CS482:
Instant Radiosity

Sung-Eui Yoon
(윤성의)

Course URL:
http://sglab.kaist.ac.kr/~sungeui/ICG

KAIST

Announcement

- **Mid-term exam**
  - **Closed book**
  - **1:00pm on Oct-20 at the class room**

Coming Schedule and Homework

- Declare the team at the noah board by Oct-5
- Browse recent papers (2012 ~ 2015)
  - You need to present two papers at the class
- Declare your chosen 2 papers at the board by Oct-14 (Wed.)
  - First come, first served
  - Paper title, conf. name, publication year
- Decide our talk schedule on Oct.-15 (Th)
- Student presentations will start right after the mid-term exam
  - 3 talks per each class; 20 min for each talk

KAIST

Modified tentative schedule

10월

27일 Q and A

29일 Student Presentation 1

11월

3 Student Presentation 2
5 Student Presentation 3

12 Mid. Project Presentation 1

17 Mid. Project Presentation 2
19 Student Presentation 1

24 Student Presentation 2

12월

1 Student Presentation 3
3 Q and A

8 Final Presentation 1

10 Final Presentation 2

KAIST

Presentation Guideline: Expectations

- **Good summary, not full detail, of the paper**
  - **Talk about motivations of the work**
  - **Give a broad background on the related work**
  - **Explain main idea and results of the paper**
  - **Discuss strengths and weaknesses of the method**

High-Level Ideas

- Deliver most important ideas and results
  - Do not talk about minor details
  - Give enough background instead

- Spend most time to figure out the most important things and prepare good slides for them
  - If possible, re-use existing slides/videos with ack.

Overall Structure

- **Prepare an overview slide**
  - **Talk about most important things and connect them well**

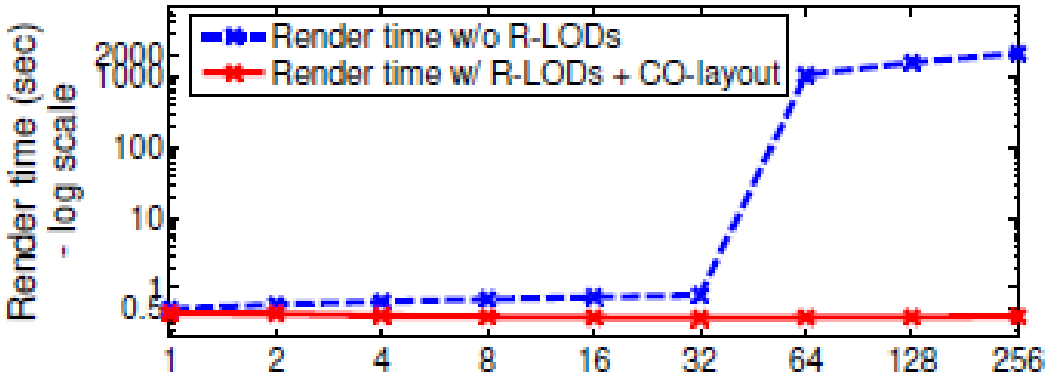
Be Honest

- Do not skip important ideas that you don’t know
  - Explain as much as you know and mention that you don’t understand some parts
- If you get questions you don’t know good answers, just say it
- In the end, you need to explain them before the semester ends

KAIST

Result Presentation

- **Give full experiment settings and present data with the related information**



- **After showing the data, give a message that we can pull of the data**
- **Show images/videos, if there are**

KAIST

Prepare a Quiz

- Give two simple questions to draw attentions
  - Ask a keyword
  - Simple true or false questions
  - Multiple choice questions
- Grade them in the scale of 0 and 10, and send the score to TA

Audience feedback form

Date:
Talk title:
Speaker:

A. Was the talk well organized and well prepared?
5: Excellent 4: good 3: okay 2: less than average 1: poor

B. Was the talk comprehensible? How well were important concepts covered?
5: Excellent 4: good 3: okay 2: less than average 1: poor

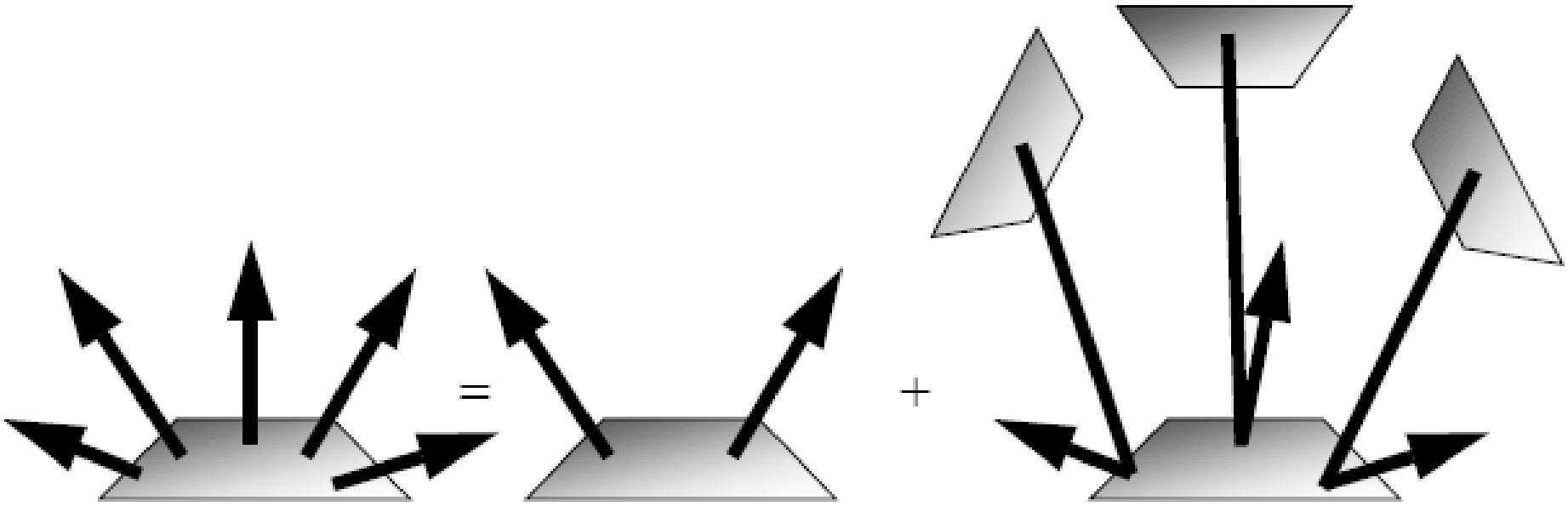
Any comments to the speaker

KAIST

Class Objective

- **Understand instant radiosity**
  - **Its general procedure**
  - **Its computational bottlenecks: shadow maps**
  - **Use imperfect shadow map and some recent techniques**

Radiosity Equation

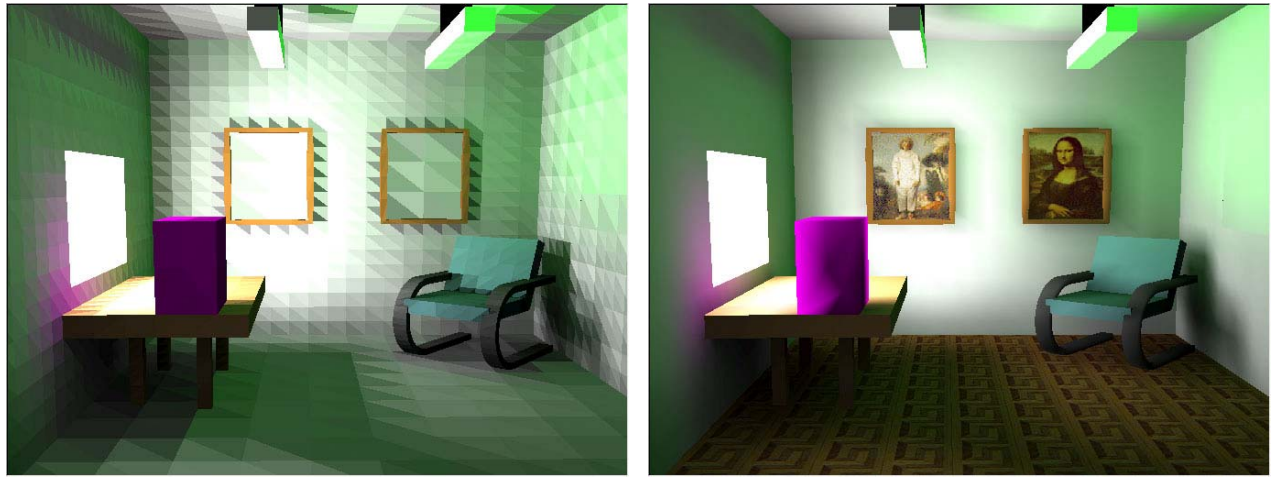


Emitted radiosity = self-emitted radiosity + received & reflected radiosity

$$Radiosity_i = Radiosity_{self,i} + \sum_{j=1}^{N} a_{j \to i} Radiosity_j$$

Radiosity Algorithm

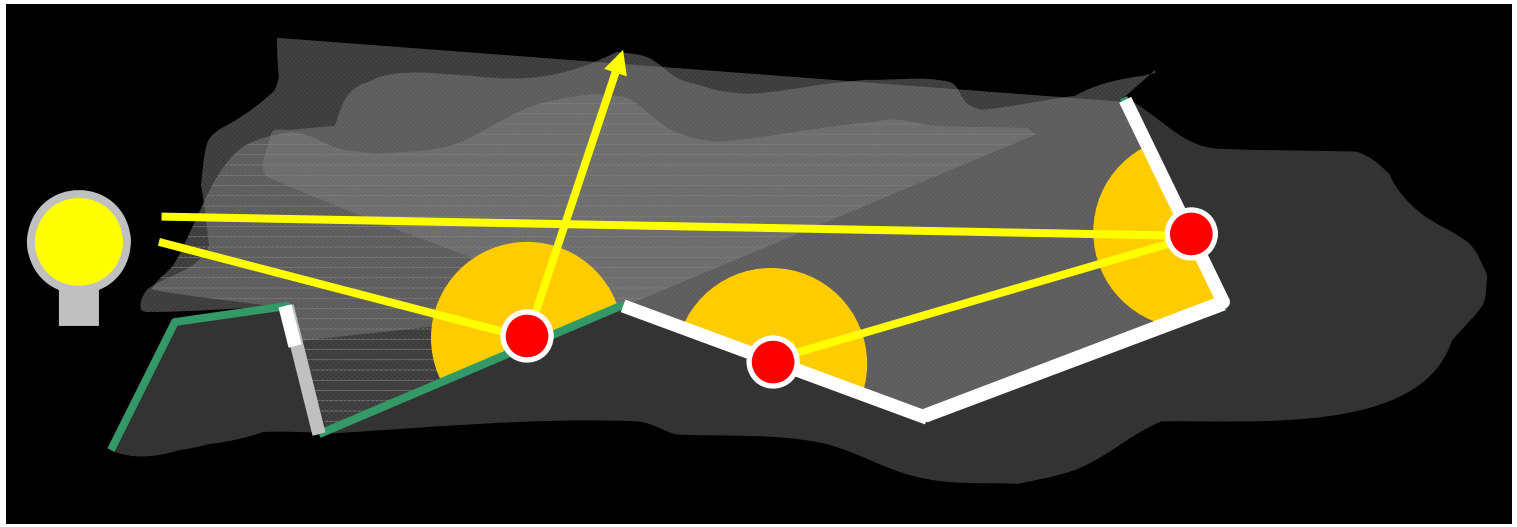
- Subdivide the scene in small polygons
- Compute a constant illumination value for each polygon
- Choose a viewpoint and display the visible polygon
  - Keep doing this process



From Donald Fong's slides

Instant Radiosity Howto

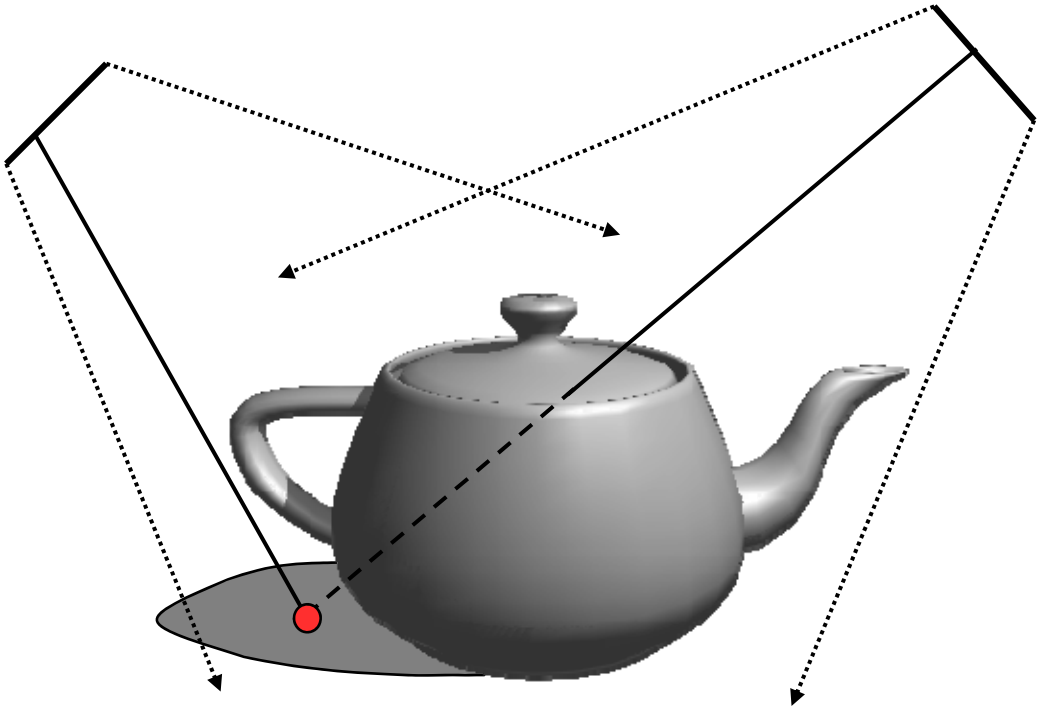
- Trace light paths from light source
- Place virtual point lights (VPLs) at intersections
- Render scene, use VPLs as 180º spots
- Global illumination ensues

