Algorithmen für die Echtzeitgrafik

Daniel Scherzer
scherzer@cg.tuwien.ac.at

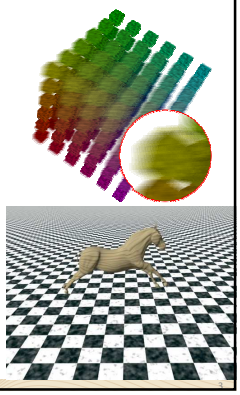
LBI Virtual Archeology

Temporal Coherence

Applications and Extensions of Reverse Reprojection

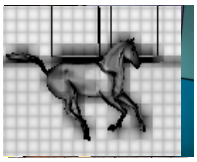
Overview

- Multi-pass rendering effects
  - Stereoscopic rendering
  - Depth of field and motion blur
- Amortized sampling
  - Theory
  - Procedual shader antialiasing



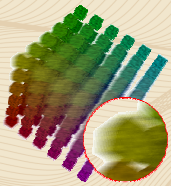
Overview

- Discrete LOD blending
- Casting shadows
  - Pixel correct shadows
  - Soft shadows
- Global illumination
  - Screen-space ambient occlusion
  - Imperfect shadow maps
- Spatio-temporal upsampling



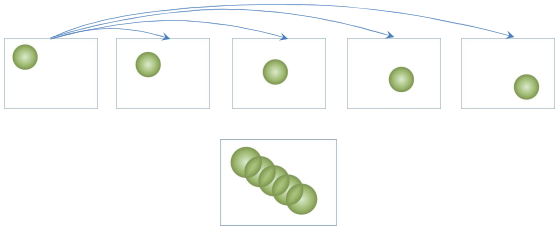
Temporal Coherence

Applications – Multi-Pass Effects



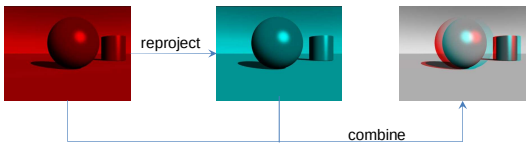
Multi-pass rendering effects

- Render a set of images with similar viewpoints
  - Shade one of them
  - Generate others using reprojection



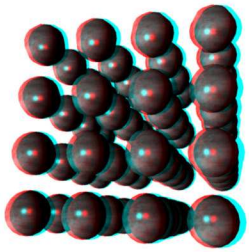
Stereoscopic rendering

- Generate images from two nearby views
- Use reverse reprojection to accelerate
  - Render the left eye normally
  - Render the right eye, fetch color from the left view
    - Cache miss: reshade
- No need to refresh



Stereoscopic rendering

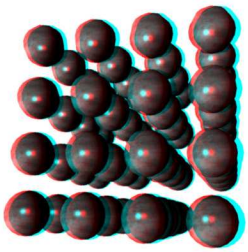
Caching view-dependent effects



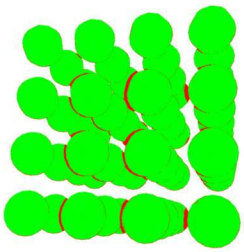
Error (artificially enhanced)

Stereoscopic rendering

Recompute view-dependent effects



Cache hit (green) miss (red)



Motion Blur

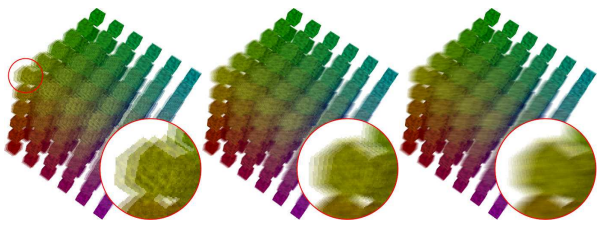
- Render scene at fractional time steps
- Blend the results
- Use reverse reprojection to accelerate:
  - Normal rendering at one time step (e.g. *t*+0.5)
  - Use reprojection when rendering others

Motion Blur

6ofps brute-force
(3 time samples)

60fps RRC
30fps brute-force
(6 time samples)

30fps RRC
(14 time samples)



Depth of field

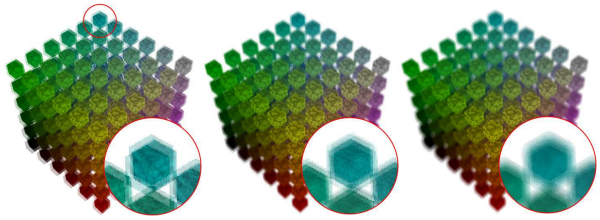
- Render from different views and blend the results
  - Sample the lens aperture region
  - Require more samples than motion blur
- Use reverse reprojection to accelerate:
  - Normal rendering at the center of aperture
  - Try to reproject when rendering other views

Depth of field

45fps brute-force
(4 aperture samples)

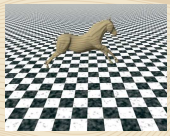
45fps RRC
20fps brute-force
(9 aperture samples)

20fps RRC
(20 aperture samples)



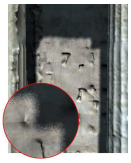
Temporal Coherence

Applications – Amortized Sampling



Amortized sampling

- Idea: use reprojection cache to amortize cost of sampling and numerical integral
- Useful in a number of applications to be covered



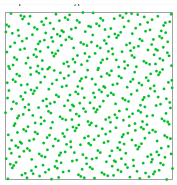
Amortized sampling

- Problem
  - Compute a Monte-Carlo estimator

$$f_N[p] \leftarrow \frac{1}{N}\sum_{i=1}^{N} s_t[p]$$

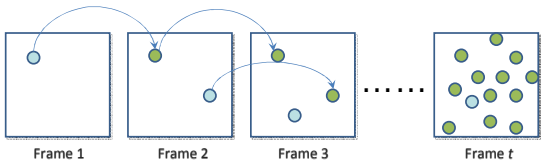
  - Can be prohibitively expensive
  - Better quality → smaller variance

$$\mathrm{Var}(f_N[p]) = \frac{1}{N}\mathrm{Var}(f_1[p])$$



Reuse sampling with reprojection

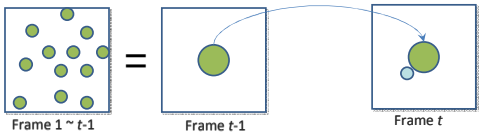
- Shading functions usually vary slowly over time
- Reuse samples from previous frames
  - Reprojection
  - Generate only one sample every frame



