Hidden Line Removal for 2D Cartoon Images

Zhanggui Zeng and Hong Yan
School of Electrical and Information Engineering
University of Sydney, NSW 2006 Australia
zzeng@ee.usyd.edu.au

Abstract

In this paper, we describe a hidden line removal scheme for 2D cartoon images. The depths are introduced to the polylines of the image. The surfaces of the image are classified and their depths updated according to the depths of polylines forming the surfaces. The hidden lines are identified by comparing the depths of the polylines with the depths of the surfaces. This algorithm has been applied successfully to our cartoon development system.

Keywords: hidden line, depth, surface, visibility, cartoon image

1 Introduction

Hidden Lines have to be removed in order to achieve realism in a computer-generated 2D or 3D image. Many algorithms have been proposed to solve this problem [Thalmann 1987, Watt 1993]. There are two basic categories of hidden line removal algorithms: object space algorithms and image space algorithms. Each has its benefits and disadvantages [Maghrabi, and Griffiths 1989]. Most researchers use hidden surface removal methods for 3D images. However, most information in use today is still presented in 2D form because 2D images are sufficient to describe most engineering and entertainment objects [Yu, and Sun 1997]. In contrast to 3D images, 2D Images including 2D cartoon images do not have depth information. This problem makes the hidden line removal difficult. Fortunately, depth can be introduced to 2D drawings by some means. The performance of some 2D hidden line removal algorithms is satisfied in many applications even if they cannot handle penetrating objects [Madi, and Walton 1999]. In this paper, we propose a scheme of hidden line removal for 2D cartoon images. This scheme can remove the hidden lines for penetrating objects. In Section 2, we introduce depth, vertical and oblique surfaces for 2D cartoon images. A visibility test and a depth update are described in Section 3. In Section 4, six experiments are described based on the hidden line scheme. Conclusions are made in Section 5.

2 Depths and surfaces

Depths of 2D drawings are the basis of hidden line removal. Some researchers introduce a varying depth for every object of a 2D image [5]. For example, Flash, the well-known animation software, uses a measurement of layers to identify the depths of objects in a 2D frame. However, when the objects and their relationship are complex, such a depth measurement is not sufficient. In fact, one object may possess many depths in order to represent being penetrated or overlapped by other objects.

Suppose that a 2D cartoon image is comprised of objects; the objects are made of polylines; and each polyline is assigned a different depth. For introducing depths to a 2D cartoon image, the storage sequence of these polylines can be regarded as their depths in a file or array. The depth of a polyline can be simply changed by changing its storage sequence in the file or array. If an object needs more depths to represent its complexity, we may break up the polylines of the object. Theoretically, any complex space relationship of objects can be described by changing the depths of polylines and breaking up polylines in a 2D cartoon image.

There are three types of objects represented by polylines in a 2D cartoon image: line objects without surface, surface objects with surfaces, and compound objects with surfaces and lines. Accordingly, there are two kinds of polylines; lines associated with line objects and edges associated with surface objects and compound objects. Lines represent line objects. Edge polylines represent object boundaries. Whether a polyline is a line or an edge can be specified by users. One or more edge polylines may form surfaces of objects. Based on the depths of the surfaces, the surfaces can be sorted into 2 classes; vertical surfaces and oblique surfaces. A vertical surface is formed by a single polyline (shown in Figure 1a). In addition, an edge polyline can be a polygon and edges (illustrated in Figure 1b). The polygon is also a vertical surface. Two or more edge polylines can join together and form a closed oblique surface with different depths. An oblique surface may be penetrated by objects as depicted in Figure 2.

