessly handles multiple light types (e.g., light probes, triangles, spheres, points)

- Picking one random light sample → quite complex:
  - First pick **which** light type
  - Then pick **which light** of that type
  - Finally pick a **point on** that light
  - Leads to execution divergence
  - Pre-randomization moves this divergence out of inner loop

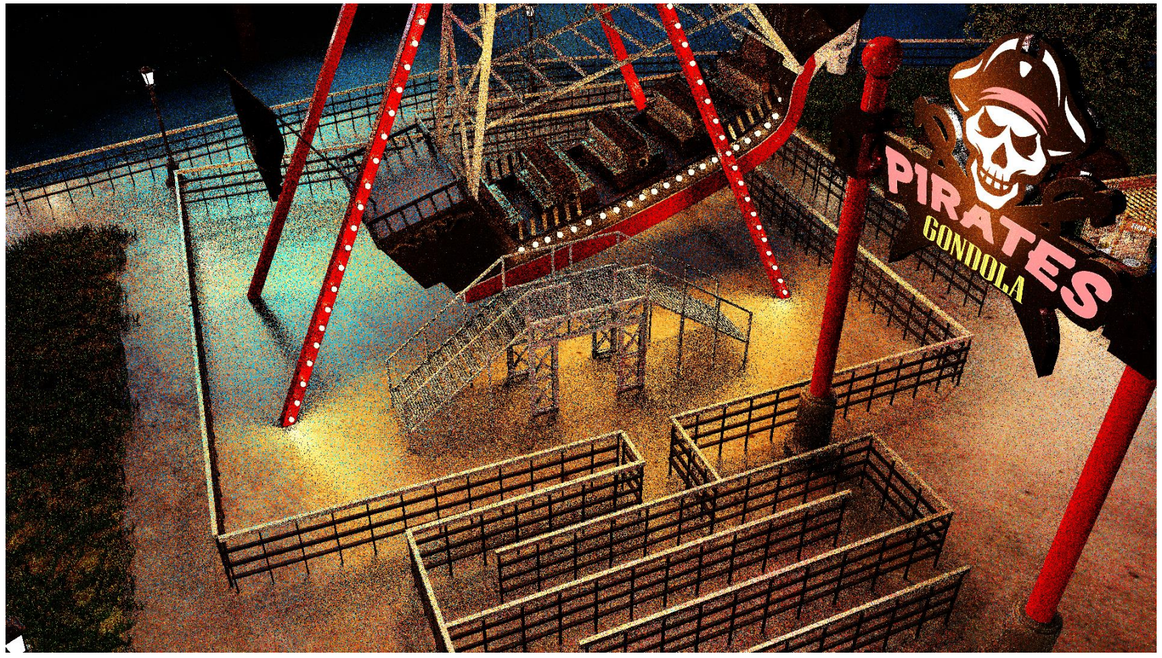
ADDITIONAL ADVANTAGE OF PRE-RANDOMIZATION

Seamlessly handles multiple light types (e.g., light probes, triangles, spheres, points)

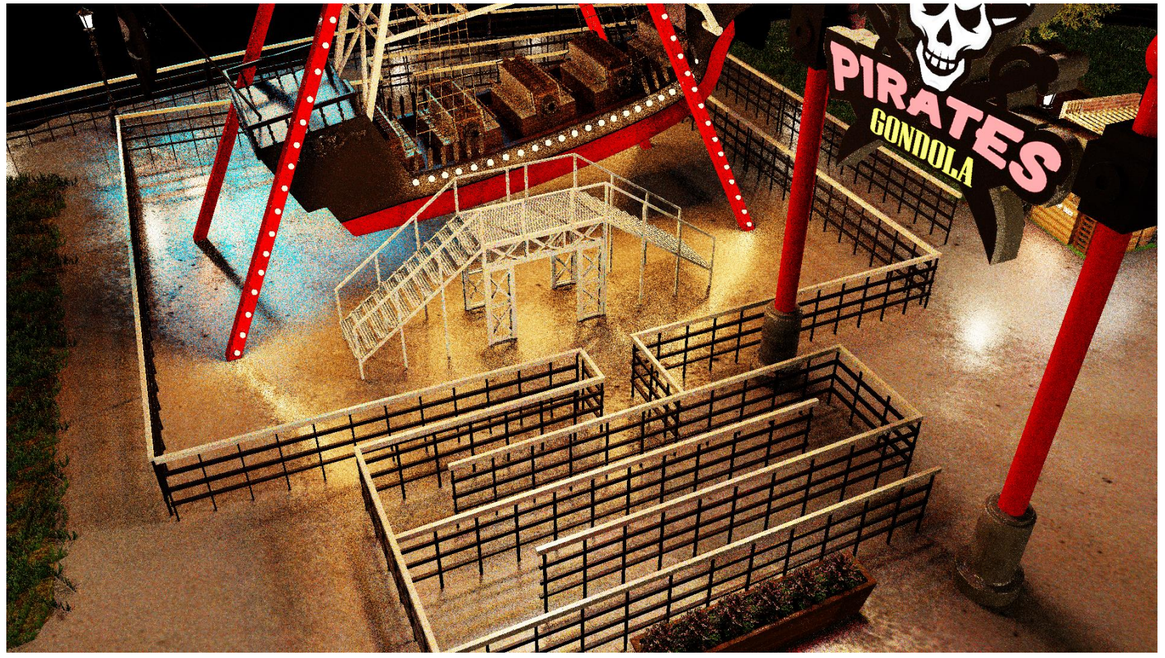
- Picking one random light sample → quite complex:
  - First pick **which** light type
  - Then pick **which light** of that type
  - Finally pick a **point on** that light
  - Leads to execution divergence
  - Pre-randomization moves this divergence out of inner loop
- Can reshape sampling multiple times
  - Sample per light type; then resample into our S_i light pools

IMPROVEMENT?

Going back to the Amusement Park



Before: ~20 ms initial light sampling
50 ms total lighting cost



After: ~0.8 ms for initial light sampling
4.6 ms for total lighting cost

AGENDA

Quick Overview

What is RIS, ReSTIR, resampling? How does it help rendering?

Memory Coherence Issues

Coherence → big impact in ReSTIR; tackle with unusual approach

---

Intelligent Compute Refactoring

Leveraging or removing redundant compute in original research

ONE GOAL: MINIMIZE RAY COUNTS

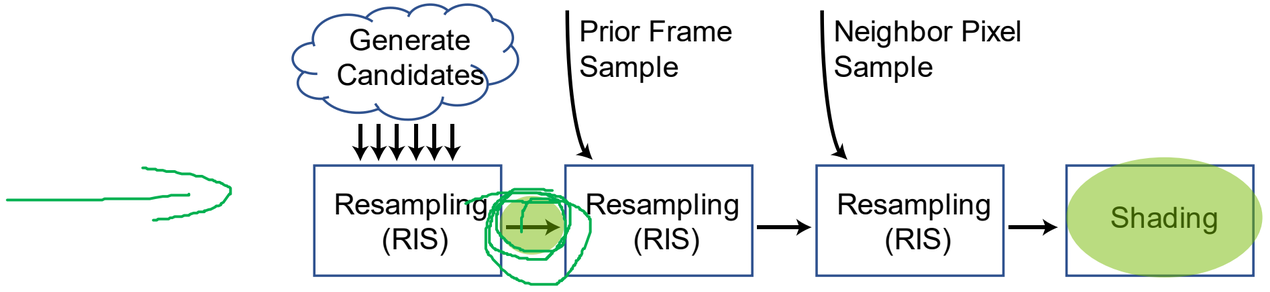
ReSTIR paper uses 5 rays per pixel

ONE GOAL: MINIMIZE RAY COUNTS

ReSTIR paper uses 5 rays per pixel

1 shadow ray
guides resampling

4 shadow rays
for final shading



ONE GOAL: MINIMIZE RAY COUNTS

ReSTIR paper uses 5 rays per pixel

1 shadow ray
guides resampling

4 shadow rays
for final shading

- Ponder: How important are all these rays? Particularly the last four…

WHAT DO THOSE RAYS DO?