Amortized sampling

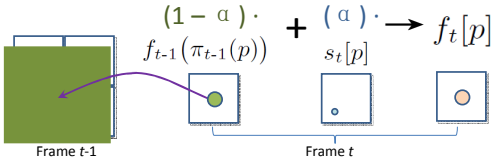
- Cannot afford to store all the samples from history
- Keep only a running accumulated result
  - Update it every frame using exponential smoothing

$$f_t[p] \leftarrow (\alpha)\boxed{s_t[p]} + (1-\alpha)\boxed{f_{t\text{-}1}(\pi_{t\text{-}1}(p))}$$



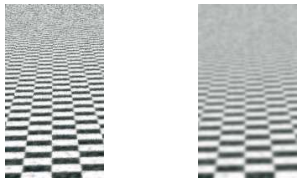
Effect of the smoothing factor α

- Larger α: less history, more aliasing/noise
- Smaller α: more history, less aliasing/noise
- Equal weight of samples: $\alpha = \frac{1}{t}$



Effect of the smoothing factor α

- Larger α: less history, more aliasing/noise
- Smaller α: more history, less aliasing/noise



Large alpha Small alpha

Effect of the smoothing factor α

- Variance reduction with constant α (steady state):

$$\lim_{t\to\infty} \mathrm{Var}(f_t^{\alpha}[p]) = \frac{1}{N_\alpha}\mathrm{Var}(f_1[p])$$

- Equivalent number of samples: $N_\alpha = \frac{2-\alpha}{\alpha}$

Procedural shader antialiasing

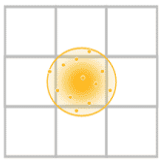
- Shading signals not band-limited
  - Procedural materials
  - Complex shading functions
- Band-limited version (analytically antialiased)
  - Ad-hoc
  - Difficult to obtain

Procedural shader antialiasing

- Supersampling
  - General antialiasing solution

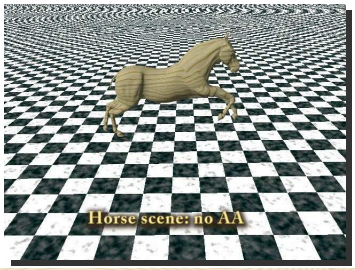
$$f_N[p] \leftarrow \frac{1}{N}\sum_{i=1}^{N} s_t[p]$$

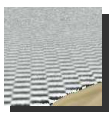
  - Amortized supersampling



Amortized supersampling

- Require small α for effective variance reduction
- Blur due to repeated bilinear interpolation



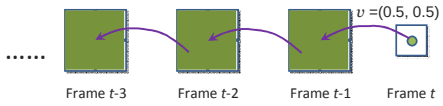


SS reproj.

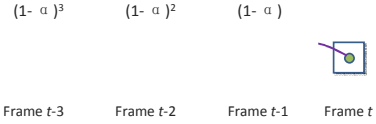
Ground truth

Factors of the blur

- Fractional pixel velocity $v = (v_x, v_y)$



- Exponential smoothing factor α



The amount of blur

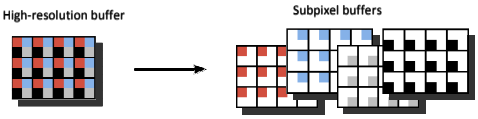
- The expected blur variance is [Yang et al. 2009]

$$\sigma_v^2 = \sigma_G^2 + \frac{1-\alpha}{\alpha}\frac{v_x(1-v_x)+v_y(1-v_y)}{2}$$

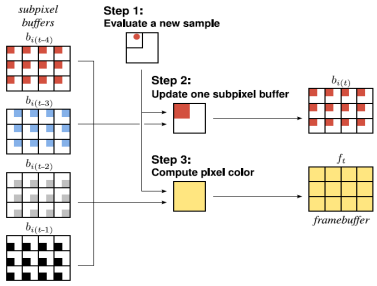
- Approaches for reducing the blur:
  1. Increase resolution of the cache buffer
  2. Avoid bilinear resampling whenever possible
  3. Limit α when needed

(1) Increase resolution

- Option 1:
  Keep a cache buffer at high resolution (2x2)
  - Have to update it every frame ☹
- Option 2:
  Keep 4 subpixel buffers at normal resolution
  - Only update one of them each frame ☺

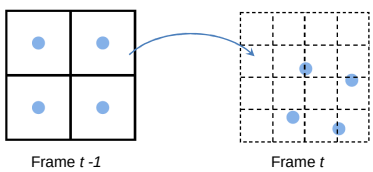


Subpixel buffers

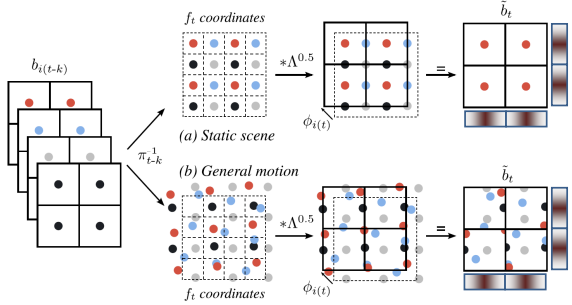


(2) Avoid bilinear sampling

- Reconstructing from subpixel buffers
  - Forward reproject the samples from 4 subpixel buffers to the current subpixel quadrant
  - Weight them using a tent function
  - GPU approximation/acceleration



