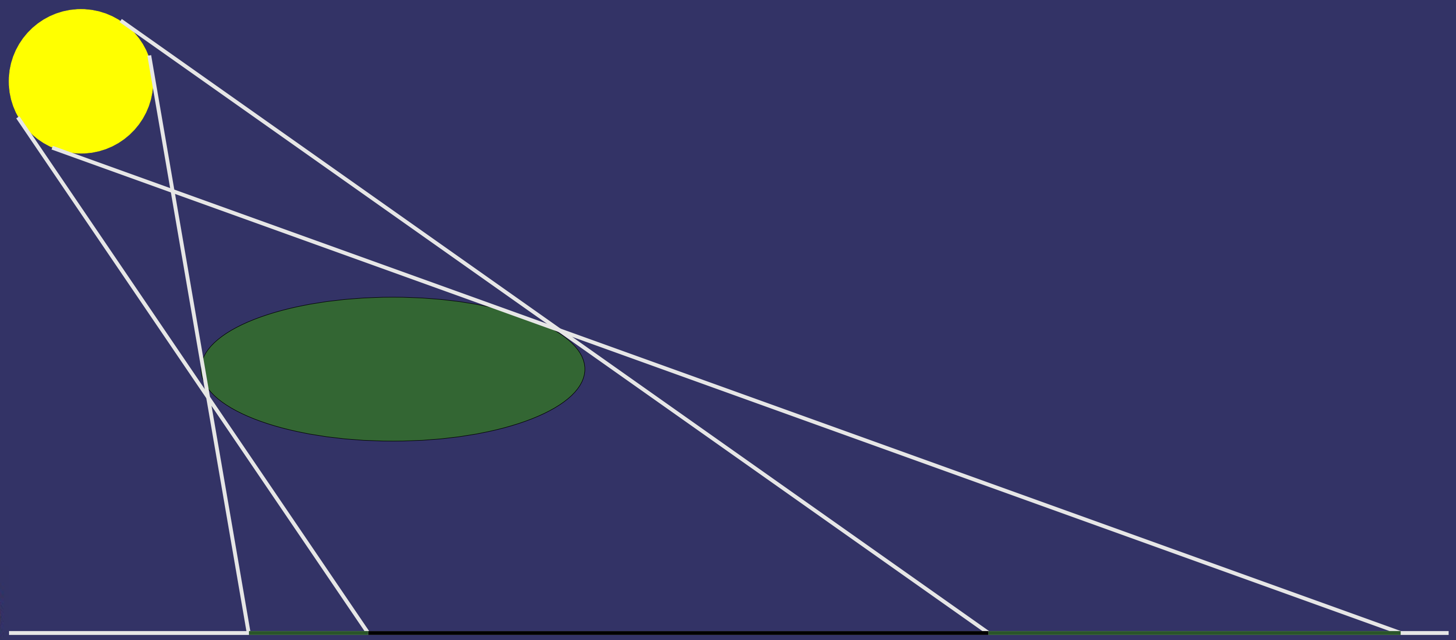
VGP353 – Week 5

- Agenda:
  - Quiz #2
  - Assignment #2 (shadow textures) due.
  - Finish shadow maps:
    - Percentage closer soft shadows (PCSS)
    - Parallel-split shadow maps (PSSMs)
  - Start assignment #3 (shadow maps)

