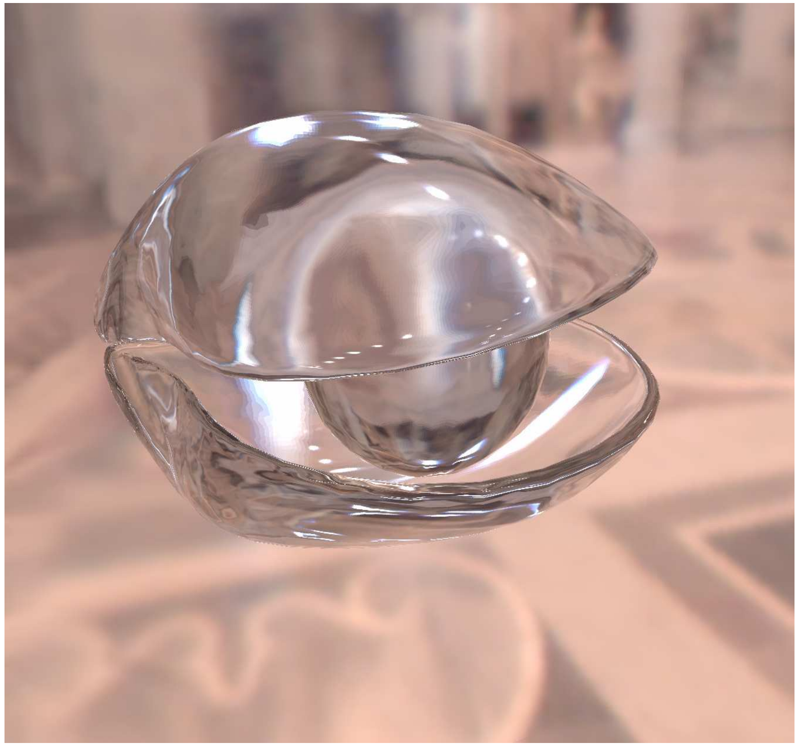
Interactive Refraction on Complex Static Geometry using Spherical Harmonics

Olivier Génevaux
Frédéric Larue
Jean-Michel Dischler

LSIIT
Strasbourg I University, France
UMR CNRS-ULP 7005

Goal

- Interactive refraction
  - Hardware accelerated
- “Complex” geometry
  - Multiple bounces
  - Multiple media

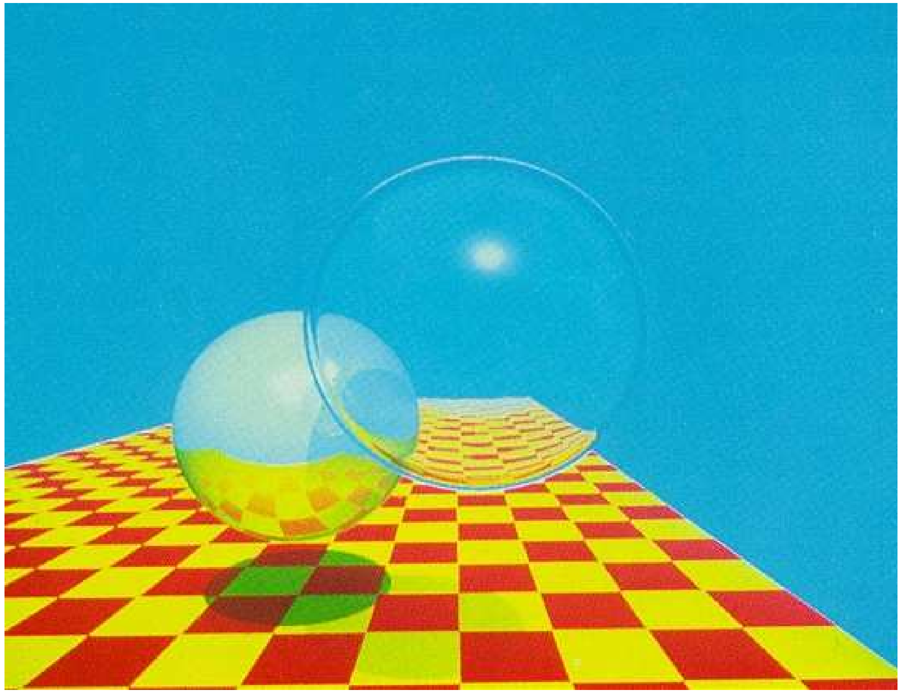


Summary

- Previous work
- Method
- Results, Limitations
- Conclusion

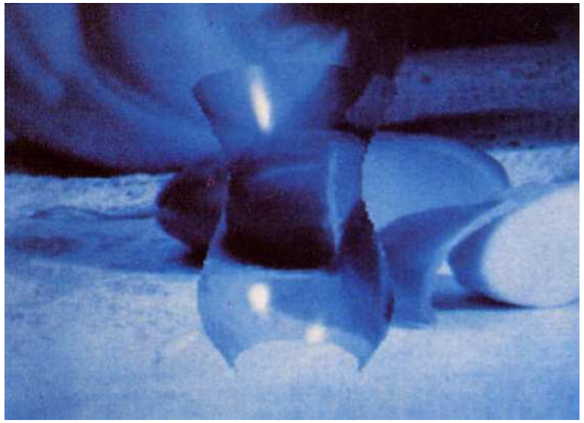
Previous work

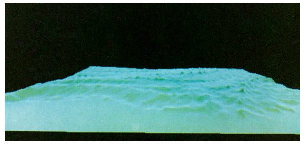
- Raytracing
  - [Whitted, 1980]



Previous work

- Feed forward pipeline
  - Rough approximation
    - [Kay & Greenberg, 1979]
  - Scene dedicated techniques
    - [Ts’o & Barsky, 1987]
    - [Guy & Soler, 2004]
  - Double sided refraction
    - [Wyman, 2005]





Previous work

- Hybrid [Hakura & Snyder, 2001]

- Offline distortion evaluation
  - Stored using lightfield parameterization [Heidrich et al., 1999]

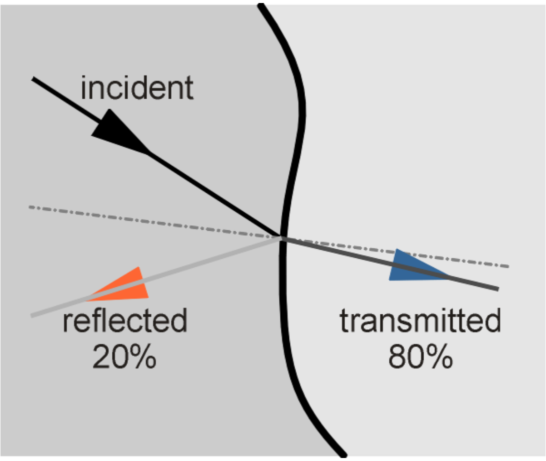
Summary

- Previous work
- Method
- Results
- Conclusion

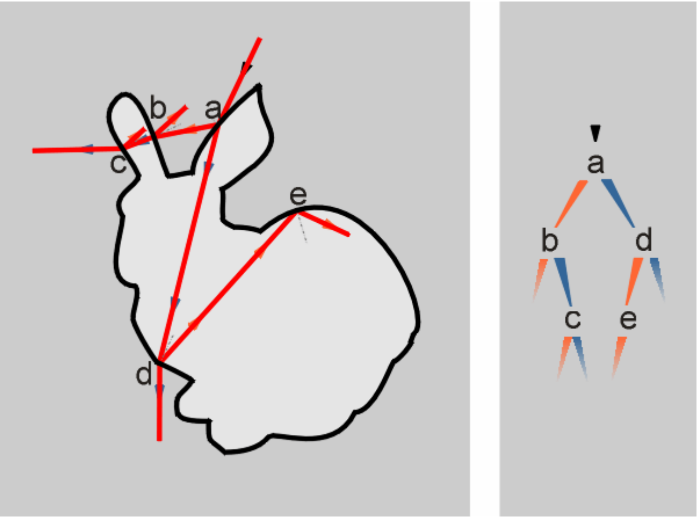
Method

Optics reminder

- Refraction
  - Fresnel equations
  - Snell's law



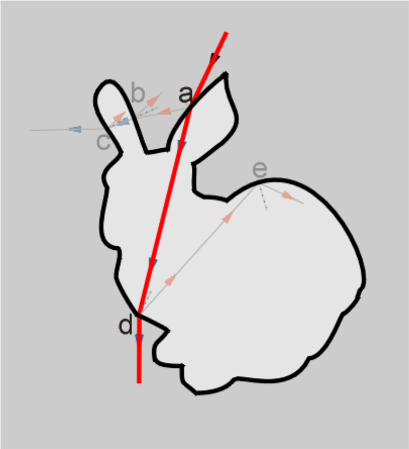
- Binary tree of rays
  - Cannot be handled interactively

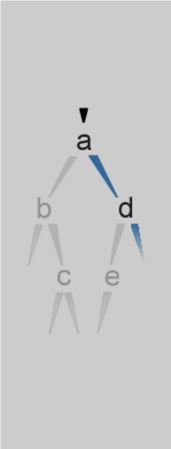


Method

Refraction model

- Model approximation
  - Pruned ray tree [Hakura & Snyder, 2001]
  - Surface x S^2 → (Surface, S^2)
- Further approximation
  - Without parallax effects
  - Output position drop
  - Surface x S^2 → S^2