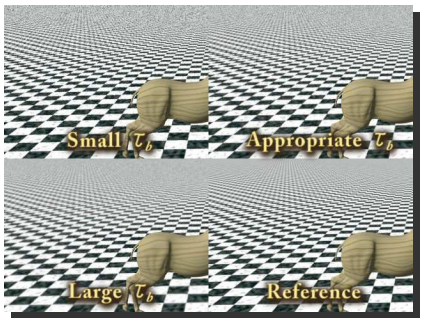
Reconstruction scheme



(3) Limiting blur via bounding α

- Derive a relationship between
  - Blur variance σ^2
  - Motion velocity v and a
- Analytic relationship is not attainable
  - Numerical simulation and tabulate
- Bound a for limiting variance
  σ^2 < a blur tolerance τ_b

Tradeoff of blur and aliasing

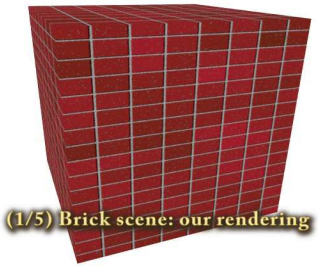


Accounting for signal changes

- Detect fast signal change
  - React by more aggressive update
- Estimate residual ε between:
  - Current sample (aliased/noisy)
  - Cache estimate
- Blur the residual estimate to remove aliasing/noise
- Bound a for limiting residual ε < threshold τ_ε

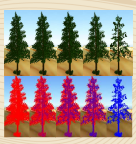
Tradeoff of signal lag and aliasing

Results



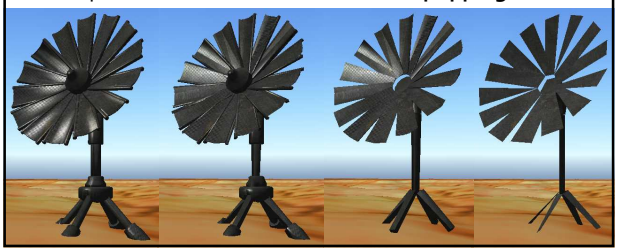
Temporal Coherence

Applications - LOD

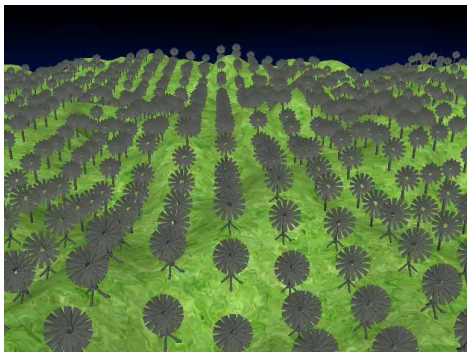


Discrete Levels-of-Detail (LOD)

- Speed up rendering by exchanging distant objects with simpler object representations (distance, size)
- Small number of object representations
- Representation switch noticeable → **popping**

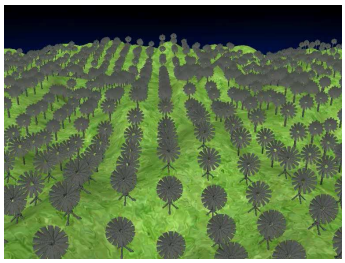


Discrete Level-of-Detail Rendering



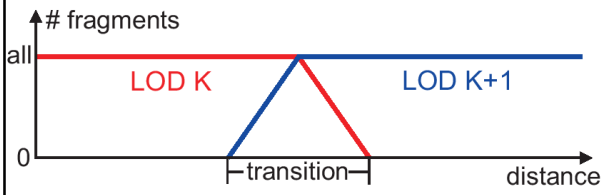
Discrete Level-of-Detail Rendering: Idea

- Interpret popping artifacts as **temporal aliasing**
- Remove hard switch

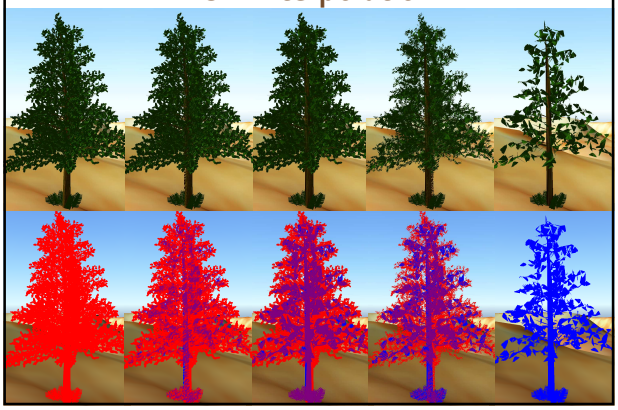


LOD Interpolation

- Introduce transition phase (distance)
  - Here both LODs are rendered (we will later avoid this with TC)
- Smoothly replace LOD K with LOD K+1
- Vary number of rendered fragments



LOD Interpolation

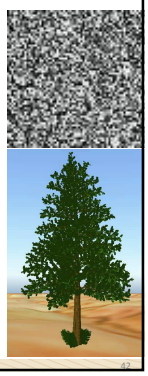


Visibility Textures

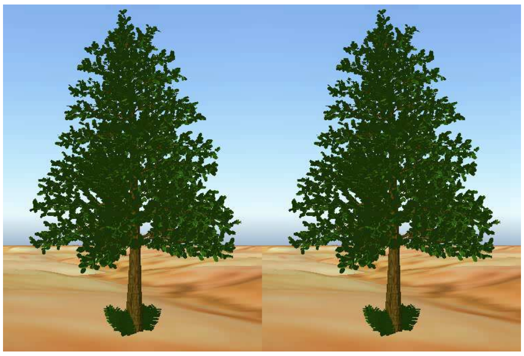
- Varying # of rendered fragments
- Encodes function of visibility
- We use noise function
- Any function possible
- Map texture to object; we use:

$$(\vec{p}.x + \vec{p}.z, \vec{p}.y + \vec{p}.z)$$

- Compare texture to threshold
- Binary result is visibility of pixel
- Varying threshold varies visibility