29-April-2008

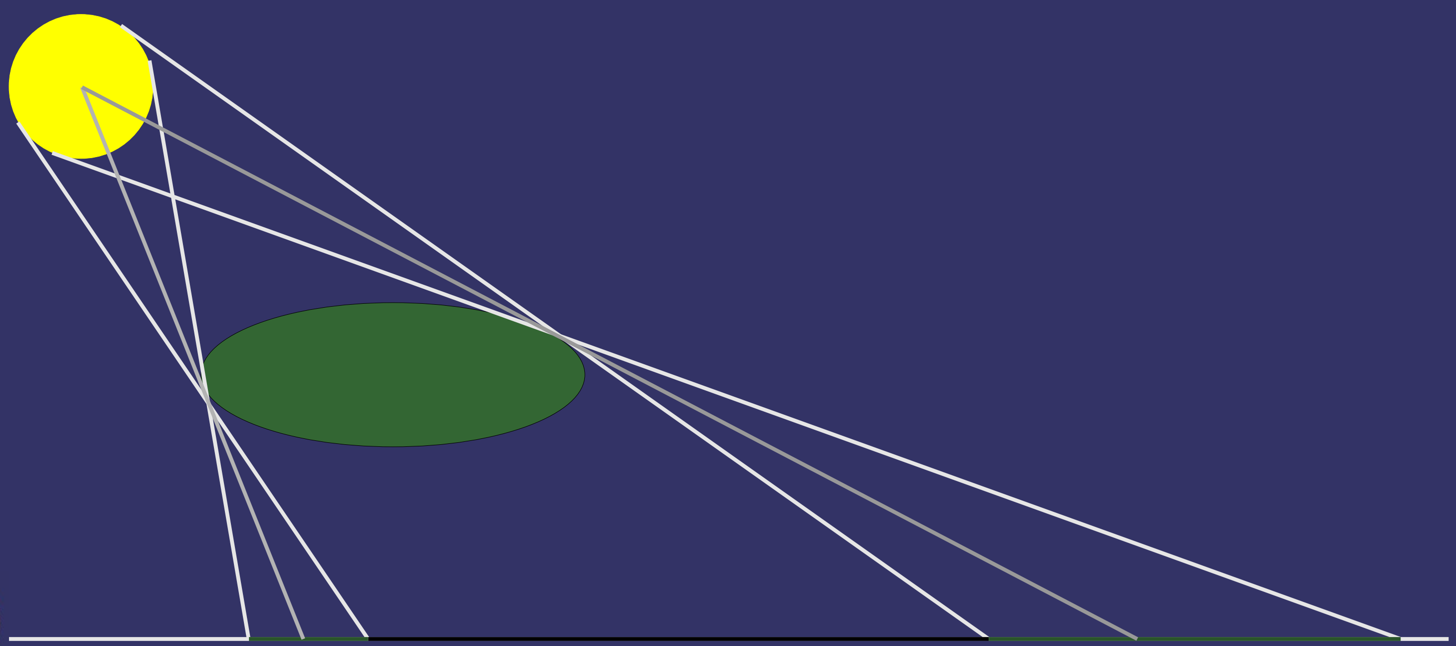
Soft Shadows

- Real lights have area
  - Since the light has area, there are regions where only a portion of the light is occluded...this is the penumbra



Soft Shadows

- Real lights have area
  - Since the light has area, there are regions where only a portion of the light is occluded...this is the penumbra
  - Shadow maps represent part of the penumbra as umbra and part as unoccluded



Soft Shadows

- Size of penumbra region varies with:
  - Size of light
  - Distance between occluder and light
  - Distance between occluder and receiver

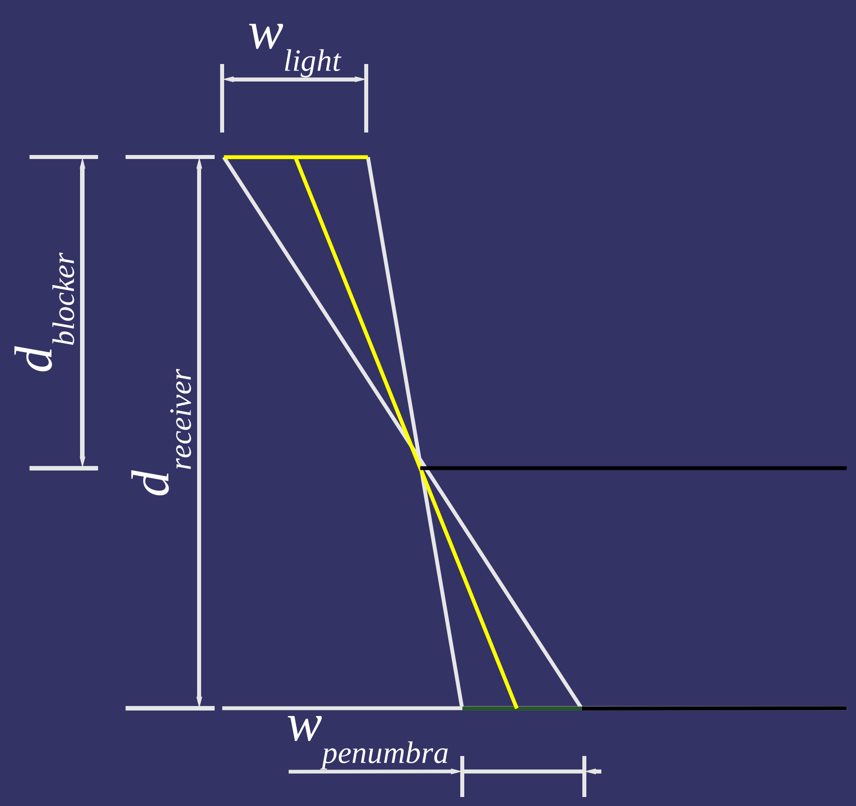
Soft Shadows

- Size of penumbra region varies with:
  - Size of light
  - Distance between occluder and light
  - Distance between occluder and receiver
- Using this information to perform *correct* light visibility calculations is hard
  - Make some simplifying assumptions!
  - Assume that all occluders, receivers, and lights are both flat and parallel to each other

