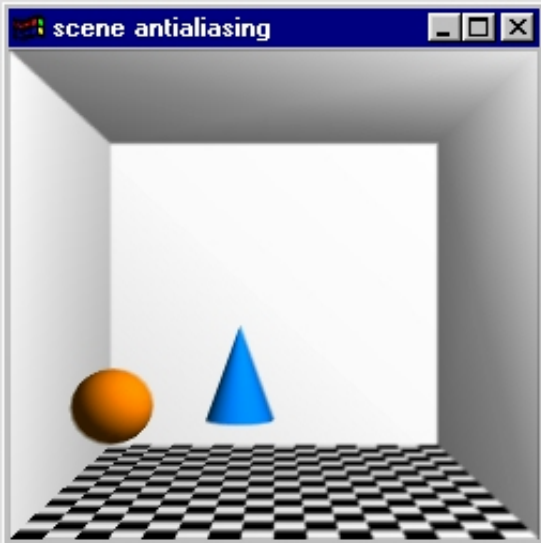
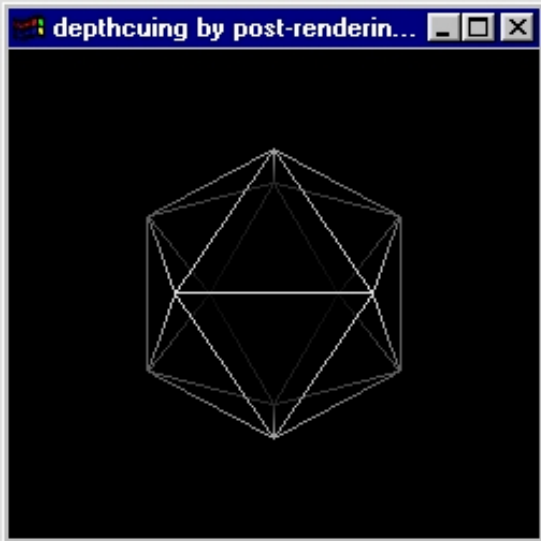


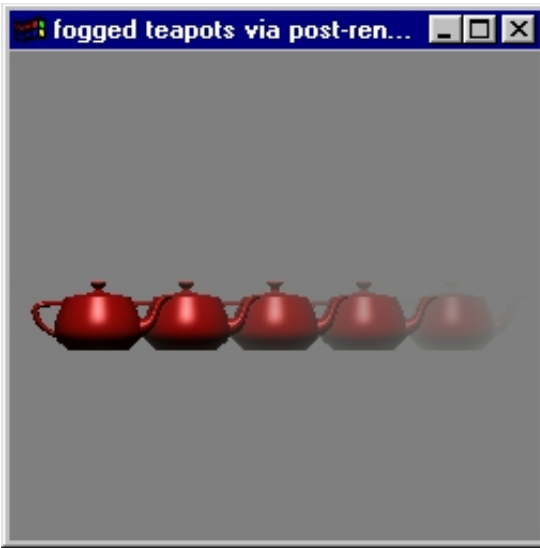
advanced.zip



 A screenshot of a window titled "scene antialiasing". It shows a 3D scene with a white background, a black and white checkered floor, an orange sphere, and a blue cone.	<p>Using the accumulation buffers to anti-alias a scene.</p> <p>Source code: accumaa.c.</p> <p>Snapshots: anti-aliased (shown), aliased.</p>
---	--

 A screenshot of a window titled "depthcuing by post-renderin...". It shows a 3D wireframe model of a dodecahedron on a black background.	<p>Adding depthcue (fog) in a second rendering pass.</p> <p>Source code: af_depthcue.c.</p> <p>Snapshots: fog added by second pass (shown), fog rendered normally.</p>
--	--





More detailed example of adding fog in a second rendering pass.

Source code: [af_teapots.c](#).