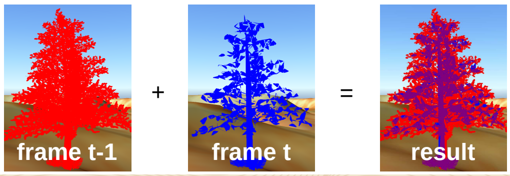


Blending vs Interpolation



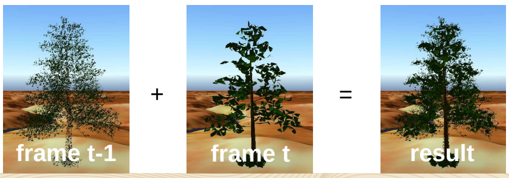
Exploiting Temporal Coherence

- Avoid rendering both LODs in same frame
  - Slower than rendering complex LOD
- Idea: render LODs in alternating frames into off-screen buffers
- Only update one buffer per frame



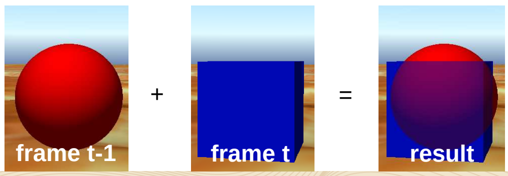
Exploiting Temporal Coherence

- Avoid rendering both LODs in same frame
  - Slower than rendering complex LOD
- Idea: render LODs in alternating frames into off-screen buffers
- Only update one buffer per frame

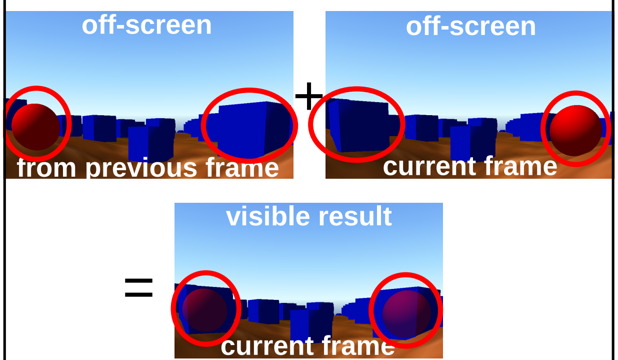


Exploiting Temporal Coherence

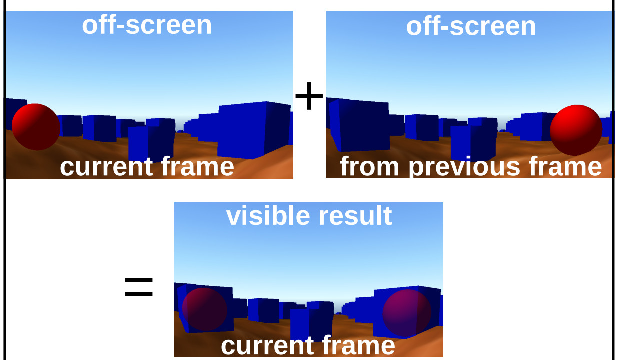
- Avoid rendering both LODs in same frame
  - Slower than rendering complex LOD
- Idea: render LODs in alternating frames into off-screen buffers
- Only update one buffer per frame



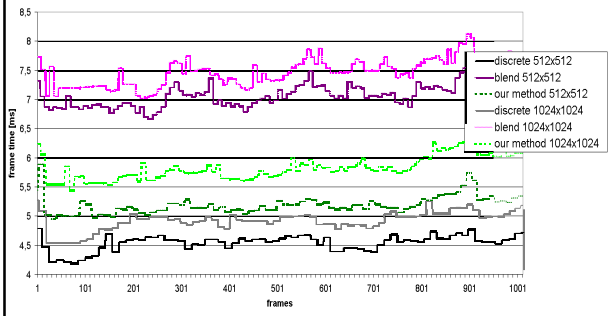
Algorithm



Algorithm: Next Frame

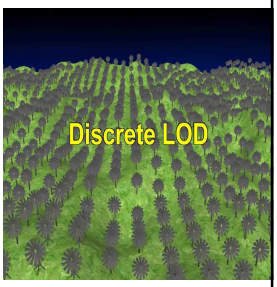


Performance



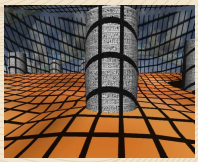
Conclusion

+ Fast
+ Arbitrary interpolations due to visibility textures
– Noise texture does not work for every type of object (do custom)



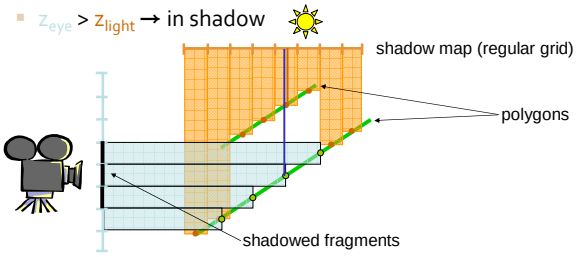
Temporal Coherence

Applications - Shadows

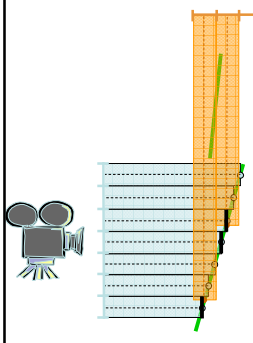


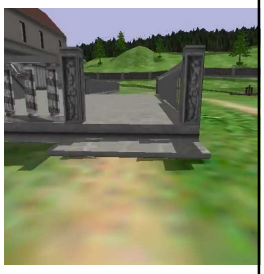
Shadow Mapping

- Shadow map stores nearest depth values
- Everything behind is in shadow
- $z_{eye} > z_{light} \rightarrow$ in shadow