Soft Shadows

- Estimate penumbra size using:

$$w_{penumbra} = \frac{(d_{receiver} - d_{blocker}) \times w_{light}}{d_{blocker}}$$

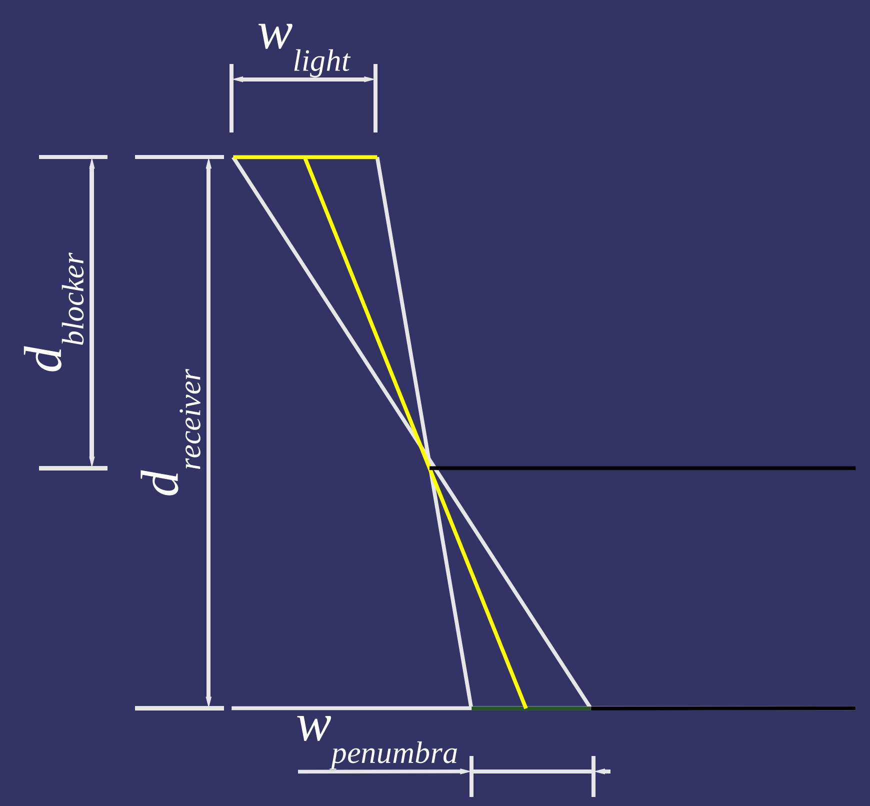


Soft Shadows

- Estimate penumbra size using:

$$w_{penumbra} = \frac{(d_{receiver} - d_{blocker}) \times w_{light}}{d_{blocker}}$$

- How do we determine $d_{blocker}$?

