VGP353 – Week 8

- Agenda:
  - SSAO:
    - Bilateral filtering
    - Horizon-based AO
    - Multi-layer dual-resolution SSAO

Bilateral Filtering

- Special filter that blurs pixels that are near each other

$$A_p = \frac{1}{k(p)} \sum_{p' \in \Omega} g_d(p' - p)\, g_r(A_p - A_{p'})\, A_{p'}$$

  - g_d sets the filter weight based on the image space distance
    - This is the usual Gaussian filter coefficients
  - g_r sets the filter weight based on the distance between the pixel *values*

Bilateral Filtering

- Special filter that blurs pixels that are near each other

$$A_p = \frac{1}{k(p)} \sum_{p' \in \Omega} g_d(p' - p) g_r(A_p - A_{p'}) A_{p'}$$

  - $k(p)$ is a normalization term:

$$k(p) = \sum_{p' \in \Omega} g_d(p' - p) g_r(A_p - A_{p'})$$

Bilateral Filtering

- What does this do?

Bilateral Filtering

- What does this do?
  - Maintains large, high-frequency elements
    - In other words, edges
  - Smooths noise in other areas

Bilateral Filtering

- How is this useful in post-processing 3D images?

Bilateral Filtering

- How is this useful in post-processing 3D images?
  - If we change the definition of g_r, we can prevent filtering across geometric edges
  - Have g_r return 0 if the parameter is above some threshold or 1 otherwise

$$A_p = \frac{1}{k(p)} \sum_{p' \in \Omega} g_d(p' - p) g_r(Z_p - Z_{p'}) A_{p'}$$

Bilateral Filtering

- Is this a separable filter?

Bilateral Filtering

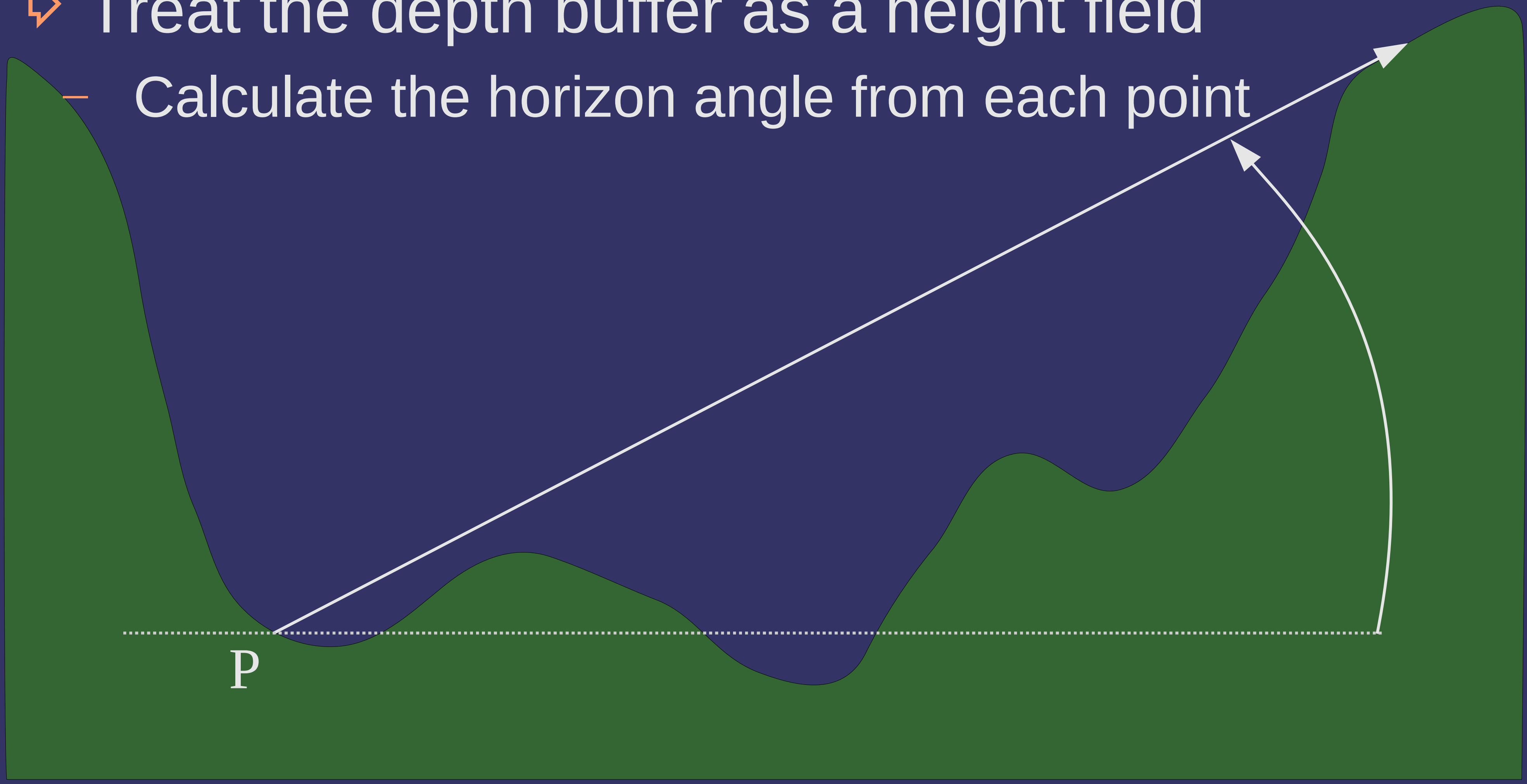
- Is this a separable filter?
  - Technically it isn't due to the g_r term
  - Many uses of bilateral filter can treat it as separable without noticeable side-effects
    - SSAO being one of those uses!