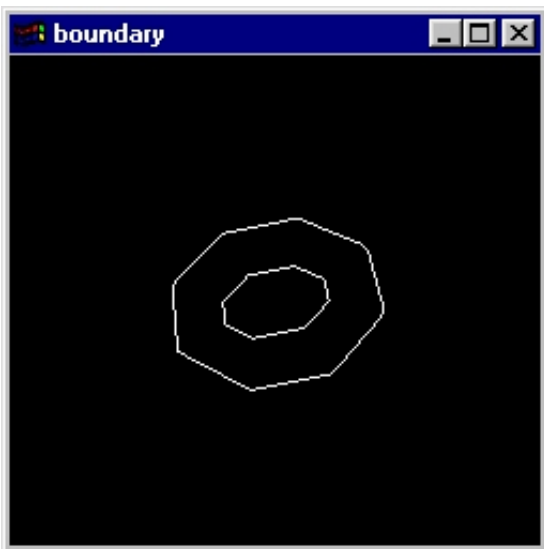
Snapshots: [linear fog \(shown\)](#), [linear fog added by second pass](#).



Using the accumulation buffer for fast convolutions.

Source code: [convolve.c](#).

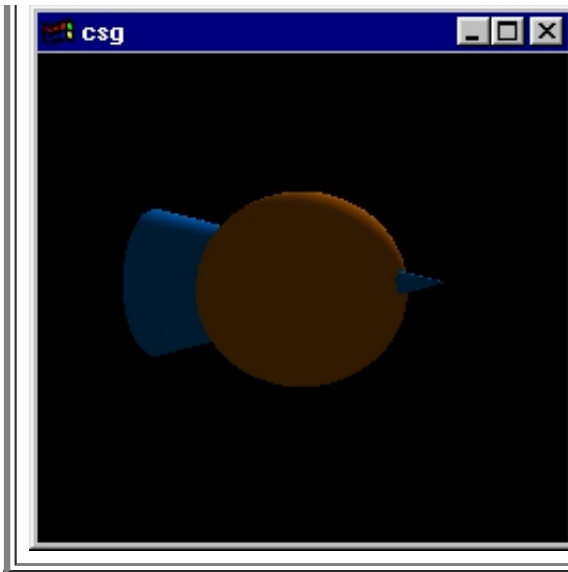
Snapshots: [5x5 blur \(shown\)](#), [laplace transform](#).



This example shows how to use the GLU polygon tessellator to determine the 2D boundary of OpenGL rendered objects. The program uses OpenGL's feedback mechanism to capture transformed polygons and then feeds them to the GLU tessellator in `GLU_TESS_WINDING_NONZERO` and `GLU_TESS_BOUNDARY_ONLY` mode.

Source code: [boundary.c](#).

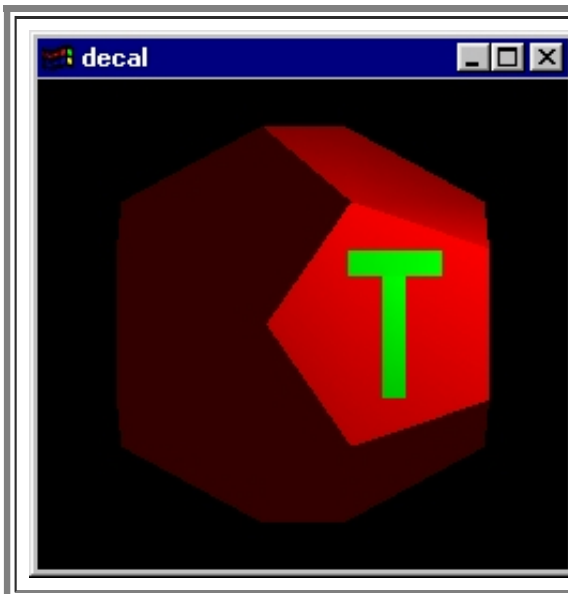
Snapshots: [torus \(shown\)](#).



Composite solid geometry using the stencil buffer.

Source code: [csg.c](#).