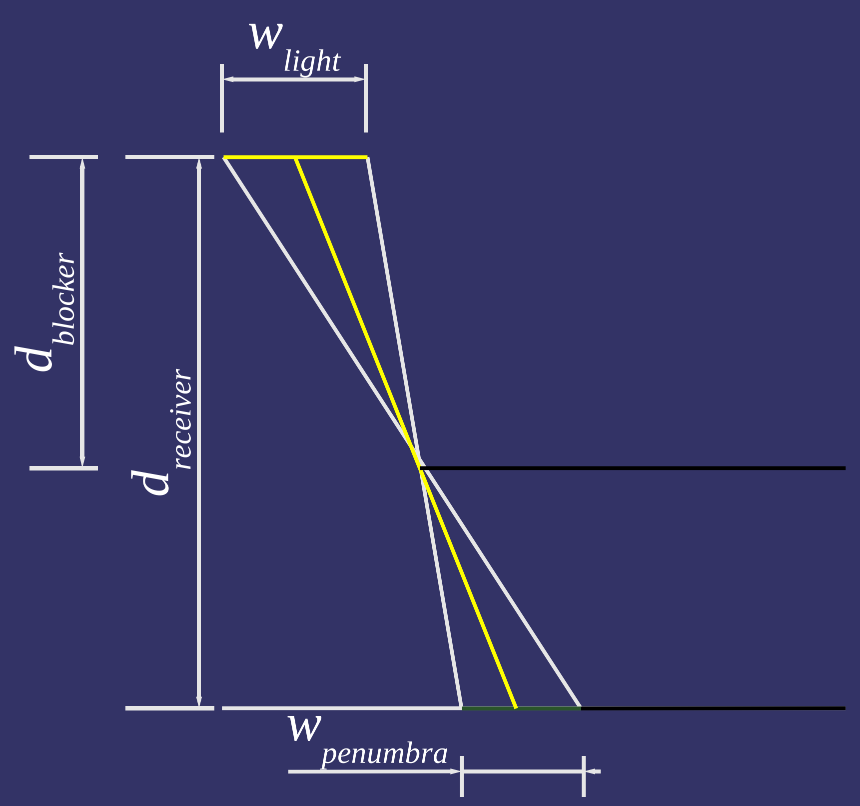


Soft Shadows

- Estimate penumbra size using:

$$w_{penumbra} = \frac{(d_{receiver} - d_{blocker}) \times w_{light}}{d_{blocker}}$$

- How do we determine $d_{blocker}$?
  - Search the shadow map for possible occluders



29-April-2008

Soft Shadows

- Examine a region around the point in the shadow map
  - Select region size based on light size and rendering budget
  - Sample values and *average* all depths less than the current fragment
    - Very similar to percentage closer filter (PCF)
- Use resulting average as $d_{blocker}$
  - $w_{penumbra}$ is the width of the PCF filter area

Soft Shadows

- Demo

Original image from
http://developer.nvidia.com/object/gdc_2005_presentations.html

29-April-2008

References

Randima Fernando. *Percentage-Closer Soft Shadows.* 2005. Game Developer's Conference. http://developer.download.nvidia.com/shaderlibrary/docs/shadow_PCSS.pdf