Reference

Petschnigg, G., Szeliski, R., Agrawala, M., Cohen, M., Hoppe, H., and Toyama, K. 2004. Digital photography with flash and no-flash image pairs. ACM Trans. Graph. 23, 3 (Aug. 2004), 664-672. http://research.microsoft.com/en-us/um/people/hoppe/flash.pdf

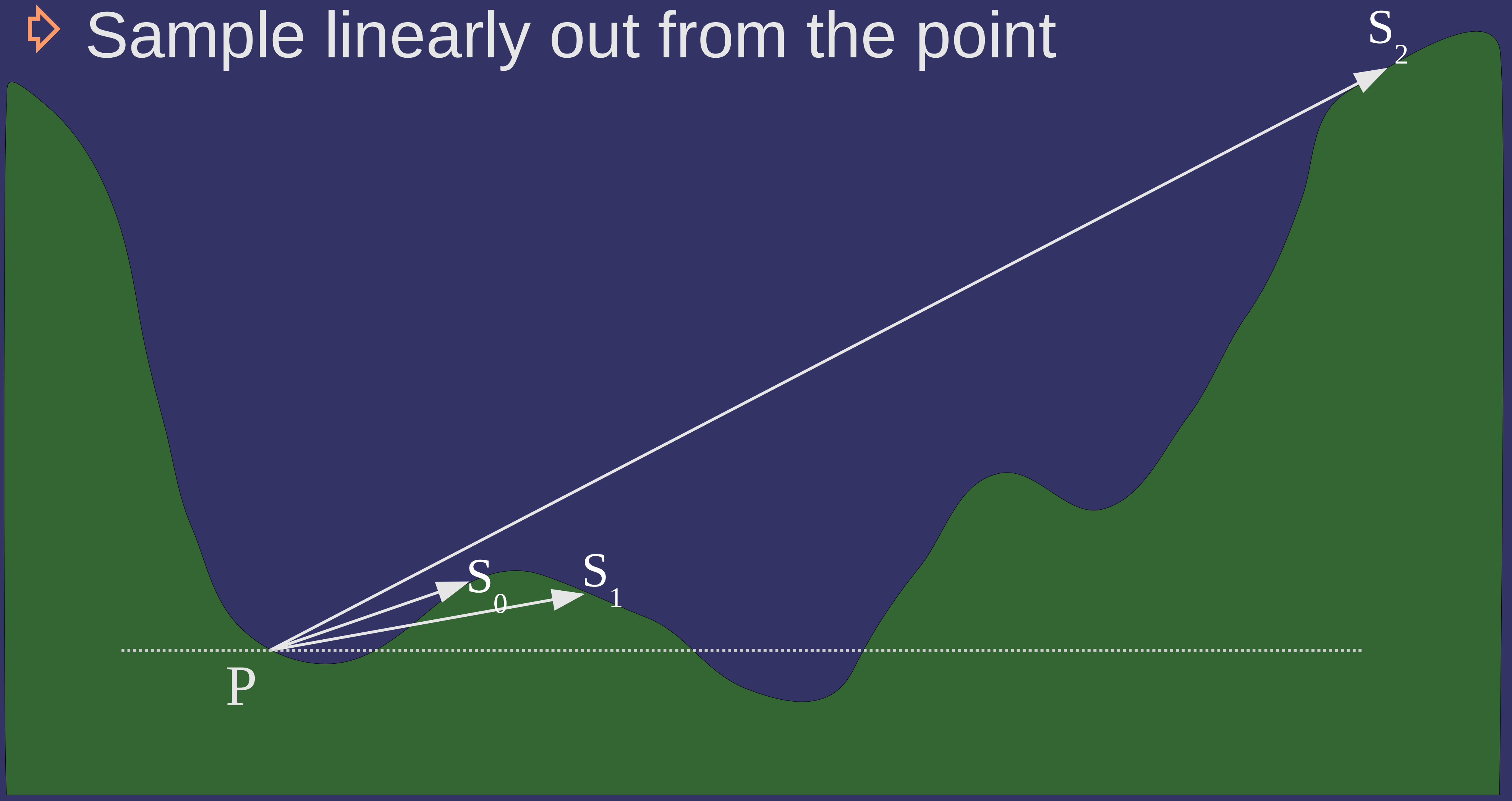
Horizon-Based AO

- Treat the depth buffer as a height field
  - Calculate the horizon angle from each point