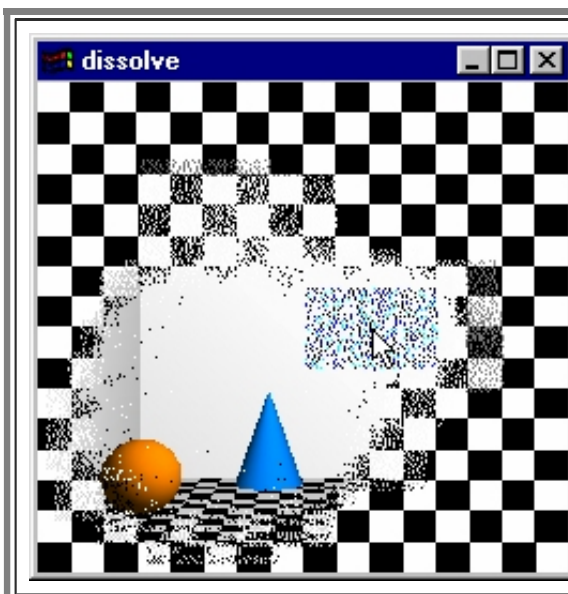
Snapshots: [A or B \(shown\)](#), [A sub B](#).



Decaling the letter "T" using the stencil buffer.

Source code: [decal.c](#).

Snapshots: [T decal\(shown\)](#).

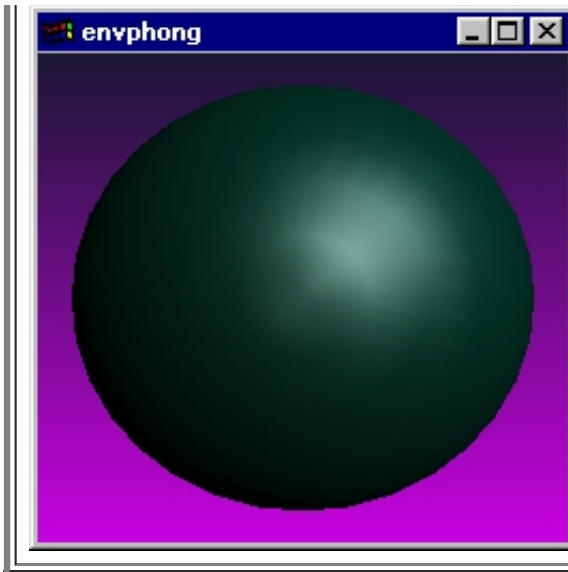


Dissolving using the stencil buffer.

Source code: [dissolve.c](#).

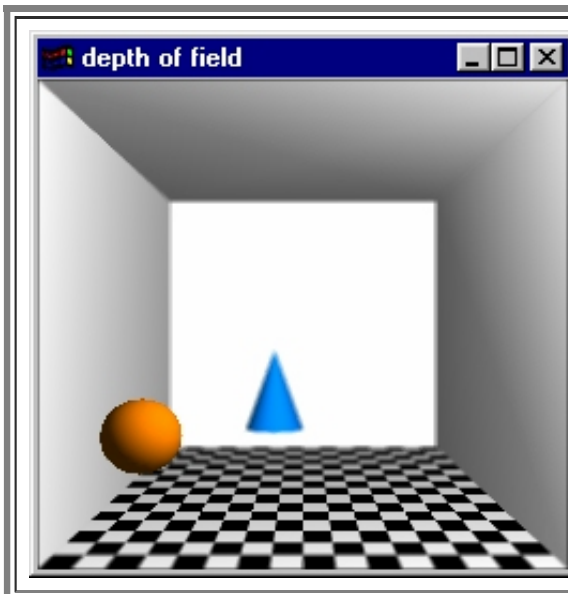
Snapshots: [dissolved checkerboard \(shown\)](#).

Demonstrates a use of environment texture mapping for improved highlight shading.



Source code: [envyphong.c](#).

Snapshots: [sphere \(shown\)](#),
[checkerboard torus](#).



Using the accumulation buffer for
depth of field (camera focus blur).

Source code: [field.c](#).

Snapshots: [depth of field\(shown\)](#),
[normal](#).

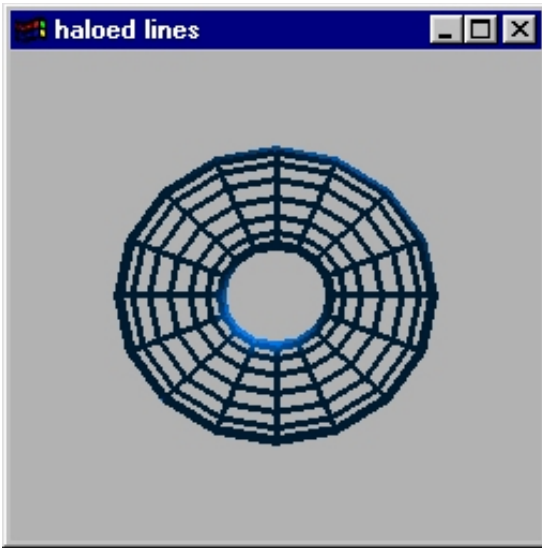


Example of how to generate texture
mipmap levels with the accumulation
buffer.