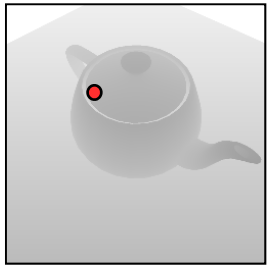


KAIST

Shadow maps [Williams 1978]

Eye





Light

KAIST

One-Bounce Indirect Illumination

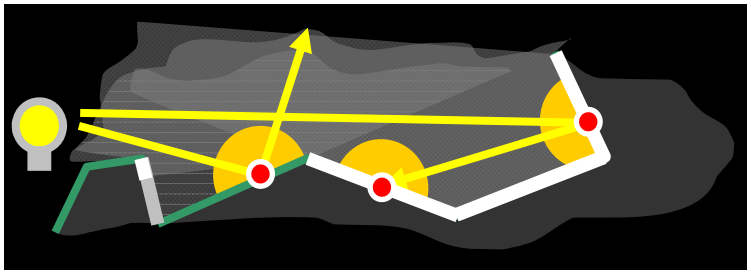
Tabellion and Lamorlette, SIGGRAPH 2004

- Officially close enough to full GI solution
- Terminate light paths at first intersection

KAIST

Baseline 1-Bounce Instant Radiosity

- Cast a bunch of rays from the light source
  - Rays must be distributed according to the emission function
- At each hit point, construct a VPL
  - Render shadow map (paraboloid)
  - Yes, that's a lot of shadow maps to render per frame
- Gather illumination from all VPLs
  - Yes, that's a lot of shadow map lookups per pixel



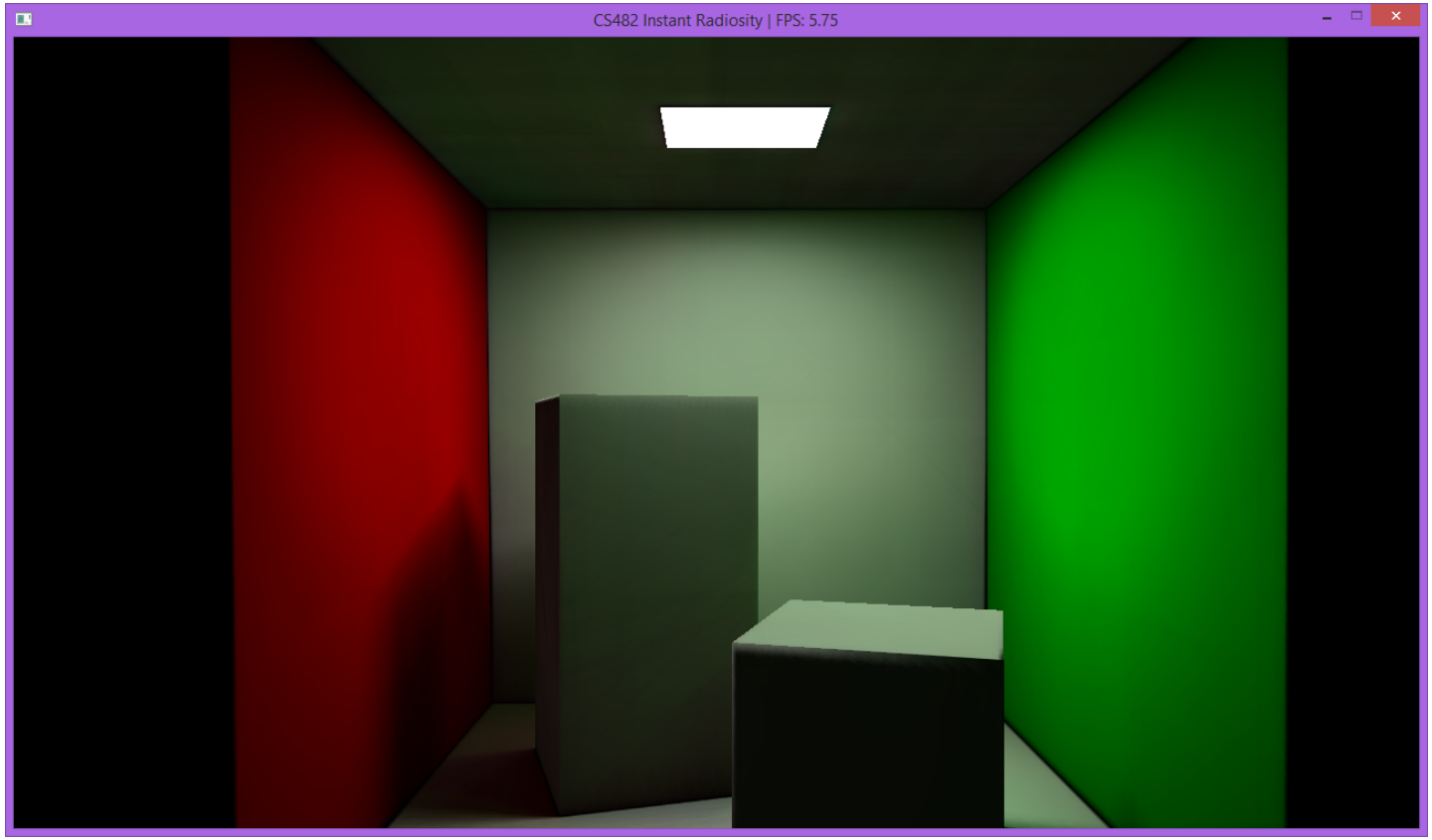
KAIST

Improved Instant Radiosity

- LOD or approximation techniques
  - Imperfect shadow maps
  - Point clouds

PA2

- **Compile and run our skeleton codes of instant radiosity**



KAIST

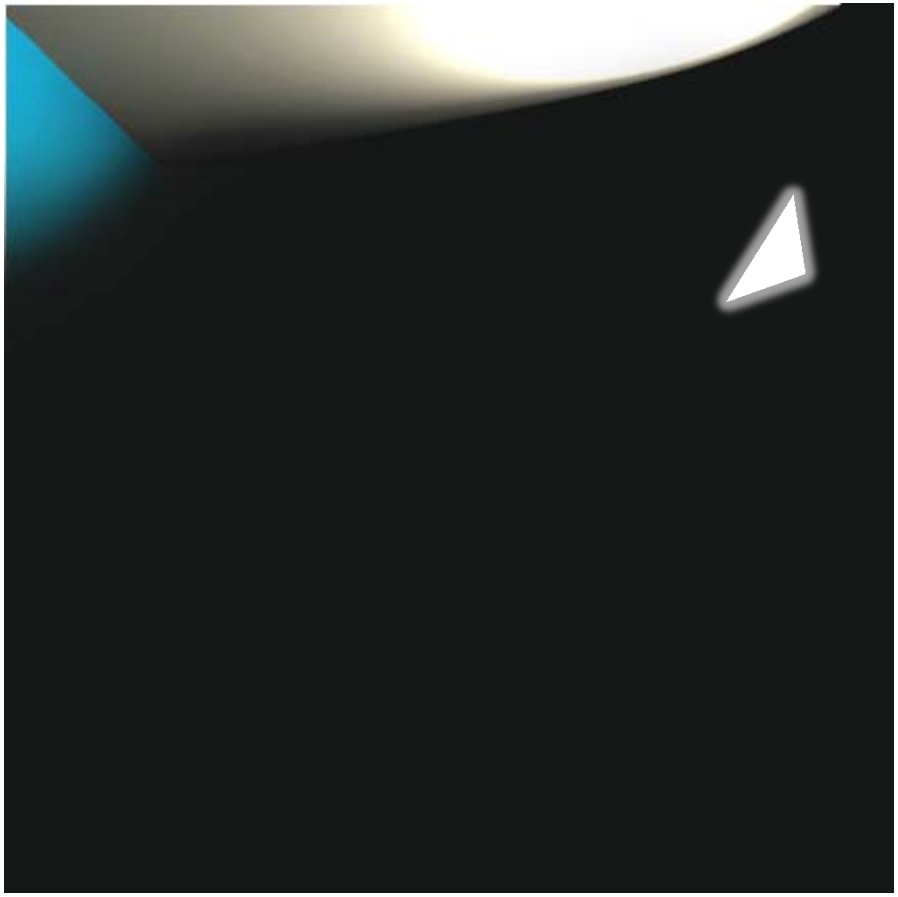
Imperfect Shadow Maps for Efficient Computation of Indirect Illumination

Tobias Ritschel et al.
Modfied from the author‘s slides

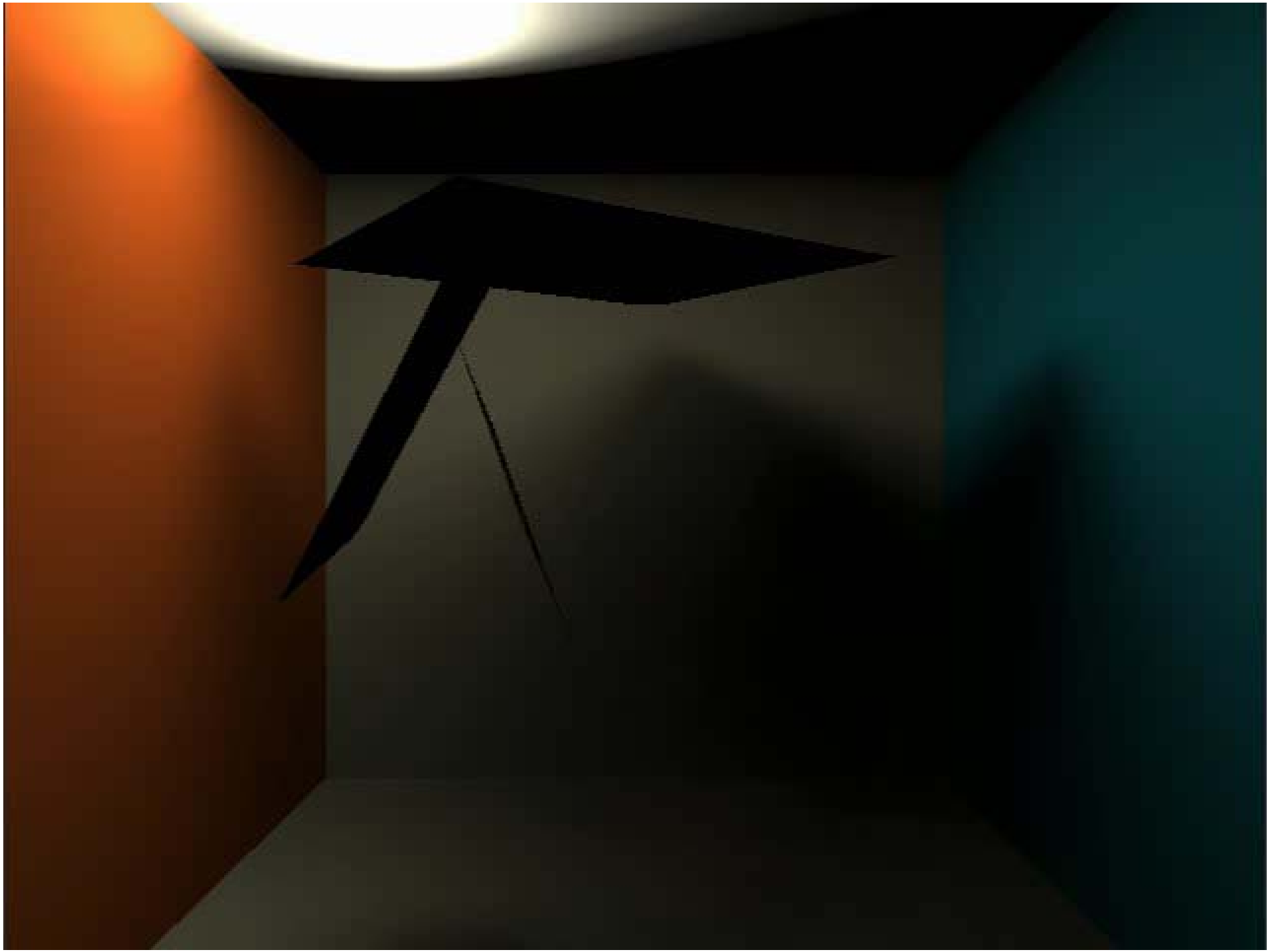
Motivation

Local illumination

Global illumination



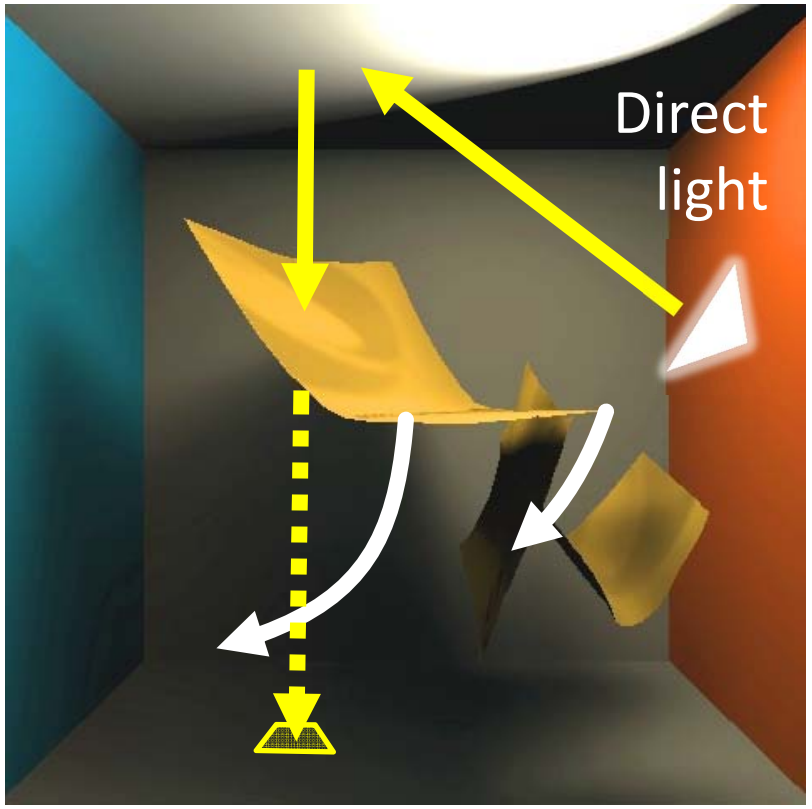
- Global illumination is perceptually important