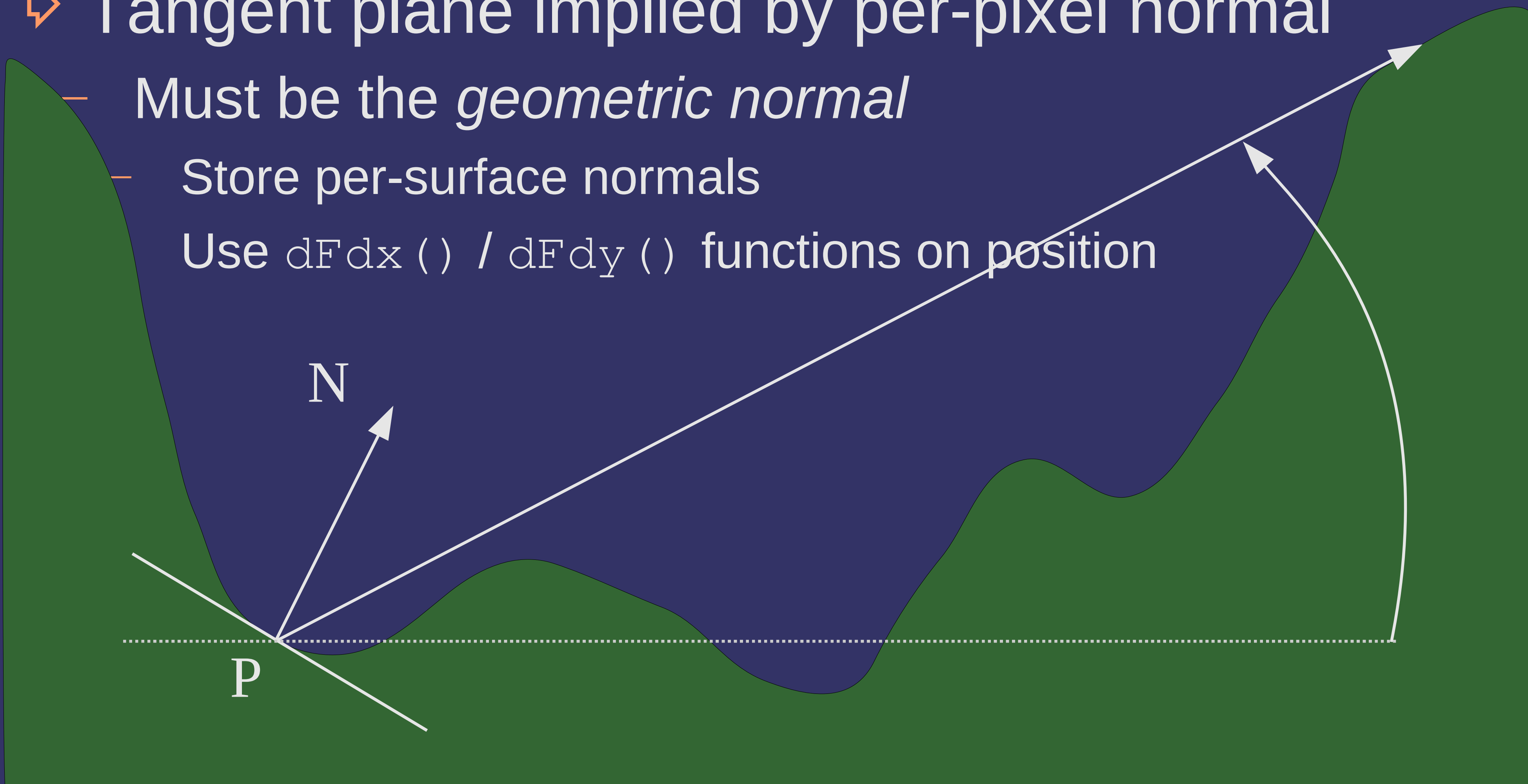
Horizon-Based AO

- Sample linearly out from the point



Horizon-Based AO

- Tangent plane implied by per-pixel normal
  - Must be the *geometric normal*
    - Store per-surface normals