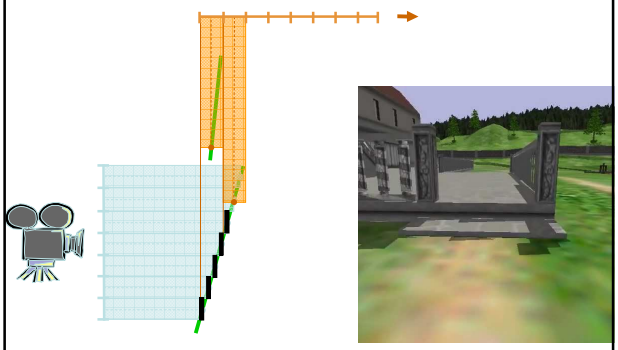


Shadow Mapping and Temporal Aliasing





Shadow Mapping and Temporal Aliasing



Shadow Mapping and Temporal Aliasing

Temporal (Exponential) Smoothing

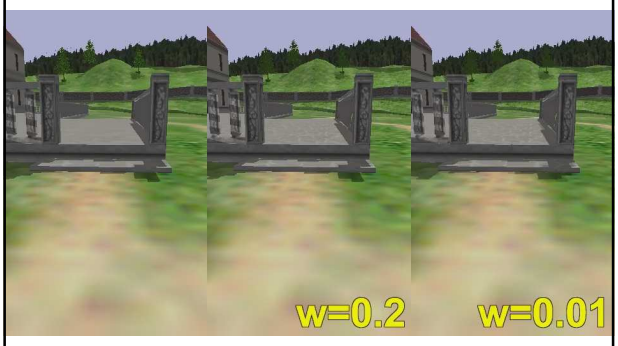
- Shadowing result of previous frame is stored in *cache*

Temporal (Exponential) Smoothing

- Shadowing result of previous frame $f_{t-1}[\mathbf{p}]$ was stored in *cache.*
- Calculate shadowing result of current frame $f_t[\mathbf{p}]$ by exponential smoothing

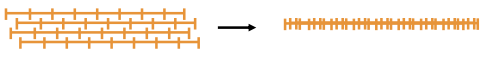
  $f_t[\mathbf{p}] := w * s_t[\mathbf{p}] + (1-w) * f_{t-1}(\pi_{t-1}(\mathbf{p}))$
  - $\pi_{t-1}(\mathbf{p})$…*transformation into previous frame*
  - $s_t[\mathbf{p}]$……*new shadow map test result, binary*
  - w……*weight (=α before), impact of current vs. previous*
    - Large w - fast update, weak smoothing
    - Small w - slow update, strong smoothing

Temporal (Exponential) Smoothing

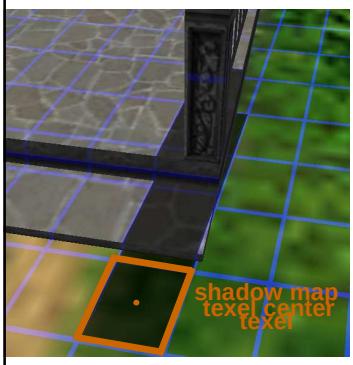


Temporal Smoothing: Observation

- Strength of smoothing depends on movement
- Stand still → smoothing disappears
- Solution: jitter shadow map each frame
  - Creates new rasterization each frame
  - Virtually infinite shadow map resolution
  - Can we take advantage of this to create more detailed shadows?

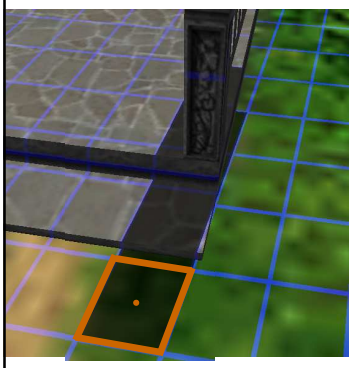


Confidence Estimation

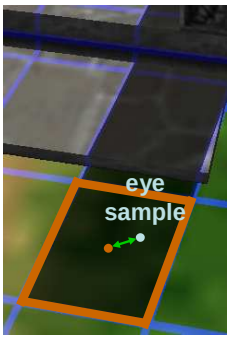


- Shadow map only correct at center
- But used for whole texel area

Confidence Estimation



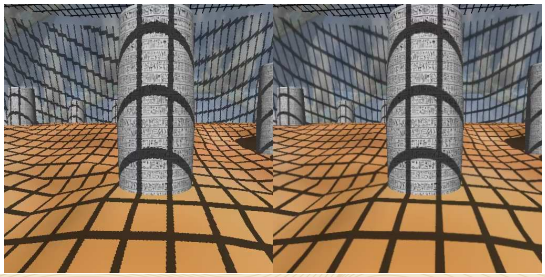
Confidence Estimation



- Shadow map test probably more correct closer to texel center
- Confidence:
  conf := 1- *dist*(eye,texel)
- Greater confidence
  → greater impact

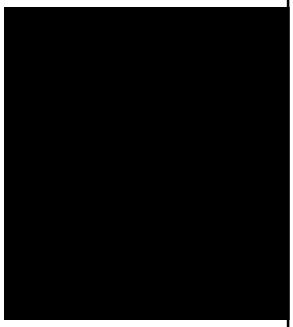
Confidence and Temporal Smoothing

- Use confidence as weight for temporal smoothing
  $f_t[\mathbf{p}] := conf * s_t[\mathbf{p}] + (1 - conf) * f_{t-1}(\pi_{t-1}(\mathbf{p}))$



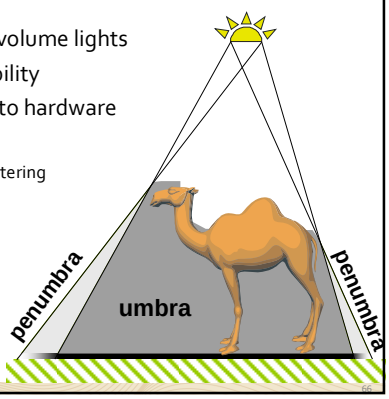
Conclusion

+ High quality shadows for static scenes
+ Overhead below 1ms in comparison to standard shadow mapping
– Standard shadow mapping quality for dynamic scenes
– No guaranteed convergence speed:
  more undersampling
  → slower convergence