Source code: [genmipmap.c](#).

Snapshots: [mandrill mipmap
\(shown\)](#).

Draw haloed lines using the stencil
buffer.



Source code: [haloed.c](#).

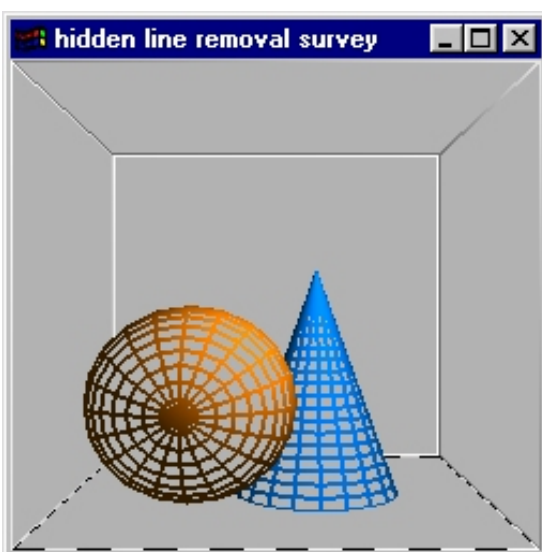
Snapshots: [torus \(shown\)](#).



Demonstrates a method of generating real-time shadows.

Source code: [hello2rts.c](#), [rts.c](#), [rtshadow.h](#).

Snapshots: [scene \(shown\)](#).



Line Rendering: Hidden line techniques.

Source code: [hiddenline.c](#).

Snapshots: [fat lines \(shown\)](#).

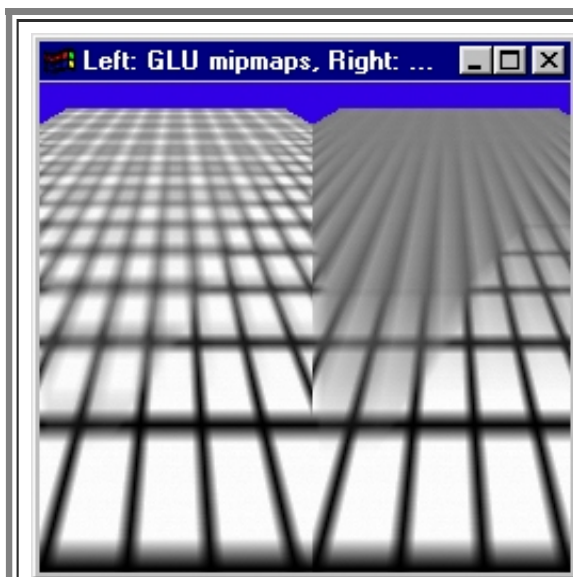
Examples of various image processing operations coded as OpenGL accumulation buffer



operations.

Source code: [imgproc.c](#).

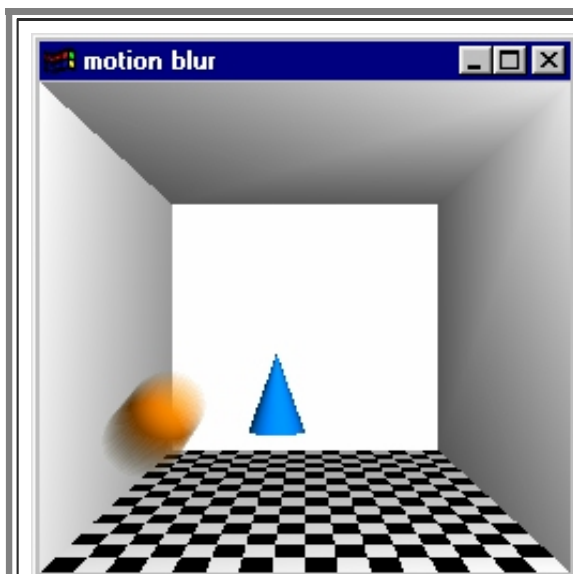
Snapshots: [saturated \(shown\)](#).