Remember: Final shade rays are the N value in RIS

WHAT DO THOSE RAYS DO?

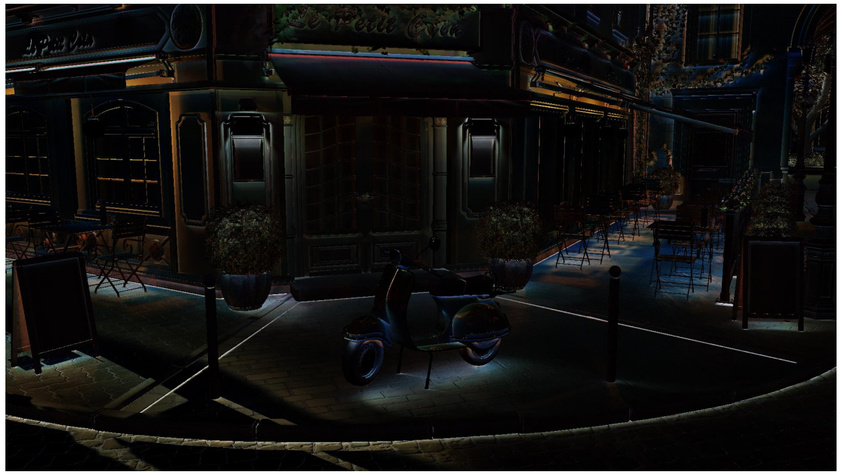
Remember: Final shade rays are the N value in RIS

$$\int f(x)\,dx \approx \frac{1}{N}\sum\left[\frac{f(x_i)}{\hat{p}(x_i)}\frac{1}{M}\sum\frac{\hat{p}(x_j)}{p(x_j)}\right]$$

- Controls variance of $f(x_i)/\hat{p}(x_i)$. Is N = 4 really much better?

With final shadow rays

No final shadow rays



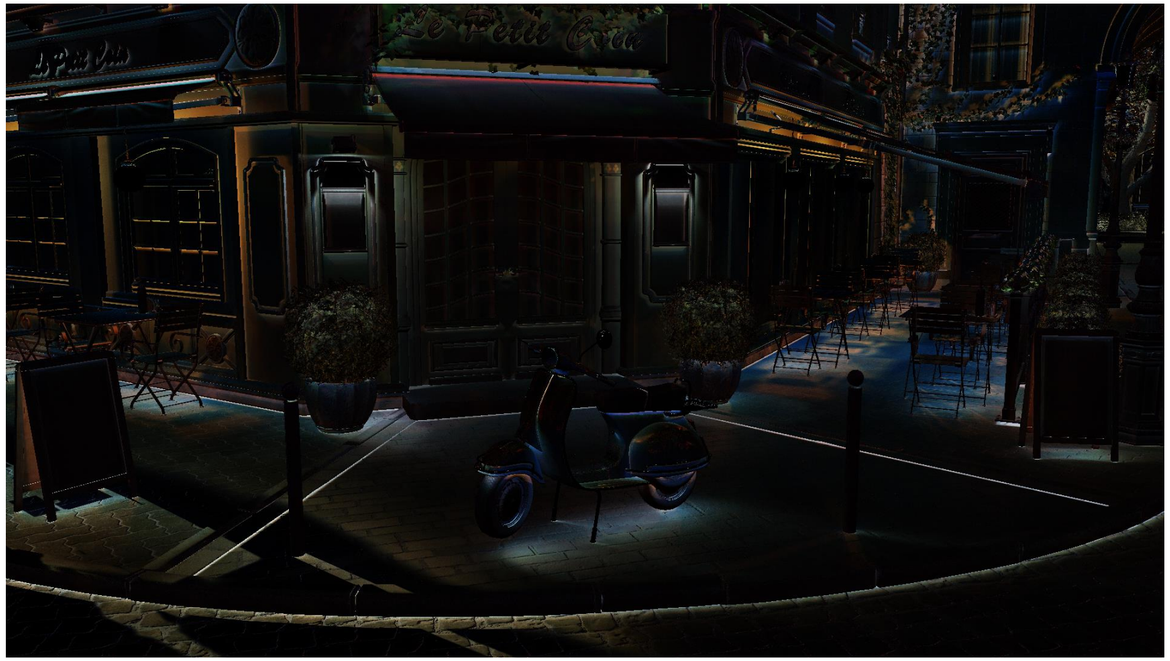
N controls variance of ***this***

LET DENOISER HANDLE THAT

- N = 1 is fine with a good denoiser
- Cuts budget to 2 rays per pixel

- Standard ways to cut further
  - Checkerboarding
  - Lower resolution reservoirs