The depths $\{d(x,y) \mid x,y \in S\}$ of a surface S are related to the depths of edges which form the surface. For a vertical surface, the depths of the pixels inside the surface are set as the depth of the polyline, i.e. $d(x,y) = i$ where $x,y \in S$ and i is the depth of the ith polylines of the image. If an oblique surface is penetrated by other objects, this oblique surface has to be divided into two or more virtual vertical surfaces which are formed by a single polyline and virtual lines (illustrated in Figure 3). The depths of pixels inside a virtual vertical surface are assigned the depth of the polyline which form the surface. If an oblique surface

does not intersect other polylines, whose depths are in-between the depths of the polylines forming the oblique surface, the depths of the pixels inside the surface are set as one depth of the polylines, i.e. $d(x, y) = i$ where $x, y \in S$ and i is the largest depth of the polylines forming the oblique surface.

If the depths of the surfaces whose depths are less than or equal to $i-1$ are settled, the hidden lines of the ith polyline can be located. By comparing the depths of the same pixels of the ith polyline with those of the surfaces, any part of the ith polyline whose depths are larger than those of the surfaces are regarded as being hidden lines.

3 Hidden line removal

Our algorithm acts in image space. Suppose that there are n polylines and their depths are ranged from 1 to n. We build a hidden line buffer to record the hidden lines for each polyline and a depth buffer or z-buffer for a 2D cartoon image [Tang, and Huang 1994]. The polylines of an image are processed individually in the order of the depths.

3.1 The process of a single polyline

Suppose that polylines $1 \sim i-1$ have been processed and the depth buffer is updated. The ith polyline is now being processed.

3.1.1 Visibility test

Suppose that the ith polyline is comprised of u points and denoted as $\{p_1,..., p_u\}$. This polyline $\{p_1,..., p_u\}$ has to be transformed to a continuous lines $\{p_1^{'},..., p_j^{'}, p_{j+1}^{'},..., p_v^{'}\}$ where $p_j^{'}$ denotes a point of the ith polyline and is adjacent to $p_{j+1}^{'}$ in the image space. The depths of each pixel of the lines $\{p_1^{'},..., p_j^{'}, p_{j+1}^{'},..., p_v^{'}\}$ are i.

By comparing the depths of continuous lines $\{p_1^{'},..., p_j^{'}, p_{j+1}^{'},..., p_v^{'}\}$ with the depths of the depth buffer, the visibility of each pixel of the polyline can be decided. If $d(x, y) < i$, then the pixel on (x, y) is invisible for the ith polyline. All the invisible points of the ith polyline can be recorded in the hidden line buffer for the ith polyline. Otherwise, the pixel on (x, y) is visible.

3.1.2 Depth update

If the ith polyline is a polygon or it forms a polygon with other edge polyline previously processed, the depths inside the polygon has to be updated. Any points of polylines behind the polygon would be invisible. The depth update includes two steps; firstly, to locate the surface, secondly, to fill the surface with appropriate depths.

Since the ith polyline may be compound, we have to decompose the ith polyline into polygons, edges, and lines. If a vertical surface is found by decomposing the ith polyline, the depths of the vertical surface can be updated as mentioned in Section 2. By searching the decomposed edges in the depth buffer, the intersections of the edges and other edge polylines can be found. By searching these intersections and the overlapped edge polylines in the depth buffer, the oblique surface can be located. By checking the depths of the intersections, we know if the oblique surface is penetrated or not. Then the depth of the oblique surface can updated as mentioned in Section 2. Please note that the visibility of the polylines, which penetrate this oblique surface, has to be tested again based on the updated image depth buffer.

3.2 The process of 2D drawings

Based on the process of single polylines, the whole process of hidden line removal for a 2D cartoon image can be outlined as follows:

```
0   Procedure Hidden_line_removal
1   Construct a depth buffer for the image
2   for each polyline
3       Transform the ith polyline
        into a contin uous lines
4       Apply the visibility test and
5       record the hidden lines to the
        hidden line buffer
6       if this polyline is edges
7           Decompose the polyline
            into vertical a surface and
            edges
8           if  there are vertical surfaces
9           then update the depths for
            the vertical surfaces
10          if there are edges
            then find the intersections
            of the  edges  and the
            processed polylines
11          Search for Oblique surface
12          if there is an oblique surface
13          then if the surface is penetrated
14              then update de pth for
                penetrated surface
```

15	Test for visibility again for the penetrating polyline
16	else
17	update the depths for th e non-penetrated oblique surface

4 Experiments

We embedded this scheme into our 2D cartoon development system and tested it on many cartoon images. The results are satisfactory in both quality and speed.