Survey of different mipmap filters.

Source code: [mipmap_lines.c](#).

Snapshots: [GLU mipmaps on the left](#), [MITCHELL mipmaps on the right \(shown\)](#).

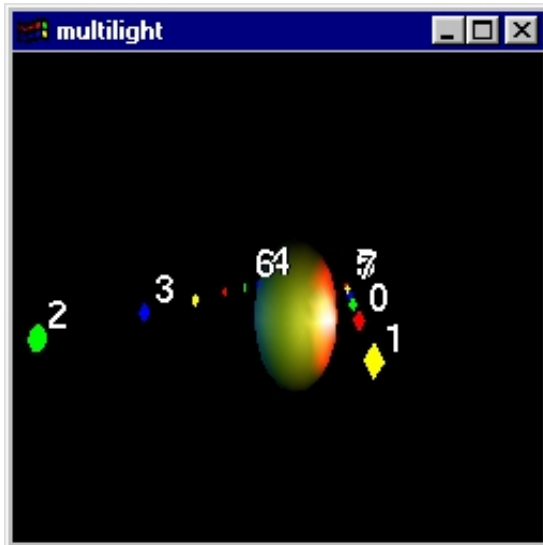


Using the accumulation buffer for motion blur.

Source code: [motionblur.c](#).

Snapshots: [motion \(shown\)](#), [normal](#).

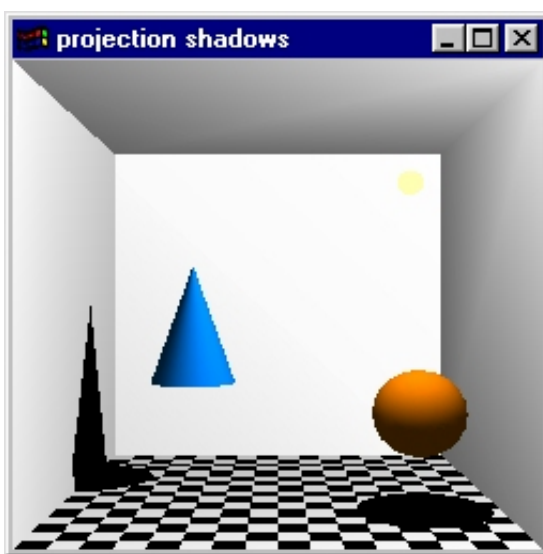
This program demonstrates virtualization of OpenGL's lights. The idea is that if an object is lit by many



lights, it is computationally more efficient to calculate the approximate lighting contribution of the various lights per-object and only enable the "brightest" lights while rendering the object. This also lets you render scenes with more lights than the OpenGL implementation light (usually 8). Two approaches are used: The "distance-based" approach only enables the 8 closest lights based purely on distance. The "Lambertian-based" approach accounts for diffuse lighting contributions and approximates the diffuse contribution.

Source code: [multilight.c](#).

Snapshots: [scene \(shown\)](#).



Rendering shadows using projective transforms.