    - Use `dFdx()` / `dFdy()` functions on position



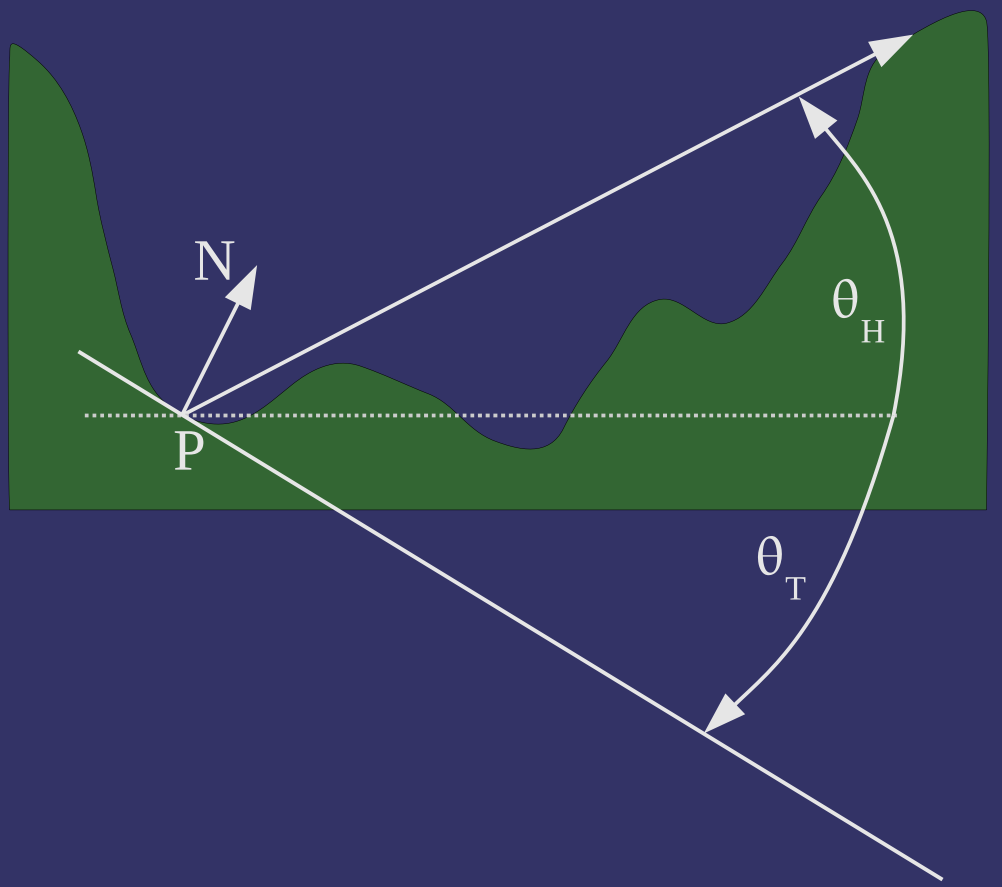
Horizon-Based AO

- Calculate two angles:
  - Horizon angle:

$$\theta_H = atan\left(\frac{H_z}{|H_{xy}|}\right)$$

  - Tangent angle:

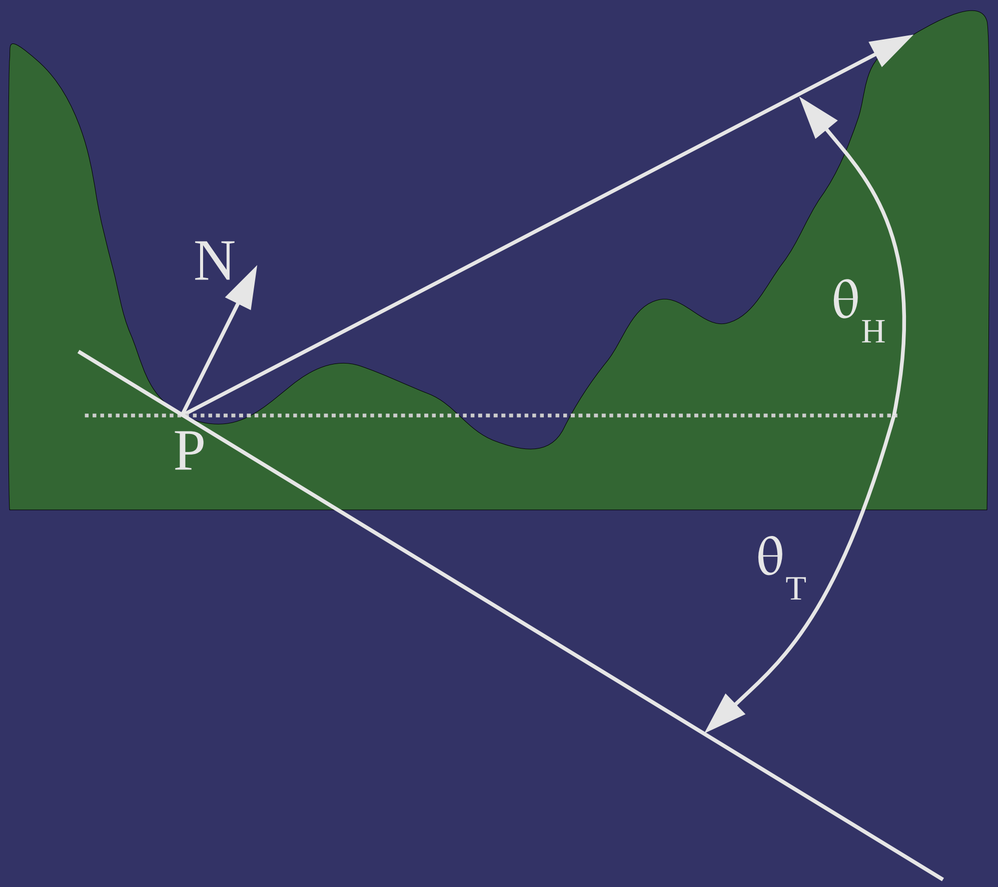
$$\theta_T = atan\left(\frac{T_z}{|T_{xy}|}\right)$$



Horizon-Based AO

- Calculate AO from those angles:

$$AO = \sin\theta_H - \sin\theta_T$$



Horizon-Based AO

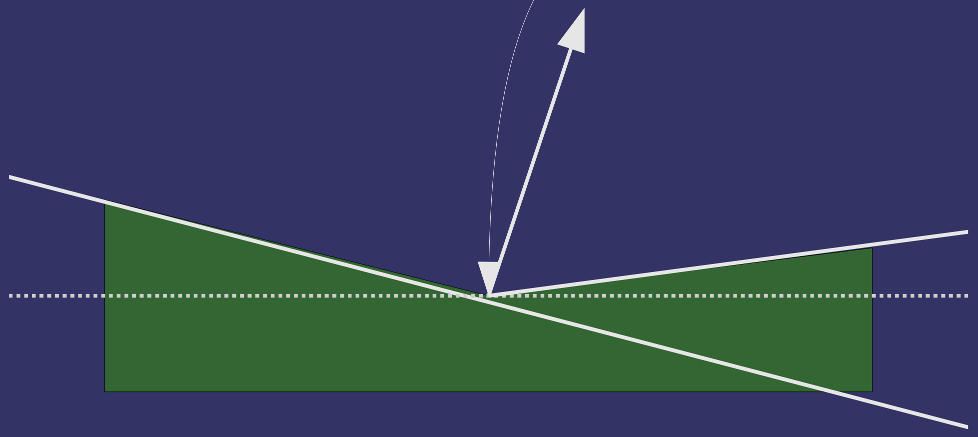
- Sampling is in screen space, but ray tracing is typically done in object space
  - Calculate a sphere in eye space
  - Project that sphere into screen space
  - Use this to set the filter radius