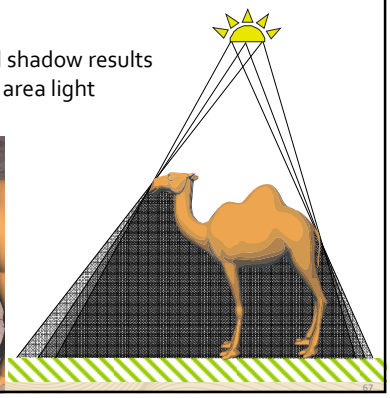
Soft Shadows

- Created by area/volume lights
- From region visibility
- Does not fit well to hardware
- Fake techniques
  - Single sample + filtering
    - PCSS
    - Backprojection
    - …



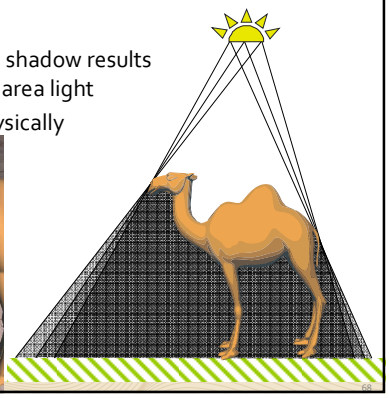
Light Source Area Sampling

- Accumulate hard shadow results from samples on area light



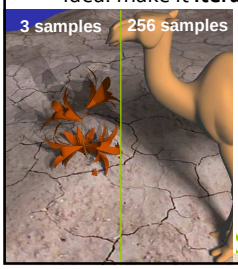
Light Source Area Sampling

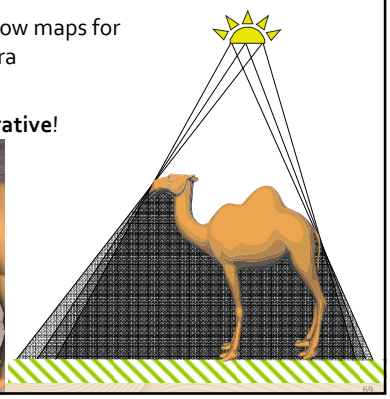
- Accumulate hard shadow results from samples on area light
- Converges to physically



Light Source Area Sampling - Problems

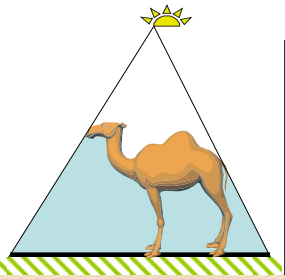
- Need many shadow maps for smooth penumbra
- Slow
- Idea: make it **iterative**!

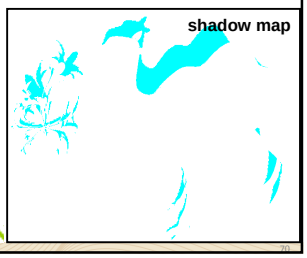




Iterative Light Source Area Sampling

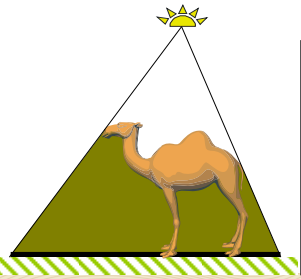
- Create only one shadow map each frame
  - At a random position on the light source

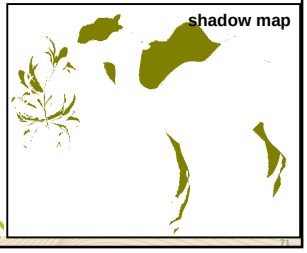




Iterative Light Source Area Sampling

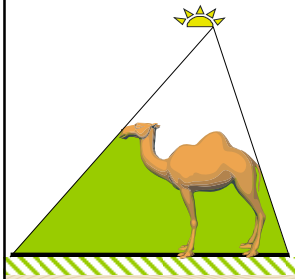
- Create only one shadow map each frame
  - At a random position on the light source

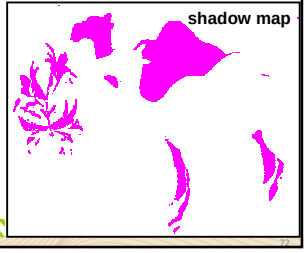




Iterative Light Source Area Sampling

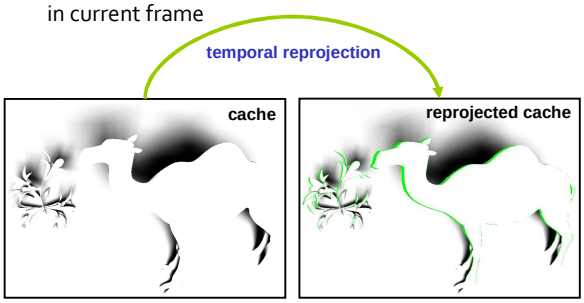
- Create only one shadow map each frame
  - At a random position on the light source





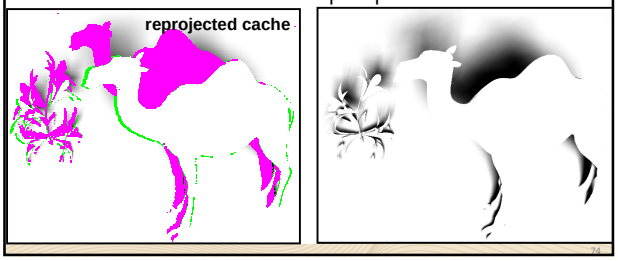
Iterative Light Source Area Sampling

- Idea: Use shadow information from previous frame in current frame



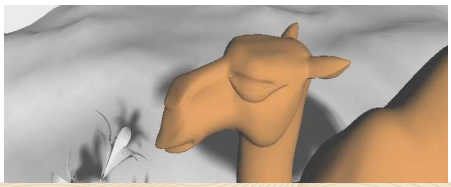
Iterative Light Source Area Sampling

- Create only one shadow map each frame
- Combine with reprojected cache
- How to combine is the complex part



