Practical use of Screen Space Ambient Occlusion

Michał Drobot
Visual Technical Director
Reality Pump
Talk Outline

- Motivation
- Theory
- General Idea
- Practical Solution in Details
  - Screen Space to Camera Space
  - Raycasting
  - Blocker Test
  - Filtering
  - Compositing with HDRi
Talk Outline

• Tackling Problems
  • Deffered Renderers
  • Forward Renderers
• Performance & IQ Optimizations
• Further Development
• Summary
• Questions?
Motivation

• Multiplatform
• HQ seamless land/city-scape rendering
  • Realistic lighting model:
    • Dynamic day cycle
    • Dynamic lights
    • Dynamic environments
    • Global Illumination Approximation - taking into consideration all above = PROBLEMATIC!
Motivation
Motivation
Motivation

- Solutions:
  - PRT - maps
  - Per Vertex SH
  - Point Clouds
  - Inhouse off-line solutions...

- All failed:
  - Memory consumption (X360)
  - Performance
  - Flexibility
Motivation

- Needed new solution
- Dynamic
- Scalable
- Max Performance/Memory ratio

Screen Space Solution
- Image Depth enhancement publications
- Hybrid rendering engine
- Z-based post processing
Motivation
Theory

- Ambient Occlusion - Technically
- Global Illumination approximation
- Omnidirectional lighting
- Accessibility shading
- Film Industry proven prepass Skylight
Theory

• AO - visual appeal
  • Artists using Skylight bake in textures
  • Enviromental effect - surface reaction to:
    • Dirt/weathering
    • Light
  • Prepass for Global Illumination
  • Enhances scene by depth, curvature, spatial proximity clues
Theory

- Integral of blocker function over the hemisphere

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

- Point A is not occluded
- Point B is darkened (high geo. Proximity)
General Idea
General Idea

- Screen Space = simplified calculation domain (performance)
- Depth buffer
  - Resembles scene in SS
  - Generally available
  - Preferably linear - high precision (>FP16)
- Pixel \((x,y)\) in SS & DB val = \((x,y,z)\) in CS
- Camera Space = simplified calculation domain (complexity)
Theory

• For each pixel in SS
  • For 1 to N samples
    • Transform to CS domain
    • Raycast in CS in random direction on point’s normal hemisphere
    • Gather irradiance information from surface hit check
    • Cumulate results considering atmosphere attenuation, radiance energy
  • Return resulting irradiance

Composite final SS result layer with HDRi
Practical Solution in Details
Screen Space to Camera Space

- SS pixel \((x,y)\) \& depth value = \((x,y,z)\) in CS
- Min 1 mul = fast
- Keep your depth high precision to avoid artifacts
- Preferable 32FP linear depth (depends on your scene depth range)
- Our solution is based on 16FP linear depth transformed from original log Depth Buffer (gives filterable artifacts)
Raycasting

- Raycast by raymarch
- Generate normalized hemisphere of randomly distributed points with uniform distribution ie. Poisson
- Rotate hemisphere so its’ horizon is perpendicular to point’s normal (for now we assume to have it)
- Scale the hemisphere for desired effect range

Offset CS original point by generated points
Raycast

Transform new point from CS to SS

- Acquire points z value from our scene in SS
- We can also acquire additional info like diffuse color, radiance energy etc.
Blocker Test

- In the same domain we’ve got
  - Depth of ‘possible’ hit (pd)
  - Real depth at (x,y) of ‘possible’ hit point (rd)
- We want to check if we hit something with our ray
- If something exists in our depth buffer at SS coordinates of ‘pd’ and at the same depth we’ve got a HIT
Blocker Test

- We need a comparison function considering attenuation, limited samples, possible precision errors
  - Real Distance Proximity Evaluation
  - Depth Difference Evaluation
Blocker Test (RDPE)

- Real Distance Proximity Evaluation (RDPE)
  - \( \text{Abs}(pd-rd) < E \)
    - If TRUE = HIT
      - Accumulate results - Occlusion += \( \frac{1}{1+ (\text{length}(\text{ray}) \times \text{scale})^2} \)
    - If FALSE = miss
      - Discard results - SamplesNo--;
        - Alternativly for high accuracy repeat whole process of raycasting using linear/binary search
  - finalOcclusion = Occlusion/SamplesNo
Blocker Test (RDPE)

• Pros
  • conceptually precise results
  • Superb quality with linear/binary raymarching (matching offline renderers)
  • Ideal for GI
  • Global (full screen) solution

• Cons
  • Discarding MISSes leads to undersampling, requiring more sample with good distribution
  • Requires ray length calculations
  • Insane sampling rate with raymarching solution: SampleNo * PerSampleMissMissSampling
Blocker Test (RDPE)