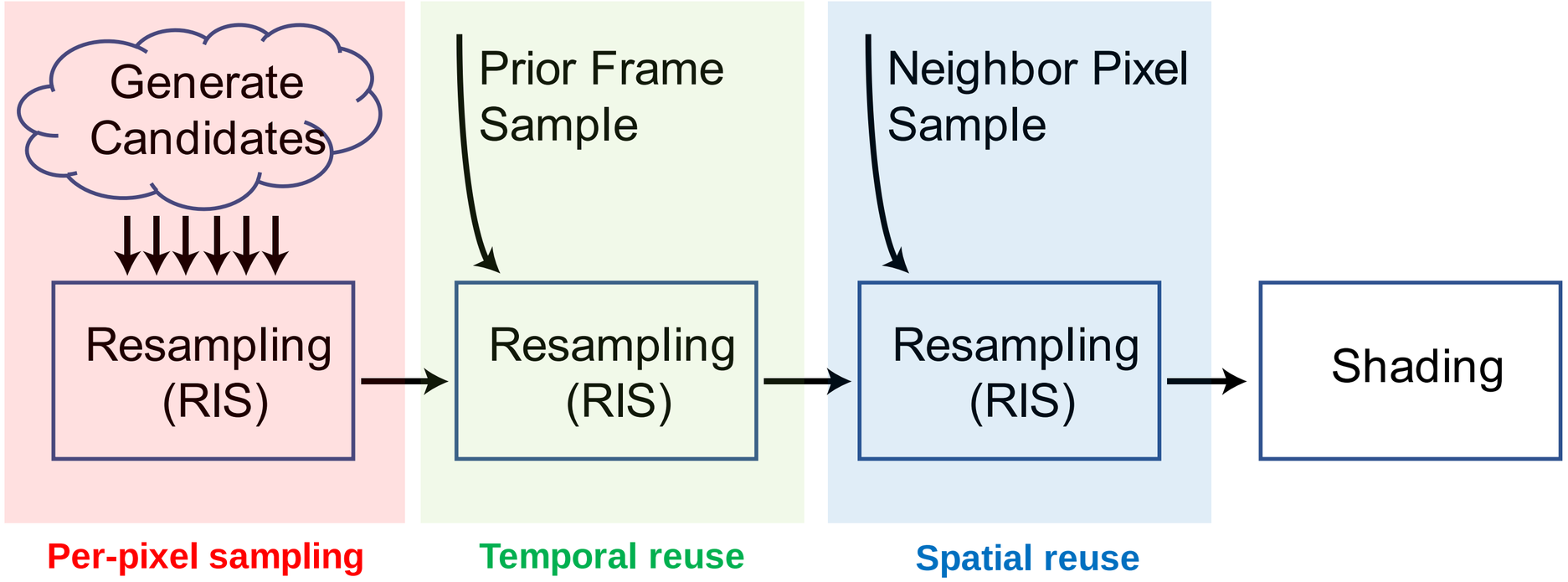
- Plus some additional tricks



REUSING SAMPLES MORE INTELLIGENTLY

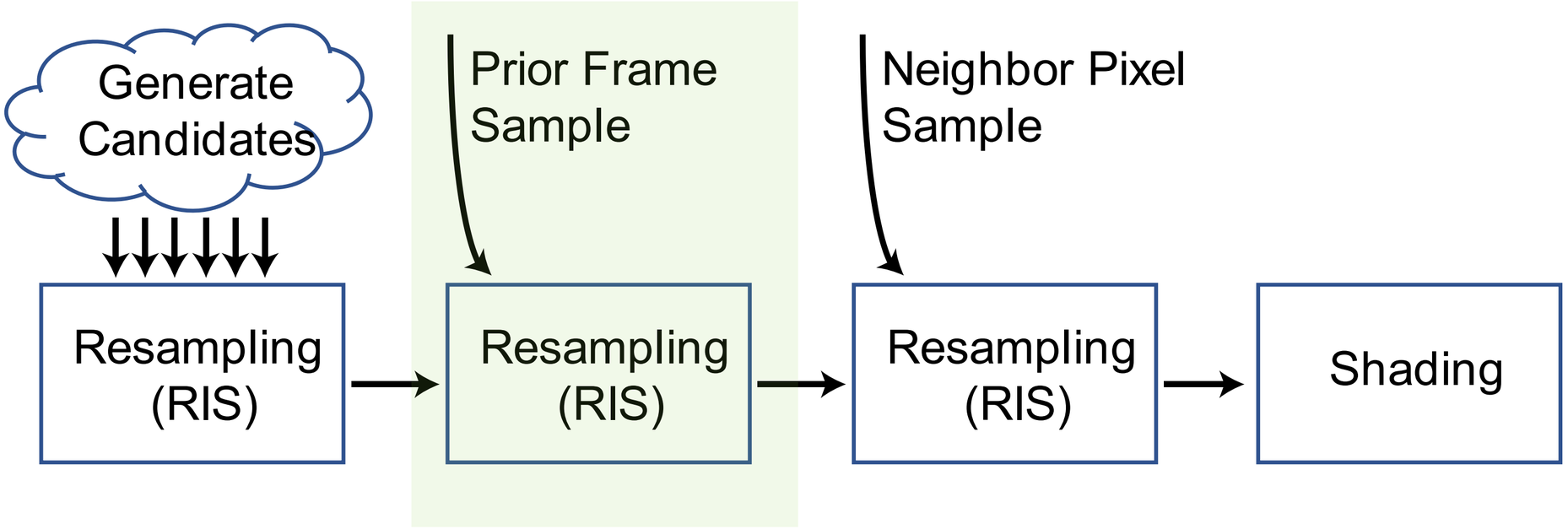
REUSING SAMPLES MORE INTELLIGENTLY

Consider the ReSTIR pipeline



REUSING SAMPLES MORE INTELLIGENTLY

Consider the ReSTIR pipeline

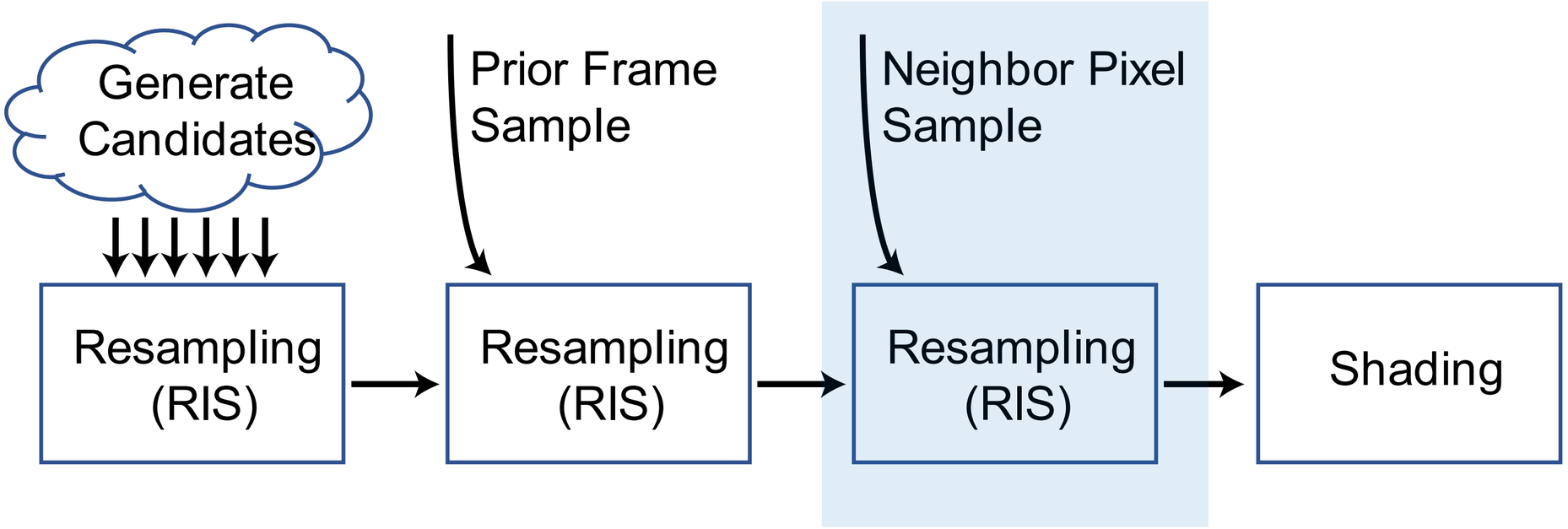


Interesting observation:
Temporal reuse most important


Initial Candidates Only
Spatial Reuse
Temporal Reuse
Both
MENU

REUSING SAMPLES MORE INTELLIGENTLY

Why use spatial reuse at all?



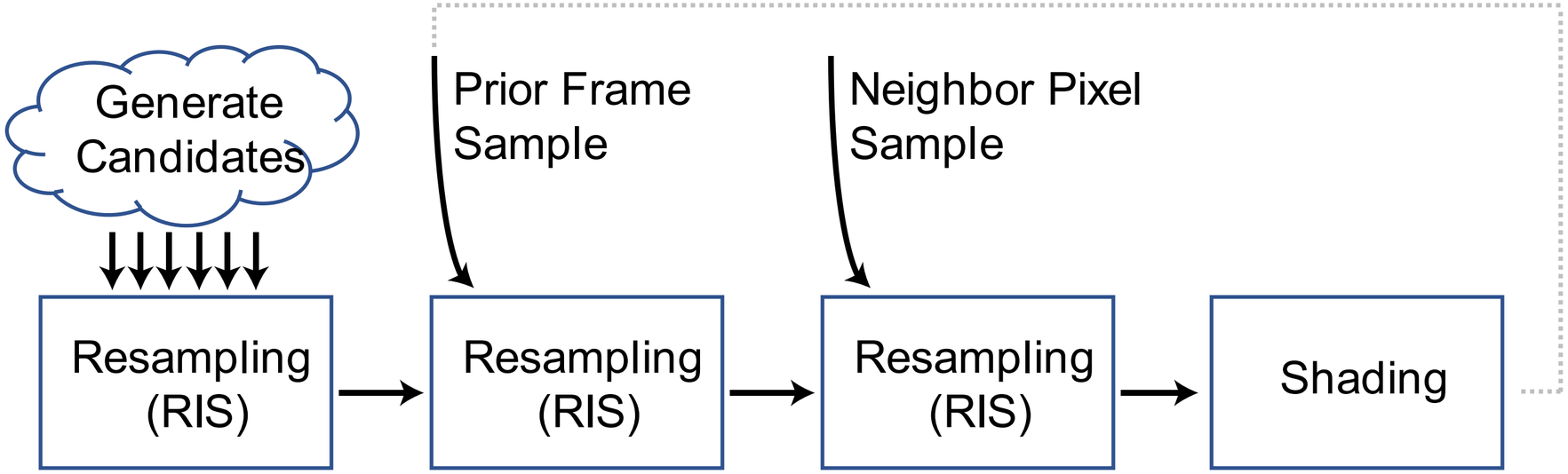
- Spatial reuse dithers out issues with temporal-only reuse
- Minimize this reuse; exactly one spatial tap seems sufficient (though 2 or 3 help with noise)