Combination

- Each shadow map creates a sample for each visible fragment
- Fragment longer visible → more samples → more correct soft shadow
- Idea: statistical approach (standard error)



Combination: Disoccluded Areas

- No previous shadow information
  - Standard error high (only 1 sample)
  - Use PCSS or similar approach as first sample
- Neighbourhood-aware filter

Combination: Disoccluded Areas

- Some shadow information
- High standard error: neighbourhood-aware filter
- Low standard error: blend to only sampled shadows

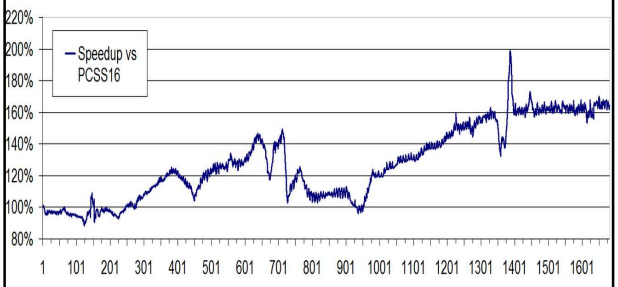
Results: Adaptation Over Time

Iterative area sampling

Iterative area sampling
+ PCSS start
+ neighbourhood filter

Performance vs Optimized Soft Shadow

- PCSS16: very fast trick used in games; artifacts;



Conclusion

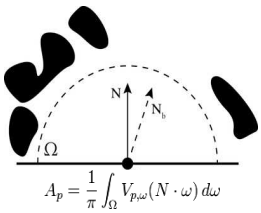
+ High quality soft shadows for static scenes
+ Even faster than fixed kernel size PCSS
– Fall back to PCSS quality for dynamic scenes
– No guaranteed convergence speed: bigger penumbra → slower convergence

Temporal Coherence

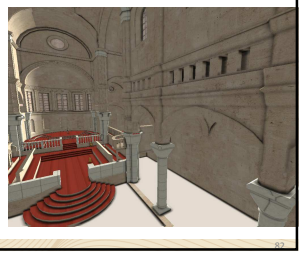
Applications - GI

Ambient Occlusion

- Cheap approximation of global illumination
- % of hemisphere that is blocked
- Integrate binary visibility function V

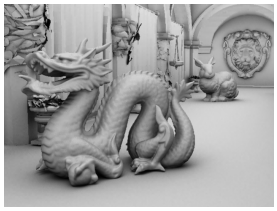


$$A_p = \frac{1}{\pi}\int_{\Omega} V_{p,\omega}(N \cdot \omega)\, d\omega$$



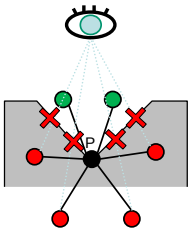
Screen-Space Ambient Occlusion (SSAO)

- Rasterize scene
- Use depth output as **discrete scene approximation**
+ Only visible parts are calculated
- Loses some occluder information (outside of screen)



SSAO: Method of Crytek

- Cast 8 – 32 samples on a sphere
- Farther away than surface → direction occluded
- Evaluate % of occluded samples



Temporal SSAO

- Combine SSAO + temporal reprojection (TSSAO)
- Reuse AO

Happy Buddha
(~1M triangles)

25 FPS

Screen-Space Ambient Occlusion (SSAO)

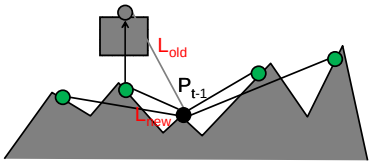
Detecting invalid old pixels

- When is old pixel invalid?
  1. Disocclusion (sufficient for static scenes)
  2. Change in the neighborhood (dynamic scenes)

Translational
movement

Invalidation: Detect changes

- Only relative changes important
- Cast invalidation samples
- Reproject samples into frame $_{t-1}$
- Compare old to new length

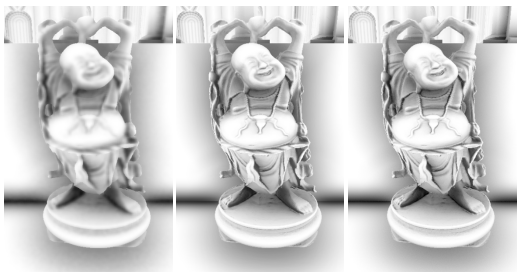


Frame $_{t-1}$

Adaptive Convergence-Aware Filter

- Filter **only** undersampled pixels
- Depending on number of **accumulated samples**
  - Filter size
  - Sample weight

Results: Happy Buddha (~1M triangles)



SSAO
23 FPS

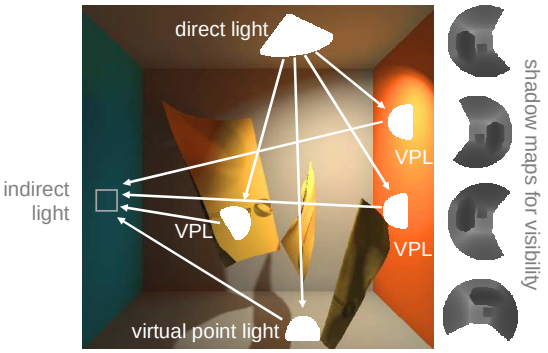
TSSAO
45 FPS

Reference
2.5 FPS

Conclusion

+ Faster than SSAO
+ Higher quality than SSAO
– Artists can often hide SSAO artifacts by careful texture design
– Only faster if scene moderately dynamic
– Worst case (quality and speed): SSAO