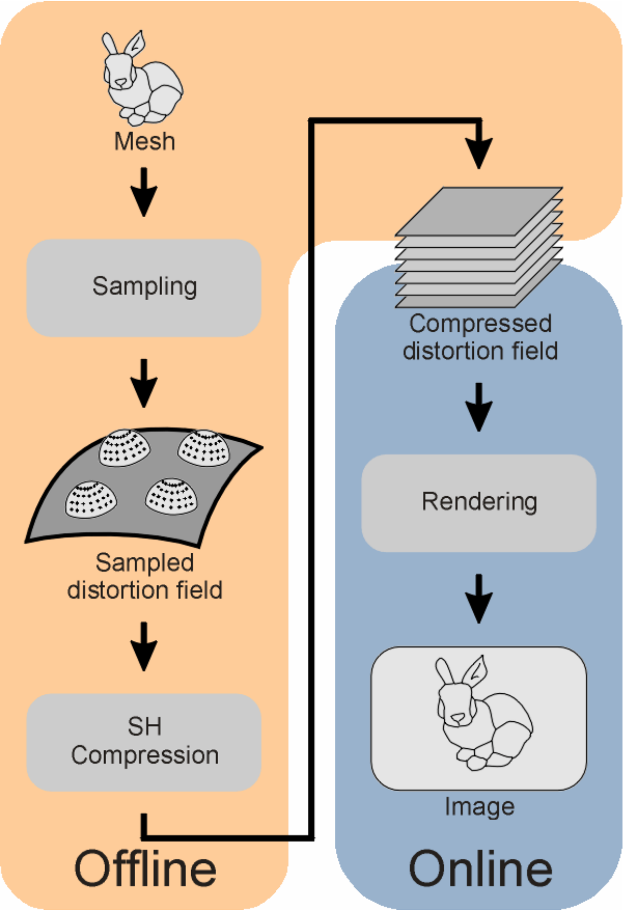
- Refraction reduced to a view dependant information over the object surface

- For each point on surface and each viewing direction, a single ‘refracted direction’ is defined: distortion field

Method

Technique outline

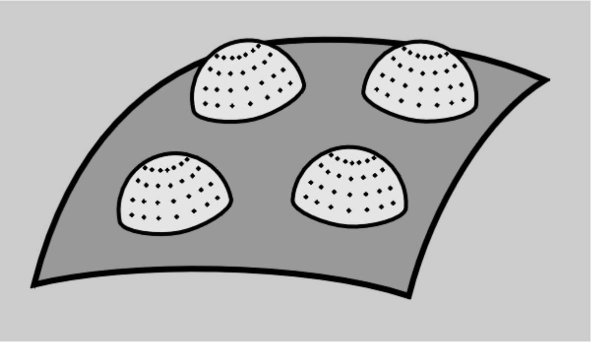
- Offline: Distortion field sampling
  - Static geometry
  - Evaluated using ray tracing
  - Compressed
- Online: Rendering
  - Uncompress wrt. current viewpoint
  - Index environment map
- HW friendly storage
  - Stored on surface
  - Directional information
    - Spherical harmonics



Method

Precomputation

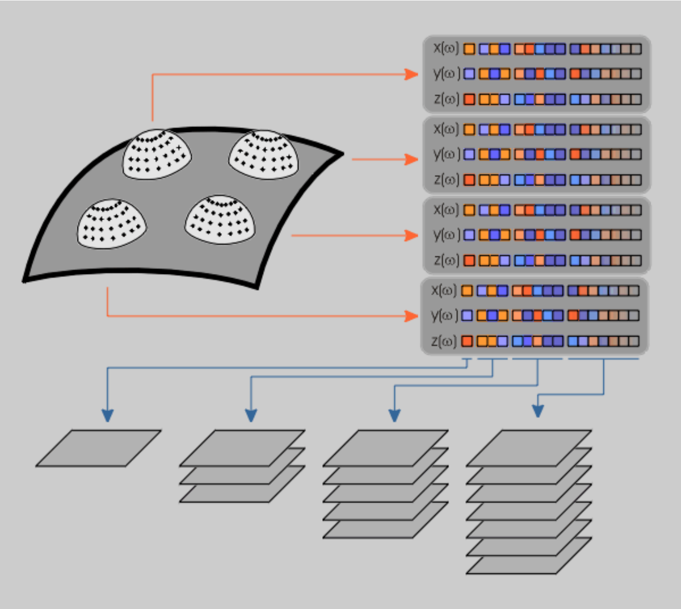
- Sampling
  - Whole surface
  - Incident directions hemisphere
    - Above each sample
  - Large data
- Example: bunny
  - 35k surface samples
  - 2048 directions
  - ~ 1.1GB data
- Compression scheme: Spherical Harmonics
  - Convenient for directional variations



Method

Stored data

- At each surface sample
  - Output direction [x, y, z](ω_{in})
    - 3 view dependant functions
    - 3 SH coefficients vectors
  - Hardware storage
    - One texture per SH basis function
    - XYZ → RGB channels
    - 8 bits / channel quantization
      - No visual loss



Method

Decompression

- Data to be used directly
  - No PRT-like convolution [Sloan et al. 2002]
  - Requires actual decompression

$$\begin{bmatrix} x_{out} \\ y_{out} \\ z_{out} \end{bmatrix}(\omega_{in}) = \sum_i \begin{bmatrix} w_i^x \\ w_i^y \\ w_i^z \end{bmatrix} \cdot Y_i(\omega_{in})$$

- Decompression: series expansion
  - SH polynomials evaluated wrt. current viewpoint
    - Basis functions Cartesian definition

$$Y_l^m(x,y,z) = K_l^m \sum_{p,q,s} \frac{1}{p!q!s!} \left(-\frac{x+iy}{2}\right)^p \left(\frac{x-iy}{2}\right)^q z^s$$

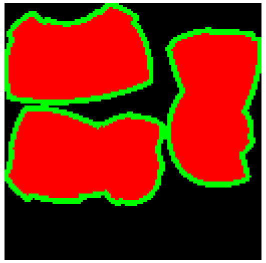
$$(x,y,z) = \omega_{in}$$

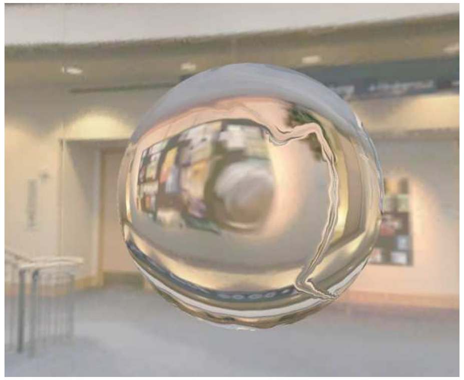
  - Multiple rendering passes
    - Offscreen: data space
    - Count related to basis functions #
      - SH order 8: 7 passes (DirectX PS 2.0b)

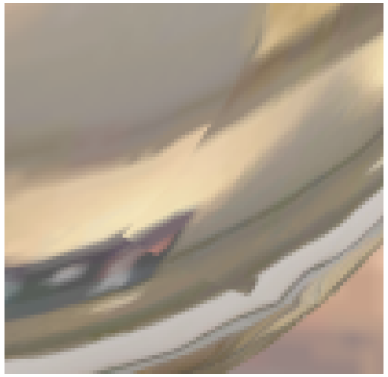
Method

Rendering – Straightforward technique

- Samples distribution given by mesh unfolding
- Output directions correctly located for hardware bilinear filtering
- Discontinuity artifacts
  - Chart borders
  - Gutters
- No perfect continuity





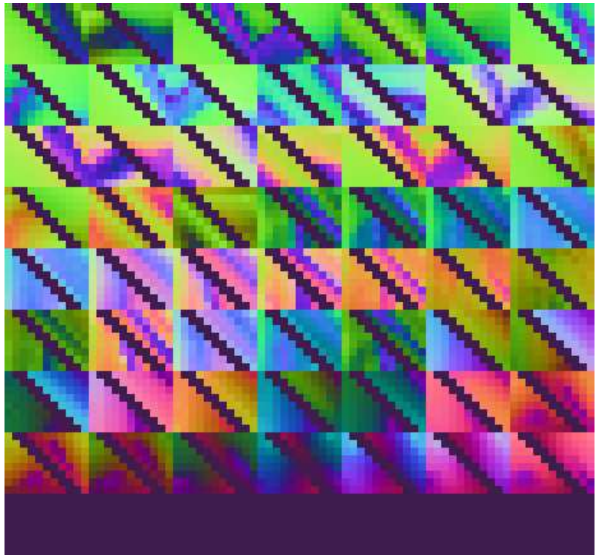


Method

Rendering – Revised technique

- Full mesh split
  - Automatic
  - Increased memory consumption
- Adequate correction procedure
  - Details in paper