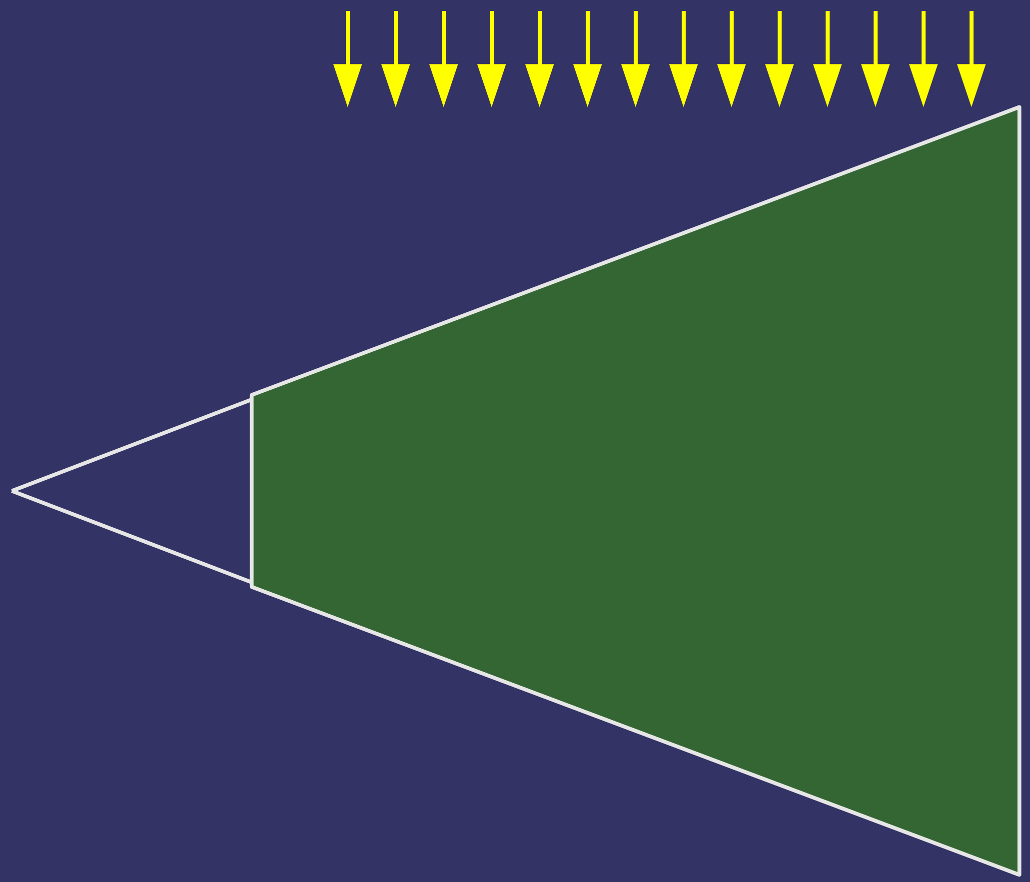
Break

Perspective Shadow Maps

- Some significant problems:
  - Shadow map *quality* is view-dependent
  - Several special cases that must be handled depending on light direction / position
  - Difficulties handling shadow casters behind the camera
- Introduced some good ideas:
  - Re-parameterizing the scene based on the camera / light frusta
  - Quantitatively determining when aliasing will occur

