Interactive Perspective Cut-away Views for General 3D Scenes

Chris Coffin, Tobias Höllerer

Department of Computer Science
University of California,
Santa Barbara, CA 93106
{ccoffin,holl}@cs.ucsb.edu

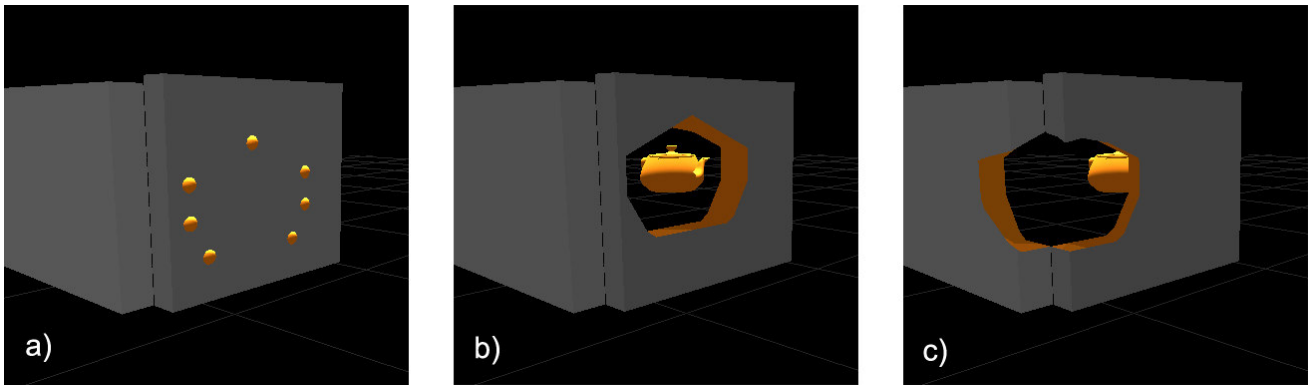

Figure 1: a) defining a "lathe" cutaway view on a plane in the scene, b) cutaway view through the occluder, c) the cutaway views sweeps correctly over multiple surfaces

ABSTRACT

We present a technique that allows a user to look beyond occluding objects in arbitrary 3D graphics scenes. In order to control this form of virtual x-ray vision, the user interactively cuts holes into the occluding geometry. The user can rapidly define a cutout shape or choose a standard shape and sweep it over the occluding wall segments to reveal what lies behind them. Holes are rendered in the correct 3D perspective as if they were actually cut into the obstructing geometry, including border regions that give the cutout shape physical depth, simulating penetration of a physical wall that possesses some generic thickness.

CR Categories and Subject Descriptors:

H.5.2 [Information Interfaces and Presentation]: User Interfaces – *interaction styles*.

I.3.6 [Computer Graphics]: Methodology and Techniques – *interaction techniques*.

I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism – *virtual reality*.

Additional Keywords: Cut-away view, interactive cutout, x-ray vision

1 INTRODUCTION

We present a new method for providing viewers of a 3D graphics scene with virtual x-ray vision — the ability to see through walls or other solid objects.

The motivation for this work is derived from the need for effective methods for allowing participants in a virtual or augmented environment to gain accurate spatial knowledge about objects behind one or more layers of occlusion. Our goal is to provide a simple interaction technique which makes use of only the geometry information in any 3D scene. Our method should work in absence of any semantic information about the objects in the scene (but may take advantage of such information if it is available).

Virtual X-ray vision easily suffers from confusing geometric relationships between occluders and revealed occluded infrastructure [7]. Our primary goal is to give the user a better understanding of the 3D scene by allowing the cuts themselves to impart information about the geometry through which the user is looking. To this end we define our points of interest and our cutout regions in 3D perspective on the actual surfaces of occluding objects. Through the apparent deformation of the cutout objects based on correct alignment, distance, and border geometry, we aim to give the user more information about the 3D scene. The ultimate goal of this technique is to make it appear to the user as if he was simply sweeping some cartoon-style hole of constant shape and area over the occluding geometry, which would deform to align with the objects being cut, and free up the view behind those objects. A cutout region should cling to the occluding geometry like a piece of cloth flattening to the respective surface geometry.

1.1 Related Work

The idea of revealing layers of information behind or about certain objects was explored early on in the Magic Lenses metaphor [3][4], a generic 2D interaction technique employing transparent user interface elements as overlays.