Motivation

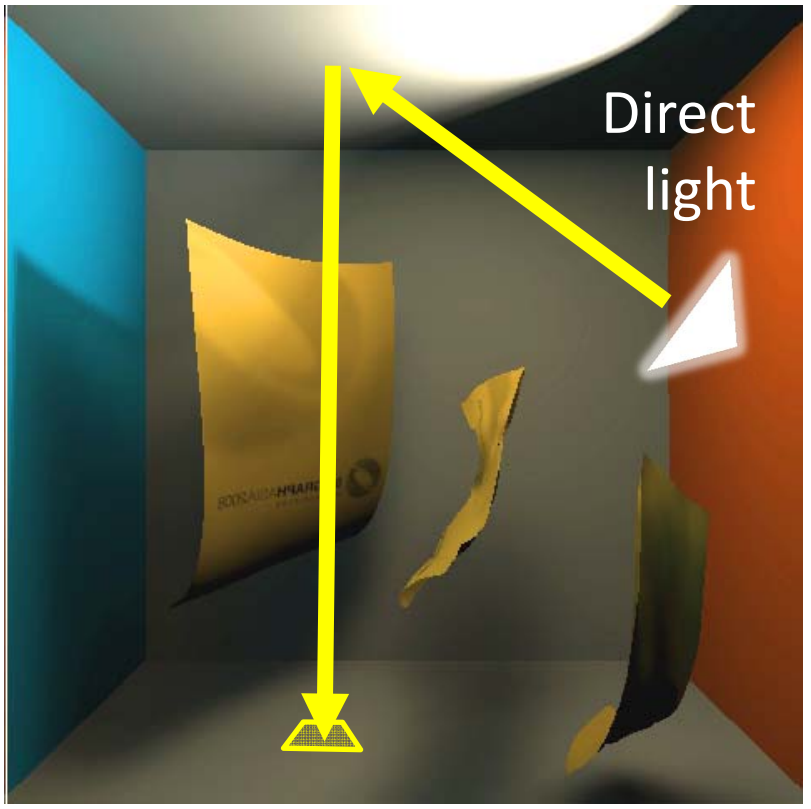
- Challenging: Dynamic indirect visibility