Parallel-Split Shadow Maps

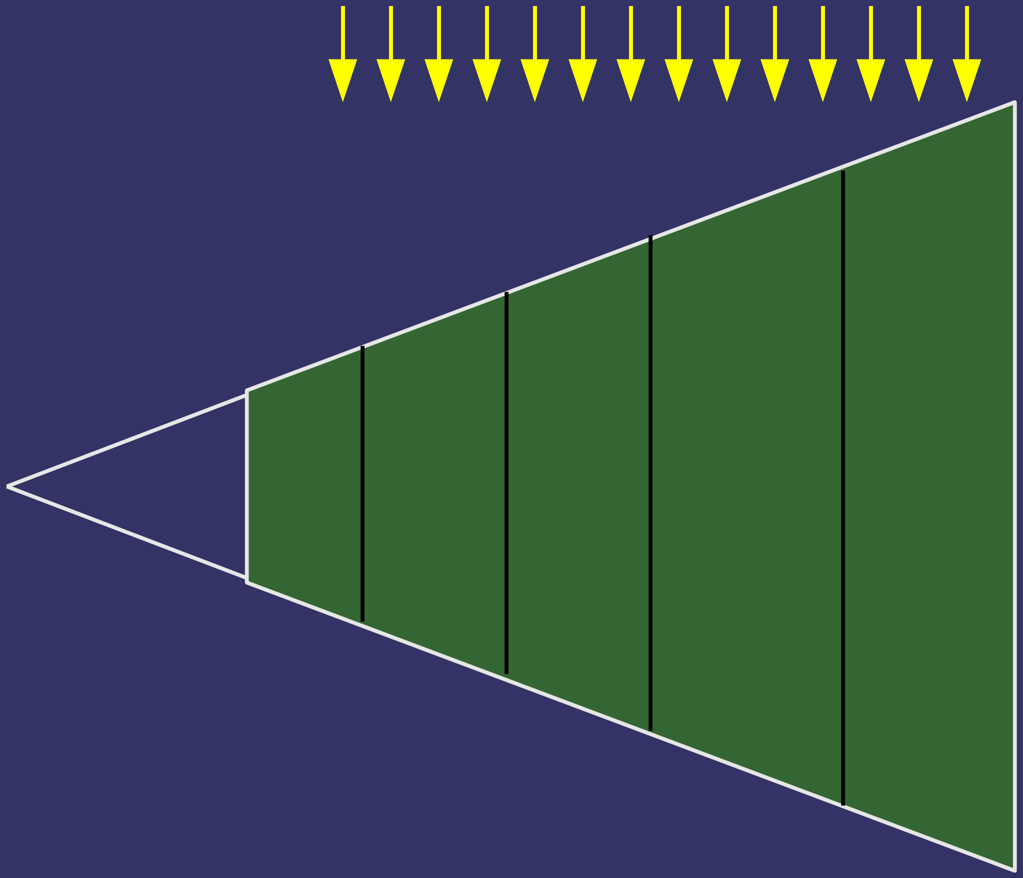
- PSSMs solve most of these problems



29-April-2008

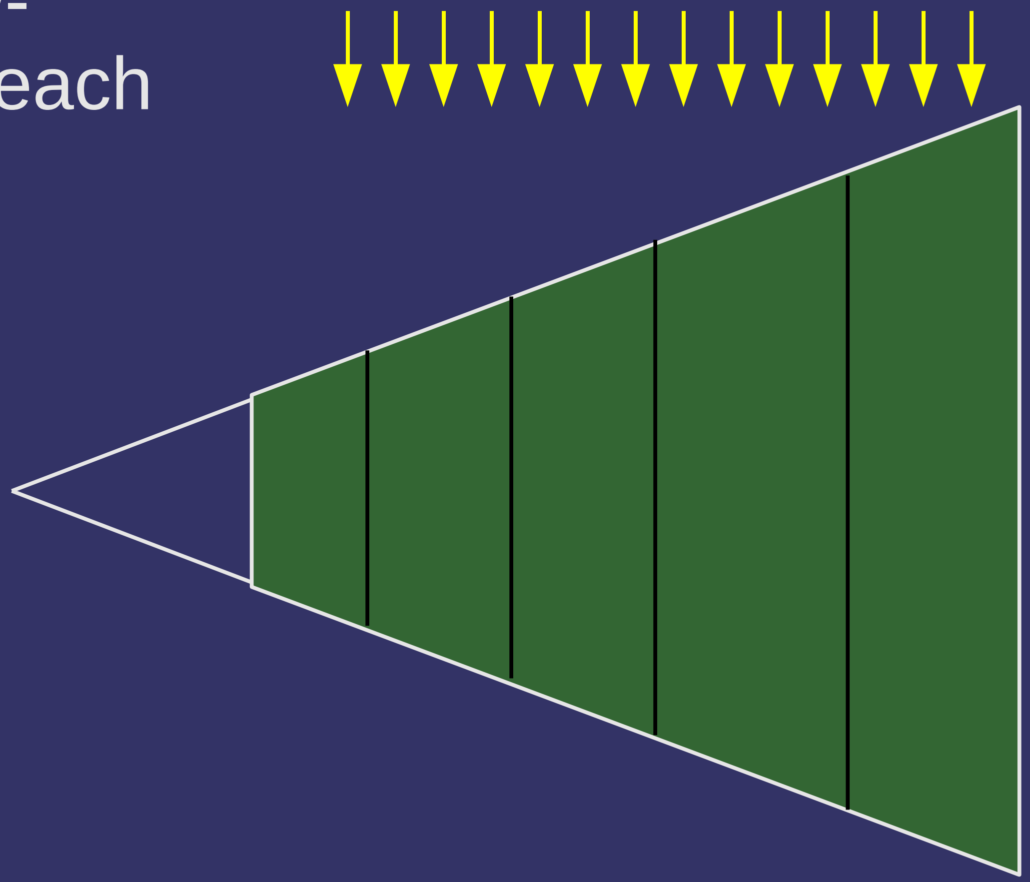
Parallel-Split Shadow Maps

- PSSMs solve most of these problems
  - Split view frustum into n parts with planes parallel to the near / far plane