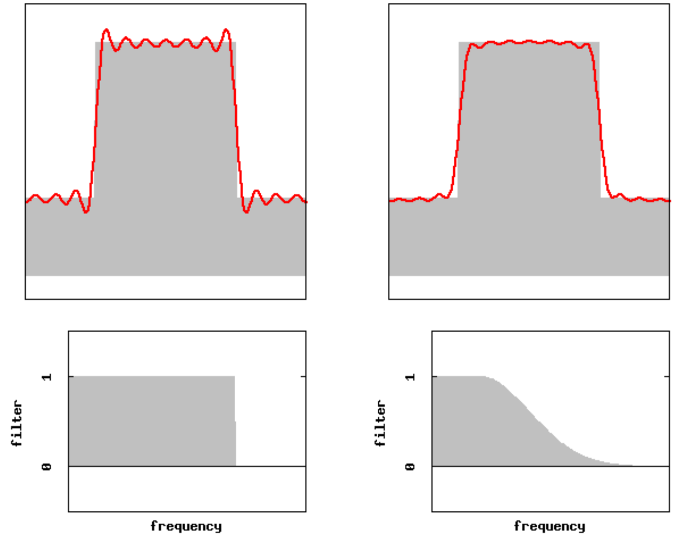
➢ Perfect continuity



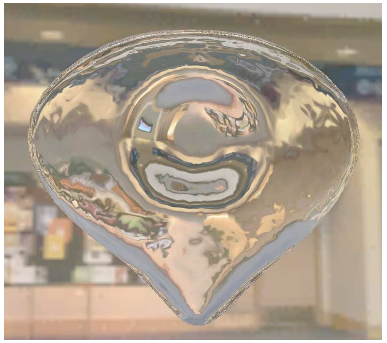
Method

Visual improvements – Smoothing

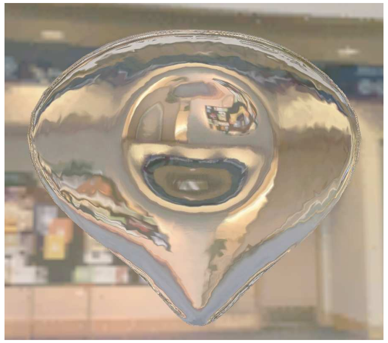
- Frequency domain
  [Westin, 92]
  - Directly on SH coefficients
  - Free of charge
- Spatial domain