Viega et al. [11] extended the Magic Lens metaphor to 3D. However, as far as application for X-ray vision is concerned, their method is mostly used to switch between preset layers of

IEEE Symposium on 3D User Interfaces 2006
March 25 - 26, Alexandria, Virginia, USA
1-4244-0225-5/06/$20.00 ©2006 IEEE

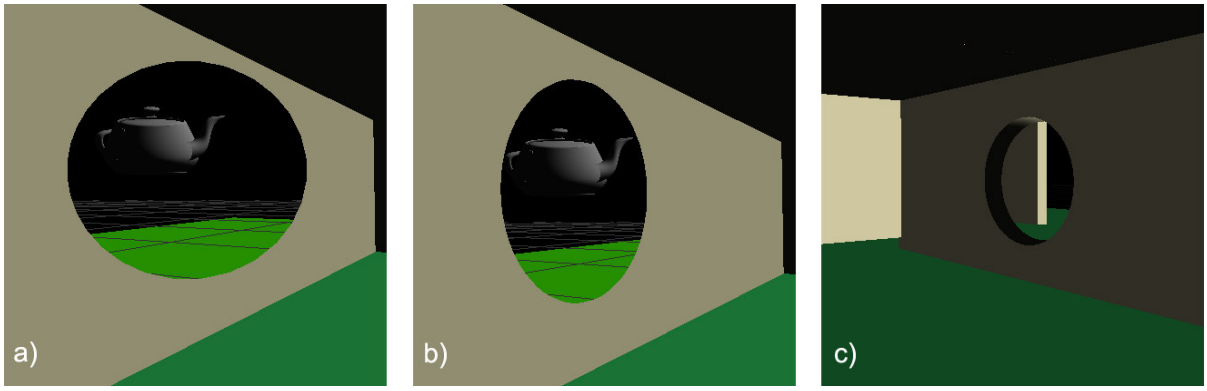

Figure 2: Increasing level of 3D perception: a) view-aligned circular cutout region, b) scene-aligned circular cutout region, c) scene-aligned circular cutout region with (artificial) border area

geometry and does not explore automatic positioning/adaptation of the cutting region. They also do not focus on obtaining an optimal shape for the cutout geometry in eliminating occlusions. Also, using OpenGL stencil buffer techniques (also employed in [4] and [8]), we can define much more irregularly shaped cutout regions than would be efficiently feasible using clipping planes (also used by [1]).

Our interaction technique is based on the notion of cutaway views [5], as previously employed in AR by the KARMA system for an equipment maintenance application. Our own recent previous work for application in mobile augmented reality focused on interactive tools that were either view-aligned or relied on semantic knowledge in the virtual scene (e.g., the concept of *rooms*) [1].

The work done by Diepstraten at al. [4] also uses the stencil buffer as a technique for eliminating occluded structures by intersection with cutout geometry. However, our method does not require knowledge about the main axis of the outside object in order to be effective. In fact, our method allows a user to place arbitrary cutting regions dynamically along surfaces of an object without requiring the recognition that the polygons involved form an object at all. Also, we generate or simulate cutout walls, one of the "requirements for cutaways" that their stencil-buffer approach did not address.

Looser et al's method for determining a 3D cutout region [8] is based on the user's view as through a view-plane aligned magic lens, while our work adapts the cutting region to the surfaces itself. We are also able to deal with much more dynamic cases of 3D geometry as the cuts are not based simply on preset layers of geometry. In fact, our method is in this respect similar to the otherwise completely differently focused approach of [10], where arbitrary OpenGL geometry is analyzed and categorized in layers, which are then presented in an exploded view.

2 METHOD

We present here a method for dynamic surface-oriented cutaway views. The algorithm for creating such cutouts is defined as follows: First, the user specifies a cutout shape, which is internally represented as a 3D cutout geometry, then, on a frame-by-frame basis a target point in the 3D scene is determined and a view-specific CSG operation is performed at that target point using the stencil buffer [9], removing all scene geometry falling within the 3D cutout geometry placed at that point.

The only information our method relies upon is the pure geometry of the scene. Connectivity and grouping information are useful for more efficiently determining intersections of cutout geometry with occluders, but are not essential.

After a desired surface and target point are selected, the cutting object automatically repositions and orients itself based on the surface's normal vector. Any intersecting polygons will then be cut based on the current state and depth of the cutout geometry.