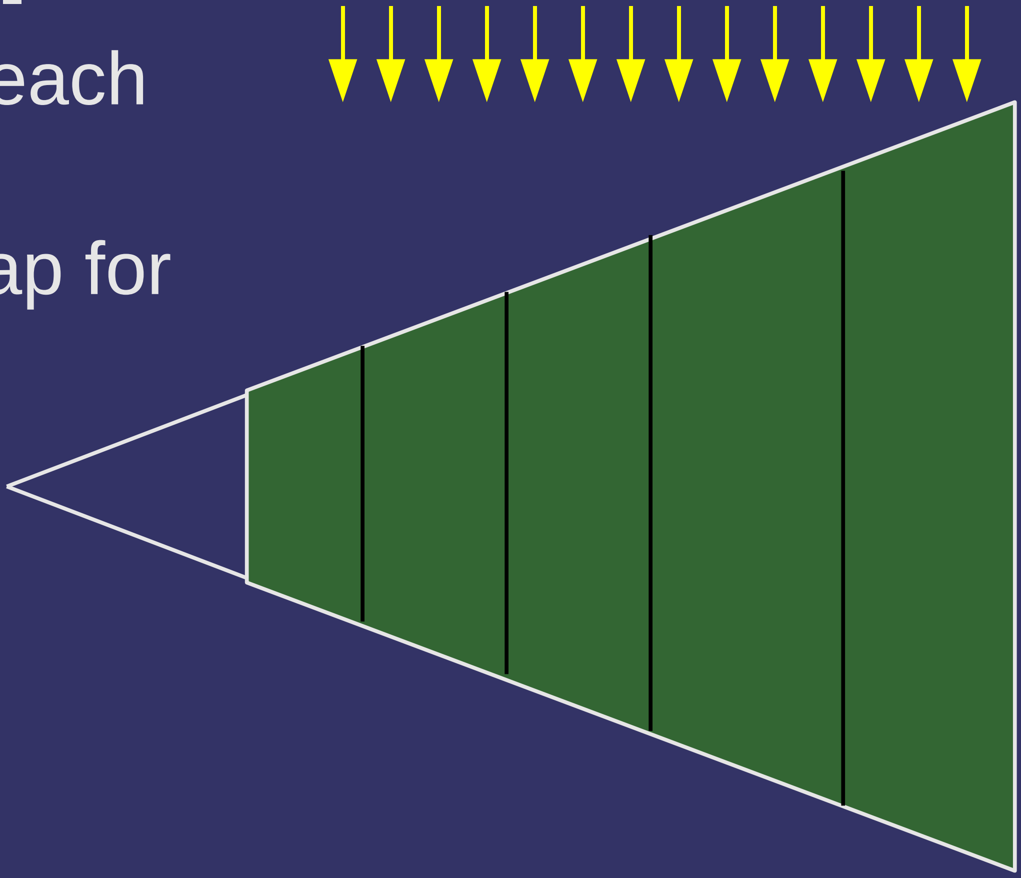
Parallel-Split Shadow Maps

- PSSMs solve most of these problems
  - Split view frustum into n parts with planes parallel to the near / far plane
  - Calculate light's view-projection matrix for each split region



Parallel-Split Shadow Maps

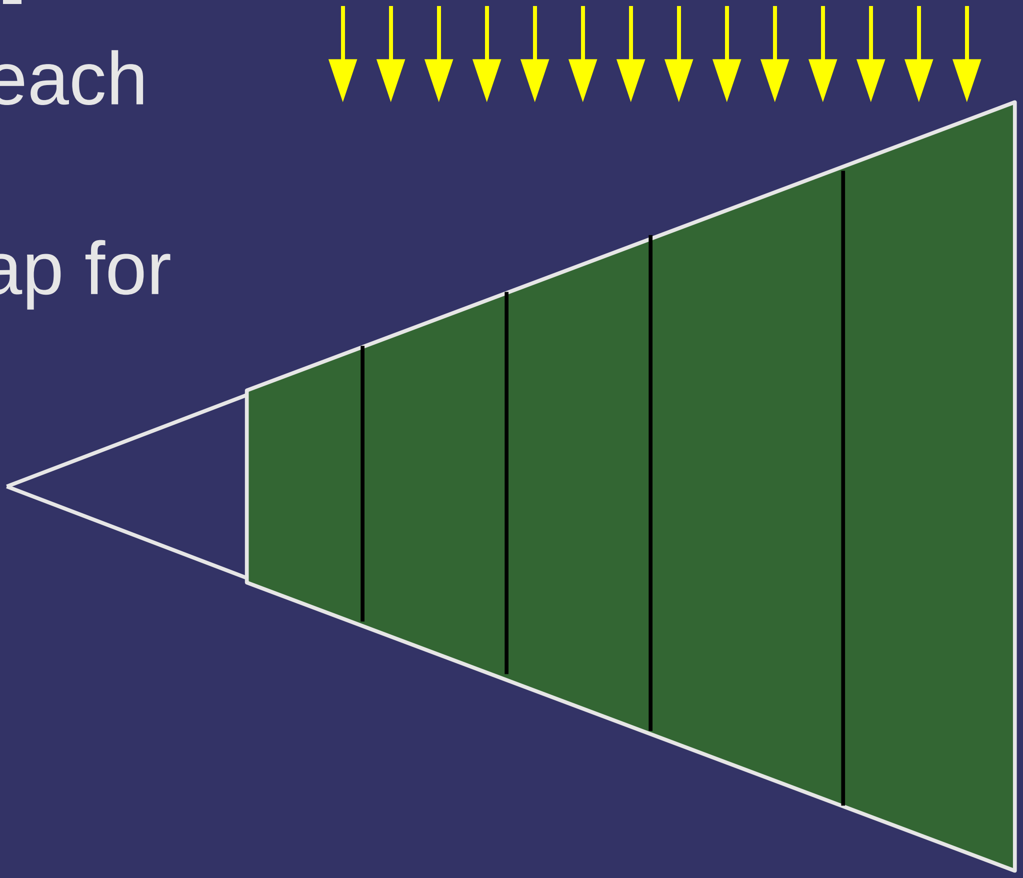
- PSSMs solve most of these problems
  - Split view frustum into n parts with planes parallel to the near / far plane
  - Calculate light's view-projection matrix for each split region
  - Generate shadow map for each split regions



