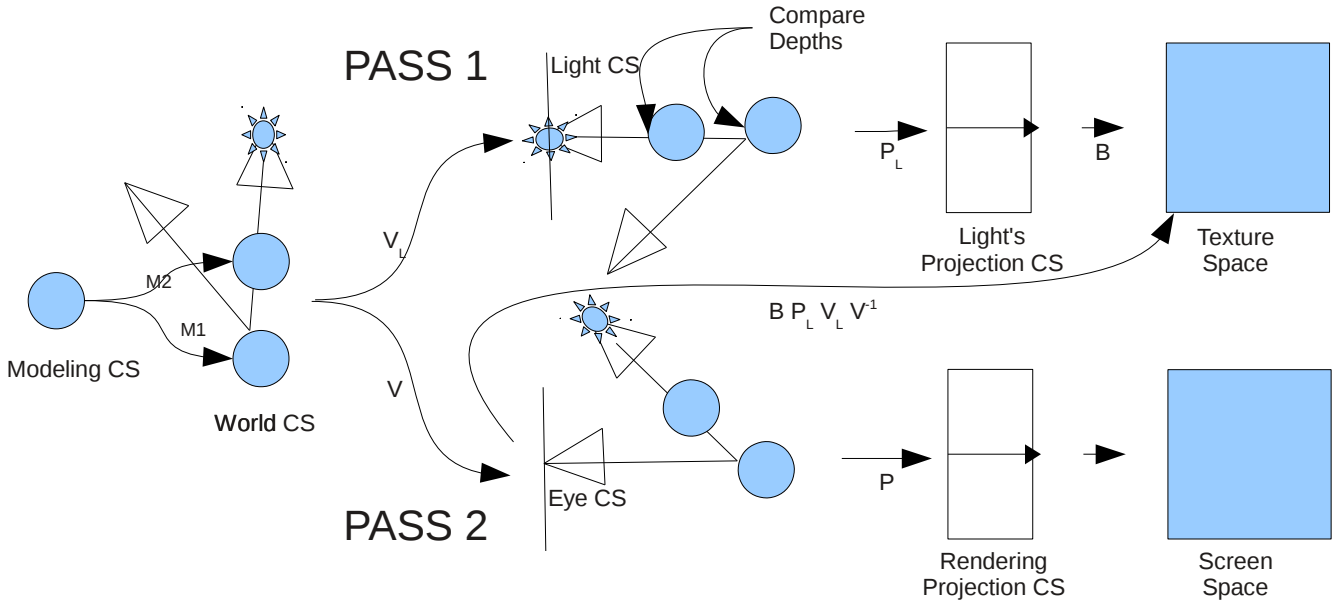
Shadow Maps



Shadow maps

Pass 1: Render from light's POV, to a Frame Buffer Object (FBO)

Viewing: V_L = gluLookAt from light's POV

Projection: P_L = glFrustum from light's POV

Vertex processor:

Usual gl_Position = ftransform() for the scan conversion

varying **position** = gl_Position;

Pixel processor:

Write **depth** (that is position.w) to gl_FragColor or gl_FragData[0]

(alternately: write full position and extract itsw comp in pass 2)

Pass 2: Render from eye's POV to screen:

Viewing: V = gluLookAt from eye's POV

Projection: P = glFrustum from eye's POV

ShadowTr: $B\ P_L\ V_L\ V^{-1}$ (Used to transform position in eye coordinates to shadow-texture coordinates)

Vertex processor:

Usual ftransform() for the scan conversion

Varying **position** = gl_ModelViewMatrix * gl_Vertex

Fragment shader:

shadowCoord = ShadowTr***position**

(For efficiency sake, consider moving this multiplication to the vertex shader.)

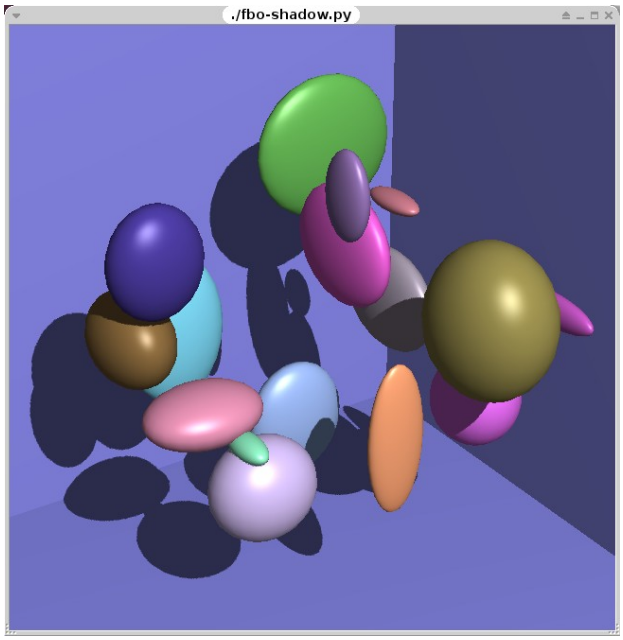
Light-depth = w coordinate from indexing shadow-texture at shadowCoord.xy/shadowCoord.w