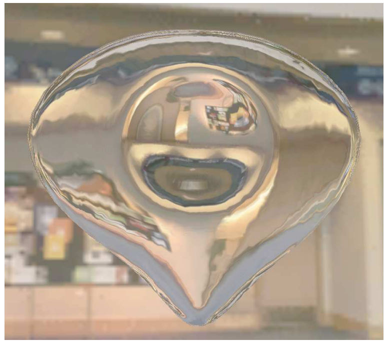
Raw



Frequency

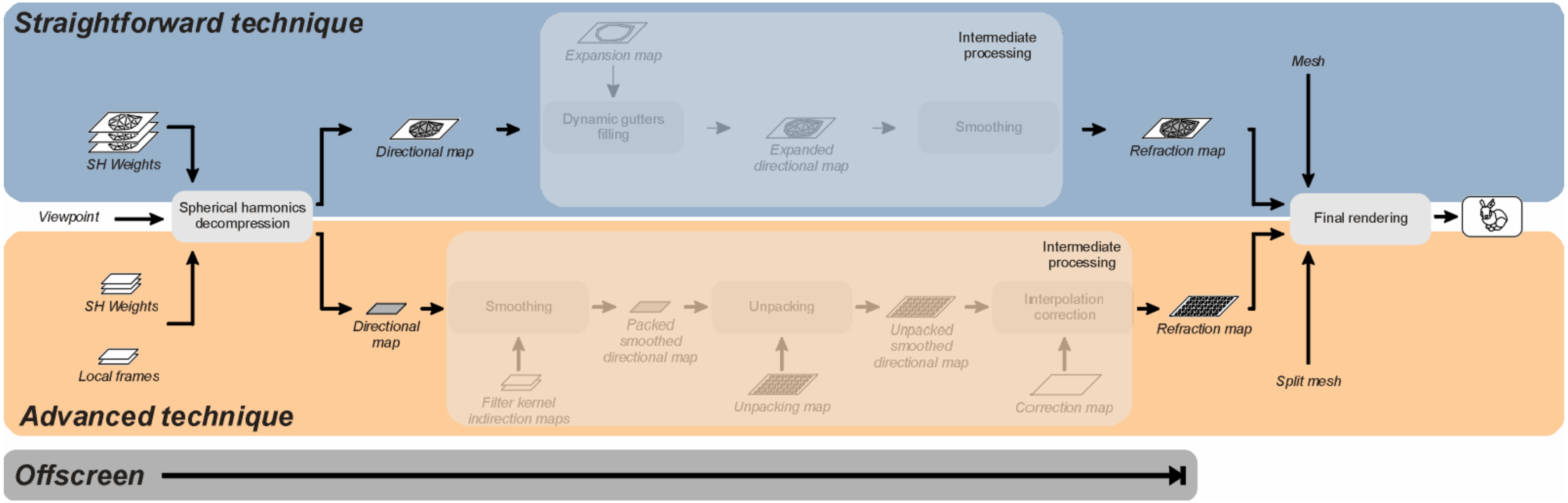


Spatial (3 pts)
+ Frequency



Method

Rendering summary

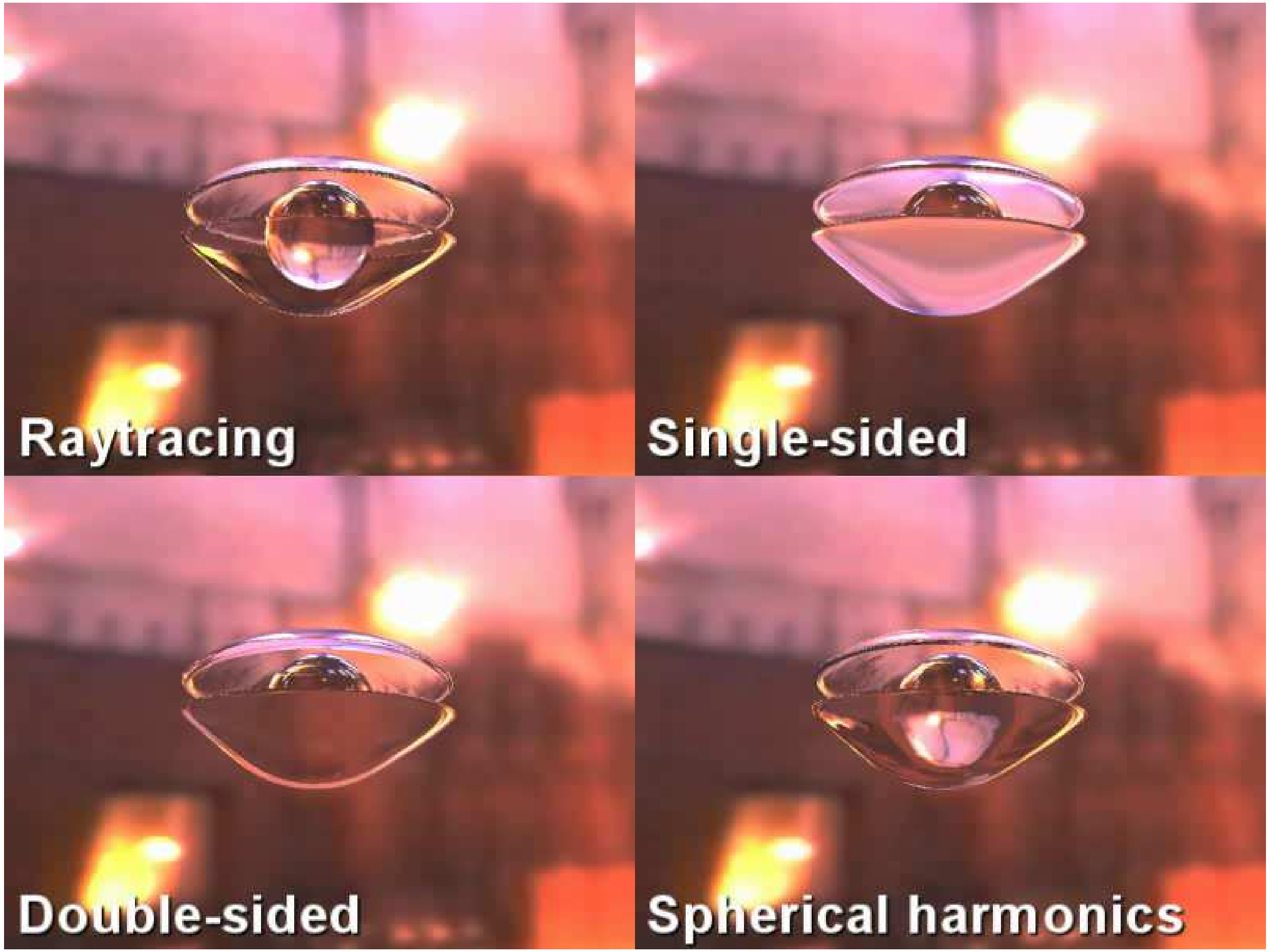


Summary

- Previous work
- Method
- Results
- Conclusion

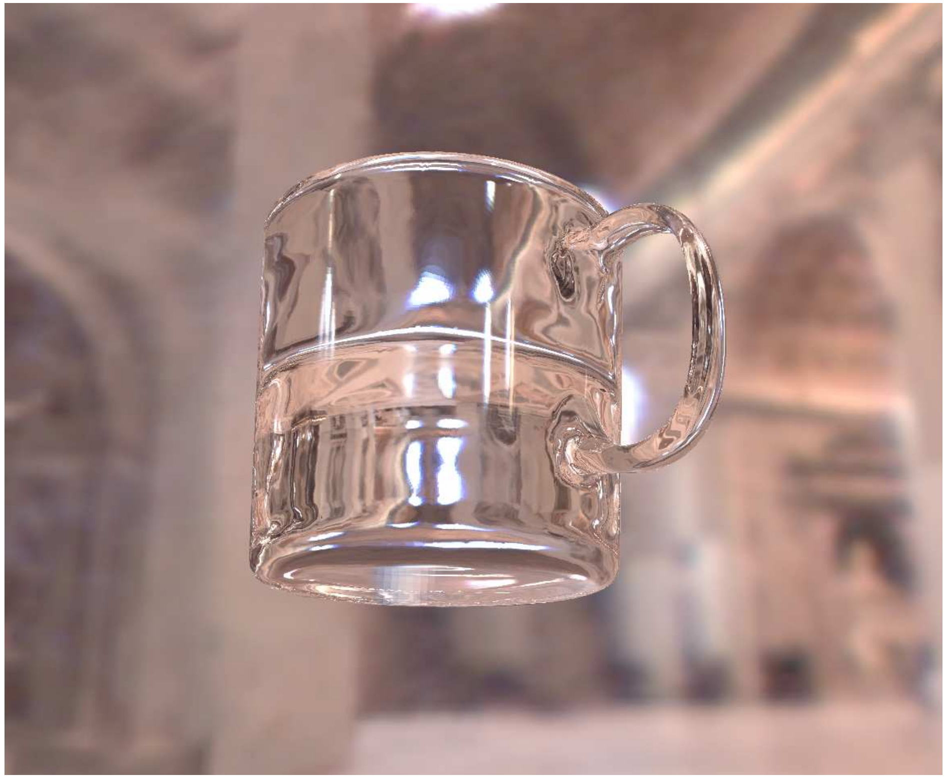
Results

Visual Comparison

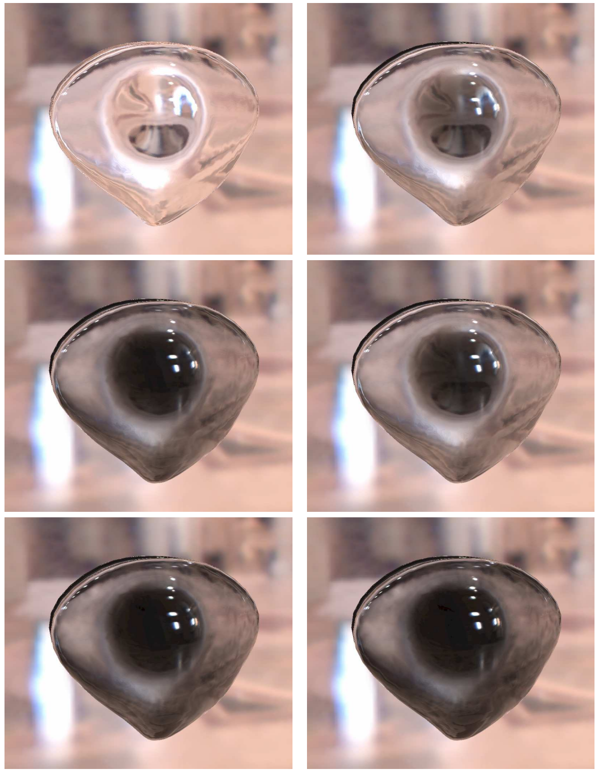


Results

Complex media



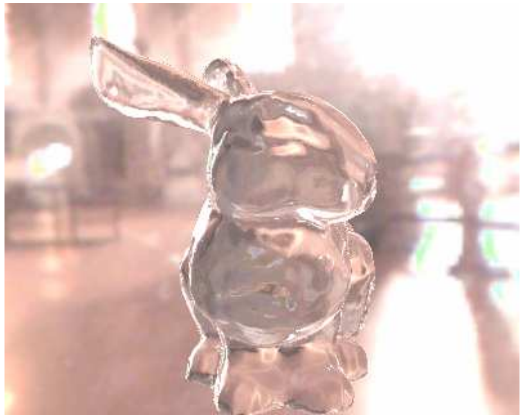
Multiple media: Air, Glass & Water

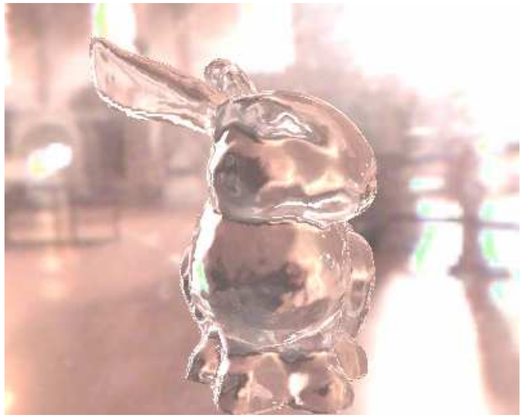


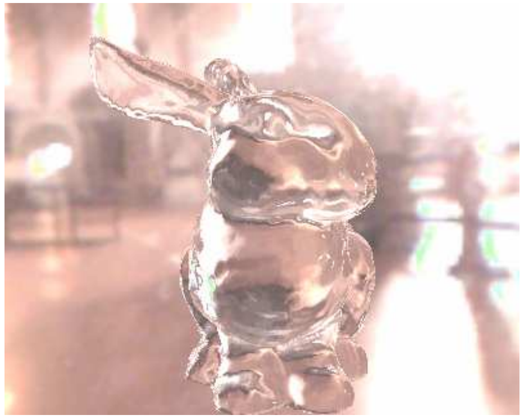
Interactive view-dependant attenuation

Results

SH order influence



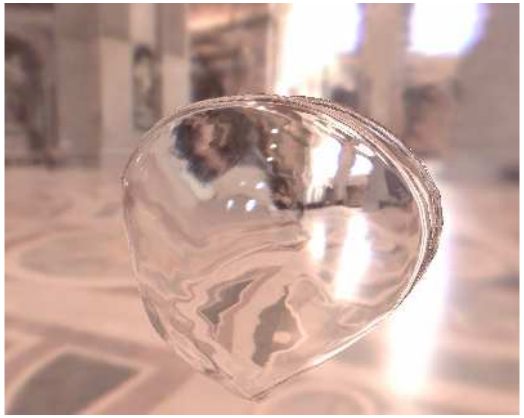


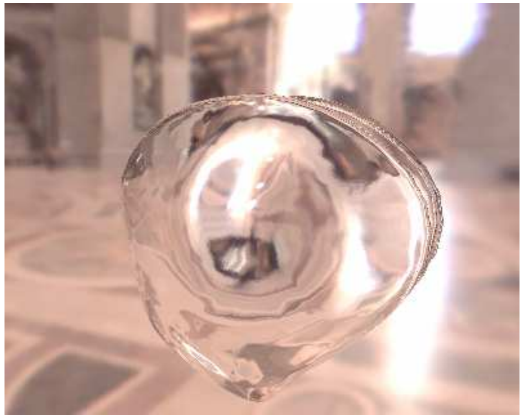


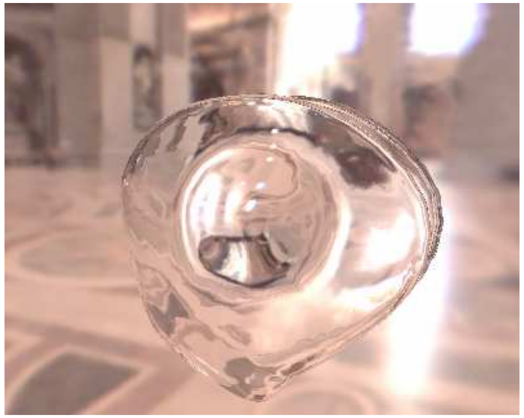
l=1: 4 functions

l=4: 25 functions

l=8: 81 functions

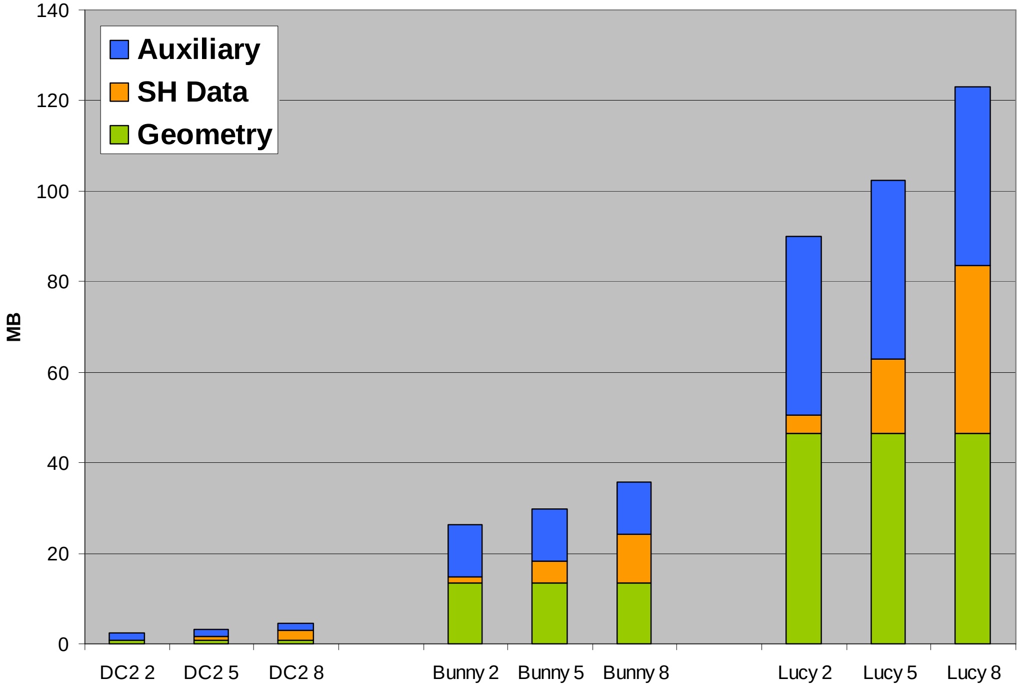


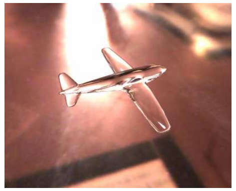




Results

Memory consumption





7k samples
3.5k triangles

35k samples
70k triangles

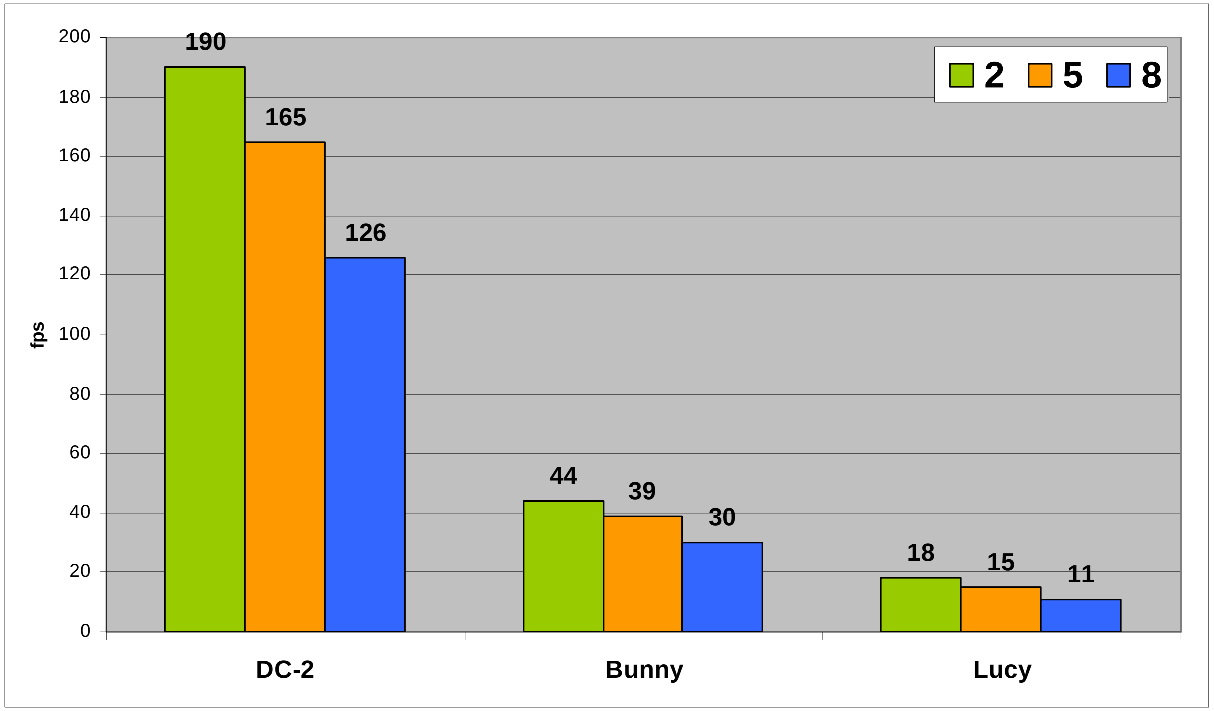
120k samples
239k triangles

Revised technique: geometry duplication

