Stenciled Shadow Volume Optimizations (1)

- Use GL_QUAD_STRIP rather than GL_QUADS to render extruded shadow volume quads
  - Requires determining possible silhouette loop connectivity
- Mix *Zfail* and *Zpass* techniques
  - Pick a single formulation for each shadow volume
  - *Zpass* is more efficient since shadow volume does not need to be closed
  - Mixing has no effect on net shadow depth count
  - *Zfail* can be used in the hard cases

NVIDIA.

Stenciled Shadow Volume Optimizations (2)

- Pre-compute or re-use cache shadow volume geometry when geometric relationship between a light and occluder does not change between frames
  - Example: Headlights on a car have a static shadow volume w.r.t. the car itself as an occluder
- Advanced shadow volume culling approaches
  - Uses portals, Binary Space Partitioning trees, occlusion detection, and view frustum culling techniques to limit shadow volumes
  - Careful to make sure such optimizations are truly correct

NVIDIA.

Stenciled Shadow Volume Optimizations (3)

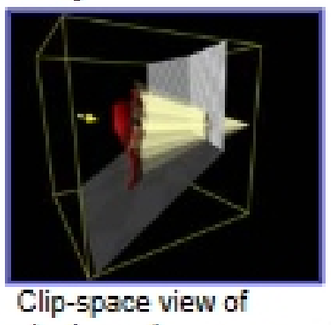
- Take advantage of ad-hoc knowledge of scenes whenever possible
  - Example: A totally closed room means you do not have to cast shadow volumes for the wall, floor, ceiling
- Limit shadows to important entities
  - Example: Generate shadow volumes for monsters and characters, but not static objects
  - Characters can still cast shadows on static objects
- Mix shadow techniques where possible
  - Use planar projected shadows or light-maps where appropriate

NVIDIA.

Stenciled Shadow Volume Optimizations (4)

- Shadow volume's extrusion for directional lights can be rendered with a GL_TRIANGLE_FAN
  - Directional light's shadow volume always projects to a single point at infinity

Scene with directional light.



Clip-space view of shadow volume

NVIDIA.

Hardware Enhancements: Depth Clamp (1)

- What is depth clamping?
  - Boolean hardware enable/disable
  - When enabled, disables the near & far clip planes
  - Interpolate the window-space depth value
  - Clamps the interpolated depth value to the depth range, i.e. [min(*n*,*f*),max(*n*,*f*)]
    - Assuming glDepthRange(*n*,*f*);
  - Geometry "behind" the far clip plane is still rendered
    - Depth value clamped to farthest Z
    - Similar for near clip plane, as long as w>0, except clamped to closest Z

NVIDIA.

Hardware Enhancements: Depth Clamp (2)

- Advantage for stenciled shadow volumes
  - With depth clamp, P (rather than Pinf) can be used with our robust stenciled shadow volume technique
  - Marginal loss of depth precision re-gained
  - Orthographic projections can work with our technique with depth clamping
- NV_depth_clamp OpenGL extension
  - Easy to use
    glEnable(GL_DEPTH_CLAMP_NV);
    glDisable(GL_DEPTH_CLAMP_NV);
  - GeForce3 & GeForce4 Ti support (soon)

NVIDIA.

Hardware Enhancements: Two-sided Stencil Testing (1)

- Current stenciled shadow volumes required rendering shadow volume geometry twice
  - First, rasterizing front-facing geometry
  - Second, rasterizing back-facing geometry
- Two-sided stencil testing requires only one pass
  - Two sets of stencil state: front- and back-facing
  - Boolean enable for two-sided stencil testing
  - When enabled, back-facing stencil state is used for stencil testing back-facing polygons
  - Otherwise, front-facing stencil state is used
  - Rasterizes just as many fragments, but more efficient for CPU & GPU

NVIDIA.