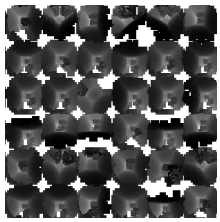
Instant Radiosity



Courtesy of Tobias Ritschel

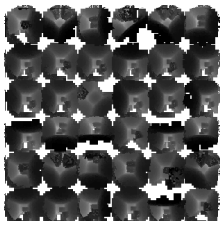
Imperfect Shadow Maps

- Make it fast:
  - Assume mostly diffuse lighting
  - → VPLs: low-res shadow maps
  - Keep all in big tiled texture



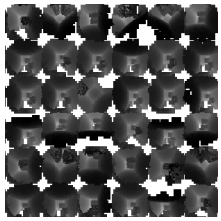
Imperfect Shadow Maps

- Make it fast:
  - Create from point cloud representation of scene
  - Splat points into big tiled texture
  - Fill holes



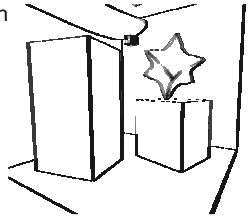
Imperfect Shadow Maps: Problems

- VPLs may change each frame
  - → Temporal aliasing
  - → Introduce temporal coherence



TC Imperfect Shadow Maps

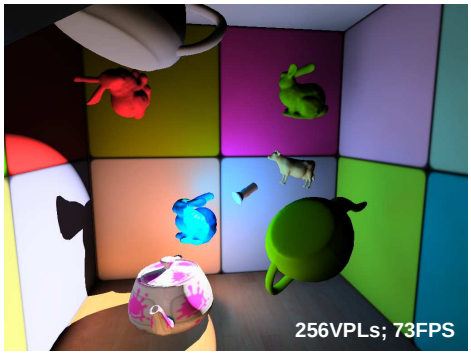
- Store illumination information from previous frame
- Introduce smooth invalidation scheme:
  Confidence is maximum
  - Position change
  - Normal change
  - Illumination change
- Used as weight in exponential smoothing



ISM (134FPS) vs TC ISM (125FPS)

64VPLs

TC Imperfect Shadow Maps

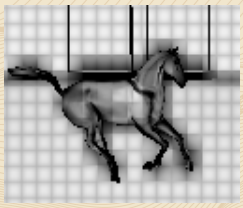


Conclusion

+ Better quality than ISM for same number of VPLs
+ Avoids temporal aliasing prominent in ISM
– After glow artifacts for fast moving objects or sudden light source occlusion
– Method is very scene dependent: confidence formula has a number of user parameters

Temporal Coherence

Applications - Spatio-temporal upsampling

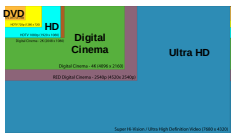


Motivation

- Increasing display resolution
- Increasing display frame rate
- Stereo viewing

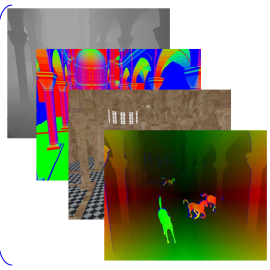
More coherent in time and space!

- More realistic and complex
- But more and more pixels to process!



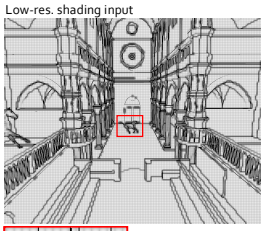
GPU Pixel Shading: Observation

- Shading correlates with geometry
- All world information behind pixel given for "free"
  - Depth (position)
  - Normals
  - Materials, Textures
  - Geometric motion flow
- Final shading is bottleneck



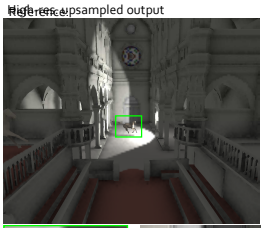
Joint-Bilateral Upsampling

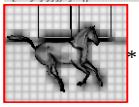
Low-res. shading input



High-res. upsampled output

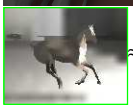
Reference





*

=

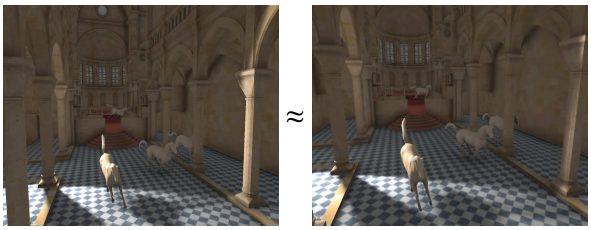


≈

Courtesy of Robert Herzog

Exponential Temporal Caching

- Exploit temporal coherence in shading
  - i.e., reuse pixel shading of previous frame(s)



t ≈ t + 1

Spatio-Temporal Upsampling

- Depending on situation one method is better than the other

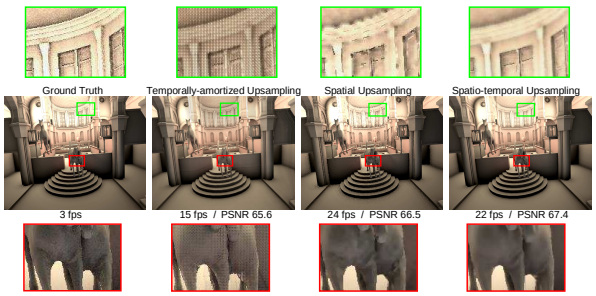
→Combine spatial upsampling with temporal caching

- How to choose weight in exponential smoothing?
  - Adapt exponential weights to **temporal gradient**

Results Dynamic Scene + Camera

(1280x1024, 4x4 upsampling)

- SSAO (12x8 samples) + indirect light (1000 VPLs)



Conclusion

+ Reduction of pixel shading cost with constant overhead
+ Combines benefits of spatial upsampling and temporal reprojection caching
– Flickering for high frequency textures and shading (upsampling)
– Higher gains with special purpose methods possible (TSSAO)