5 points spatial smoothing

Results

Rendering speed



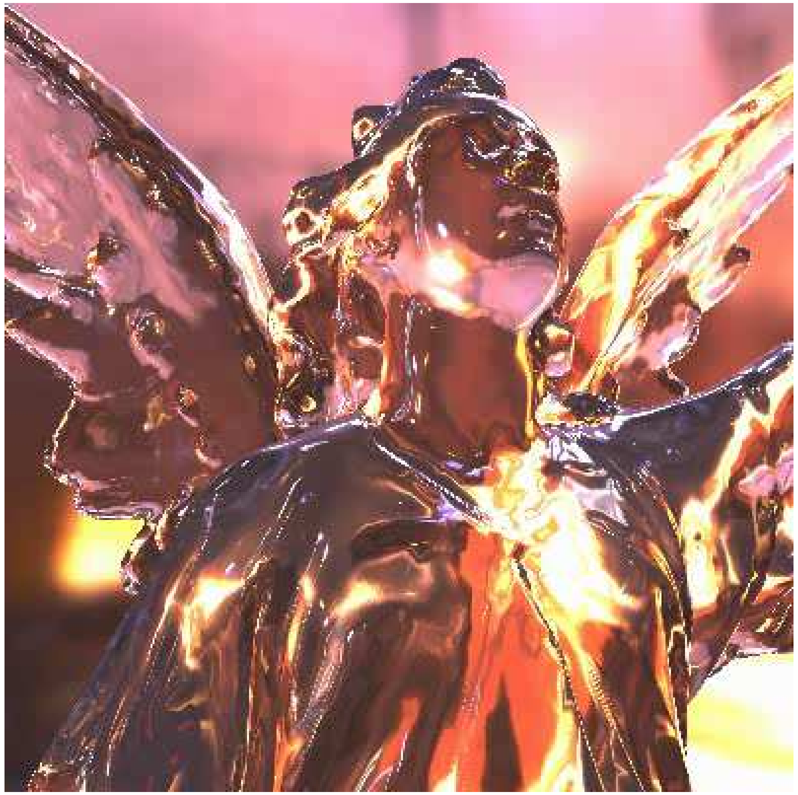
GeForce 6800GT

~ 1350 x 1100

Results

Limitations

- Noise prone
  - Fixed sampling set on surface
    - Mip-mapping unavailable
  - Point sampling of environment
    - High curvature area

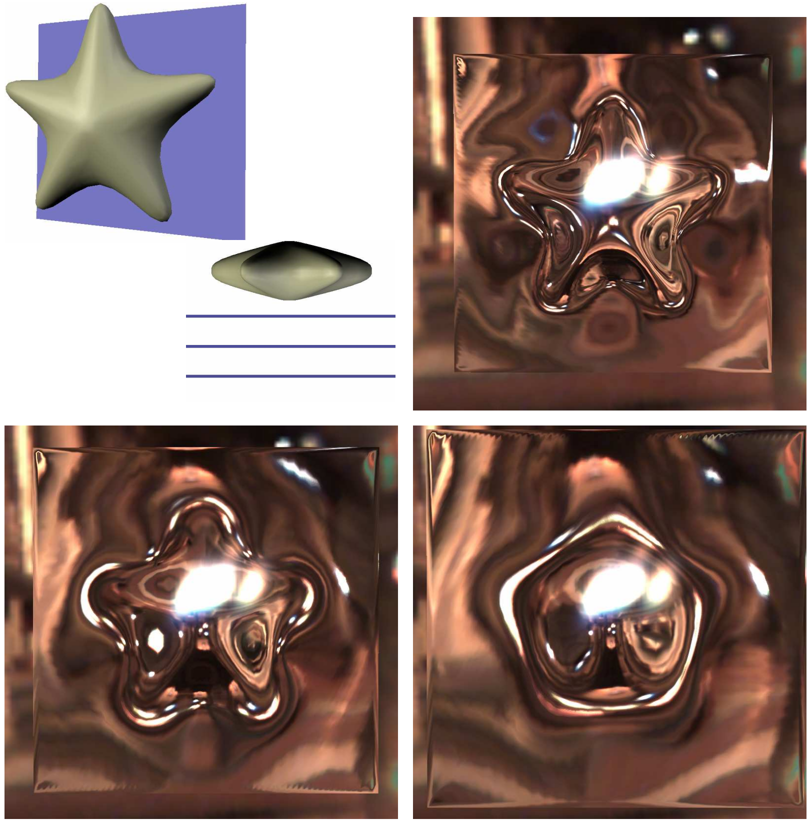


Worst case

Results

Limitations

- Low frequency variations capture
- Example
  - Increasing distance between star and glass pane
  - Details lost when frequency increase



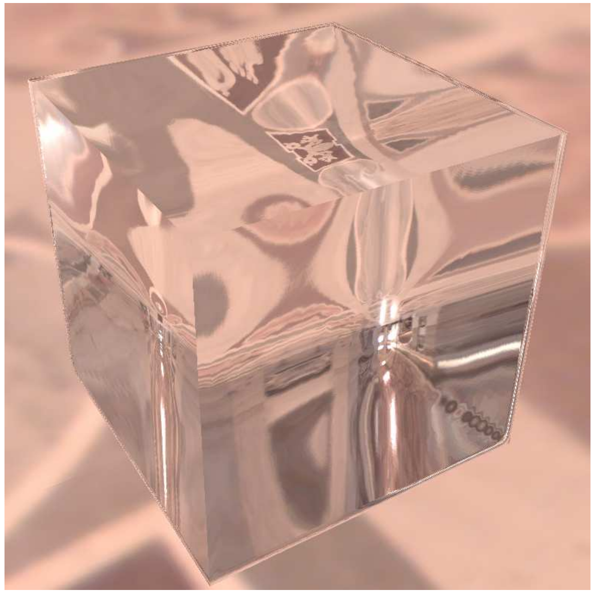
Summary

- Previous work
- Method
- Results
- Conclusion

Conclusion

Pros & Cons

- Cons
  - Quality issues
    - Sparkling
  - SH: low frequency
    - Blurry
    - Precision requires too many resources
  - Static geometry



Conclusion

Pros & Cons

- Pros
  - Pleasing refraction approximation
    - Reasonable cost on many objects
      - Multiple media
    - At interactive framerate
    - Continuous behavior
      - Believable even if not well captures
      - Major bounces captured
  - Progressive decompression