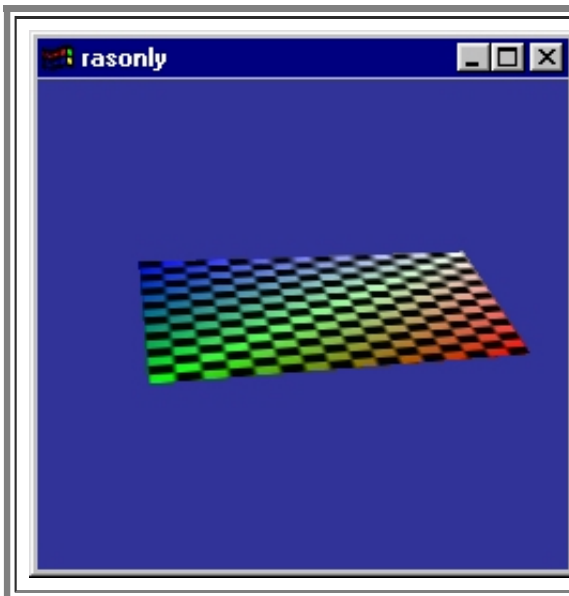
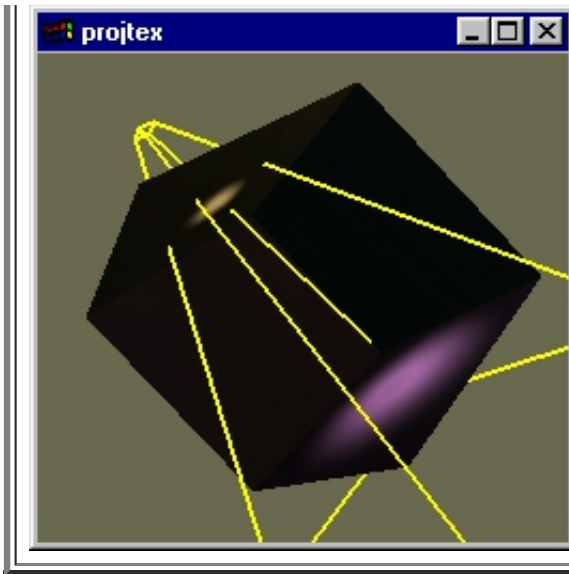
Source code: [projshadow.c](#).

Snapshots: [shadows \(shown\)](#), [no shadows](#).

Demonstrates simple projective texture mapping.

Source code: [projtex.c](#).

Snapshots: [cube \(shown\)](#), [three sides](#).



Demonstrates the use of OpenGL for rasterization-only, with perspective-correct texture mapping.

Source code: [rasonly.c](#).

Snapshots: [textured plane \(shown\)](#).

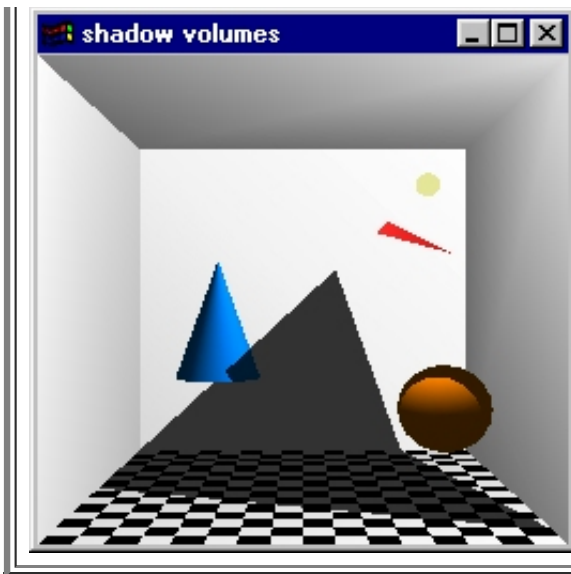


This program demonstrates a light source and object of arbitrary geometry casting a shadow on arbitrary geometry. The program uses OpenGL's feedback, stencil, and boundary tessellation support.

Source code: [shadowfun.c](#).

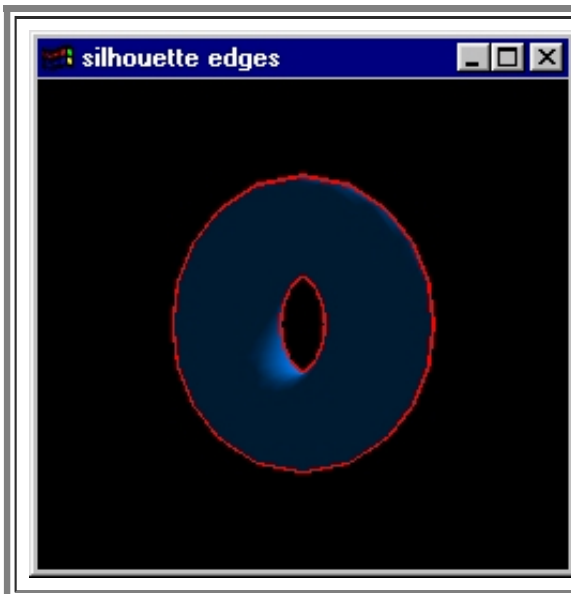
Snapshots: [shadows \(shown\)](#), [view from light](#).

Demonstrate shadow volume techniques.