In many 3D file formats rough groupings of polygons are easily identifiable. Often, these low-level groupings are designed for easy manipulation of a given object part or to conserve space in situations where information is repeated. Our current implementation defines groupings only slightly higher in structure than the coordinates for the geometry. When available, we are using shape nodes from VRML/X3D files to define our object groupings. We currently do not make use of any other hierarchy in the models themselves.

We define the scene as groupings of objects to optionally allow somewhat semantically intelligent cutting procedures. That is, in some cases it may be desired to only cut through one occluding object while leaving all the nearby occluding geometry intact. For example, a user may wish to eliminate only a large portion of the lower half of a wall without cutting into the floor.

2.1 Specifying the Cutout Geometry

We implemented several different ways for the user to specify or select a cutout shape: 1. defining a planar hole on a surface in the 3D scene, 2. defining a rotational sweep object on a surface in the 3D scene (termed "lathe object)", and 3. selecting one of a standard set of 3D cutting shapes.

2.1.1 Planar cutouts

Our first approach at creating a useful cutout region allowed the user to specify a polygonal surface along the surface of the obstructing geometry by drawing an outline directly onto a planar part of it. We then generated a 3D object by extruding the polygonal surface a certain distance along its surface normal. This "thickness" of the cutout geometry is kept constant for a given 3D scene and is user-specified. This method works well when cutting one surface only, or in a case where the angles between cut surfaces changes gradually. In the case where there is a sharp change in the angle of the surfaces being cut, the cutout region may appear to the user in a way which is non-intuitive. This is especially true if the edge of the surface is along a right angle (see Figure 3b)).

2.1.2 "Lathe" Cutouts

We observed that the desired effect can be achieved quickly and easily by using a piece of 3D geometry defined by a rotational sweep (termed "lathe") as the cutout region.

Proceedings of the 2006 IEEE Symposium on 3D User Interfaces (3DUI'06)
1-4244-0225-5/06 $20.00 © 2006 IEEE

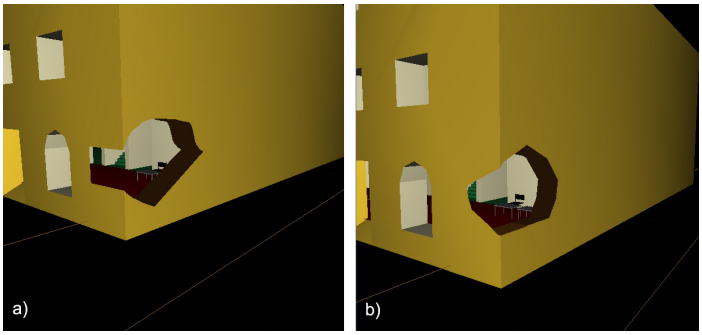

Figure 3: a) Problem with planar cutouts: non-intuitive wrapping around corners, b) "lathe" objects wrap more intuitively

Once the surface being selected is identified it is a simple matter to determine the surface normal and from that to allow the user to specify a series of points on the surface of the object. We decided not to have the user specify the points along the image plane, as in some cases the cutting shape will be defined specifically for the selected surface or object and we believe that defining the cut directly on the surface of the object itself is most beneficial for this purpose.

Figure 1a) shows a still from the procedure of defining a lathe object: we first define a central point for our radial axis (the projection of the vertical viewing axis onto the surface). We then set down points on one side of the radial axis and the opposing points are generated automatically. Once we have specified the shape of the cutout region, the solid geometry is rendered and the stencil test (see 2.2.2) begins eliminating occluding geometry.

Figure 3 illustrates how the previously mentioned problem with extruded planar geometry is addressed by lathe cutouts: In contrast to the non-intuitive way in which an irregularly shaped planar cutout defined on the right wall of the house results in a rectangular region being cut from the front wall of the house (3a), the cutout area stemming from the lathe object used in Figure 3b) appears to wrap nicely around the corner.

2.1.3 Standard Cutout Shapes

Irregular lathe cutouts can still be unpredictable in a subset of the cases in which the planar geometry fails. For example, in cases for which the lathe object extends over multiple surfaces, for one of which the rotational axis is close to the surface normal of the intersected geometry, the cutout region may not appear as expected (imagine applying the cutout of example 3b) to a 3D room corner including a ceiling plane). Using an object that is rotationally symmetric along multiple axes eliminates this problem. A sphere or a similar shape with randomly defined extrusions (for a more irregular look that emphasizes the non-permanent ad-hoc nature of the cutouts) is well-suited. The examples in Figures 2b) and 2c) use a spherical lathe object.