Frame *t*



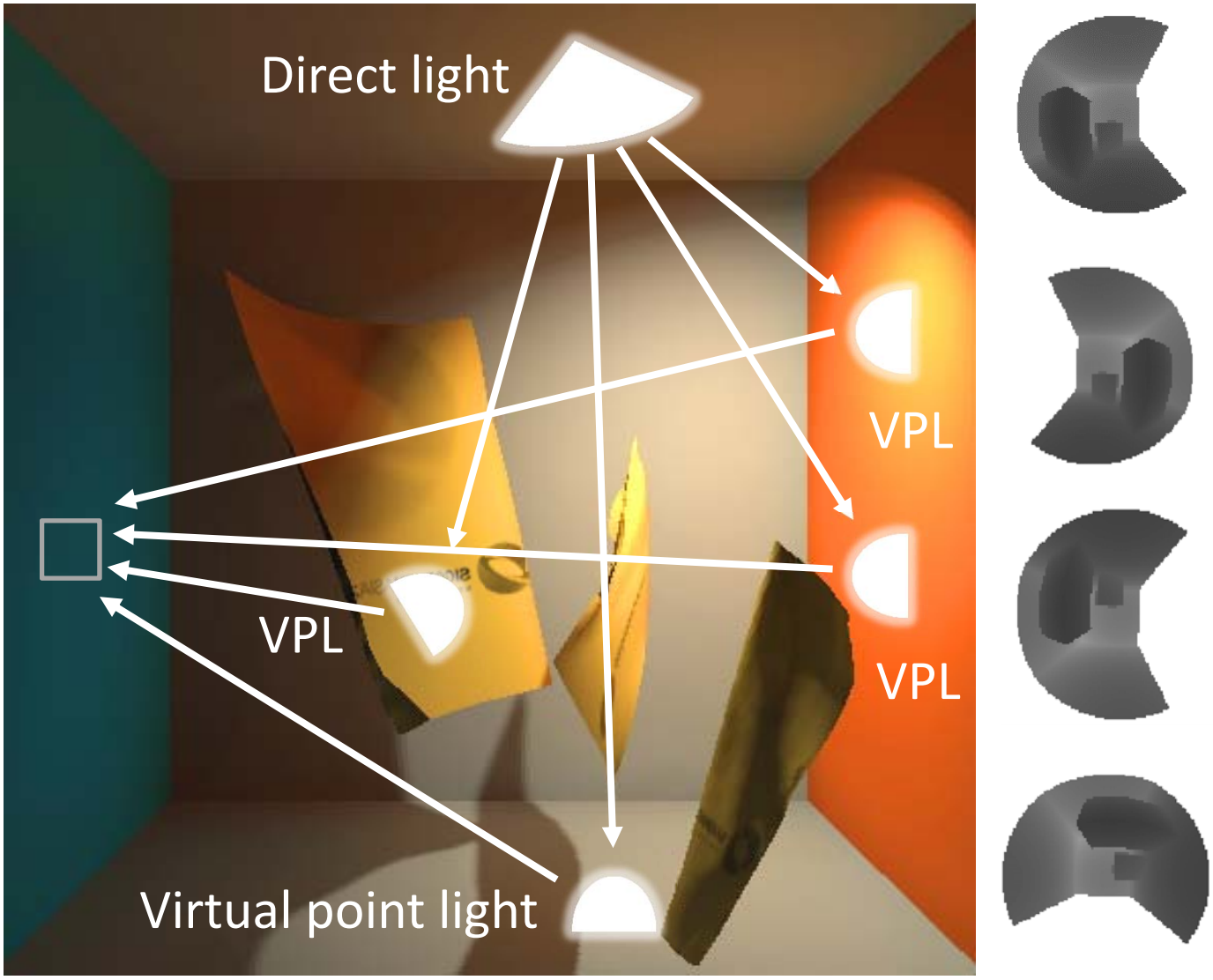
Indirect shadow

Frame *t*+1



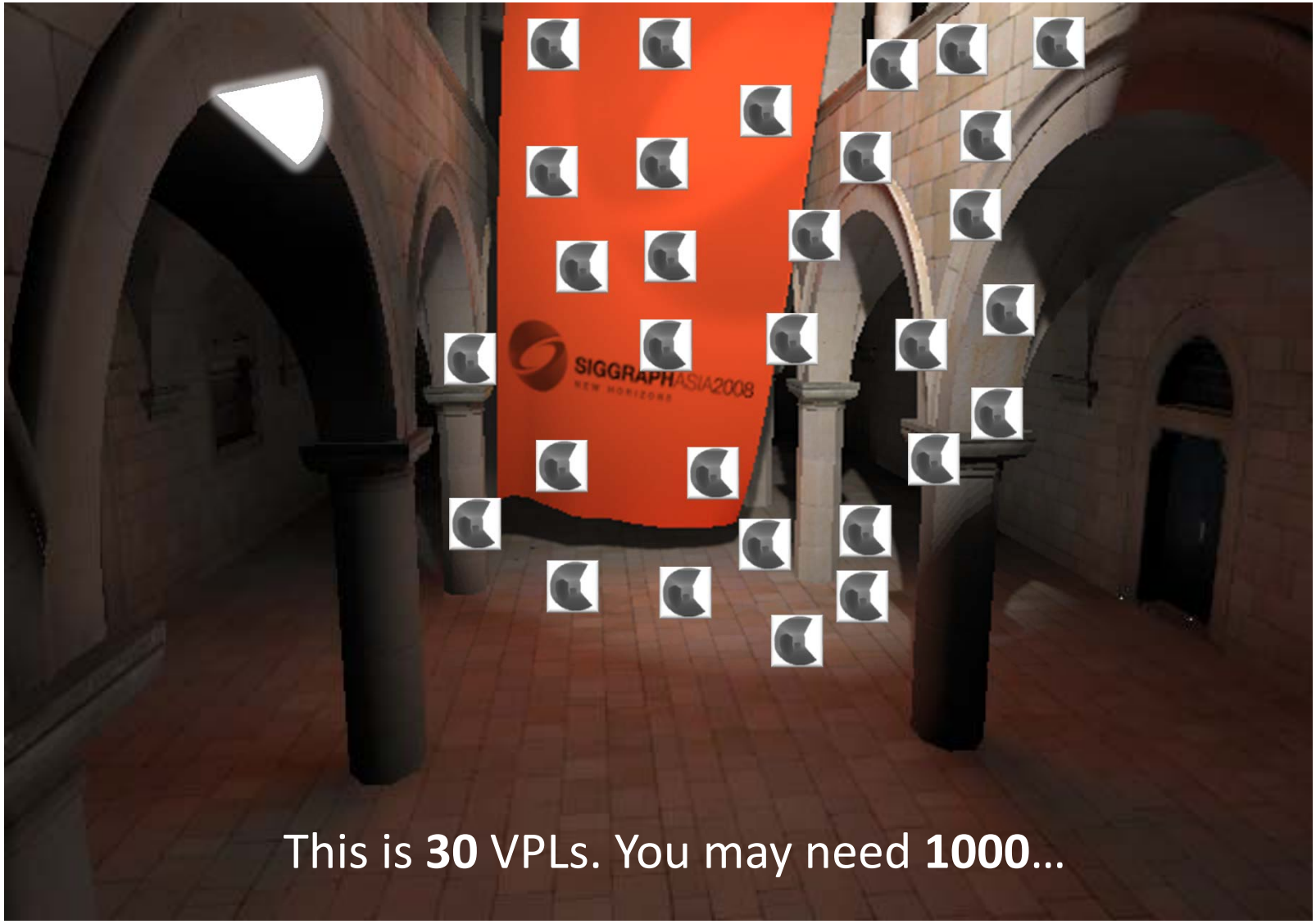
No indirect shadow

Instant Radiosity

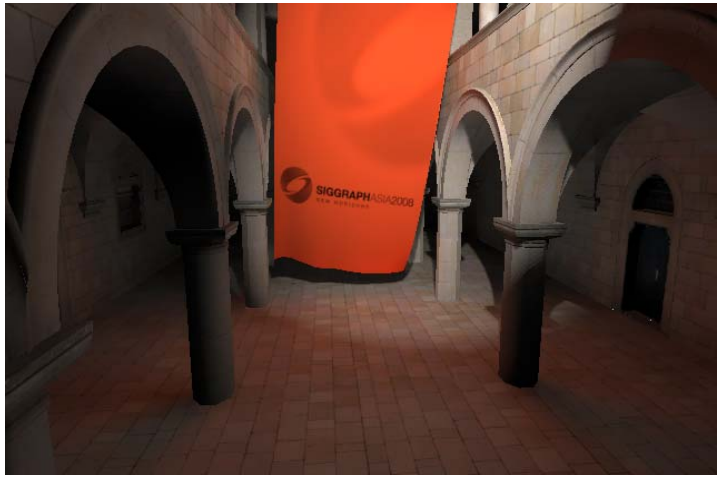


Indirect light

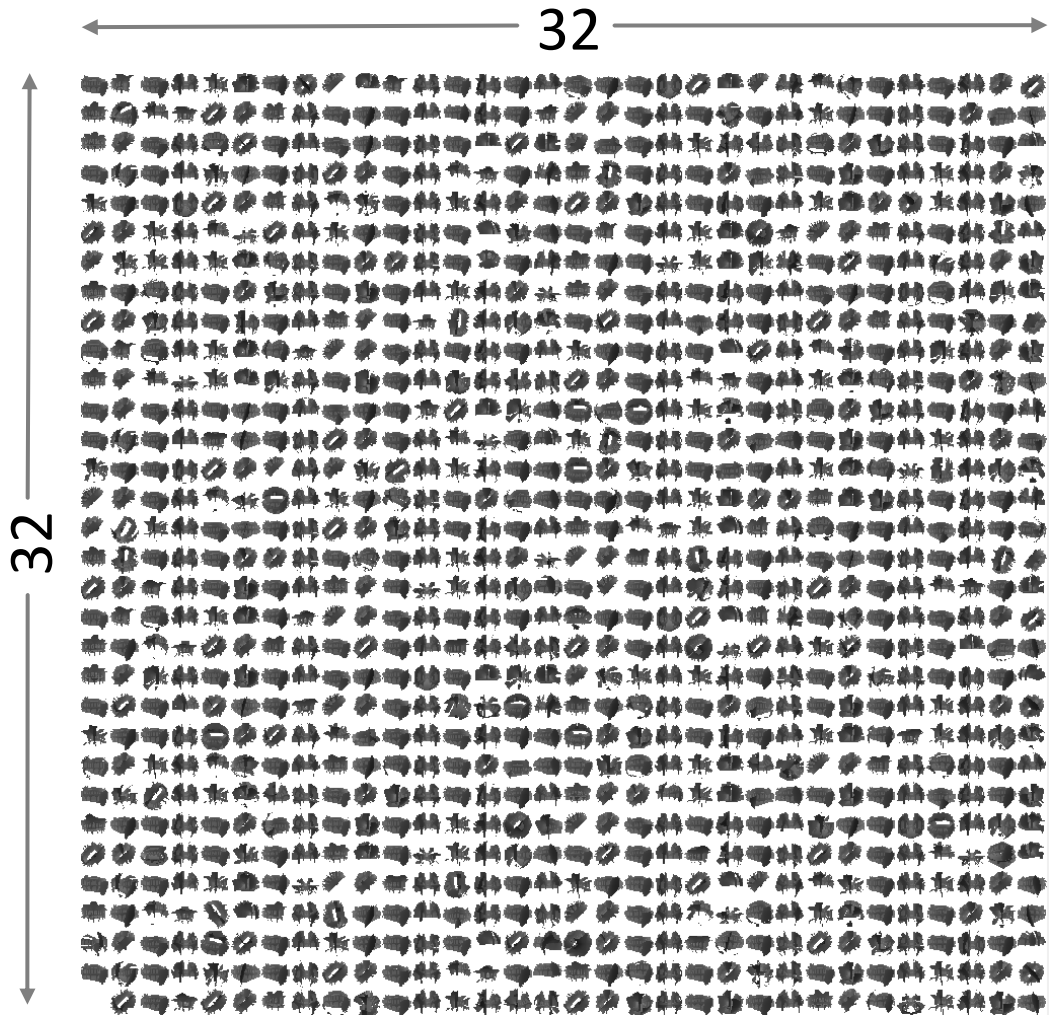
Instant Radiosity



Instant Radiosity bottleneck



- 1024 VPLs
- 100k 3D model
- 32x32 depth map
- 100x overdraw



Imperfect shadow maps

- Our **observations**:
  Low quality (imperfect) depth maps sufficient for many faint VPLs that form smooth lighting
- Our **contribution**:
  Efficient generation of low quality depth maps
- Main steps (detailed next)
  1. VPL generation
  2. Point-based depth maps
  3. Pull-push to fill holes
  4. Shading

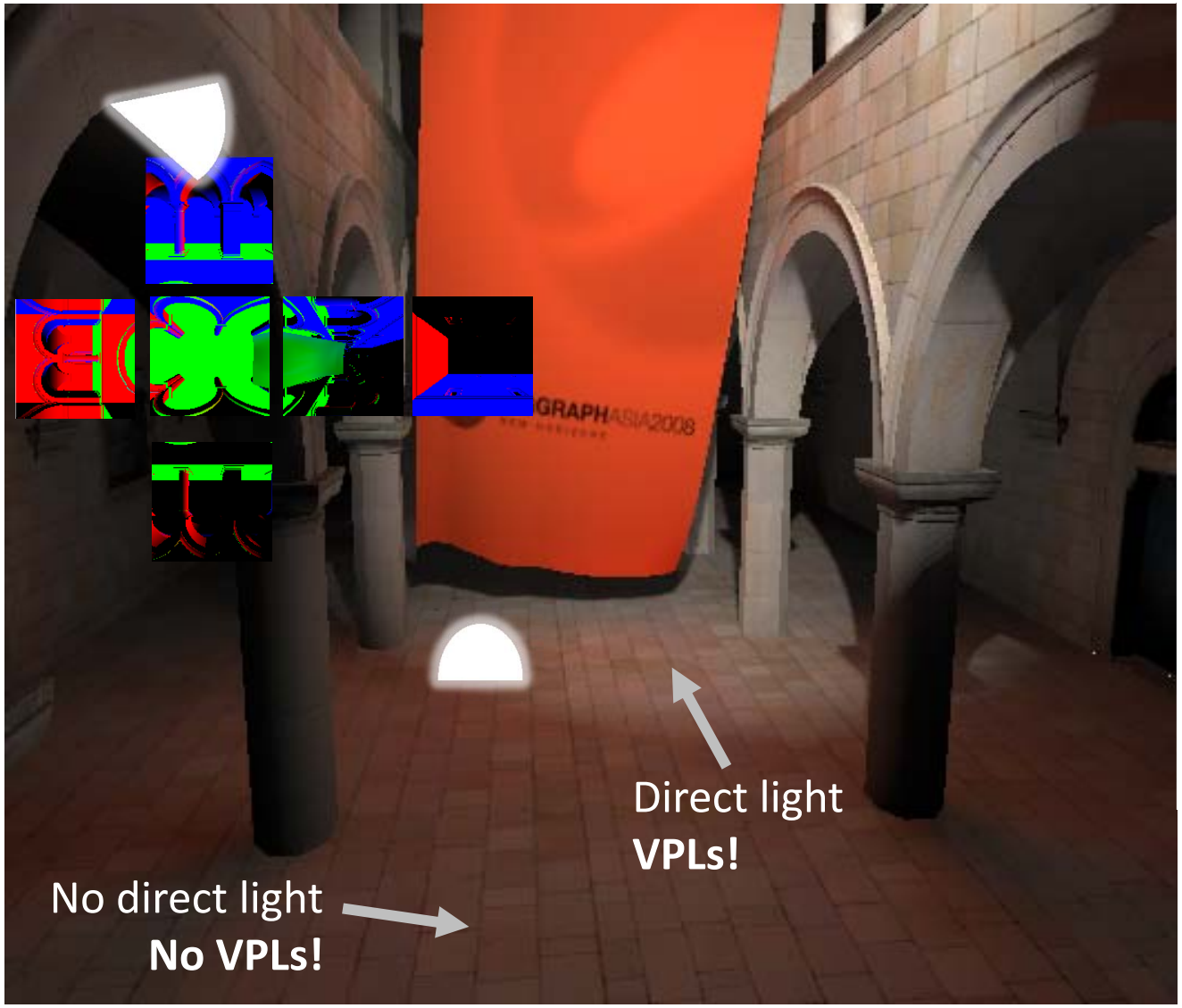
Step 1 VPL generation

Step 2 Point based depth maps

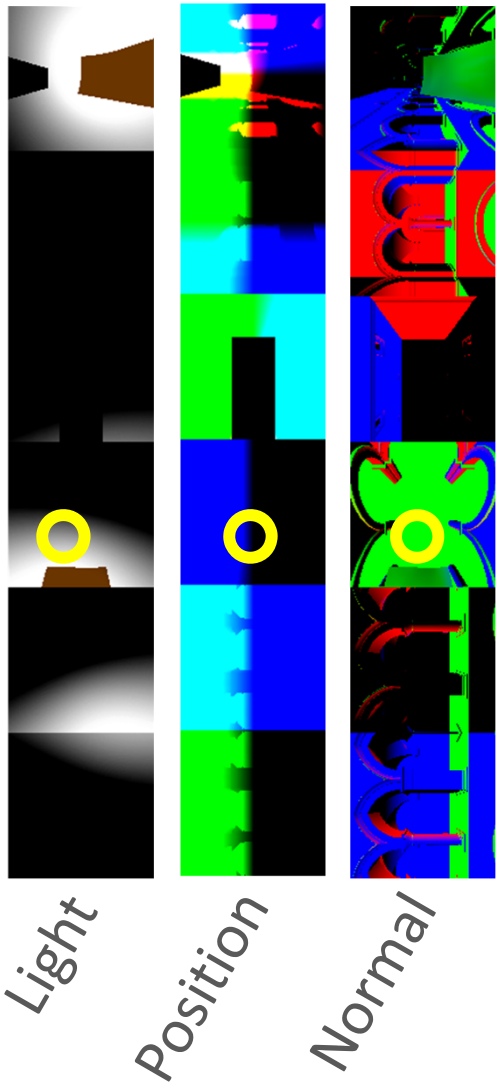
Step 3 Pull push

Step 4 Shading

Step 1: VPL generation



Render the scene into the cube map



Environment or Shadow Mapping

- Cube Maps
  - Low distortion
  - Accelerated by GPU
  - Introduces Seams

Environment or Shadow Mapping

- Dual Paraboloid

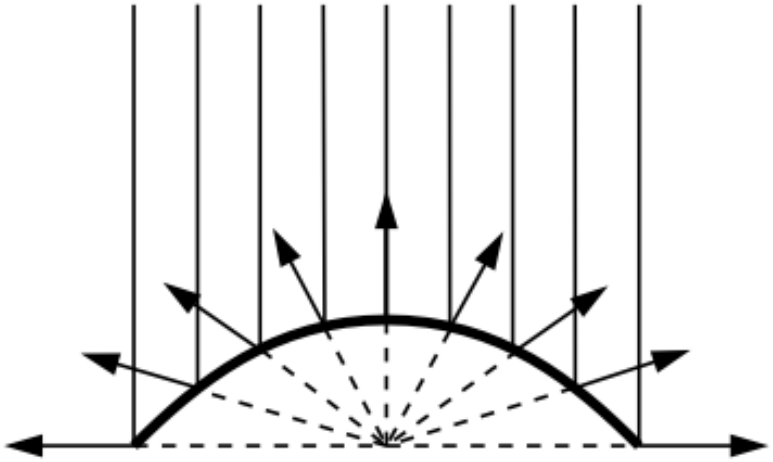
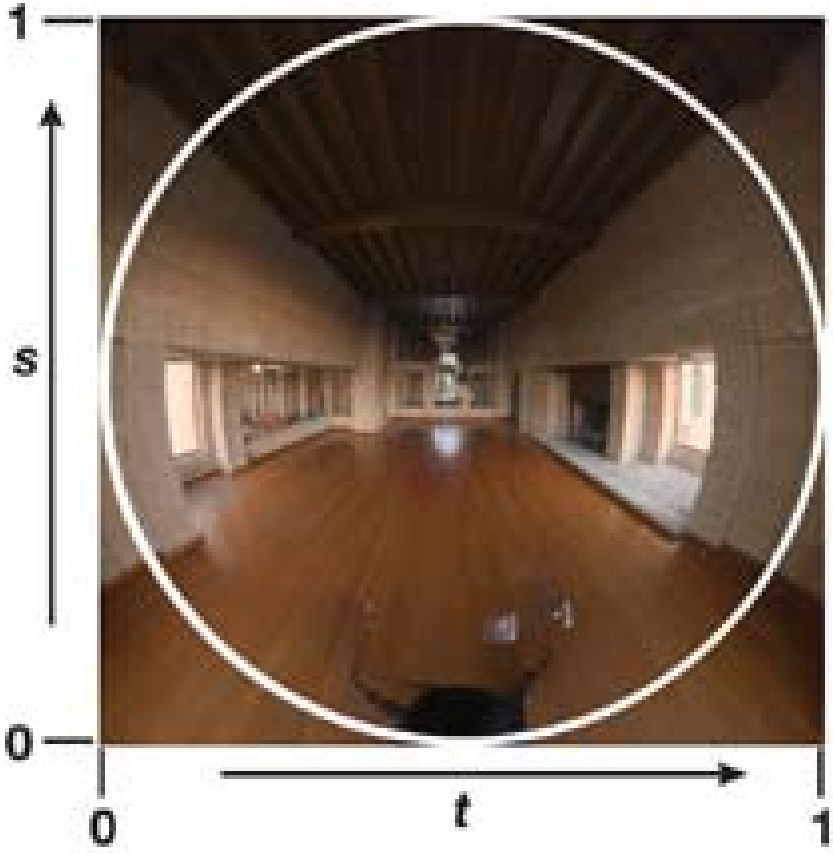


Figure 2: The rays of an orthographic camera reflected on a paraboloid sample a complete hemisphere of directions.