- Scaleability
  - E - Epsilon / Sample No Ratio
  - Iteration/E limit for raymarching
  - Still Raymarched solution needs at least 16 subsamples on avg (using flow control)

- Practically useless due to undersampling artifacts while using sane number of samples
Blocker Test (DDE)

- Depth Difference Evaluation (DDE)
  - Use all depth difference data with attenuation function
    - Occlusion = \( \frac{1.0}{1.0 + (pd - rd)^2} \)
    - FinalOcclusion = Occlusion/SamplesNo
Blocker Test (DDE)

**Pros**
- Fast
- Simple
- Perceptually accurate

**Cons**
- Lacks accuracy
- Gives more local results
- Generally too local for GI
Blocker Test (DDE)

- Scaleability
  - Sample rate tweaking
  - Scale samples with Z - depth (flow control)

- Practically DDE is a WIN
- But... only for this generation...
Filtering

- Acquired Occlusion layer due to undersampling artifacts should be filtered.
- Due to AO low-frequency nature denoise blurring filters are the weapon of choice.
- Should smooth out results, denoise the signal and retain the details.
Filtering
Filtering

- Gaussian Filter
  - Weighted average of sampled values
  - Requires preprocessed kernel weights
  - Kernel of N x N pixels
Filtering

- Gaussian Filter
  - 2 pass filtering with interpolator usage
  - Very fast $O(N)$
  - Good smooth and denoise ability
  - Removes details, blurs edges
  - AO bleeding occurs
  - Introduces new artifacts (halos, bleeding)
Filtering

Gaussian 18x18
Separable
18 taps
Filtering

- Median filter
  - Median of sampled values
  - Kernel of NxN pixels
  - Heavy use of vector arithmetics for value sorting
Filtering

- Median filter
  - 1 pass artifact free filter is $O(N^2)$
    - That’s too slow for reasonable kernel size
    - Possible use of combined multipass filters of small size
    - IQ better than gaussian, but too slow
  - 2 pass (V/H) is only $O(2*N)$
    - Fast enough
    - Not correct
    - May introduces block artifacts
    - Usable as multipass/varied kernel size filter
Filtering

Median 18x18 Separable 18 taps
Filtering

Median 2x(6x6)
Non Separable
18 taps
Filtering

- Bilateral filtering
- 3D filter
- Filtering in domain of spatial proximity and intensity
- Possible fast implementation using vector arithmetic

\[ BF[I]p = \frac{1}{Wp} \sum_{q \in S} (G[d](\|p-q\|) \cdot G[i](\|I_p-I_q\|) \cdot I_q) \]
Filtering

- We can simplify BF using same Gaussian for proximity and intensity domains.

- For each pixel $p$:
  - Normalize Sum of:
    - For each pixel $q \in S$:
      - $\text{Gaussian}(||p-q||) \times \text{Gaussian}(||p_l-q_l||) \times l_q$
Filtering

- Bilateral filter is NOT separable
  - But results of separable filtering are acceptable
- We can use linear filtering for bigger kernels
- Optimized BF is $O(n)$ and as fast as gaussian when done right
Filtering

Bilateral 18x18
Separable
18 taps
Filtering

Bilateral 9x9
Separable
18 taps
Filtering

- Bilateral Filter
  - Gives very good smoothing
  - Retains details
  - Doesn’t affect edges
  - Prevents bleeding
  - Is fast
  - Is our choice
Filtering

Original

Gaussian 18x18

Bilateral 18x18

Median 2x(6x6)

Median 9x9

Bilateral 9x9
Compositing with HDRi

- **Forward rendering**
  - Compose as post process by multiply with HDRi before tone mapping
    - Exposure will need adjustment
  - Post HDR - AP soft light layer processing (or other chosen by artists)

- **Deferred rendering**
  - Do it first
  - Use it as ambient occlusion term during material shading and lighting pass
Tackling Problems
Tackling Problems

- **Deffered Rendering**
  - easy access to WS normal layer
  - Normal steps as stated before
  - Gives highest quality and stability

- **Optimization of hemisphere generation**
  - Generate sphere of points
  - Mirror all along plane perpendicular to pixel’s normal (transformed to CS)
Tackling Problems

- Forward Rendering
  - We have NO acces to normal layer
  - We CANNOT generate hemisphere of point
  - Our only solution is to generate a sphere
Tackling Problems

- FR Sphere sampling problems
  - RDPE
    - Raytracing ‘inside’ surface
      - Performance loss / undersampling in RDPE
  - DDE
    - Can’t find backface rays
    - Backface rays return false results
    - Lack of stability due to non uniform front/backface sampling rate
    - Are we doomed?
Tackling Problems
Tackling Problems