2.2 Using the Cutout Geometry

Now that we have defined a 3D cutout shape, the algorithm to generate the dynamic cutouts simply picks a target point on a surface in the scene and performs a real-time CSG operation, removing the volume of the correctly aligned cutout geometry from the scene geometry using the stencil buffer. This produces the desired effect of the cutout sweeping over the surfaces in the scene when the user is looking around or moving the cursor around.

2.2.1 Selection of a target point

In the 3D desktop implementation of our method, we determine the interest point through an image-space selection technique, namely as the first intersection of a ray orthogonal to the view-plane through the 2D mouse cursor with the perspectively rendered scene geometry. In alternative implementations, we could also pick along the user's line of sight, or a line extending from some pointing device.

This approach saves considerable time and effort compared with the 3D cursor-based selection of our previous work [1], since there is no need for manually determining distance.

2.2.2 CSG using the stencil buffer

We efficiently calculate the cutout region using the stencil buffer, as in [9]. Going beyond the application of stencil-buffer CSG as applied in [4], we then define border geometry for the cutout region as follows: If the occluding plane is already defined by front and back faces, a border around the cutout region will automatically be generated, reflecting the thickness of the object. If the surfaces are defined in only a loose grouping then we allow the user to switch to a mode which artificially produces a border of some chosen thickness by rendering a displaced copy of the front-facing polygon where we would normally assume the back face of the object in the above algorithm. This is happening in addition to, and independently of, the depth of the 3D cutout geometry. The thickness of border regions simulated for cutting through stand-alone flat polygons and the thickness of the 3D cutout shape are two independent variables, kept at different constant values for a specific 3D scene.

3 Results

We achieve interactive framerates on a Gateway laptop with 512MB of ram, 64 MB ATI mobility 9000 and 1.6 Intel Centrino processor and the scene from Figure 3, which represents a multi-story house consisting of about 10K polygons.

The speed of our method depends on the speed of calculating the intersection of the cutout region with the occluding geometry once per frame. This is important as the naive approach of performing the stencil test against each piece of geometry in the scene is prohibitive due to the large number of passes required.

The screenshots in Figure 1 b) and c) and 3c) illustrate the advantage of the lathe method over planar extruded geometry, as the cutout region smoothly wraps over to the adjacent plane without generating confusing rectangular patterns.

In Figure 2, we demonstrate the respective benefits of the different implemented depth cues. The user-oriented view of part a) (as also used by [1] and [8]) provides a larger view into the occluded scene, but the surface-oriented cutout of part b) results in a much improved understanding of the surface geometry, the exact location of the cut, and, arguably, increased immersion into the 3D scene. Part c) demonstrates the positive effect of using a border region around the cutout geometry on 3D scene interpretation.

3.1 Discussion

The size or depth of any solid cutout geometry is an important consideration independent of the apparent size of the cutout region on the surface. In some cases, the object being cut may not be fully penetrated by the cutout geometry. In this case the stencil buffer algorithm will fail. In other instances the cutout geometry may be much larger than the user anticipates and it may cut into objects the user wishes to stay visible. Currently, we have left the

determination of the size and depth of the cutout geometry to the user.

Our method also has problems similar to those described in [4], in that some concave surfaces as well as nested surfaces can cause our algorithm to produce unexpected visualizations when using the front and back face stencil buffer test. Methods for automatically choosing suitable depth for the cutout geometry, as briefly addressed in Section 4, would allow us to get around these problems.

The user currently defines the cutout region on a single surface. This causes problems if there is no surface large enough for the desired cutout region shape and size. We are currently implementing support for sketching the cutout region over arbitrary 3D geometry as in [6]. Alternatively, the user could define the cutout region on the view plane and/or scale a previously defined region.

4 Conclusions and Future Work

We have presented a method for interactively applying perspective cutouts to arbitrary 3D geometry, facilitating virtual x-ray vision that integrates unobtrusively with the 3D scene.

In the future, we plan to allow a more accurate orientation of the 3D geometry along surfaces in the scene by rotating the objects based on the average surface normals within the section of the screen covered by the cutout geometry. Pixel normals for that region can be calculated by a fragment shader program and an average speedily determined. This will allow for a smooth transition of, e.g., extruded planar geometry around sharp edges of obstructing geometry, providing another solution for the problem depicted in Figure 3). We are also looking into methods of quickly and accurately representing flexible cutouts by using projective texturing to determine the region cut. This should allow for an even greater range of objects which can be accurately cut.