Step 1 VPL generation

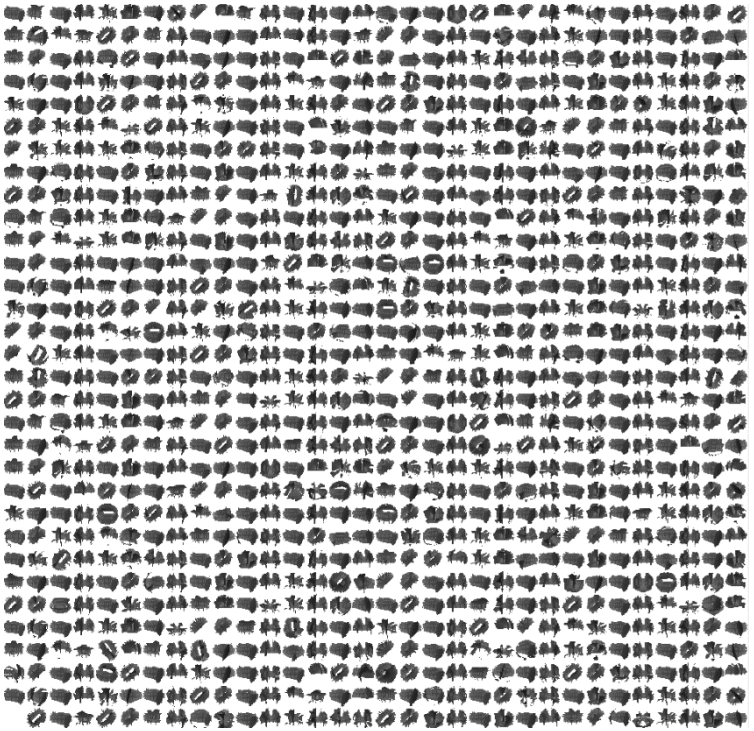
Step 2 Point based depth maps

Step 3 Pull push

Step 4 Shading

Step 2: Point-based depth maps

- **Goal:** Fill ~1000 depth maps for every frame?
  - Classic approach takes ca. 500ms for “Sponza”
  - As correct as possible
  - Using only little bandwidth
- **Solution:** Simplify!
  - Use points (no connectivity)
  - As many as there are pixels



Step 2: Point-based depth maps

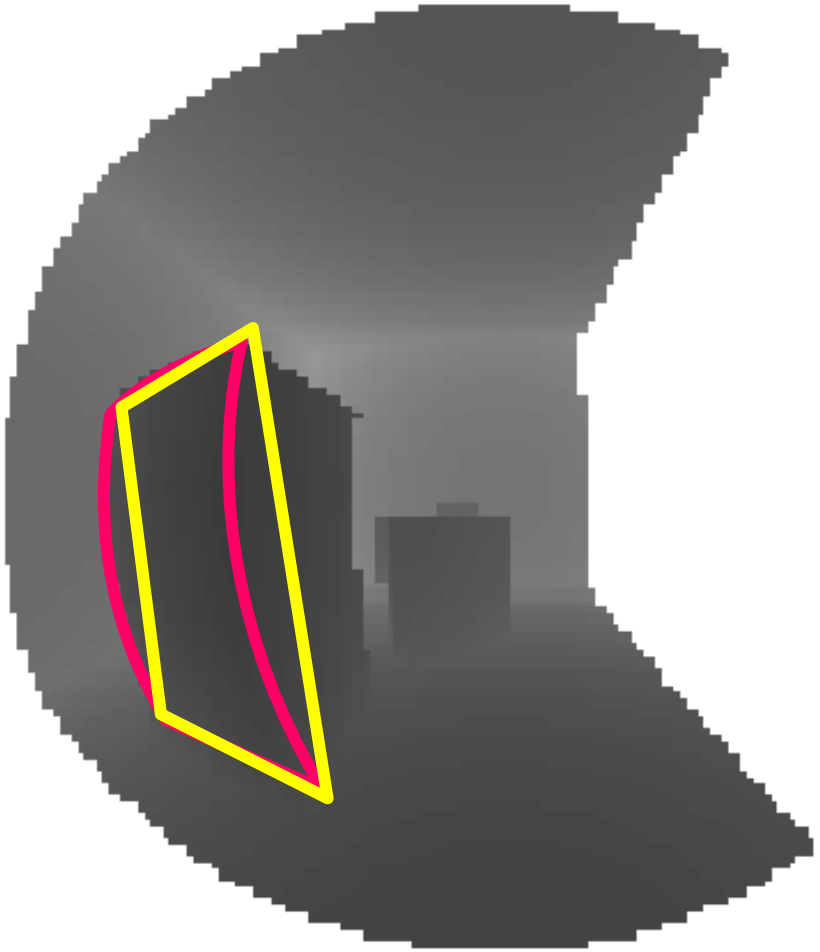
Classic

Imperfect

Imperfect
Smaller points
Less points

Step 2: Point-based depth maps

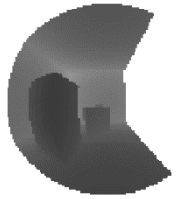
- Paraboloid depth maps
  - Single pass
  - Only with areas (e.g. tris)
- No tessellation problem
  - Tris need to be subdivided
  - Not with points



Step 2: Point-based depth maps

- **Pre-process**:
  Distribute points on surface
  - ~8k points for every VPL
  - Different set for every VPLs
- **At runtime**:
  Deform this distribution

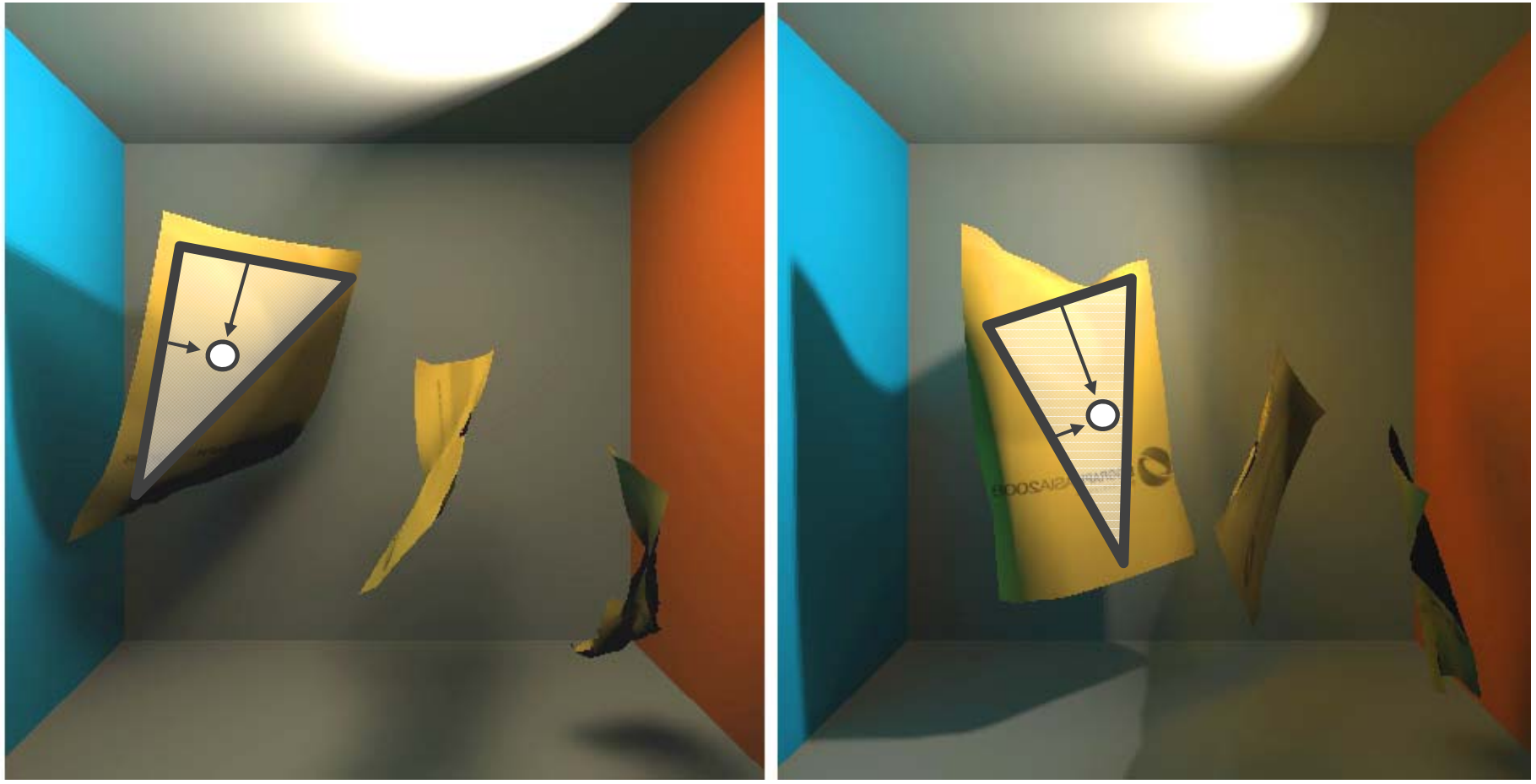
VPL / Depth map



Step 2: Point-based depth maps

Frame *t*

Frame *t*+1



- Store points relative to tris: deforms on-the-fly

Step 1 VPL generation

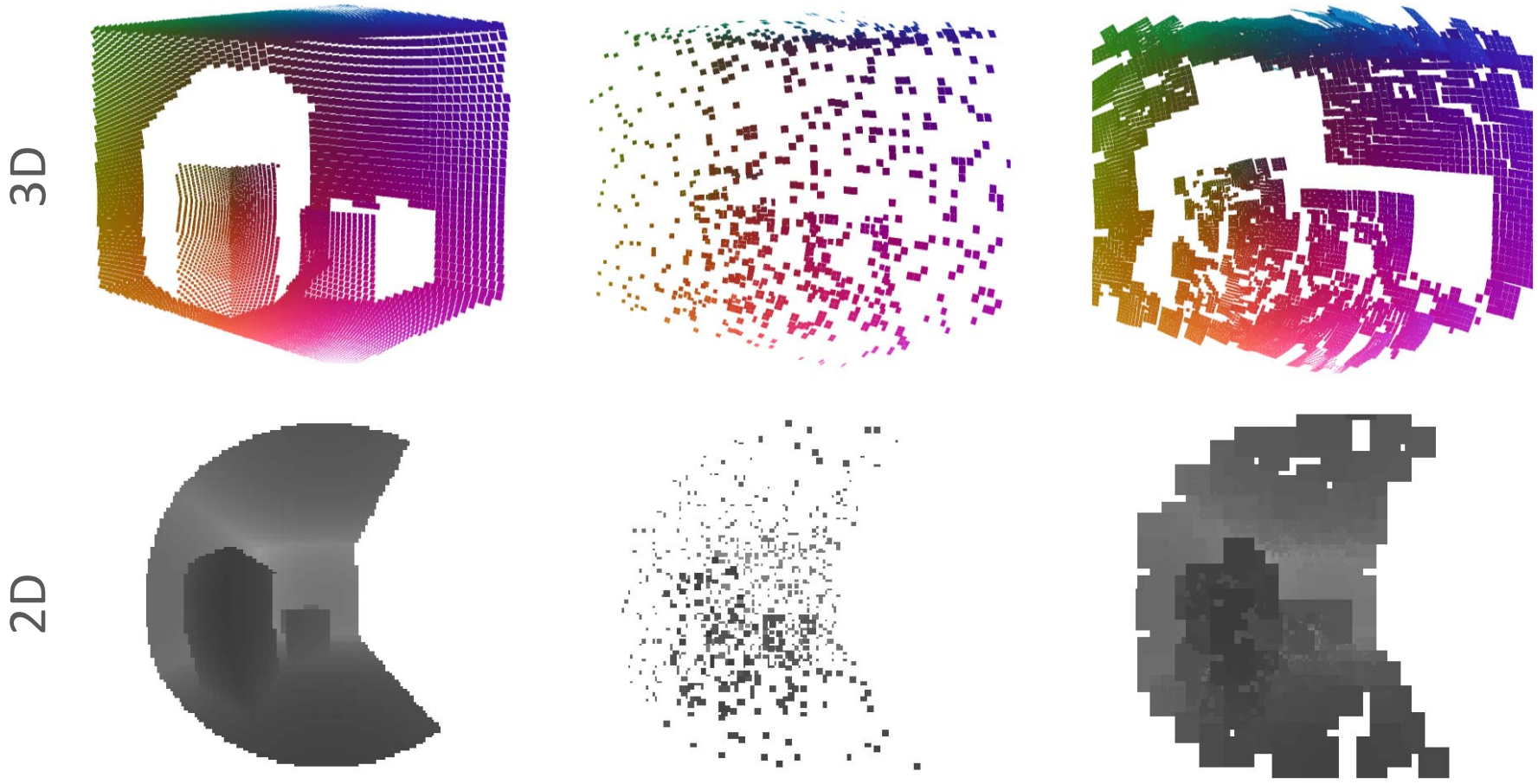
Step 2 Point based depth maps

Step 3 Pull push

Step 4 Shading

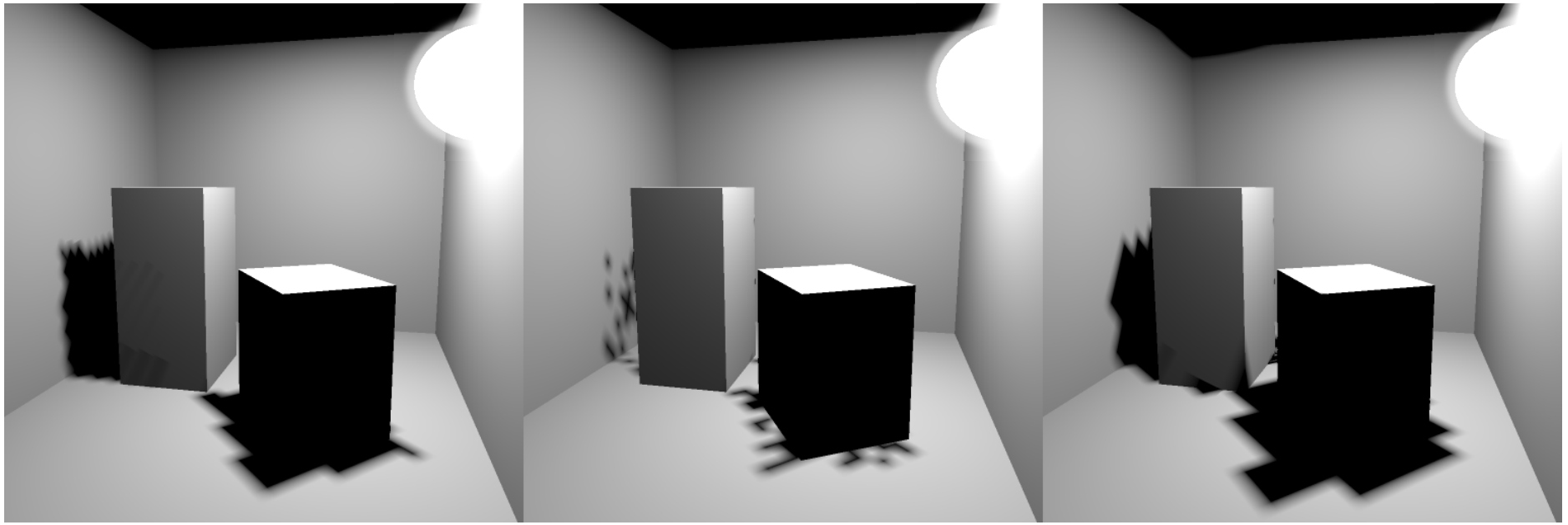
Step 3: Pull-Push

- Depth maps from points have holes



Step 3: Pull-Push

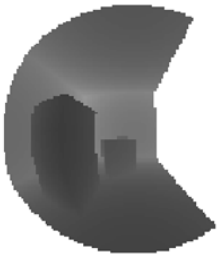
- We fill those holes using pull-push ..