Source code: [shadowvol.c](#).

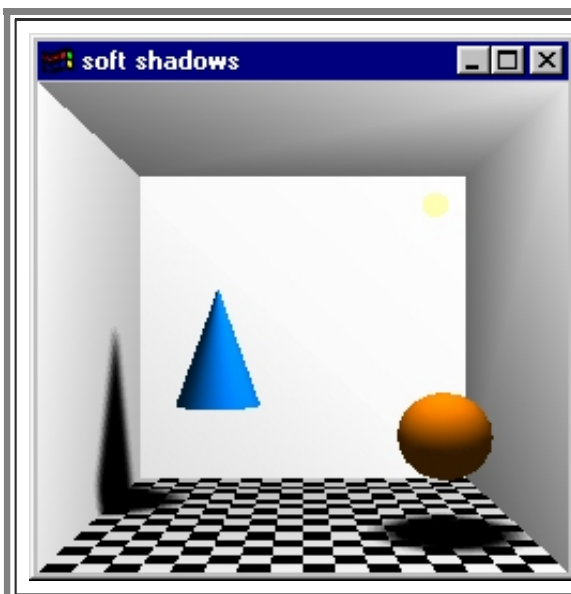
Snapshots: [shadow \(shown\)](#), [volume](#).



Generating silhouette edges with the stencil buffer.

Source code: [silhouette.c](#).

Snapshots: [torus \(shown\)](#).

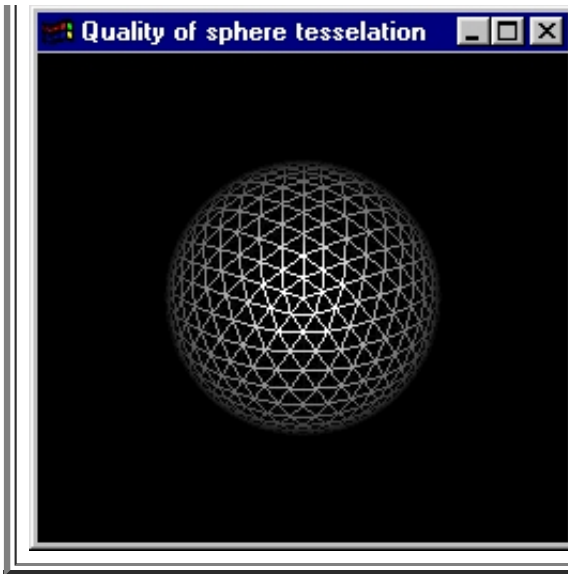


Demonstrate the use of accumulation buffer to create soft shadows.

Source code: [softshadow.c](#).

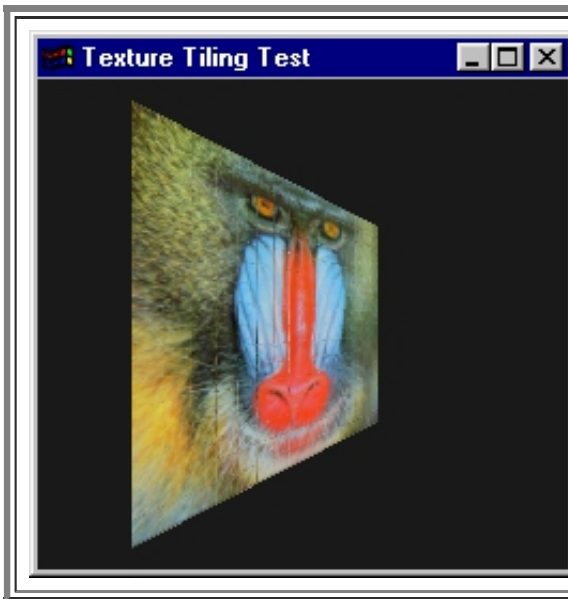
Snapshots: [soft shadows \(shown\)](#), [no shadows](#).

Different tessellations for a spherical shape.



Source code: [tess.c](#).

Snapshots: [geodesic \(shown\)](#),
[longitude/latitude](#).



Demonstration of how to "tile"
together small textures with a result
identical to if a much larger texture
was drawn.

Source code: [textile.c](#).

Snapshots: [tiled \(w/o borders\)](#)
[\(shown\)](#).