Horizon-Based AO

- Sample from the point in a few uniformly spaced directions
  - Four directions of the compass work well
  - As usual, randomize sampling per-pixel
    - Rotate sampling directions
    - Jitter samples off the true sample direction

Horizon-Based AO

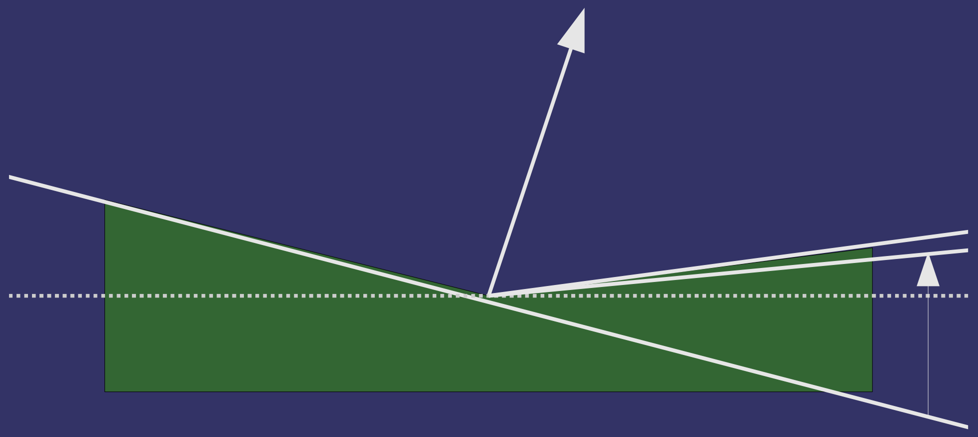
- Can get over-shadowing in curved areas due to too little tessellation



Horizon-Based AO

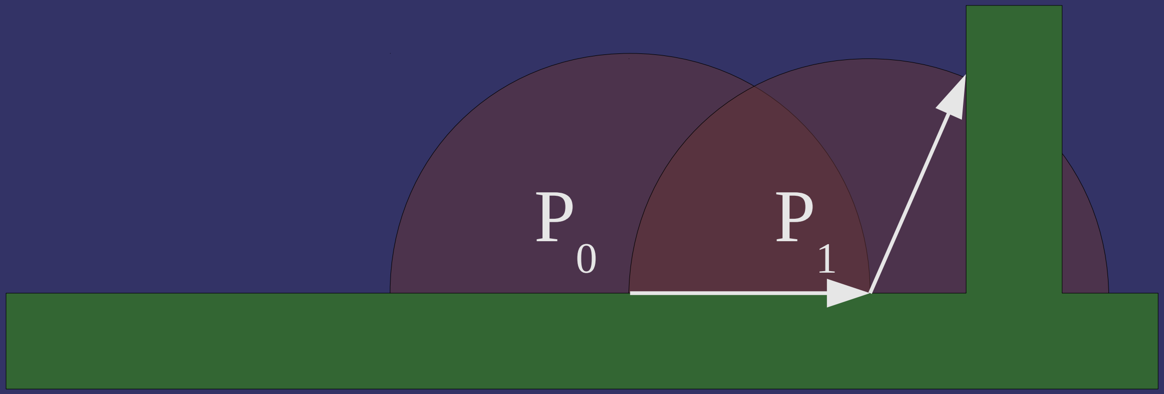
- Can get over-shadowing in curved areas due to too little tessellation
  - Fix this by setting an angle bias on θ_T

$$AO = \sin\theta_H - \sin(\theta_T + \theta_{bias})$$



Horizon-Based AO

- Discontinuities between neighboring pixels
  - Consider P_0 and P_1:
    - P_0 has 0 occlusion
    - P_1 has a very high occlusion