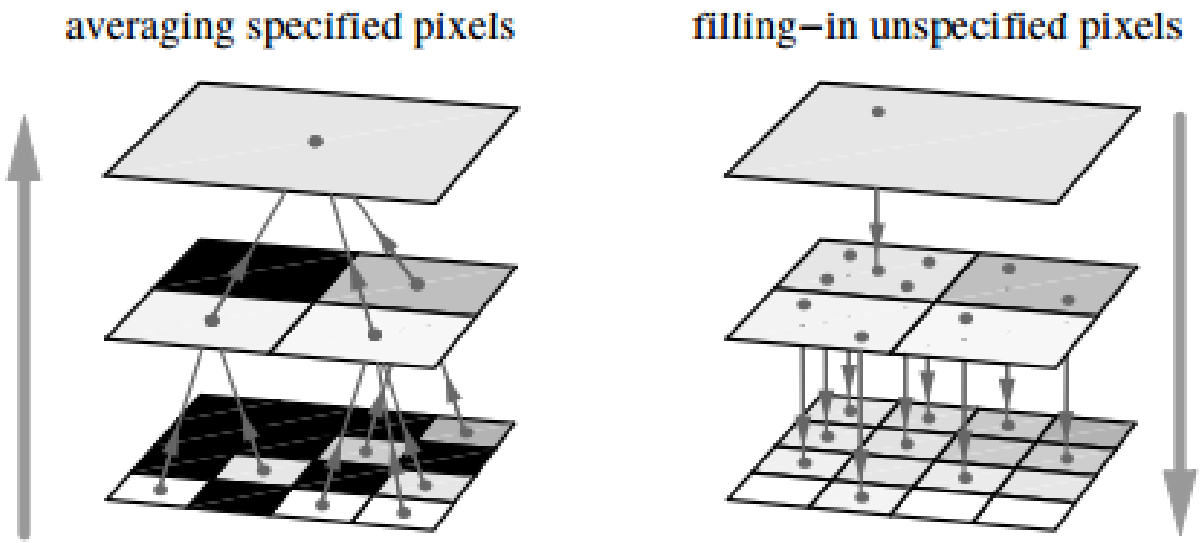
Classic

Without pull-push

With pull-push



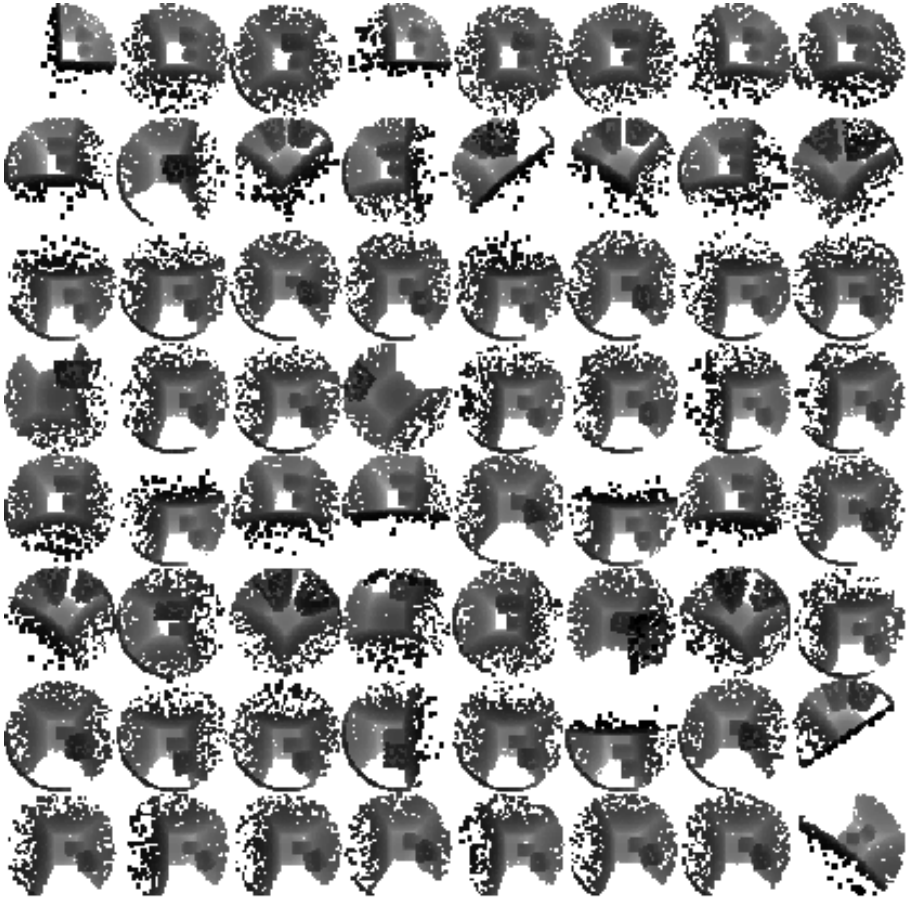
Pull-Push Method



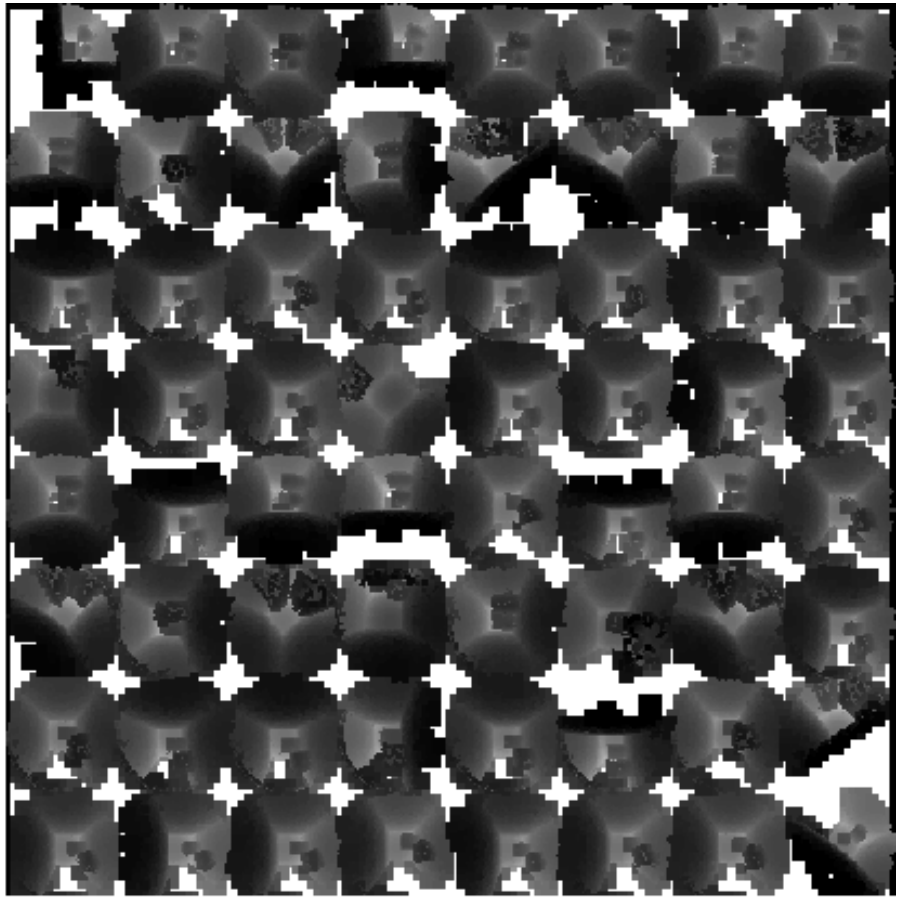
Remove black holes by using the pyramid and by pushing values in the top-down manner

Step 3: Pull-Push

- .. on all depth maps in parallel.



Without pull-push



With pull-push

Step 1 VPL generation

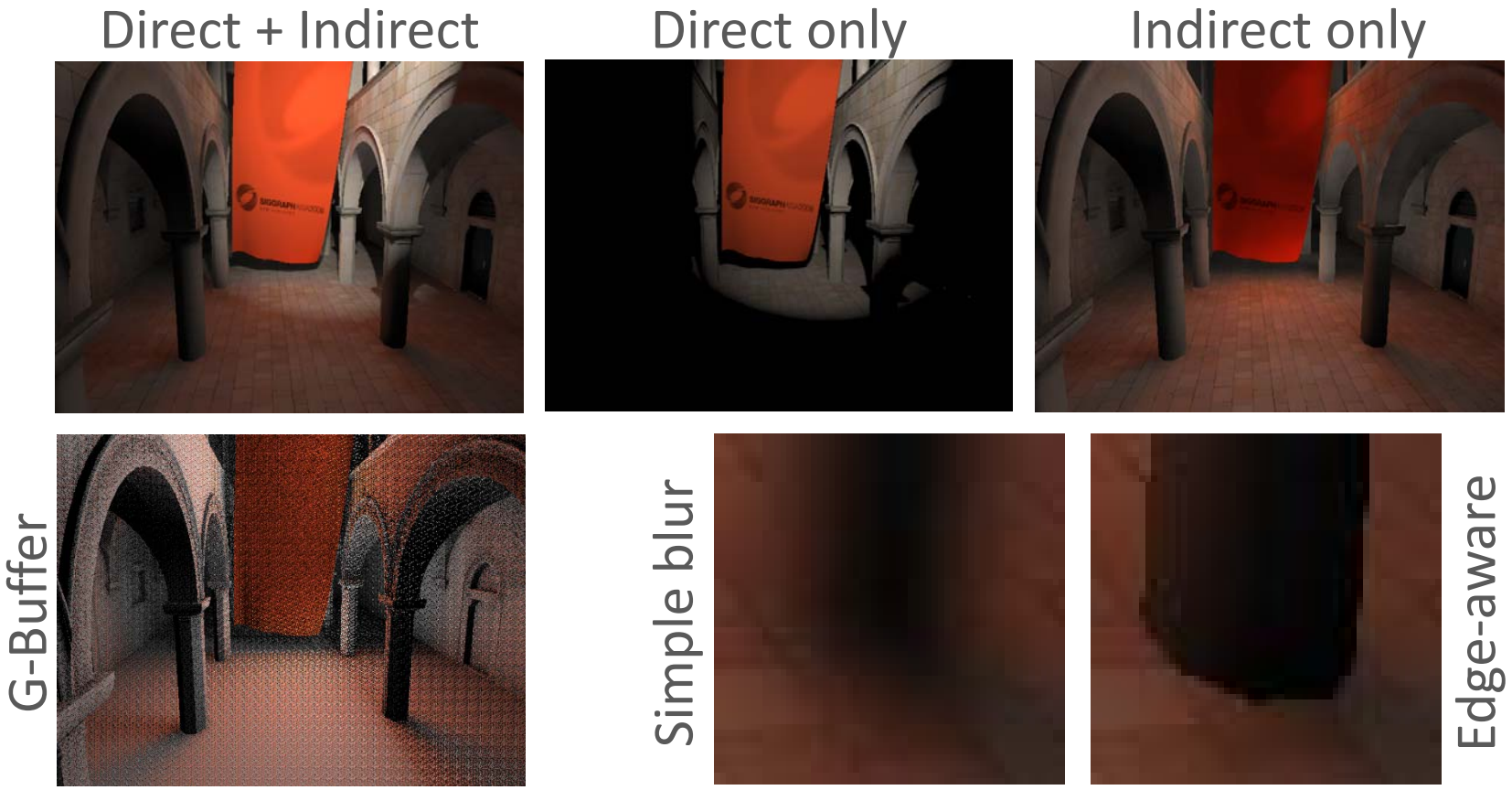
Step 2 Point based depth maps

Step 3 Pull push

Step 4 Shading

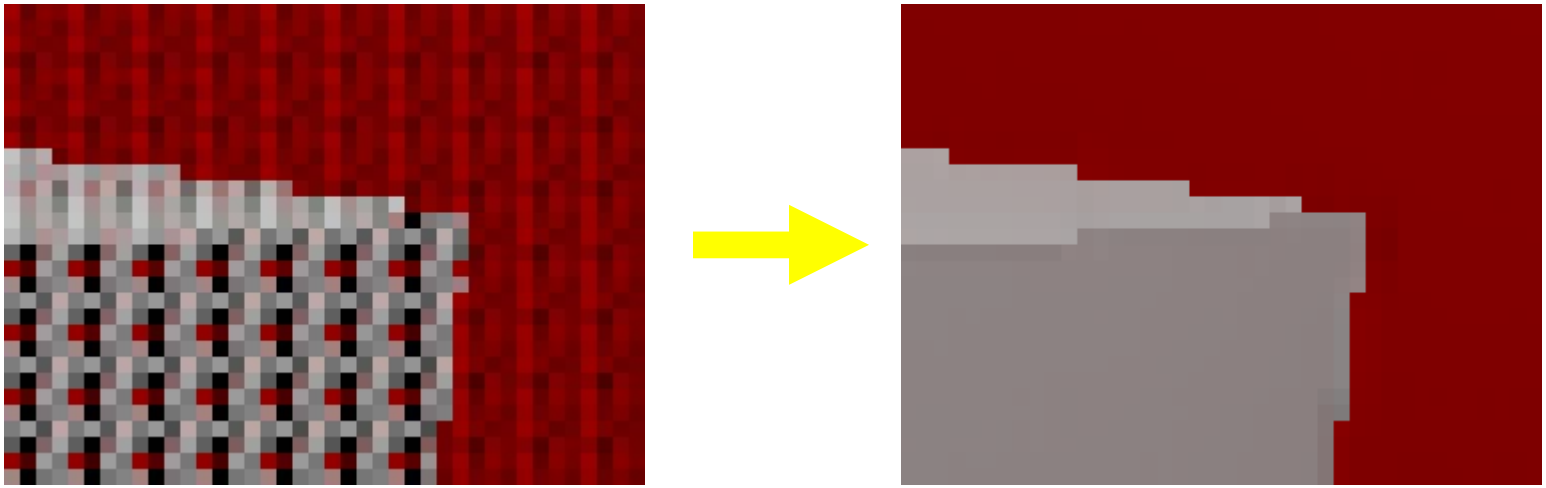
Step 4: Shading

- Separate direct and indirect, both deferred
- Indirect: Interleaved sampling, geometry aware blur filter