Hardware Enhancements: Two-sided Stencil Testing (2)

NV_stencil_two_side OpenGL extension

- Enable applies if GL_STENCIL_TEST also enabled
  glEnable(GL_STENCIL_TEST_TWO_SIDE_NV);
  glDisable(GL_STENCIL_TEST_TWO_SIDE_NV);
- Control of front- and back-facing stencil state update
  glActiveStencilFaceNV(GL_FRONT);
  glActiveStencilFaceNV(GL_BACK);
- Existing stencil routines (glStencilOp, glStencilMask, glStencilFunc) update the active stencil face state
- glClear and non-polygon primitives always use the front-facing stencil state

- Expect on future GPUs

NVIDIA.

Usage of NV_stencil_two_side & EXT_stencil_wrap

OLD SCHOOL

```
glDepthMask(0);
glColorMask(0,0,0,0);
glEnable(GL_CULL_FACE);
glEnable(GL_STENCIL_TEST);
glStencilMask(~0);
glStencilFunc(GL_ALWAYS, 0, ~0);
// Increment for back faces
glCullFace(GL_BACK);
glStencilOp(GL_KEEP,   // stencil test fail
            GL_INCR,   // depth test fail
            GL_INCR);  // depth test pass
renderShadowVolumePolygons();
// Decrement for front faces
glCullFace(GL_FRONT);
glStencilOp(GL_KEEP,   // stencil test fail
            GL_DECR,   // depth test fail
            GL_KEEP);  // depth test pass
renderShadowVolumePolygons();
```

NEW SCHOOL

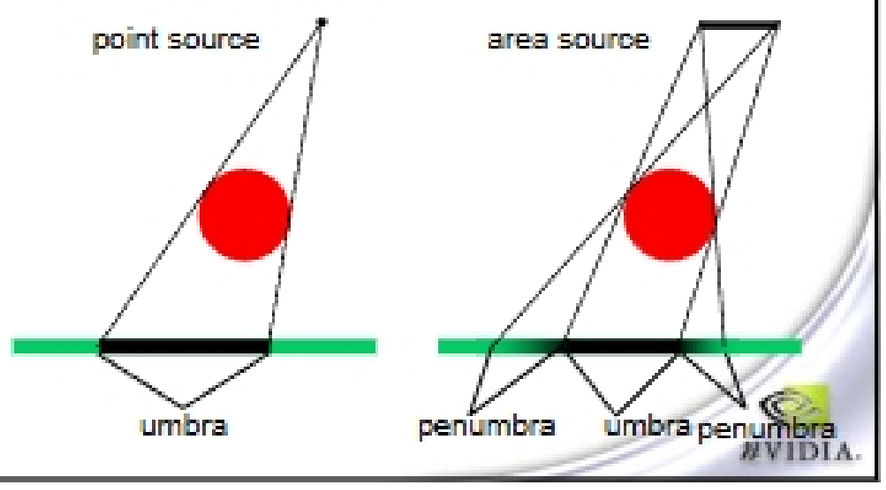
```
glDepthMask(0);
glColorMask(0,0,0,0);
glDisable(GL_CULL_FACE);
glEnable(GL_STENCIL_TEST);
glEnable(GL_STENCIL_TEST_TWO_SIDE_NV);
glActiveStencilFaceNV(GL_BACK);
glStencilOp(GL_KEEP,          // stencil test fail
            GL_INCR_WRAP_EXT, // depth test fail
            GL_KEEP);         // depth test pass
glStencilMask(~0);
glStencilFunc(GL_ALWAYS, 0, ~0);
glActiveStencilFaceNV(GL_FRONT);
glStencilOp(GL_KEEP,   // stencil test fail
            GL_DECR_WRAP_EXT, // depth test fail
            GL_KEEP);  // depth test pass
glStencilMask(~0);
glStencilFunc(GL_ALWAYS, 0, ~0);
renderShadowVolumePolygons();
```

New approach calls renderShadowVolumePolygons() just once.

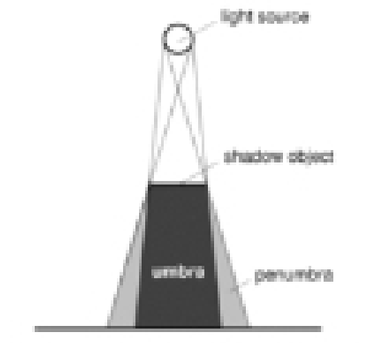
NVIDIA.