- FR DDE continued...
  - Naive solution - acceptable results
  - Discard all results for rays ‘behind’ point we are shading
    - Pros
      - Gives acceptable results
    - Cons
      - Undersampling due to 50% loss of data on average
      - Not accurate because we’re discarding ‘good’ points
      - Lacks stability = introduces view dependency because of non uniform no of samples at higher angles
Tackling Problems

• FR DDE - Real Solution
  • Observation
    • In CS when (pd-rd)<0 (checked depth was ‘behind’ ray) there’s no occluder affecting current pixel
    • We can use it to change our blocker function
  • New blocker function
    • if (pd-rd)<0
      • TRUE - Discard results - SamplesNo--;
      • FALSE - HIT - Occlusion += 1.0/(1.0+abs(pd - rd)^2)
    • finalOcclusion = Occlusion/SamplesNo
Tackling Problems

- FR DDE - Real Solution
  - New blocker function
    - Gives very good accuracy
    - Undersampling occurs but is greatly minimized on average
    - Stil high viewdependancy due to nonuniformity of samples
  - We want to guarantee that each pixel at each angle will be shaded with same amount of samples
Tackling Problems
Tackling Problems

- FR DDE - Real Solution
  - GEOSPHERE to the rescue
  - Instead of randomly generated sphere of ray points we use randomly rotated points lying on GEOSPHERE
  - Geosphere guarantees that plane slicing it will always divide it’s point number in half
  - Guarantee of uniform samples solves view dependancy problems
  - For better accuracy we can generate geosphere in geosphere point clouds
Tackling Problems
Tackling Problems
Tackling Problems

- FR DDE - Real Solution
  - Summary
    - Use enhanced blocker function
    - Use Geosphere for ray generation
Performance & IQ Optimizations

• Use LO-res buffers for SSAO rendering
• Efficient raycasting
• Efficient filtering
• Bilateral filtering / Bilinear upsampling combo
Performance & IQ Optimizations

- Use jittering
  - Use dependant depth reads instead of high sampling rate
- Precompute Your Point Sphere
  - Geocube
  - Poisson distribution
- And rotate it by random vector every pixel
  - Reflect(SpherePos[i],RandomVector)
Performance & IQ Optimizations

• Acquire random vectors
  • Use pregenerated noise textures
  • Map them 1-1 pixel ratio in Screen Space
• Scale sample number to your needs
• Scale search radius and sample number according to depth
• Use as fast image format as you can for depth sampling depending on hardware
  • Pure SSAO (FP16, FP32)
  • Irradiance transfer (RGBA16)
Performance & IQ Optimizations
Further Development

- Reuse results from last N frame buffer
  - Efficiency/accuracy due to low frequency nature of AO
- Compute Irradiance transfer
  - Color bleeding with DDE
    - Gather diffuse color samples transformed with IR Energy transfer matrix
- GI approximation with RDPE
  - Can use NdotL diffuse layer for IR Energy transfer approximation in whole scene
Summary

• Main steps
  • Calculate SSAO
    • FR / Deferred Solution
    • RDPR / DDE
  • Filter Results
  • Compose
Summary

• Our solution uses
  • FR DDE
    • Special image enhancement blocker function for edge highlighting
    • Geosphere (18v on PC / 12v on X360) = 18/12 samples
    • Jittering
    • Flow control
    • ¼ RT size for SSAO generation and filtering
    • Separable Bilateral Filter
    • RGBA16 – RGB irradiance, A depth
Summary

- SSAO done right
  - Enhances your scene
  - Fakes lightmaps where they aren’t
  - Can be fast
  - Can fake GI color bleeding
  - Is only local thus viewdependent
    - But illusion is good enough
  - Is a good start for future more complex scene lighting solutions
Summary

- With SSAO as GI approximation
  - You don’t need additional textures for assets
    - Pure diffuse (no baked AO)
    - Normal/height map
    - Specular
    - Additional special textures
  - SSAO can substitute lightmaps in many cases
  - Leave all lighting to your engine
Summary

For more information contact me

hello@drobot.org

Slides, whitepaper and code will be available at

Drobot.org

Everybody is welcome at

Booth 20 Hall 5 GC
Summary

Special thanks goes to

Mirosław Dymek
Michał Orkisz
Mariusz Szaflik
Demo

Demo time :)

Fully RT SSAO

‘AS IS’ in our game environment

WiP
Questions
Update

• Screens show current implementation results for X360
• Visual appeal tweaked taking into consideration artistic style
• Shown screens represent output of whole SSAO pipeline with last step of image enhancement (contrast etc.)
• Tech and results are subject to change
• Code & whitepaper will be soon available :]

Two Worlds: The Temptation
Update
Update
Update