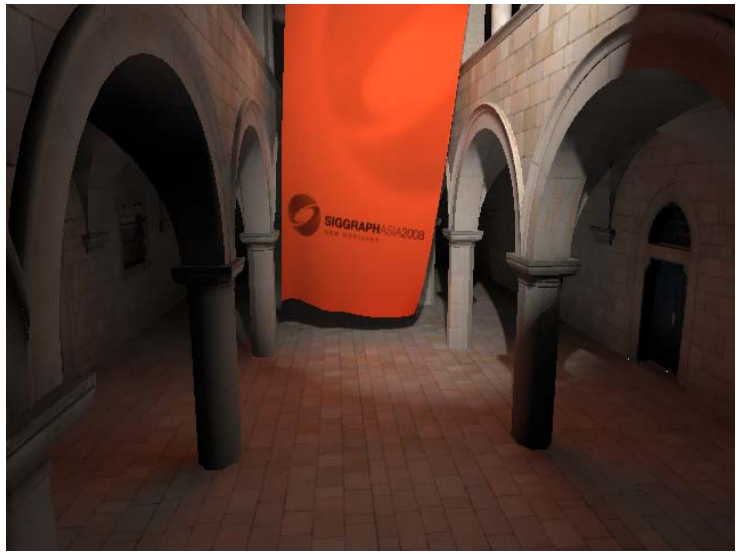


Interleaved Sampling

- Reduces the number of shadow map lookups per pixel
- For each pixel, use a subset of all VPLs
- Apply geometry-aware filtering



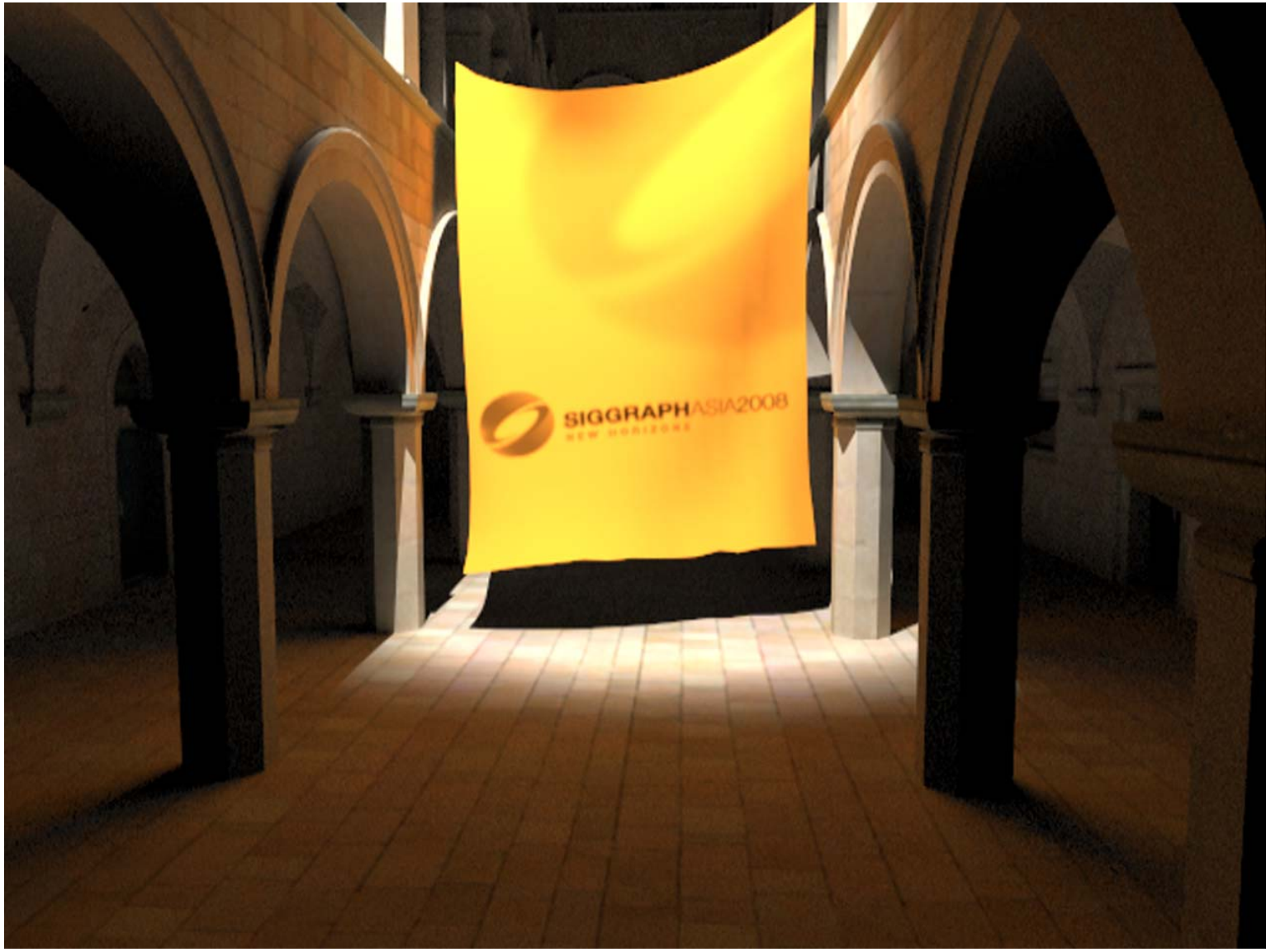
Results: Performance



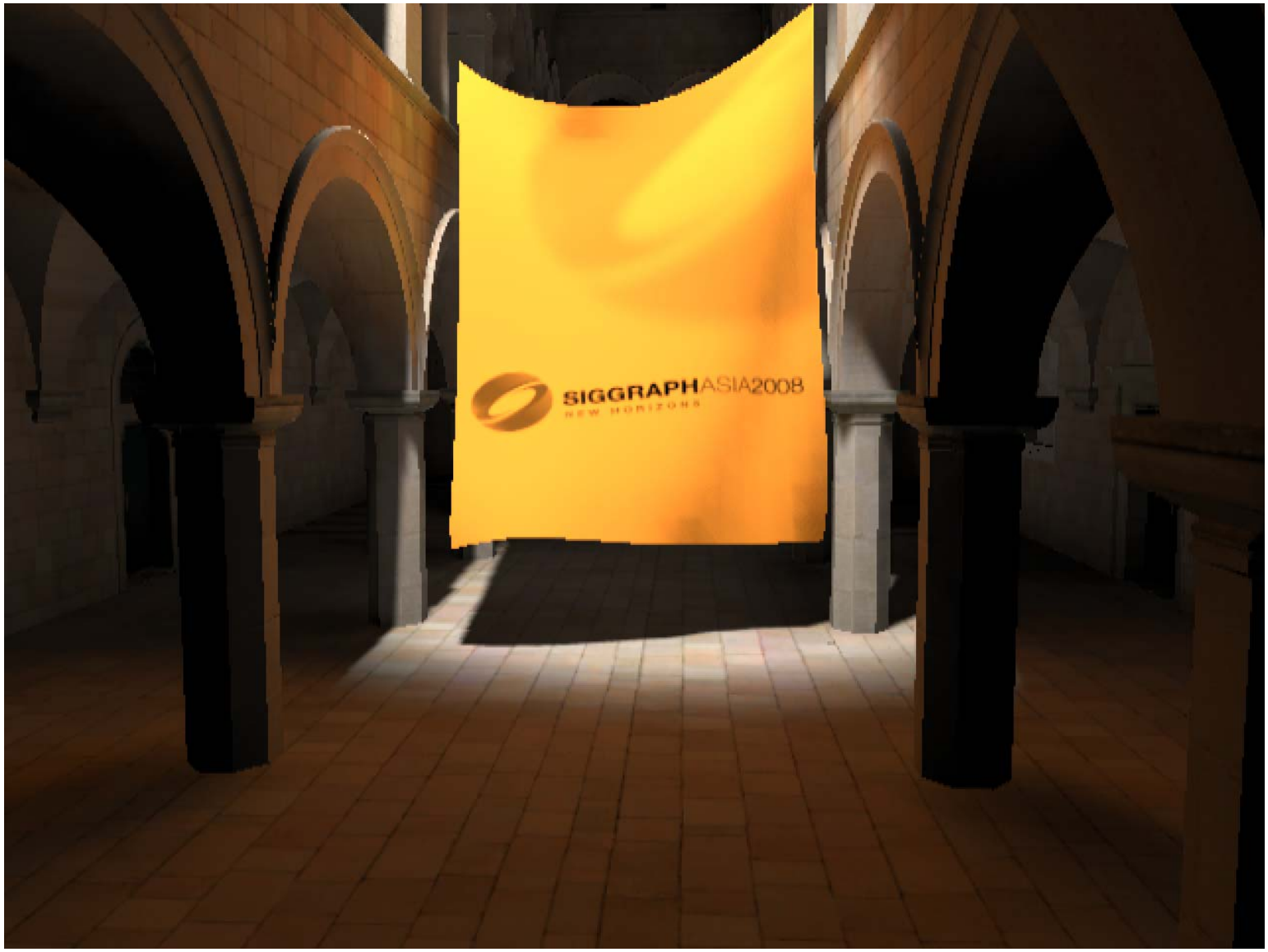
- "Christo's Sponza"
  - 70k faces, dynamic
  - 1024 VPLs
  - 256x256 depth maps
  - 8k points each

- Breakdown
  - 7 ms VPL generation
  - 44 ms ISM
  - 8 ms Pull-push
  - 15 ms Rendering
  - 4 ms G-Buffer
  - 11 ms Direct light
- Total
  - 89 ms frame time
  - 11 frames / s

Results: Quality (*PBRT, hours*)



Results: Quality (*Ours*, 11 fps)