SHADING MORE INTELLIGENTLY

Other opportunities to reduce redundancy

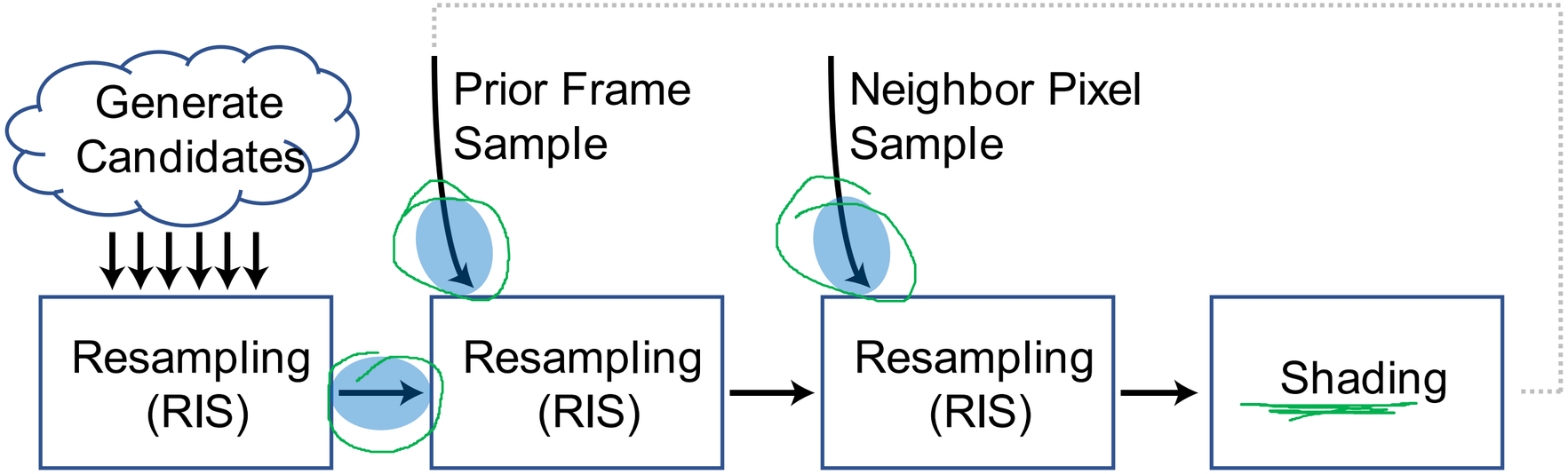
SHADING MORE INTELLIGENTLY

Other opportunities to reduce redundancy



SHADING MORE INTELLIGENTLY

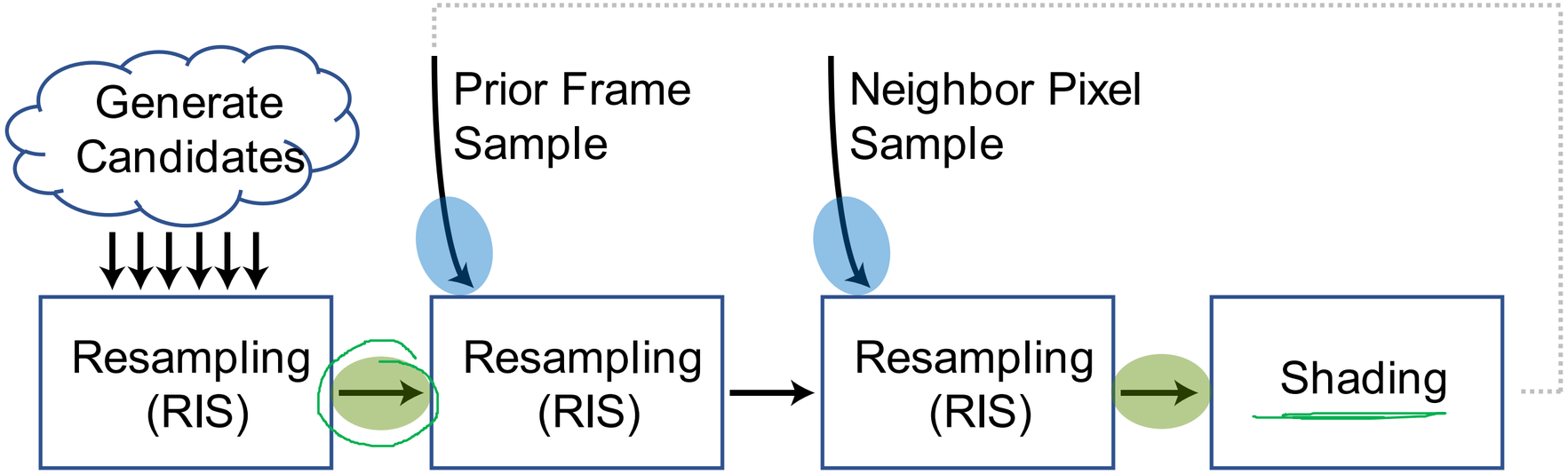
What's going on here?



- One of three samples is shaded

SHADING MORE INTELLIGENTLY

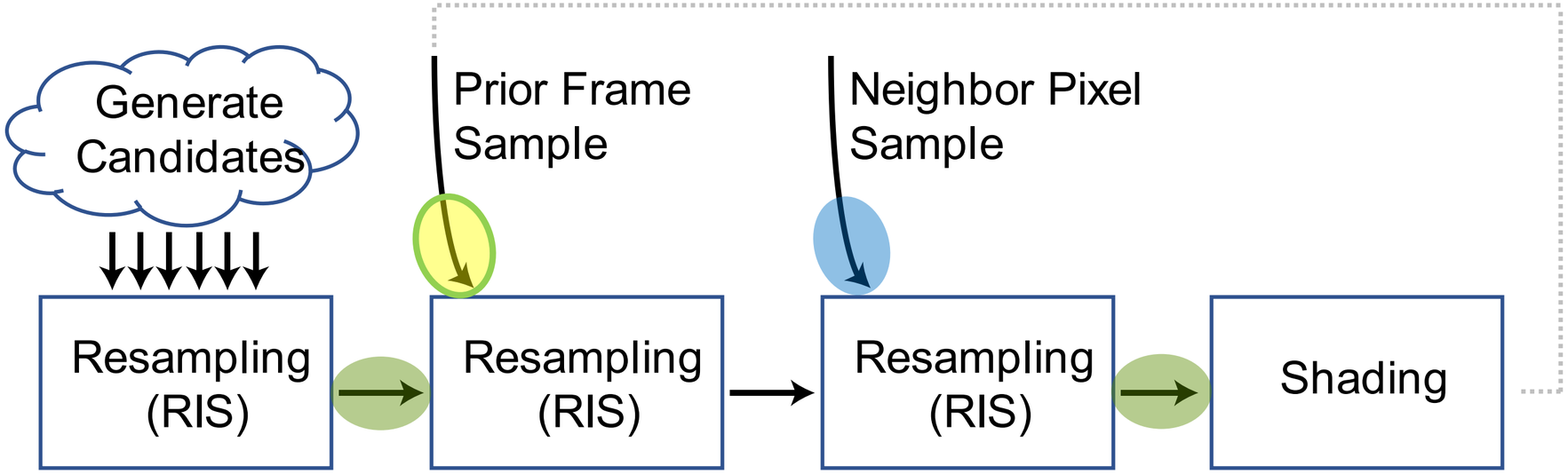
What's going on here?