Algorithms for Soft Shadows

- Hard vs. Soft Shadow



NVIDIA.

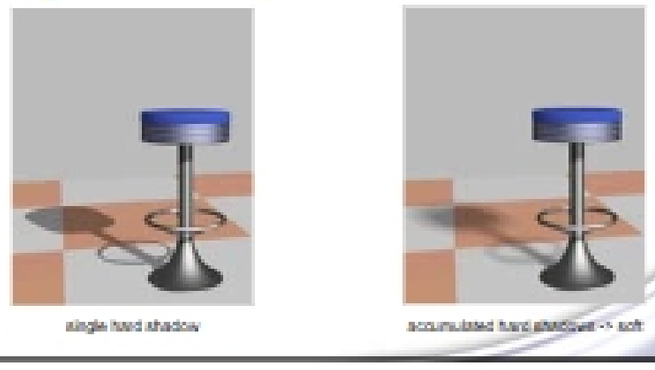


© Kavita Bala, Computer Science, Cornell University

NVIDIA.

Sampling the Light Source

- Use arbitrary hard shadow algorithm
  - Select point sample on area light source
  - Render hard shadows
  - Sum up weighted result (e.g. accumulation buffer)



single hard shadow

accumulated hard shadows -> soft

NVIDIA.

Sampling the Light Source

- Example: Ground plane shadow texture

```
1. Initialize FB (white)
2. For each sample point do
   2a. Render scene
   2b. Subtract 1/N from FB
       only once for each
       pixel (stencil) !
```

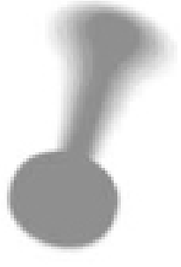


Image from ATI Developer's Site

Sampling the Light Source

- Result

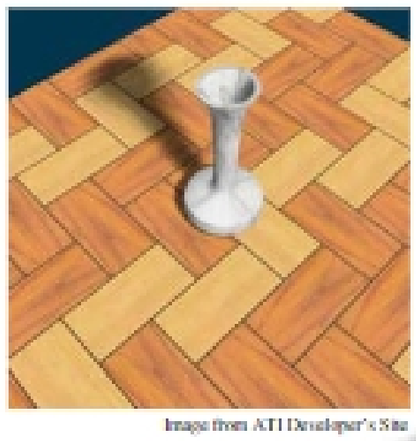


Image from ATI Developer's Site

Sampling the Light Source

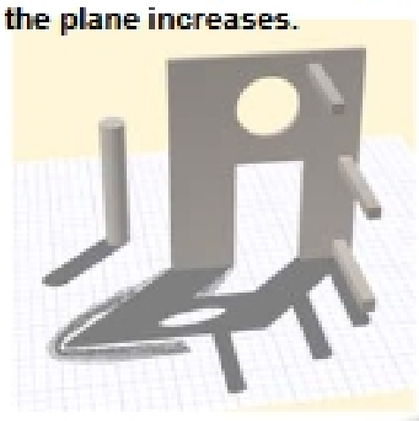
- Number of samples == number of passes
  - Problematic for complex scenes
- N samples == (N+1) levels of shadow
  - Fully lit, fully shadowed
  - (N-1) levels of penumbra
- How much samples ?

Plateau Soft Drop Shadows

- Say we want to paint a soft shadow for an image.

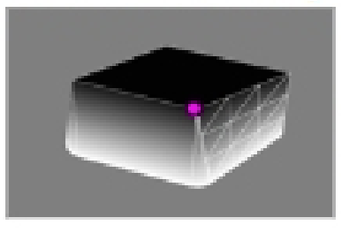
Plateau Idea

- "Paint" each shadow's edge, blurring it as its height from the plane increases.

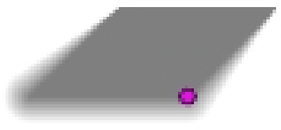


Forming Plateaus

Create the shadow object



Render from above



Apply rendering as texture