Results

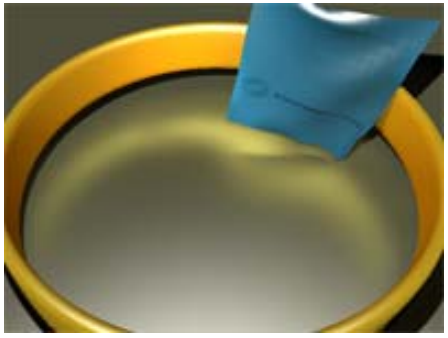
Cornell box horse

Christo‘s Sponza

Multiple bounces

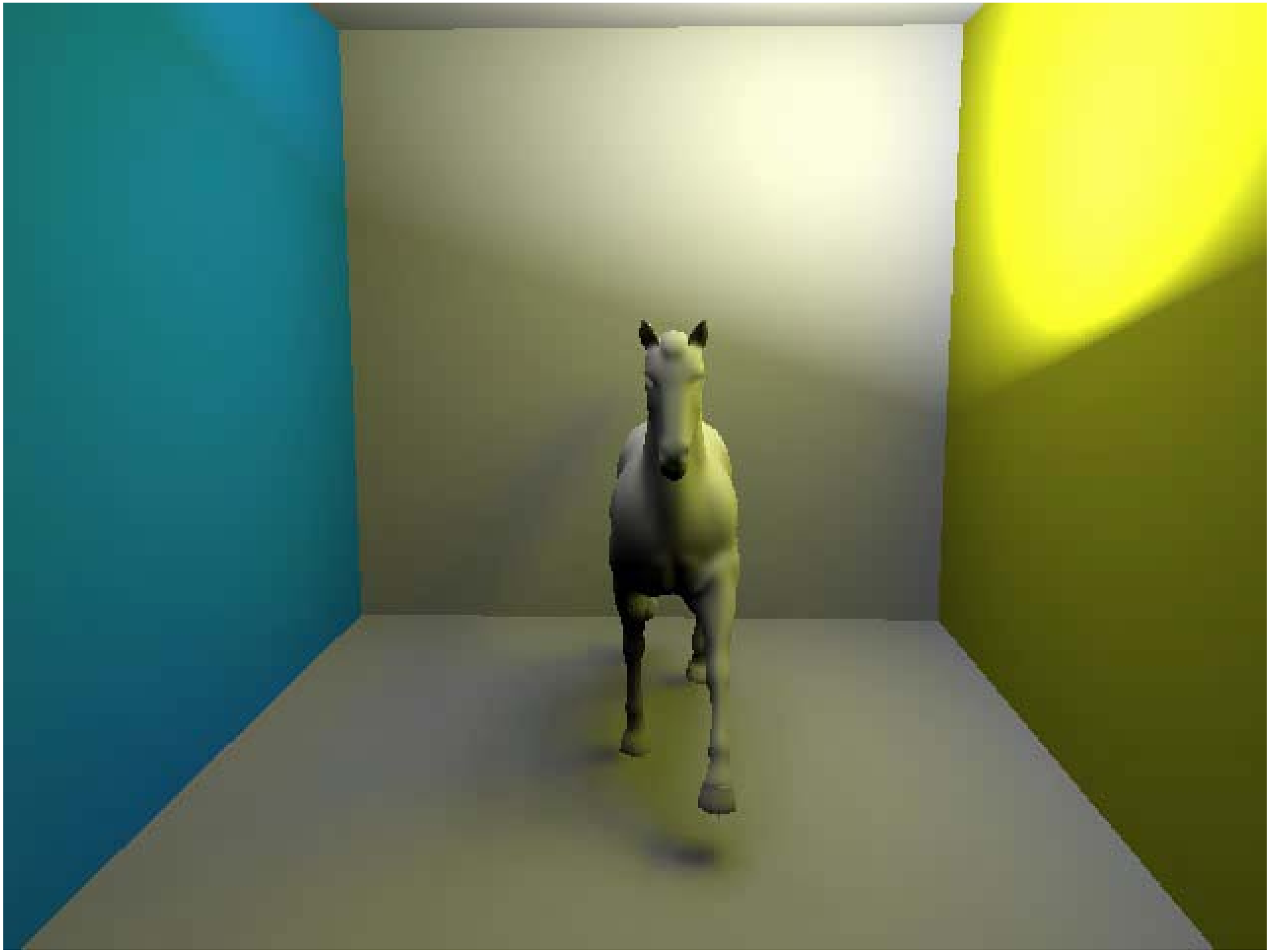
Complex, local area lights

Natural illumination

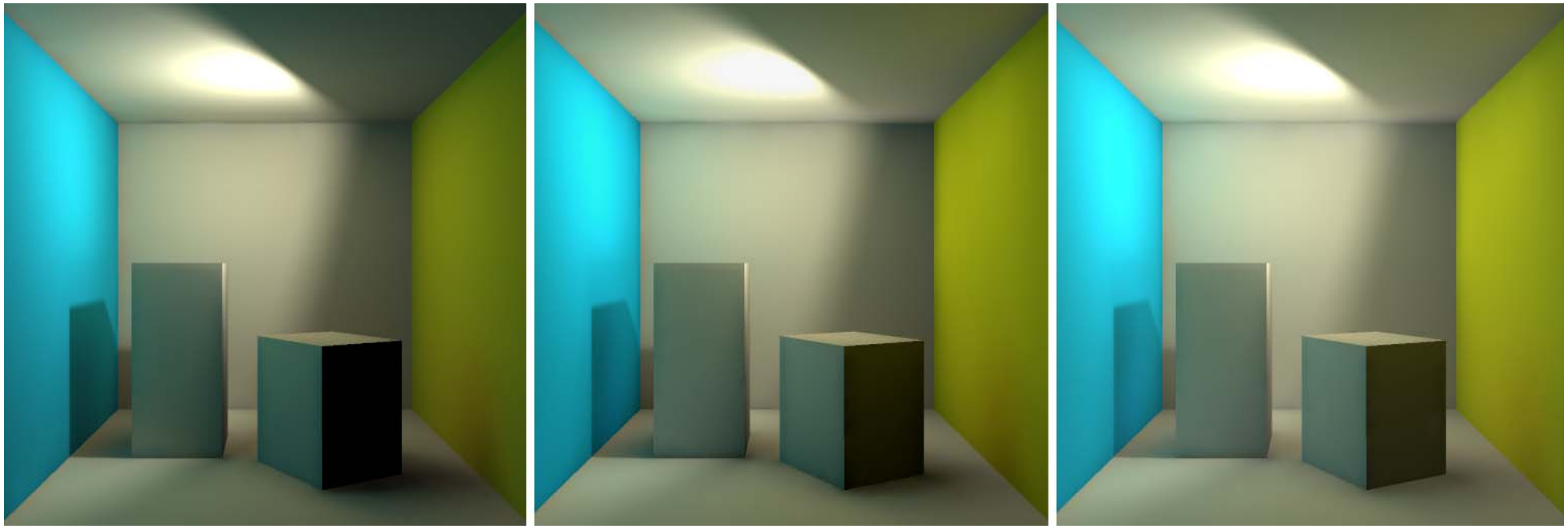


Caustics

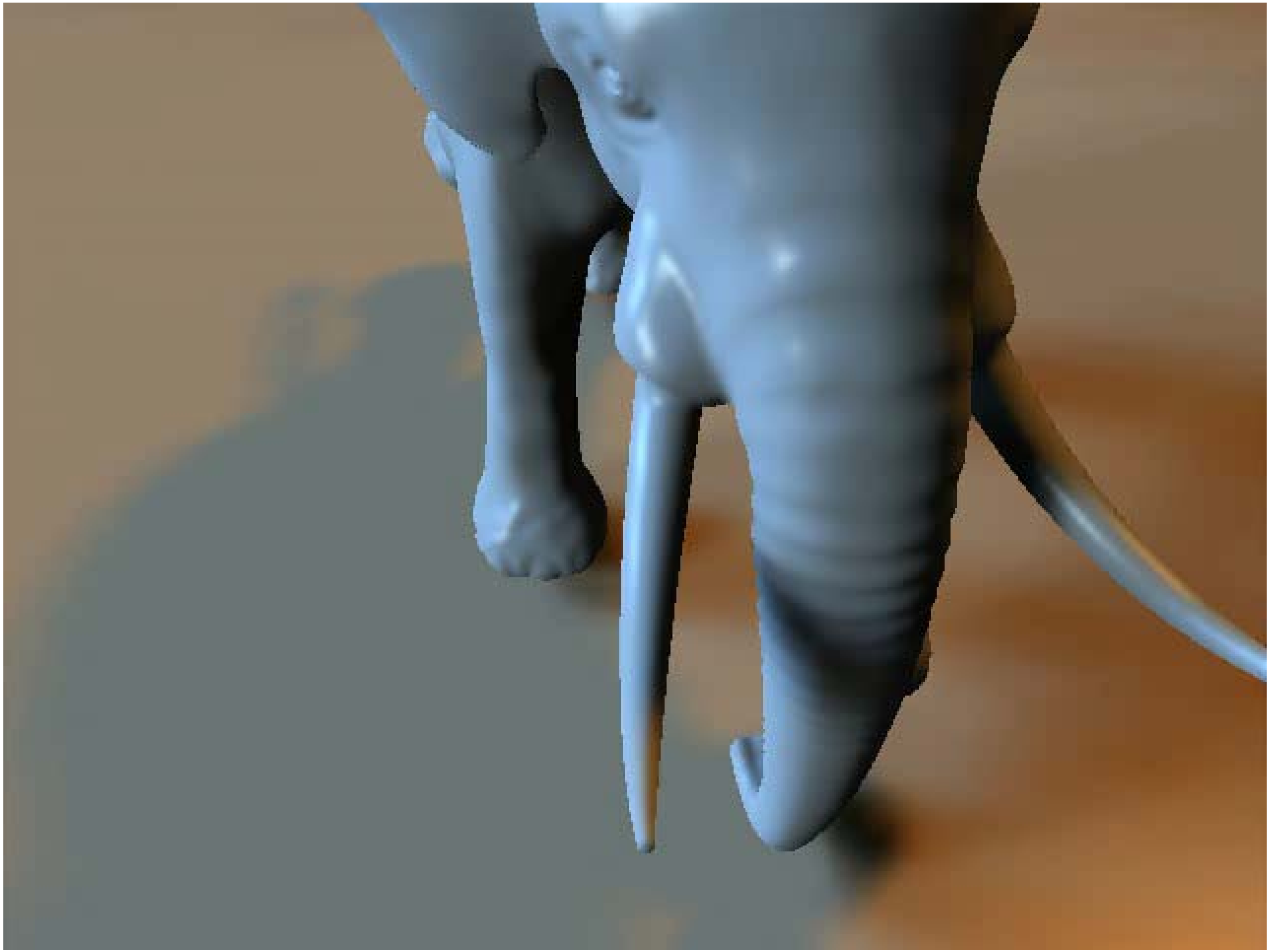
Timings: Nvidia GeForce 8800 GTX

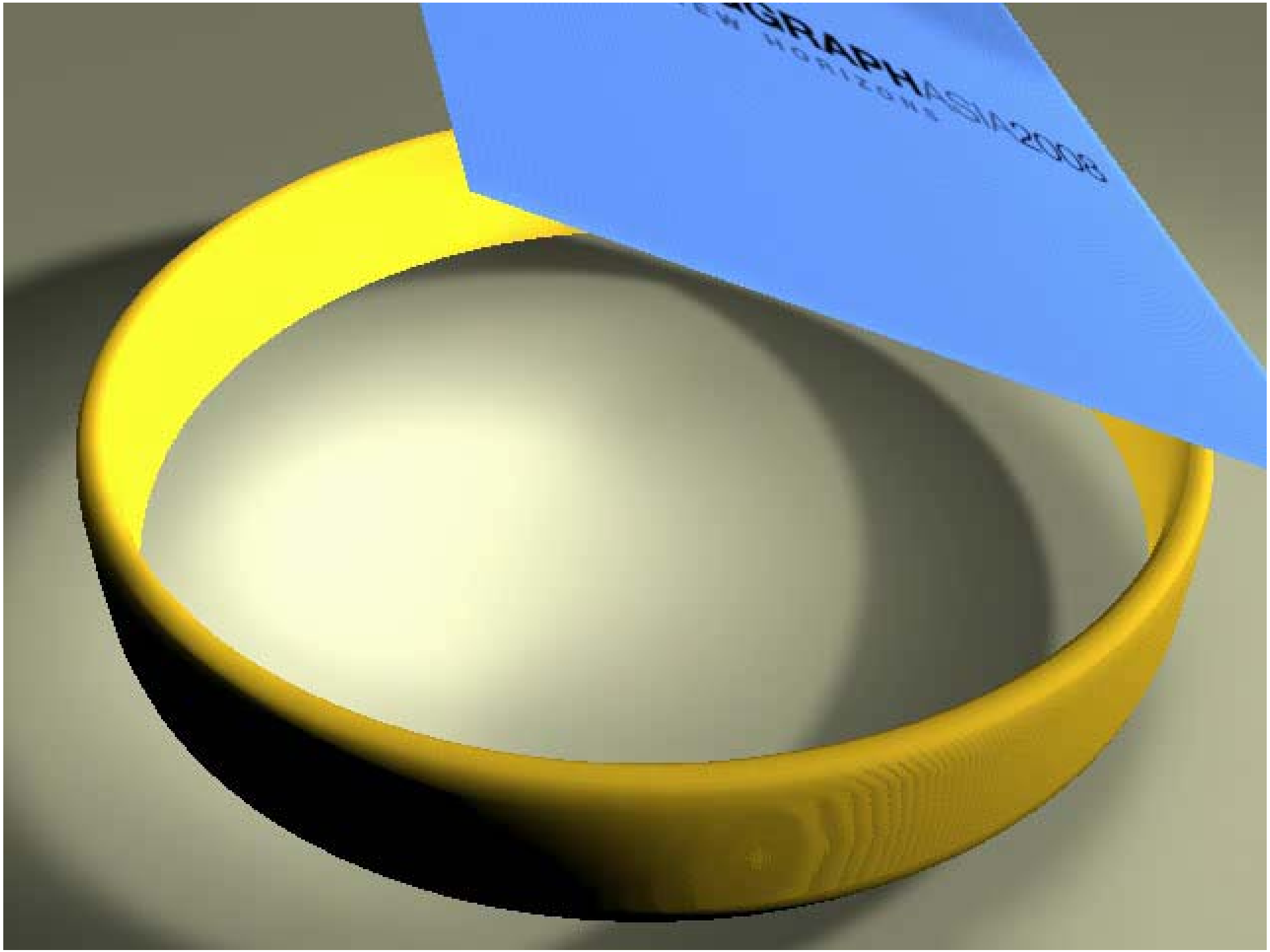


Multiple bounces



- Imperfect reflective shadow maps
  - Multiple bounces
  - Complex direct lighting





Conclusion

- Conclusion
  - Indirect visibility can be simplified
  - Imperfect shadow maps exploit this
- Future work
  - More scalable
  - Adaptive
    - Better points
    - view dependant
  - Perceptual evaluation

Class Objective were:

- Understand instant radiosity
  - Its general procedure
  - Its computational bottlenecks: shadow maps
  - Use imperfect shadow map and some recent techniques

Any Questions and HWs

- **Come up with one question on what we have discussed in the class and submit at the end of the class**
  - **Submit four times in Sep./Oct.**
  - **1 for typical questions**
  - **2 for questions that have some thoughts or surprise me**
- **Go over the next lecture slides before the class**
- **Watch 2 SIG/I3D/HPG videos and submit your summaries every Tue. class**

Next Time

- **Microrendering**