- One of three samples is shaded
- Two shadow rays traced (a reasonable chance they're duplicates)

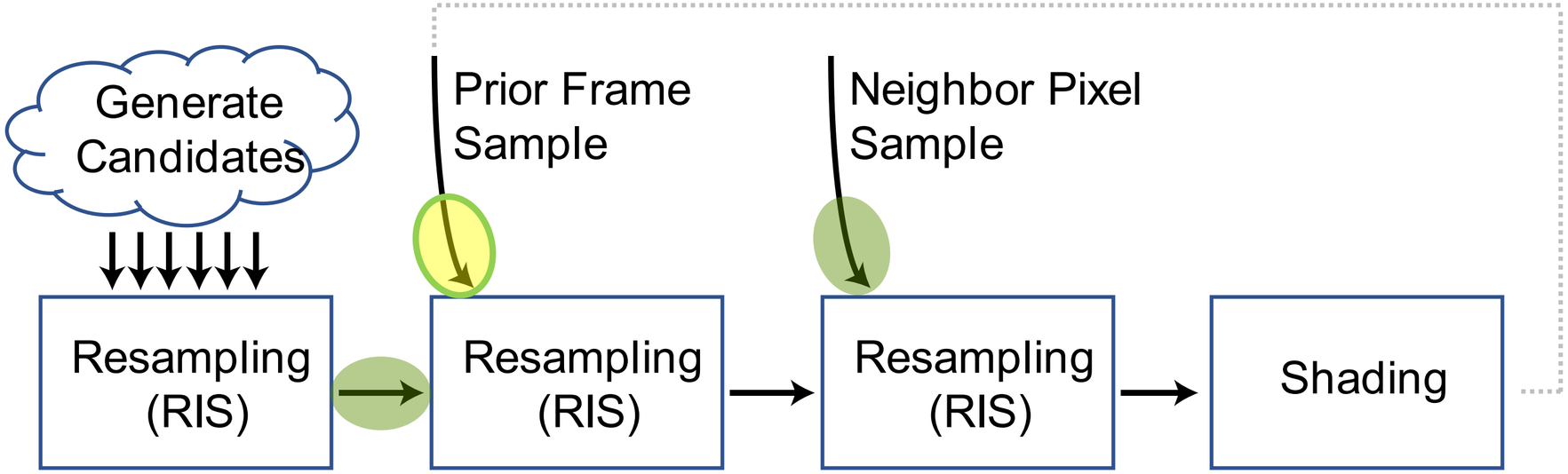
SHADING MORE INTELLIGENTLY

What's going on here?



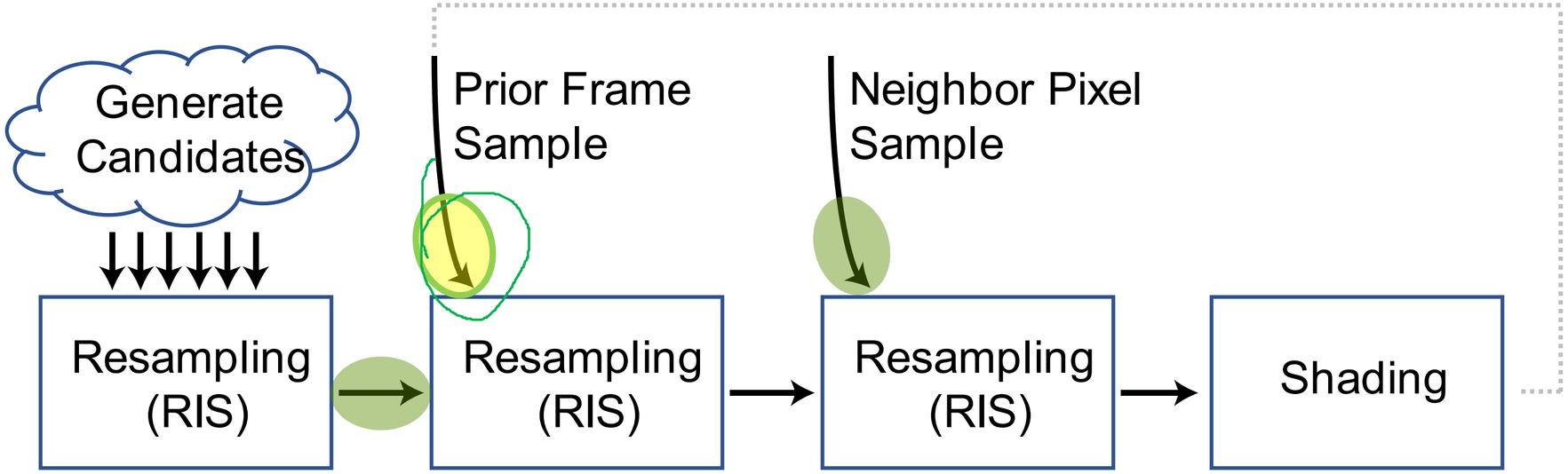
- One of three samples is shaded
- Two shadow rays traced (a reasonable chance they're duplicates)
- Have reasonable (if approximate) visibility from last frame

COULD WE REORGANIZE WHERE WE TRACE RAYS?



- Still 2 rays per pixel, but now has **3 shaded samples**

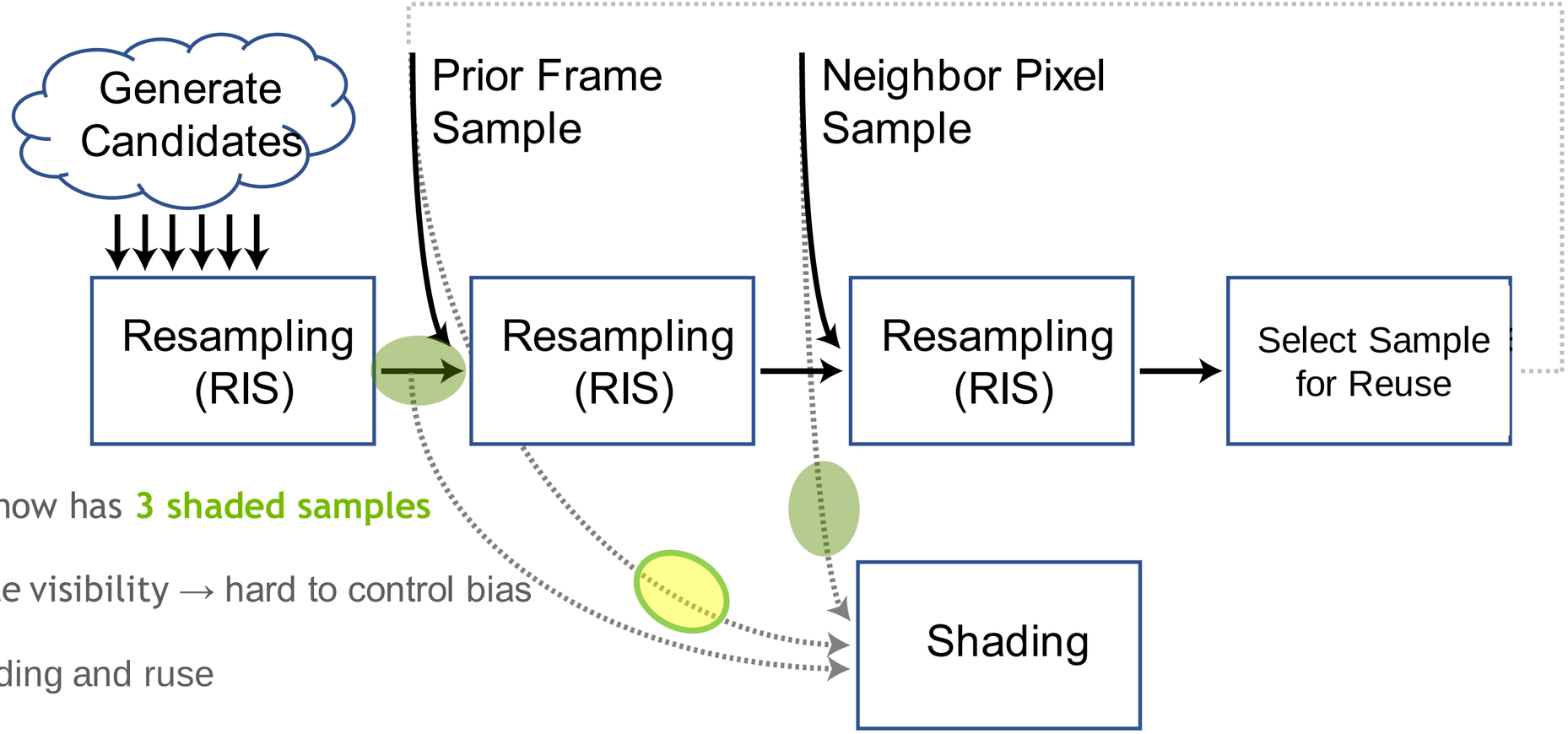
COULD WE REORGANIZE WHERE WE TRACE RAYS?