One shortcoming of the current method is that the 3D cutting shapes could intersect scene geometry in a non-intuitive manner based on their depth, which is currently fixed. In future work, we want to automatically adjust the depth based on levels of occlusions that could automatically be determined from the scene geometry and viewing position.

Selection of a point on the surface of an object enables several natural interactions, such as easily traversing along the outside face of a building. However, it is naturally a level-of-occlusion–based selection technique, and in some cases several layers may need to be cut in order to expose the desired geometry. This may prove tedious. There are obvious workarounds, such as allowing the user to adapt the size or thickness of the cutting region in order to cut through several levels quickly. We plan to explore these methods as well as combination of surface selection with a distance based selection tool as in [1]. In future work, we further hope to explore interaction techniques combining surface and object selection with several other methods of cutting. For example we would like to combine the room selection method of [1] with a surface selection tool to enable interactions such as easy selection of adjacent rooms along a hallway.

We also plan to explore transitions of one cutting method to another automatically. We aim to supply the user with the greatest possible combination of information on both the immediate geometry as well as information on the position and state of the occluded geometry. We will explore a method similar to level-of-detail displays in order to determine the selection method as well as the size and type of the cutout region and the type of orientation, either view-plane–based or surface-based.

Finally, using solid cutout geometry is only one of many possibilities for eliminating sections of occluding structures. Using transparency or wireframe rendering styles are possible alternatives and we plan to develop a framework that allows us to explore efficient combinations of all these techniques.

References

[1] Bane, R. and T. Höllerer, Interactive tools for virtual x-ray vision in mobile augmented reality. In *Proc. ISMAR 2004 (IEEE/ACM Int. Symp. on Mixed and Augmented Reality)*, Arlington, VA, Nov. 2-5 2004, pp. 231–239.

[2] Bier, E.A., M.C. Stone, K. Fishkin, W. Buxton, and T. Baudel, 1994: A taxonomy of see-through tools. In *Proc. CHI 94*, ACM press, pp. 358–364.

[3] Bier, E. A., M. C. Stone, K. Pier, W. Buxton, and T. DeRose, 1993: Toolglass and Magic Lenses: The see-through interface. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, Kajiya, J. T., editor, volume 27, pp. 73–80.

[4] Diepstraten, J., D. Weiskopf, and T. Ertl. Interactive cutaway illustrations. In: *Proc. Eurographics '03*, pages 523–532, 2003.

[5] Feiner, S. and D. Seligmann, 1992: Cutaways and ghosting: Satisfying visibility constraints in dynamic 3D illustrations. In: *The Visual Computer*, **8**(5-6), pp. 292–302.

[6] Igarashi, T. and D. Cosgrove, 2001: Adaptive unwrapping for interactive texture painting. In: *Proc. ACM I3D 2001* (Symposium on Interactive 3D Graphics), pp. 209–216

[7] Livingston, M. A., J. E. Swan II, J. L. Gabbard, T. Höllerer, D. Hix, S. J. Julier, Y. Baillot, and D. Brown, 2003: Resolving multiple occluded layers in augmented reality. In *Proc. ISMAR '03 (Int. Symposium on Mixed and Augmented Reality)*, Tokyo, Japan, 56–65.

[8] Looser, J., Billinghurst, M. and Cockburn, A. Through the looking glass: the use of lenses as an interface tool for Augmented Reality interfaces. In: *Proc. GRAPHITE*, 2004, pp. 204–211.

[9] McReynolds, T. and D. Blythe, Advanced Graphics Programming Techniques Using OpenGL, SIGGRAPH '98 Course Notes, 1998, http://www.opengl.org/resources/tutorials/advanced/advanced98/notes/node19.html (accessed June 2005).

[10] Niederauer, C., Houston, M., Agrawala, M., and Humphreys, G. Noninvasive interactive visualization of dynamic architectural environments. In *Proc. of the 2003 Symposium on Interactive 3D Graphics* (Monterey, CA, Apr. 28–30). ACM Press, New York, 2003, 55–58.

[11] Viega, J., Conway, M. J., Williams G., and Pausch, R. 3D Magic Lenses. In *Proc. UIST '96 (ACM Symp. on User Interface Software and Technology)*, 1996, pp. 51–58.

1-4244-0225-5/06 $20.00 © 2006 IEEE