29-April-2008

Parallel-Split Shadow Maps

- PSSMs solve most of these problems
  - Split view frustum into m parts with planes parallel to the near / far plane
  - Calculate light's view-projection matrix for each split region
  - Generate shadow map for each split regions
  - Apply shadow maps to scene



29-April-2008

Parallel-Split Shadow Maps

- Aliasing occurs when $d > d_i$

$$d = d_s \frac{r_s}{r_i} \frac{N \cdot V}{N \cdot L}$$

  – Rename r_i as z, and call dz the change in z relative to one unit in ds

$$d = \frac{dz}{z\,ds} \frac{N \cdot V}{N \cdot L}$$

  – Ignoring perspective aliasing, this means that we want dz / zds to be constant over the entire view

    – Call this constant ρ

Parallel-Split Shadow Maps

- Optimal shadow map distribution is:

$$\frac{ds}{z\,dz} = \rho \Rightarrow \mathrm{s}(z) = \int_0^s ds = \frac{1}{\rho}\int_n^z \frac{1}{z}\,dz = \frac{1}{\rho}\ln\left(\frac{z}{n}\right)$$

  - Since $\mathrm{s}(f) = 1$, $\rho = \ln(\mathrm{f} / \mathrm{n})$

Parallel-Split Shadow Maps

- Current hardware can't do this non-linear z transform
  - Discretely perform the mapping in steps at the split planes

$$s_i = s(C_i^{\log}) = \frac{1}{\ln(f/n)} \ln\left(\frac{C_i^{\log}}{n}\right)$$

  - Each split gets 1 / m of total texture resolution, substituting i / m for s_i

$$C_i^{\log} = n\left(\frac{f}{n}\right)^{i/m}$$

Parallel-Split Shadow Maps

- Alternately, the view frustum could be divided into equally sized pieces

$$C_i^{uni} = \frac{(f - n) \times i}{m} + n$$

Parallel-Split Shadow Maps

- Neither split strategy work very well
  - Logarithmic splitting groups split-planes too close to the near plane
  - Uniform splitting doesn't group split-planes close enough to the near plane

Parallel-Split Shadow Maps

- Neither split strategy work very well
  - Logarithmic splitting groups split-planes too close to the near plane
  - Uniform splitting doesn't group split-planes close enough to the near plane
- Instead, use a hybrid of the two

$$C_i = \lambda C_i^{\log} + (1 - \lambda) C_i^{uni}$$

  - λ is tunable parameter
  - The paper calls this the *practical split scheme*

Parallel-Split Shadow Maps

- Light transformation matrices are determined much like before
  - Calculate view-projection matrix for light relative to whole view frustum
  - Transform each split region to light's post-projection space
  - Calculate AABB for transformed split region
  - Use AABB to calculate "crop" transformation to scale and center split region to full view

Parallel-Split Shadow Maps

- To apply shadows, the shader must determine which region contains the current fragment
  - Determine the split-plane, C_s, nearest the camera but farther away than the current fragment
  - C_s determines which shadow map to apply
  - The light transforms, C_i distances, and shadow maps (samplers) must be provided to the shader as arrays of uniforms
    - m is a compile-time constant

Parallel-Split Shadow Maps

- Only directional lights have been dealt with so far

Parallel-Split Shadow Maps

- Only directional lights have been dealt with so far
  - Light transformations for each split region are calculated from the light's post-projection space

Parallel-Split Shadow Maps

- Only directional lights have been dealt with so far
  - Light transformations for each split region are calculated from the light's post-projection space
  - For point lights, transform by the light's view-projection matrix *first*
    - This effectively converts the point-light to a directional light!

References

Zhang, F., Sun, H., Nyman, O. “Parallel-Split Shadow Maps on Programmable GPUs,” in *GPU Gems 3,* ed. Hubert Nguyen, pp. 202 – 237. Boston, MA: Addison-Wesley, 2008. http://appsrv.cse.cuhk.edu.hk/~fzhang/pssm_project/

Wimmer, M., Scherzer, D., and Purgathofer, W. “Light Space Perspective Shadow Maps,” in *Proceedings of Eurographics Symposium on Rendering*, pp. 143 - 151. Norrköping, Sweden: Eurographics Association, 2004. http://www.cg.tuwien.ac.at/research/vr/lispsm/

Next week...

- Shadow volumes

Legal Statement

This work represents the view of the authors and does not necessarily represent the view of IBM 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.