- Still 2 rays per pixel, but now has **3 shaded samples**
- Sadly, reusing approximate visibility → hard to control bias

IDEA: DECOUPLE REUSE AND SHADING

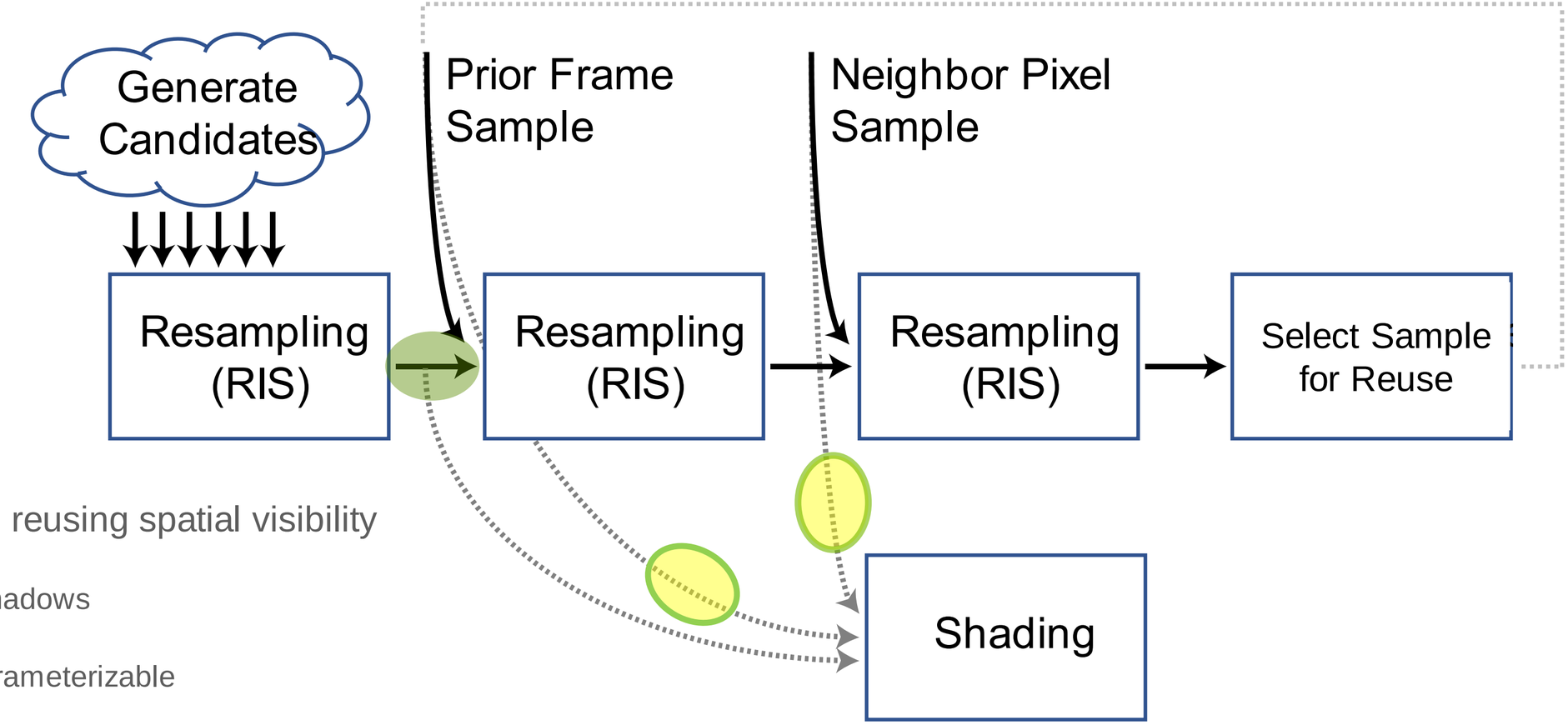
Visibility can be used ***differently*** for reuse and for shading



- Still 2 rays per pixel, but now has **3 shaded samples**
- Sadly, reusing approximate visibility → hard to control bias
- But we can **decouple** shading and ruse
  - Shading artifacts less noticeable; transient, just 1 frame