Conclusion

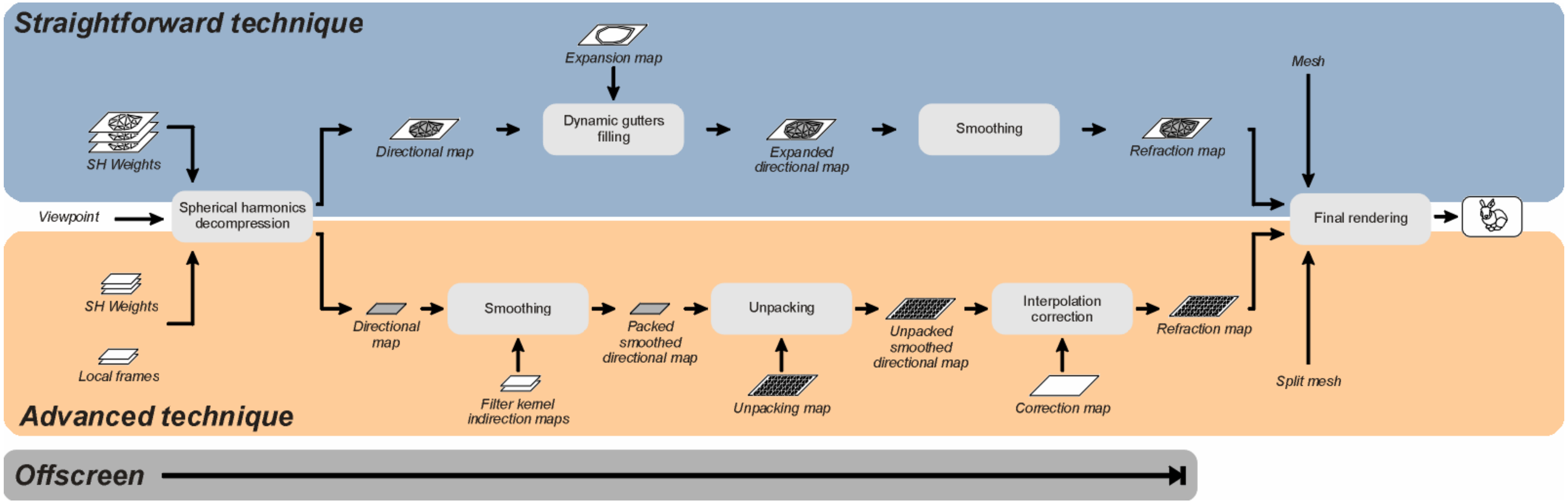
Future work

- Single rendering pass
    - Much simpler
    - Certainly faster
    - Partially alleviate sampling problems
- Continuity handling
    - Extension to straightforward approach
- Compression scheme

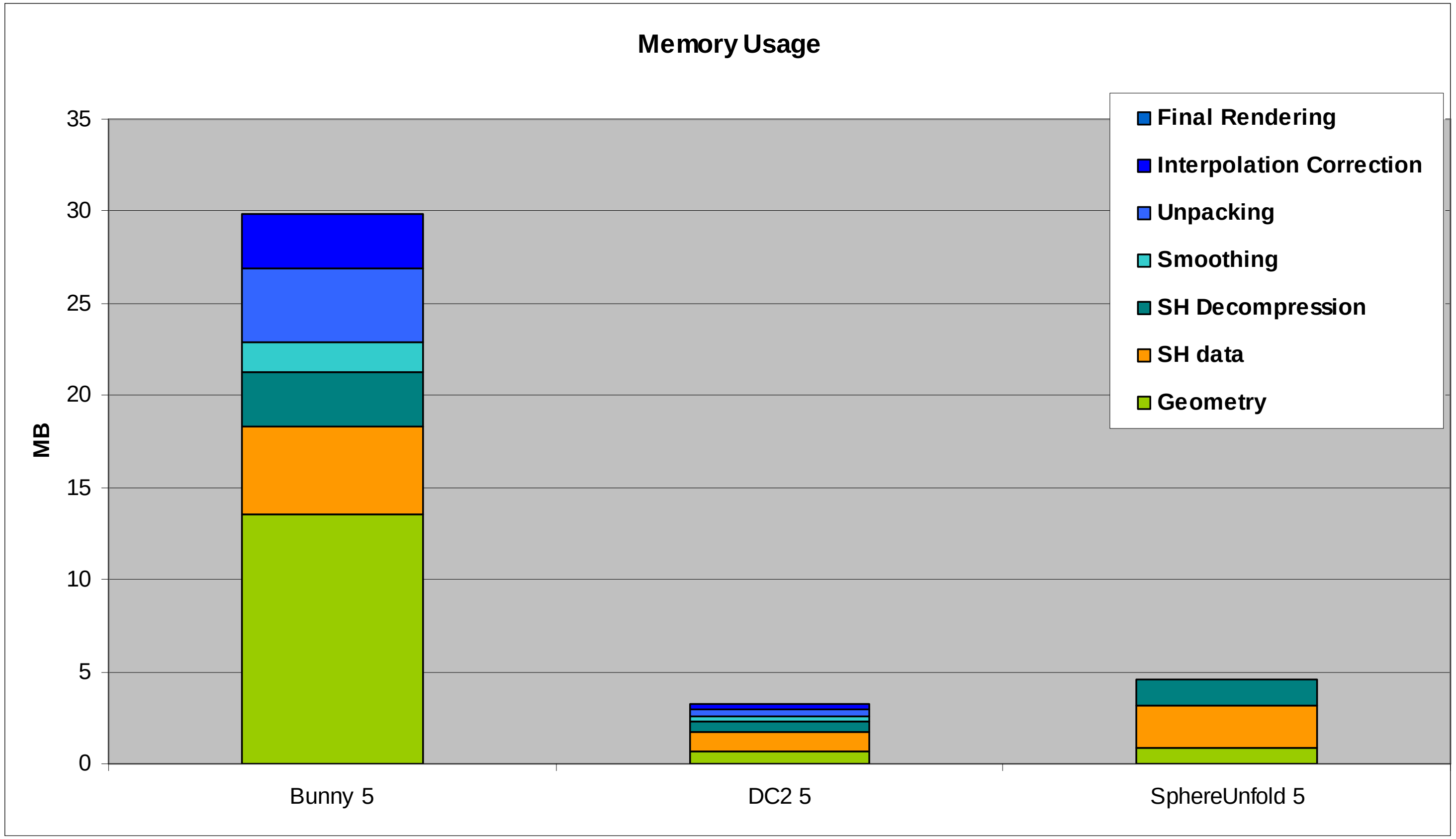
Thanks for your attention

Questions?

Tech 1

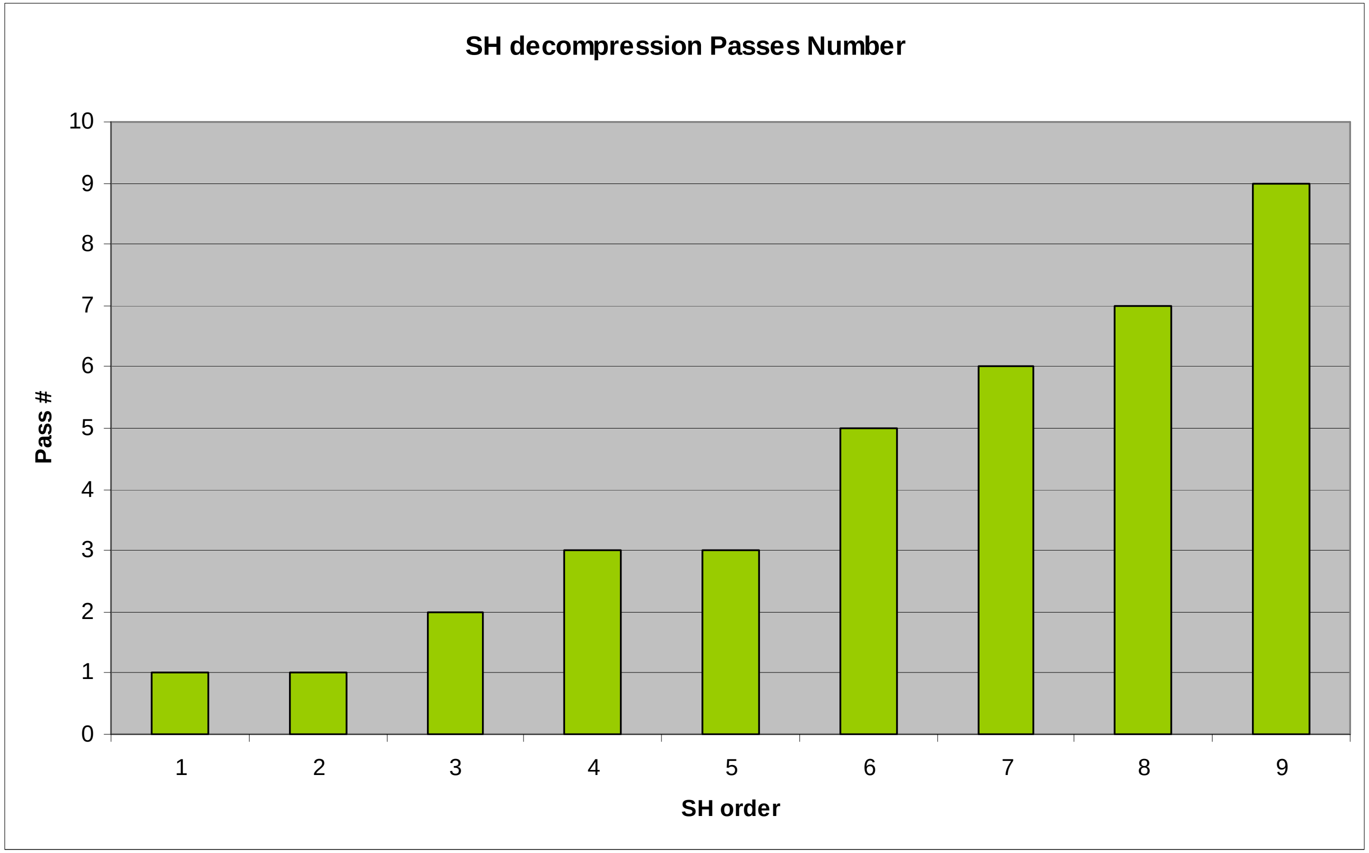

Straightforward technique
Expansion map
Mesh
SH Weights
Directional map
Dynamic gutters filling
Expanded directional map
Smoothing
Refraction map
Viewpoint
Spherical harmonics decompression
Final rendering
SH Weights
Directional map
Smoothing
Packed smoothed directional map
Unpacking
Unpacked smoothed directional map
Interpolation correction
Refraction map
Local frames
Filter kernel indirection maps
Unpacking map
Correction map
Split mesh
Advanced technique
Offscreen