Horizon-Based AO

- Use a per-sample attenuation factor:

$$r = \frac{|S - P|}{R}$$

$$W(r) = 1 - r^2$$

  - P is the position of the point being calculated
  - S is the position of the sample
  - R is the sampling radius

Horizon-Based AO

- Modify update algorithm using per-sample attenuation factor:

```
WAO = 0;   // Weighted ambient occlusion
AO_prev = 0;
horizon_prev = 0;

For all samples:
    If (horizon > horizon_prev)
        AO = sin(horizon) – sin(tangent);
        WAO += W(S)(AO – AO_prev);
        horizon_prev = horizon;
        AO_prev = AO;
```

References

Bavoil, L., Sainz, M., and Dimitrov, R. 2008. Image-space horizon-based ambient occlusion. In *ACM SIGGRAPH 2008 Talks* (Los Angeles, California, August 11 - 15, 2008). SIGGRAPH '08. ACM, New York, NY, 1-1. http://developer.nvidia.com/object/siggraph-2008-HBAO.html

Note: There is a pending patent application for this technique. http://www.faqs.org/patents/app/20090153557

SSAO Problems

- Several problems with SSAO:
  - Lots of pixels to process and filter – slow
  - Missing depth information – over or underocclusion
    - This occurs because we only know the nearest depth value at each X/Y location
  - Missing information at borders – underocclusion at edges

Multi-Layer

- Render enlarged, depth-peeled layers
  - Clamp filter kernel size with a parameter B
  - Render layers of $W \times H$ as $(W+B) \times (H+B)$
  - Enlarge the view frustum to cover this new area

Multi-Layer

- Calculate maximum AO from multiple layers
  - At each sample location determine which layer gives the maximal AO, use just that layer
- How many layers?
  - [Bavoil & Sainz 2009] suggest that 3 is *usually* good enough
  - They note that surfaces at grazing angles to the view rays (e.g., ground planes) can cause problems

SSAO Problems

- Several problems with SSAO:
  - Lots of pixels to process and filter – slow
  - ~~Missing depth information – over or underocclusion~~
    - This occurs because we only know the nearest depth value at each X/Y location
  - ~~Missing information at borders – underocclusion at edges~~

Dual Resolution

- Most AO effects are low frequency
  - Render depth and normals to half resolution buffers
  - Use bilateral filter to upscale
  - Example:
    - For a 1600×1200 display, calculate AO in a (800+B)×(600+B) buffer
    - Upscale AO buffer to 1600x1200, then apply

Dual Resolution

- In some cases, AO effects aren't low frequency
  - What happens with geometry that's less than 2×2?
    - We get both temporal and spatial aliasing effects. Yuck!
- Determine which areas need more resolution
  - Compute AO variance
  - If variance is above a certain threshold, compute at full resolution

Dual Resolution

- Variance computation is fairly expensive
  - Use (max(AO) – min(AO)) as a rough approximation
  - Compute over small kernel in half-resolution buffer
    - 3×3 or 5×5 is probably sufficient
    - If you actually calculate the minimum and maximum (instead of the difference), this is a separable filter

Dual Resolution

- Based on variance, use half-resolution value or recalculate full-resolution value
  - Setting the threshold to 0 causes all values to be re-calculated
  - Setting the threshold to 1 uses all half-resolution values
  - [Bavoil & Sainz 2009] suggest using 0.1

References

Bavoil, L. and Sainz, M. 2009. Multi-Layer Dual-Resolution Screen-Space Ambient Occlusion. In *ACM SIGGRAPH 2009 Talks* (New Orleans, Louisiana, August 3 - 7, 2009). SIGGRAPH '09. ACM, New York, NY, 1-1. http://www.sci.utah.edu/~bavoil/

Next week...

- Quiz #4
- High Dynamic-Range Rendering

Legal Statement

This work represents the view of the authors and does not necessarily represent the view of Intel or the Art Institute of Portland.

OpenGL is a trademark of Silicon Graphics, Inc. in the United States, other countries, or both.

Khronos and OpenGL ES are trademarks of the Khronos Group.

Other company, product, and service names may be trademarks or service marks of others.