IDEA: DECOUPLE REUSE AND SHADING

Visibility can be used ***differently*** for reuse and for shading

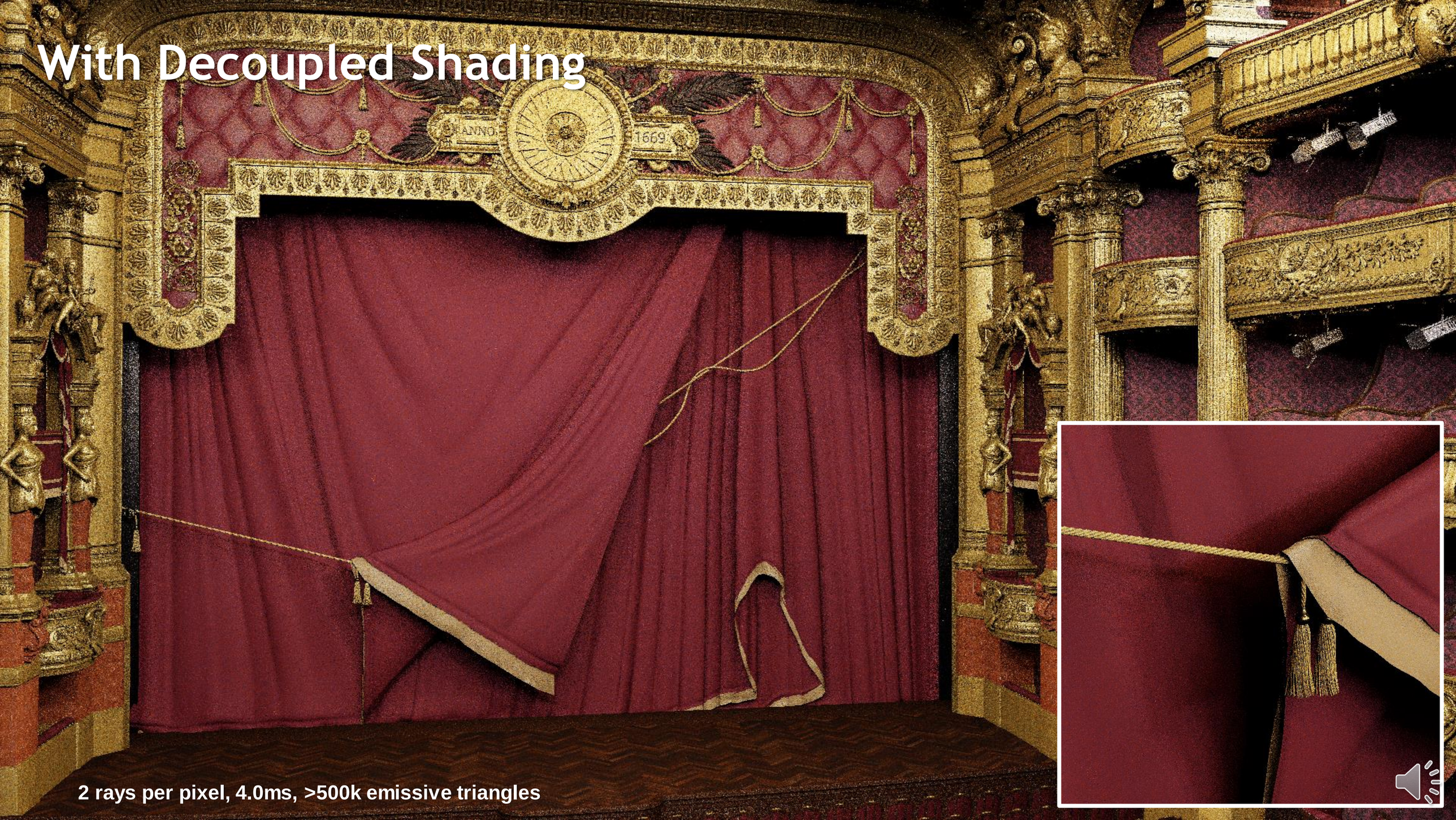


- Can be more aggressive, reusing spatial visibility
  - Noticeably lightens shadows
  - But degradation is parameterizable
  - Allows tuning down quality for fewer rays

Before Decoupled Shading

Paris Opera House, GoldSmooth from TurboSquid

With Decoupled Shading



2 rays per pixel, 4.0ms, >500k emissive triangles

SUMMARY

SUMMARY

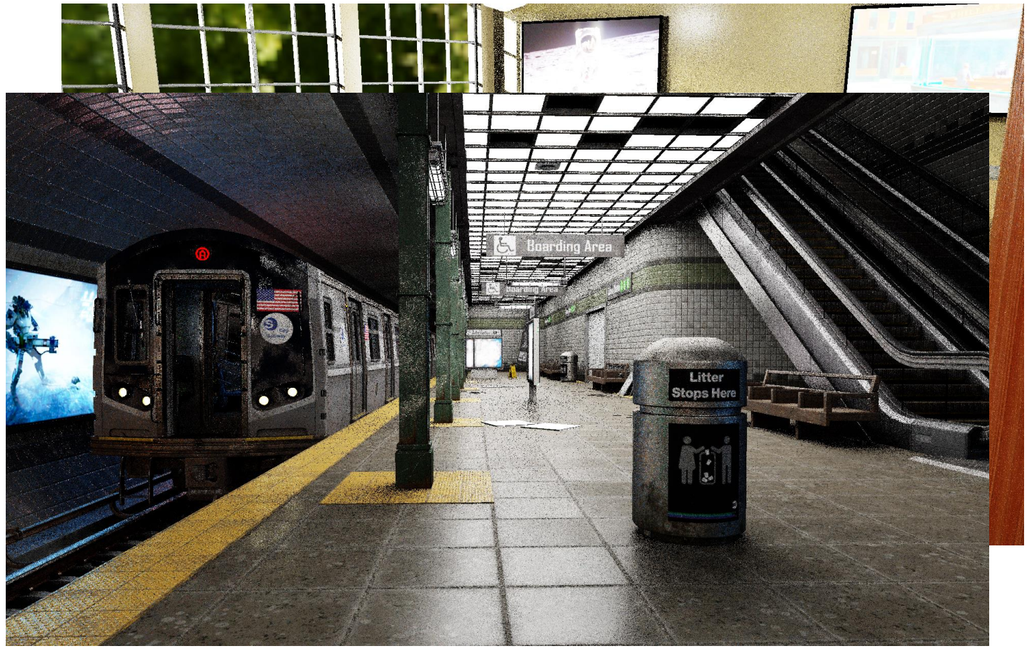
- RTXDI available now; take a look!

Classroom:
- 4 emissive meshes (13k tris) plus light probe
- 2 ms for lighting

SUMMARY

- RTXDI available now; take a look!

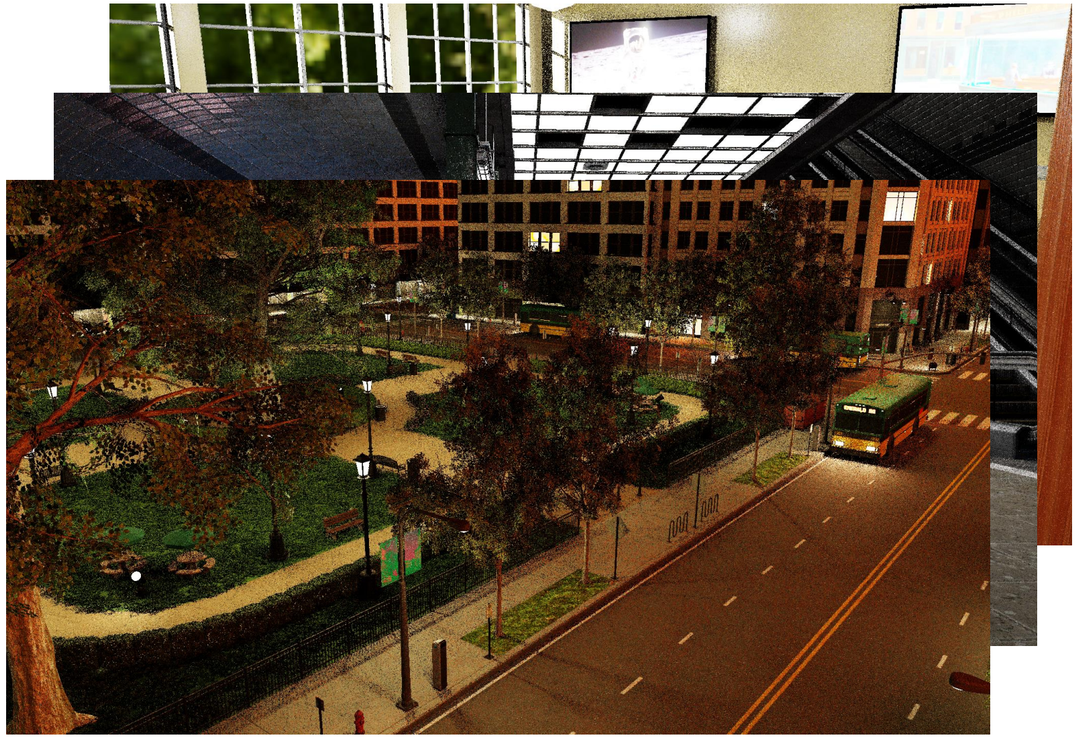
- We've seriously tuned perf since prior published numbers