Tech 2



Memory Usage
MB
35
30
25
20
15
10
5
0
Bunny 5
DC2 5
SphereUnfold 5
Final Rendering
Interpolation Correction
Unpacking
Smoothing
SH Decompression
SH data
Geometry

Tech 3



DirectX
PS 2.0b

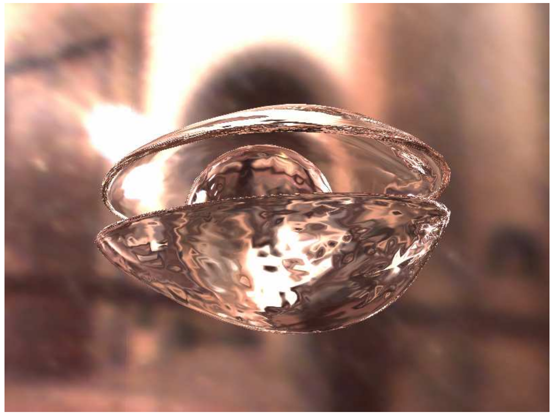
Tech 4

```
[…]

sampler2D shTc0 : register(s0);
uniform float4 scale0 : register(c0);
uniform float4 bias0 : register(c1);

[…]

uniform float4 bias8 : register(c17);
uniform float4 camPos : register(c31);
uniform float4 scaledOpticalDepth : register(c30);

float4 main(VS_OUTPUT params) : COLOR0
{
[…]

        mr0 = tex2D(mr0Sampler, params.shTC);
        mr1 = tex2D(mr1Sampler, params.shTC);
        mr2 = tex2D(mr2Sampler, params.shTC);

        lDirection.x = dot(mr0, camPos);
        lDirection.y = dot(mr1, camPos);
        lDirection.z = dot(mr2, camPos);

[…]

        dirPowers[0] = normalize(lDirection);

        for(int i = 1 ; i < 2 ; i++)
                dirPowers[i] = dirPowers[i - 1] * dirPowers[0];

        accum = (float)0;

        shc = tex2D(shTc0, params.shTC);
        shc = shc * scale0 + bias0;
        bVal = 0;
        bVal += 0.282095;
        accum += shc * bVal;

        shc = tex2D(shTc1, params.shTC);
        shc = shc * scale1 + bias1;
        bVal = 0;
        bVal += 0.488602 * dirPowers[0].y;
        accum += shc * bVal;

[…]
```

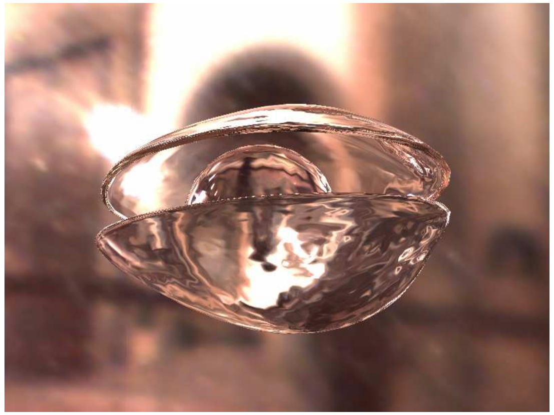
```
        shc = tex2D(shTc6, params.shTC);
        shc = shc * scale6 + bias6;
        bVal = 0;
        bVal += 0.630783 * dirPowers[1].z;
        bVal += -0.315392 * dirPowers[1].x;
        bVal += -0.315392 * dirPowers[1].y;
        accum += shc * bVal;

        shc = tex2D(shTc7, params.shTC);
        shc = shc * scale7 + bias7;
        bVal = 0;
        bVal += 1.092548 * dirPowers[0].x * dirPowers[0].z;
        accum += shc * bVal;

        shc = tex2D(shTc8, params.shTC);
        shc = shc * scale8 + bias8;
        bVal = 0;
        bVal += 0.546274 * dirPowers[1].x;
        bVal += -0.546274 * dirPowers[1].y;
        accum += shc * bVal;


        float3 dirPart = float3(accum.x, accum.y, accum.z);


        dirPart = 0.5f * normalize(dirPart) + 0.5f;


        float attPart = clamp(exp2(- accum.w * scaledOpticalDepth.w), 0.0f, 1.0f);


        return float4(dirPart.x, dirPart.y, dirPart.z, attPart);

}
```