Figure 4 represents two crossing rectangles. Figure 4(a) shows the results when the depth of the horizontal rectangle is less than the depth of the vertical rectangle. Figure 4(b) gives the results when the depth of the vertical rectangle is less than the depth of the horizontal rectangle. The dash lines are the hidden lines in the cartoon images. Figure 5 represents three moving ellipses. Figure 5(a) shows the results when the ellipse of the least depth locates at the right. Figure 5(b) illustrates the results when the ellipse of the least depth moves to the middle. Figure 6 illustrates a cube whose hidden lines are indicated by the dash lines. Some surfaces of the cube are non-penetrated oblique surfaces. Figure 7 represents a quadrilateral penetrated by a triangle. Please note that the quadrilateral is formed by two polylines. The depth of the triangle is larger than the depth of one polyline of the quadrilateral but less than the depth of another polyline of the quadrilateral. Figure 8 shows a "U" shape oblique surface penetrated by another surface. Figure 9 shows an image formed by a cross, three ellipses and a cube. The depths of the images are clearly identified by the hidden lines.

5. Conclusions

In this paper, we present a hidden line removal scheme for 2D cartoon images. By introducing depths to the polylines of the surfaces, the method can show oblique surfaces penetrated by other surfaces. The proposed scheme is simple and can only process some simple images. However, the idea can be developed into a complex hidden line scheme for 2D images.

References

Thalmann, N.M., Thalmann D. (1987): Image Synthesis Theory and Practice, Springer-Verlag.

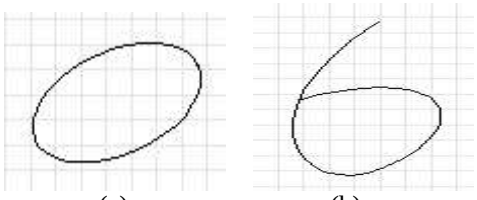
Watt, A. (1993): 3D Computer Graphics, Addison-Wesley.

Maghrabi, M., Griffiths J.G. (1989): Removal of hidden lines by recursive subdivision, Computer Aided Design, **21**: 570-576.

Yuan, X., Sun, H. (1997): P-buffer: a hidden line algorithm in image space, Computer & Graphics, **21**(3): 359-366.

Madi, M.M., Walton, D.J. (1999): Modeling and visualization of layered objects**,** Computers & Graphics, **23**(3): 331-342.

Tang, L.A., Huang, T.S. (1994): An efficient hidden-line removal method based on z-buffer algorithm, Image Processing, Proceedings of IEEE International Conference, **1**: 657-660.



(a) (b)

Figure 1. Vertical Surface, (a) formed by a whole edge polyline, (b) formed by a partial polyline.

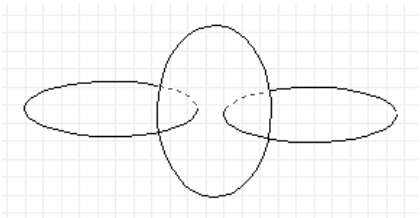


Figure 2. Penetrating oblique surfaces.