Subway:
750 emissive meshes (25k tris)
2.2 ms for lighting

SUMMARY

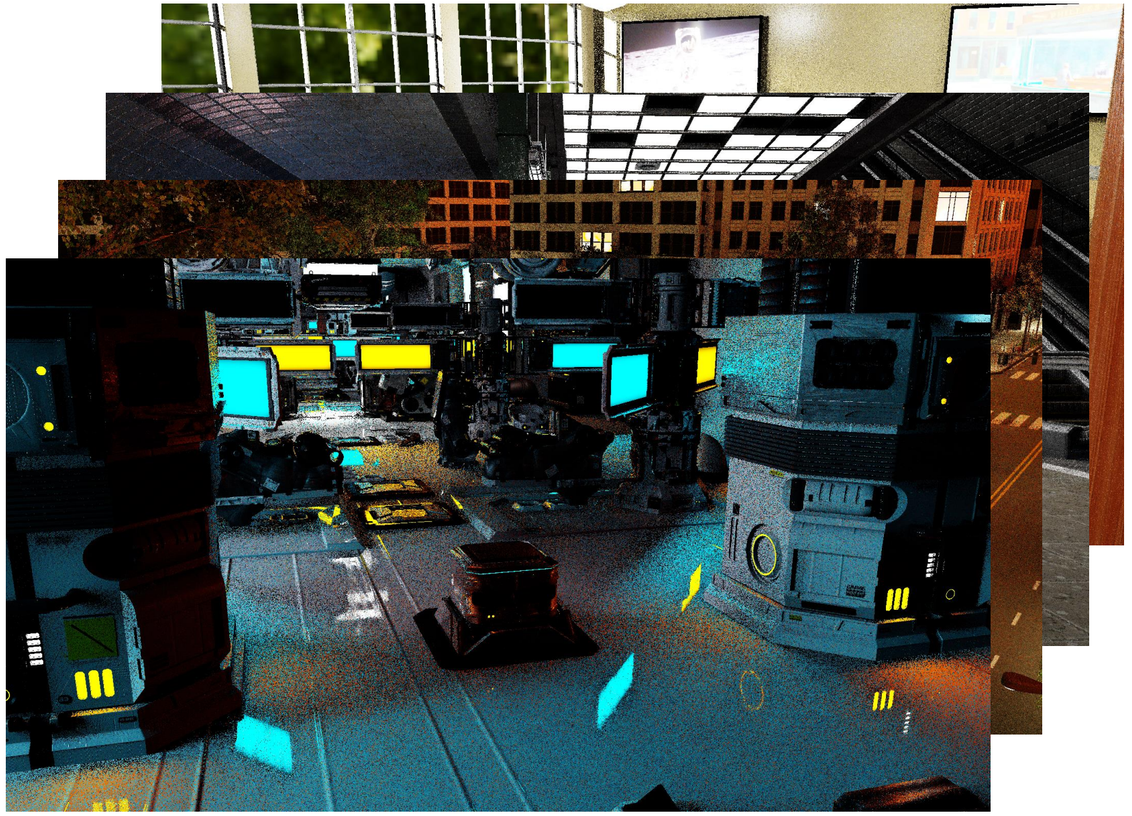
- RTXDI available now; take a look!
- We've seriously tuned perf since prior published numbers
- Big algorithmic changes:
  - Pre-randomize to reduce incoherency; remove from inner loop
  - Minimizing ray count
  - Improving quality at isoperf via decoupling



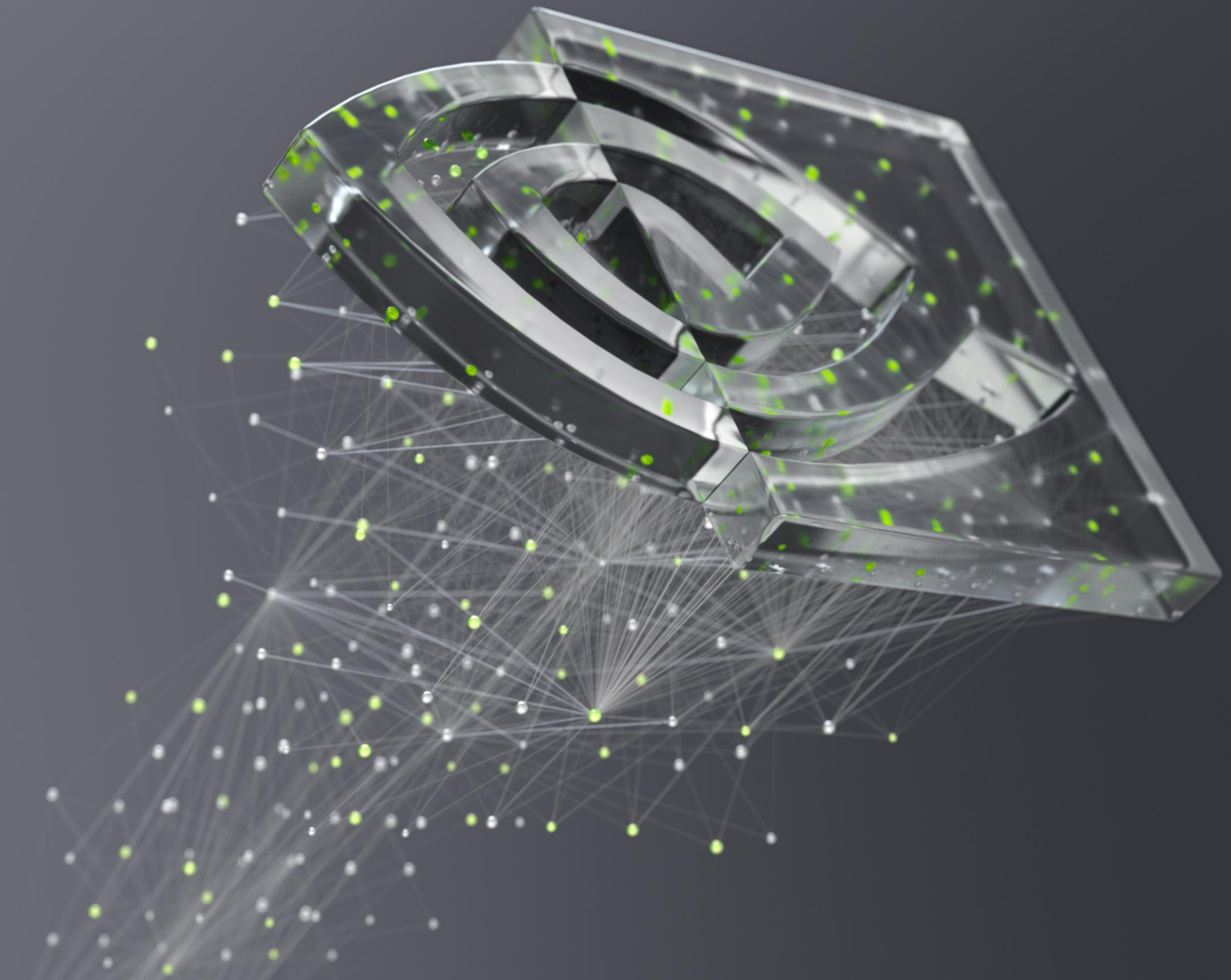
Emerald Square:
280 emissive meshes (89k tris)
5 ms for lighting

SUMMARY

- RTXDI available now; take a look!
- We've seriously tuned perf since prior published numbers
- Big algorithmic changes:
  - Pre-randomize to reduce incoherency; remove from inner loop
  - Minimizing ray count
  - Improving quality at isoperf via decoupling
- Remaining scene-to-scene perf deltas largely from ray cost



Zero Day:
384 emissive meshes (11k tris)
3 ms for lighting



More information:
https://developer.nvidia.com/rtxdi

E-mail:
cwyman@nvidia.com

Twitter:
@_cwyman_

GAME STACK LIVE

NVIDIA