Tech 5

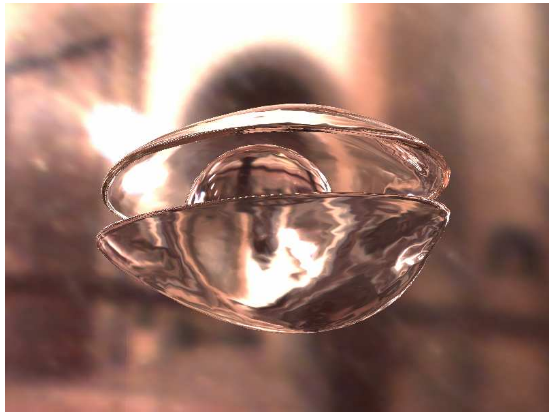
Influence of sampling density



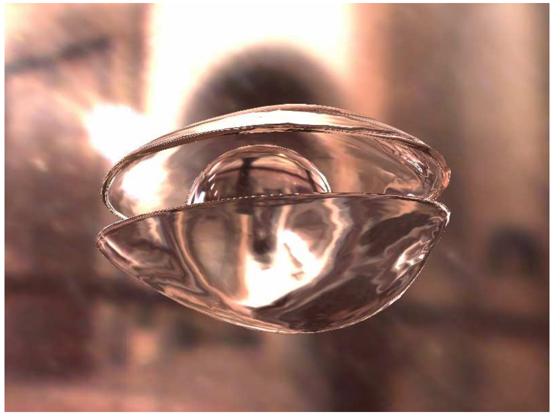
8 x 16 = 128



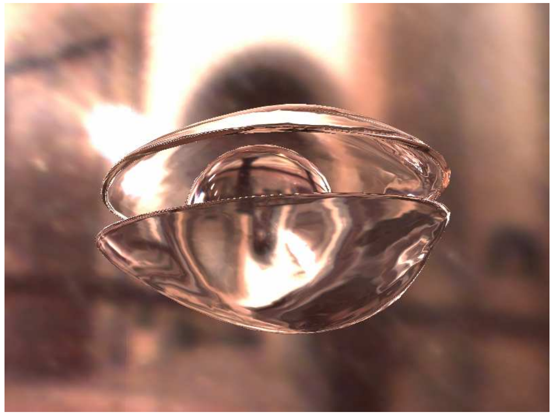
16 x 32 = 512



32 x 64 = 2048



64 x 128 = 8192

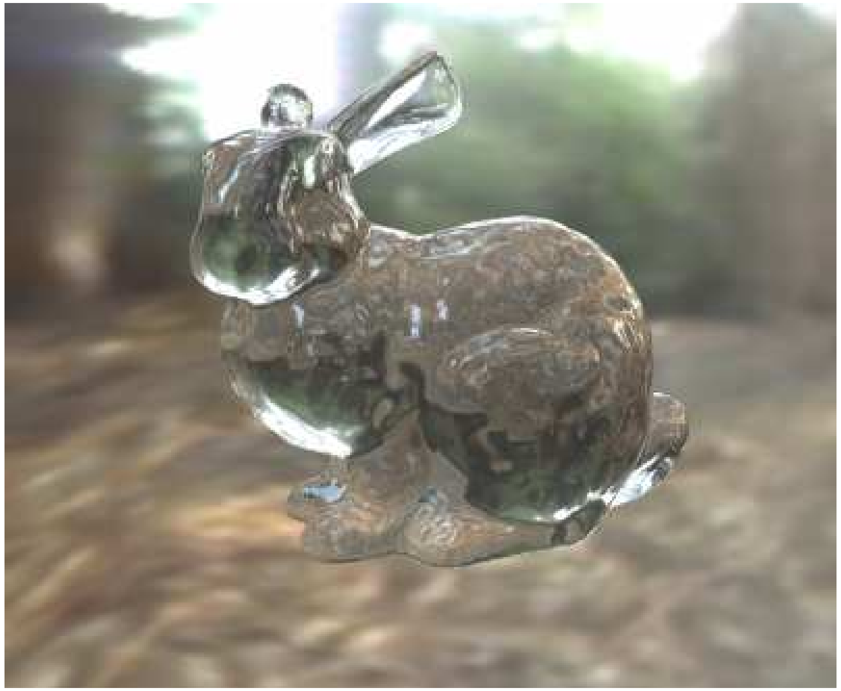


128 x 256 = 32768

Tech 6

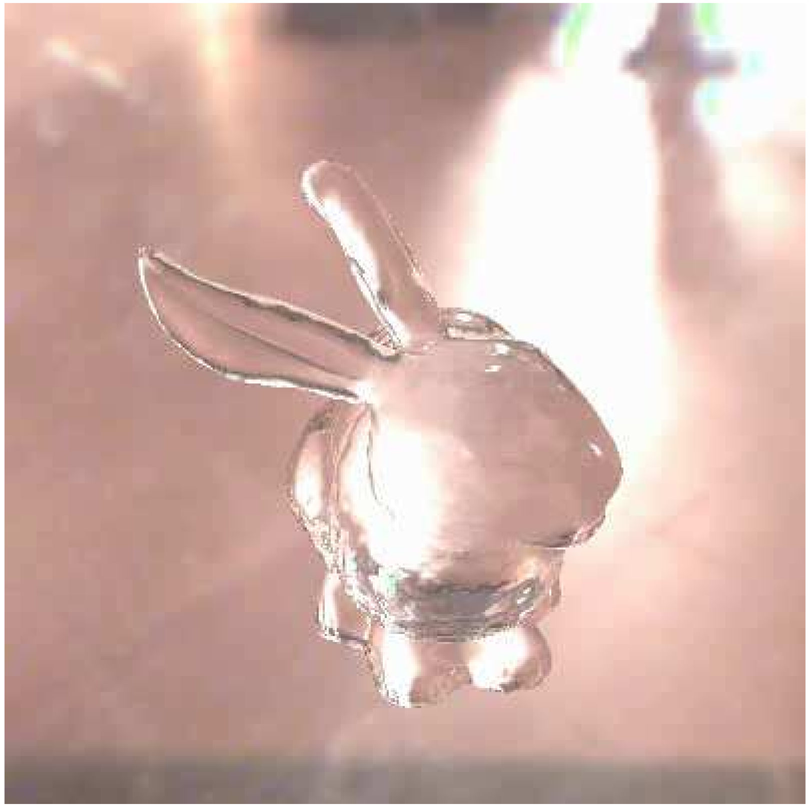
- Bunny
  - 35k surface samples
  - 2048 directions

  - 71.5M rays
  - 26 minutes
  - ~ 1.1GB data



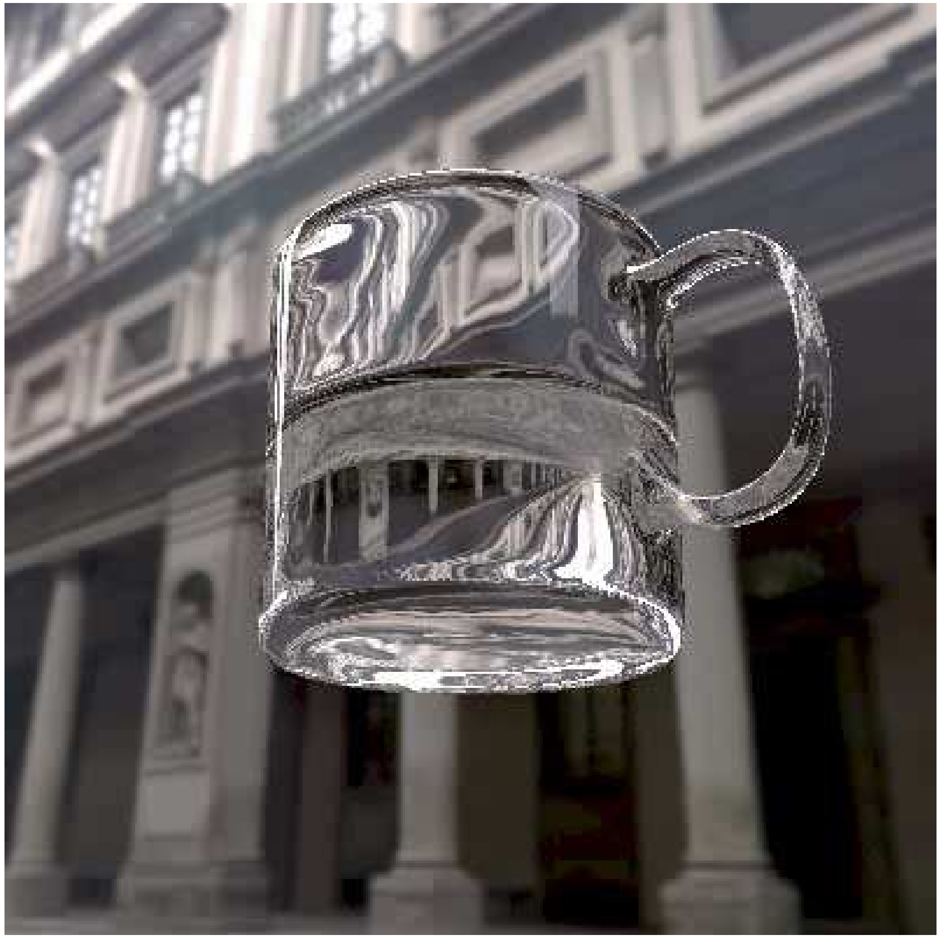
Examples 1

IOR & optical depth variation



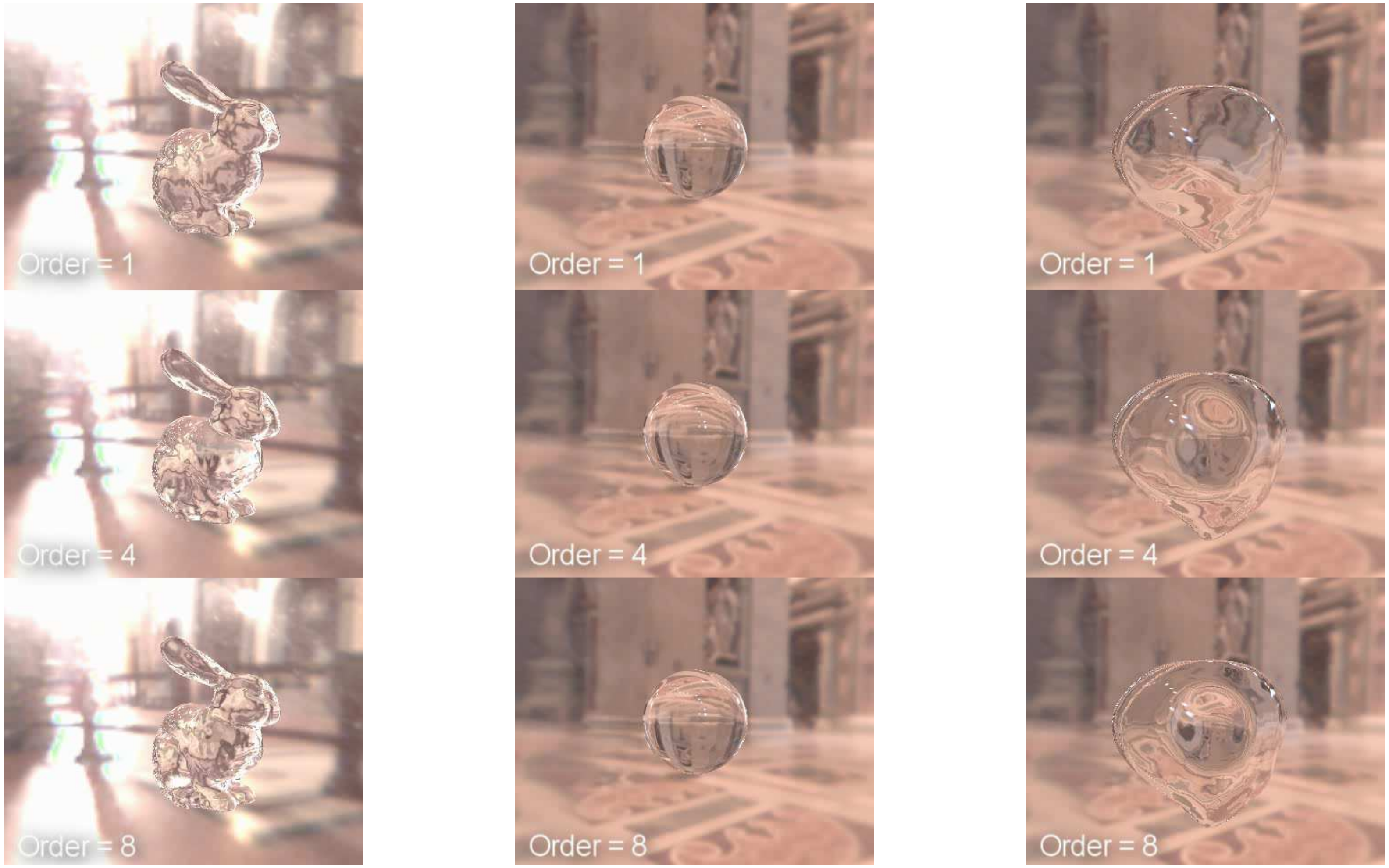
Examples 2

Multiple media



Examples 3

SH order variation



Examples 4

Rendering strategies