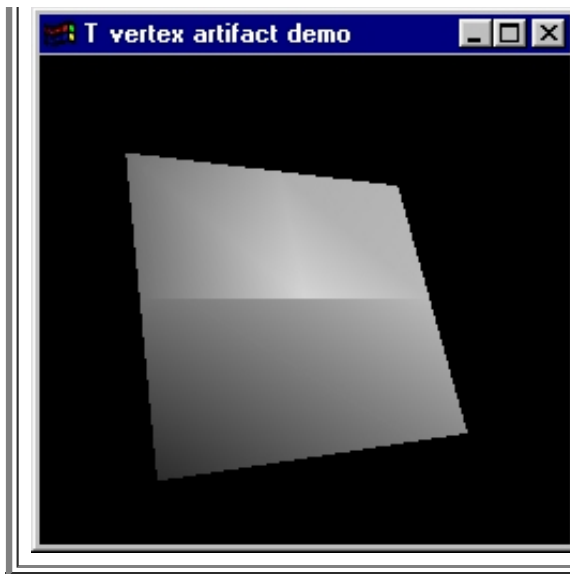


Trimming textures: demonstrates how
alpha blending or alpha testing can
be used to "trim" the shape of
textures to arbitrary shapes.

Source code: [textrim.c](#).

Snapshots: [alpha tested \(shown\)](#), [no](#)
[alpha blending/testing](#).

T-vertex artifacts example. The
moral: Avoid vertex edge junctions
that make a T-shape.



Source code: [tvertex.c](#).

Snapshots: [shaded T artifact \(shown\)](#), [wireframe](#).

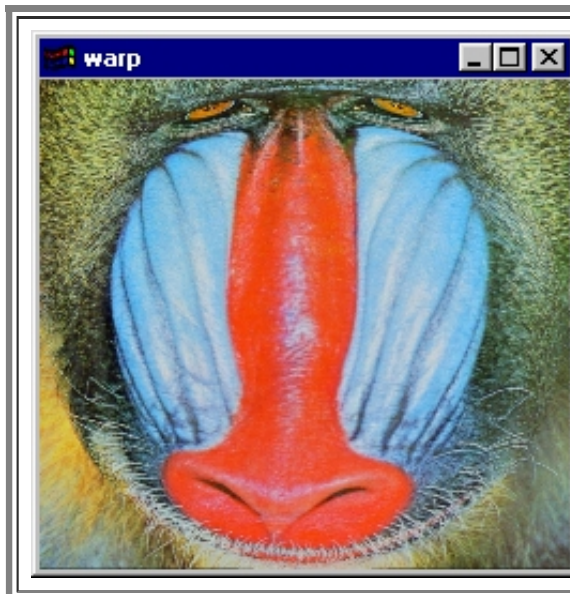
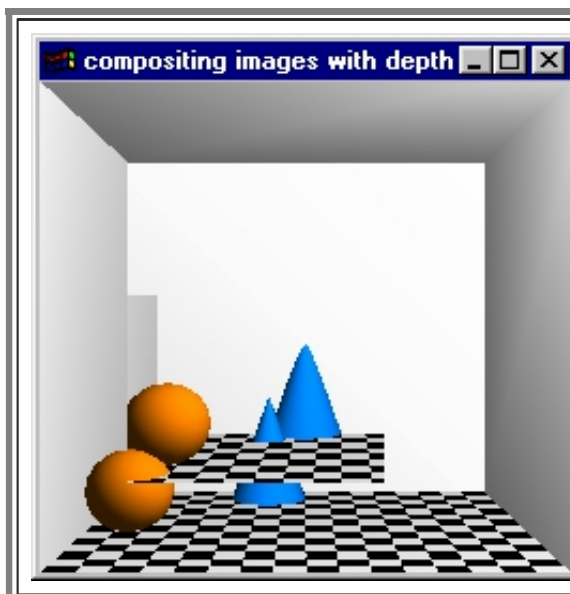


Image warping operations can be done via OpenGL texture mapping.

Source code: [warp.c](#).

Snapshots: [warped mandrill \(shown\)](#).



Compositing images that include depth information.

Source code: [zcompose.c](#).

Snapshots: [composed scene \(shown\)](#).