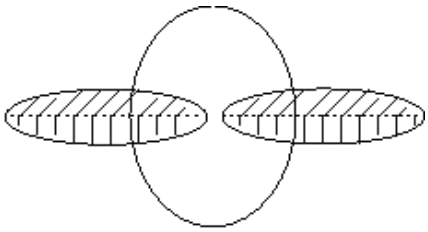
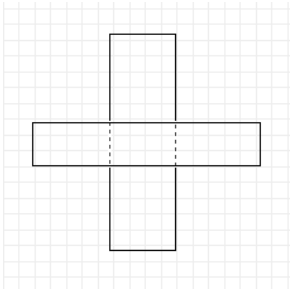
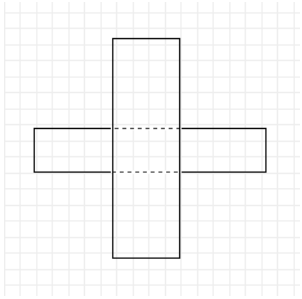


Figure 3. The division of oblique surfaces.

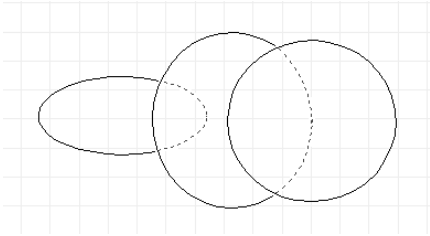


(a)

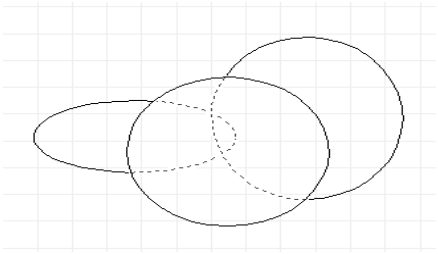


(b)

Figure 4. Rectangles, (a) hidden vertical rectangle, (b) hidden horizontal rectangle.



(a)



(b)

Figure 5. Ellipses, (a) three ellipses, (b) three moved ellipses.

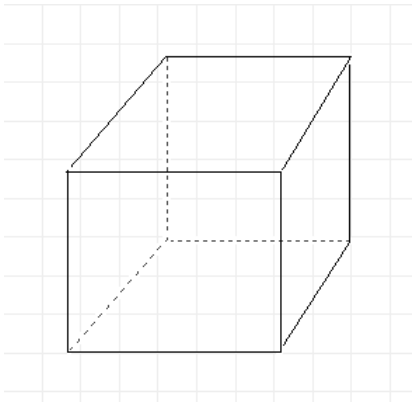


Figure 6. Cube.

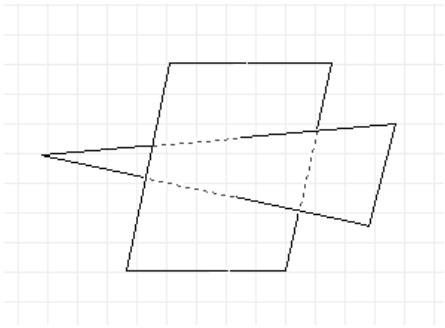


Figure 7. Penetrating triangle.

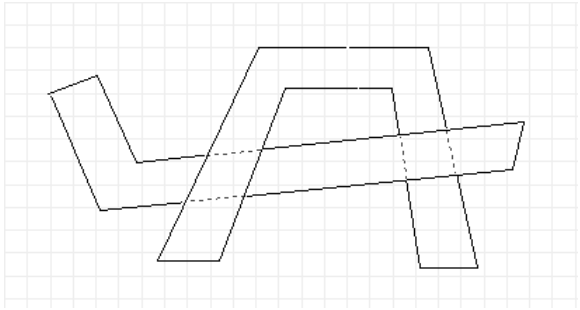


Figure 8. Penetrated “U” shape.

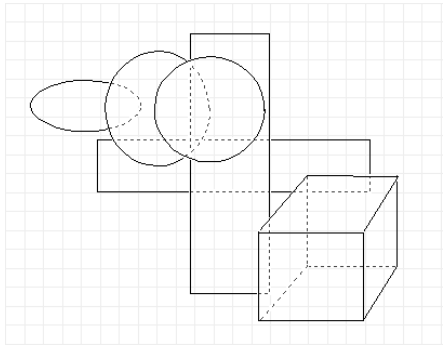


Figure 9. Hidden line removal for several objects.