Pixel-depth = shadowCoord.w

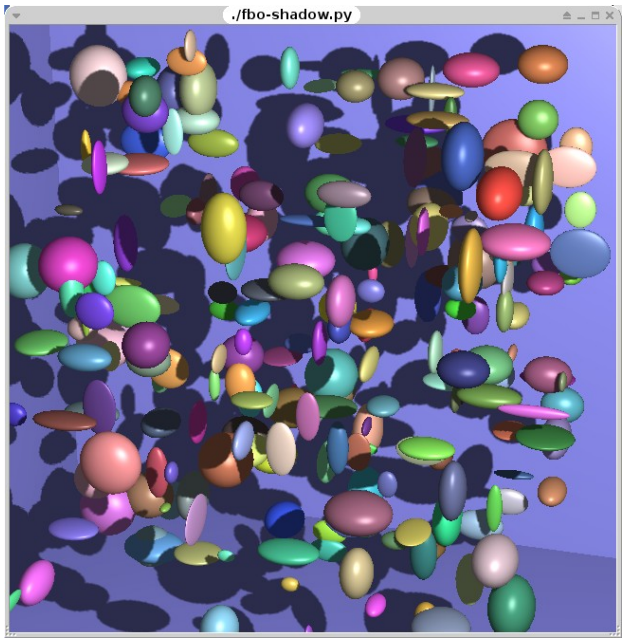
Pixel is in shadow if: Pixel-depth > Light-depth

Color pixel accordingly: ambient only (if in shadow), ambient+diffuse+specular (if not in shadow).

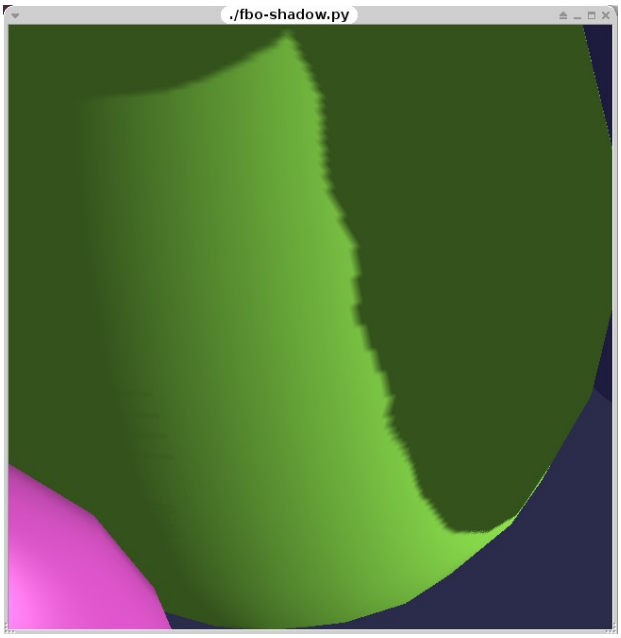
Examples



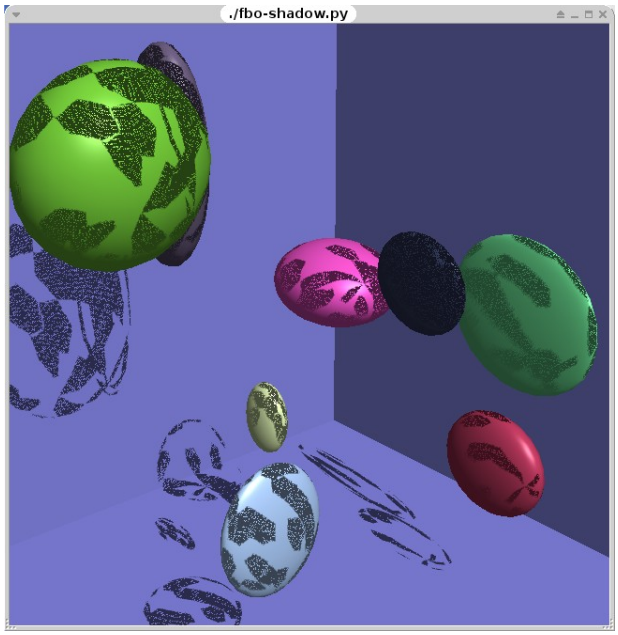
Example



Many shadows cast on many objects



Closeup showing pixelation of shadow map



Error due to depth precision problems

Problems and solutions

- Depth precision problems:
  - The shadow test compare two float for equality. This must be done carefully to allow for round-off errors.
- Depth fighting occurs at the front surface of any object
  - Front face culling, so shadow starts at back of surface
  - Fighting at back of surface is OK since

    unlit due to geometry == unlit due to shadow