Resolution Independent Rendering of Deformable Vector Objects using Graphics Hardware

Yoshiyuki Kokojima*
Toshiba Corp.
Research & Development Center

Kaoru Sugita
Toshiba Corp.
Research & Development Center

Takahiro Saito
Toshiba Corp.
Semiconductor Company

Takashi Takemoto
Toshiba Corp.
Semiconductor Company

1 Introduction

Resolution independent rendering is important for many applications such as text rendering and rendering vector images. This area has received quite some interest in recent years due to the growing popularity of Flash and SVG-based applications. This sketch presents a new method for resolution independent rendering of vector images suitable for programmable graphics hardware. We have enhanced a previous method [Loop and Blinn 2005] by using a stencil buffer and *transparency multisampling* [ATI 2005; NVI 2004], so that our method has the following advantages:

- efficient rendering of dynamically deformable geometry,
- improved handling of geometric (self) intersections,
- order independent rendering with anti-aliasing.

2 Algorithm

We assume that a given vector object is defined by closed paths consisting of vector primitives such as line segments and Bézier curves, see Figure 1(a). In this figure, filled dots represent the end points of the vector primitives, and hollow dots represent Bézier control points. This figure contains only quadratic curves for simplicity, but our method is also able to handle cubic curves. Our method consists of the following steps (see Figure 2):

1. *Triangulation*: *Line-edged triangle fans* (Figure 1(b)) are generated from the given vector object, each of which is defined by the arbitrarily selected pivot point (each hollow box in Figure 1(b)) and the filled dots of the individual closed path. In addition, *curve-edged triangles* (Figure 1(c)) are also generated by connecting the end points and the control point of each Bézier curve.

2. *Stencil buffer processing*: A stencil buffer is cleared to zero, and the line-edged triangle fans (Figure 1(b)) are drawn into the buffer using a bitwise-inversion operator such as GL_INVERT. If a pixel is covered an odd number of times by the triangle fans, its stencil value will be nonzero (black), otherwise zero (white), as shown in Figure 1(d). Subsequently, the convex regions in the curve-edged triangles (Figure 1(e)) are drawn into the stencil buffer, which changes the content of the stencil buffer from Figure 1(d) to Figure 1(f). We determine if a pixel is inside or outside the convex region by evaluating an implicit equation [Loop and Blinn 2005] in a pixel shader program.

3. *Alpha assignment*: Our method uses multisampling for anti-aliasing. However, aliasing artifacts on the curved edges remain since multisampling does not anti-alias inside triangles [ATI 2005]. In order to reduce these artifacts, we assign alpha values to pixels close to the curved edges using a signed distance function [Loop and Blinn 2005]. These alpha values will be used for calculating fragment coverage values in step 5.

4. *Color buffer processing*: A polygon large enough to enclose the whole region of the given vector object is drawn into a color buffer with a stencil mask that accepts fragments only where the stencil buffer is nonzero.

5. *Alpha to coverage*: For anti-aliasing of the curved edges, we map the alpha values assigned in step 3 to the coverage values using transparency multisampling.

3 Advantages

The previous method employed constrained Delaunay triangulation and subdivision of overlapping triangles as preprocesses on a CPU.

* e-mail: yoshiyuki.kokojima@toshiba.co.jp

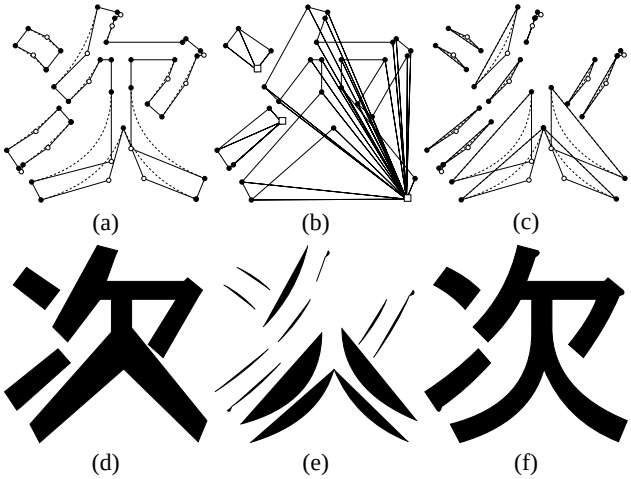

Figure 1: Example of generating a stencil mask.

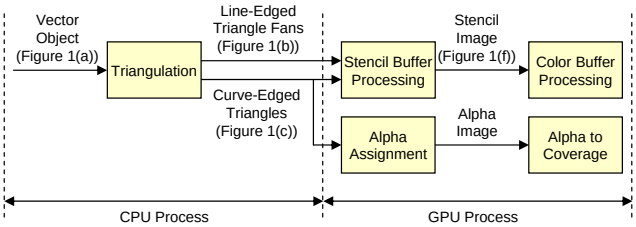

Figure 2: The process flow of our method.

If the given vector objects are deformed dynamically, the performance of the previous method will degrade since further CPU intervention is needed for re-triangulation and re-subdivision. In contrast, our method has no expensive CPU processes, and therefore it can render dynamically deformable vector objects efficiently.

Another issue for the previous method is that the subdivision may not terminate if the given objects have (self) intersections. Even for such cases, our method guarantees termination of the processes since the subdivision is unnecessary.

For anti-aliasing, the previous method employed alpha blending. Thus, it was necessary to sort all the objects from back to front since the result of alpha blending depends on a drawing order. In contrast, our method uses transparency multisampling instead of alpha blending. Therefore, sorting is unnecessary.

4 Results

We applied our method to rendering of Japanese TrueType font outlines embedded in a three dimensional space. Our method rendered over three hundred dynamically deforming characters at about 40 fps. Under such conditions, our method was more than 10 times faster compared to the previous method.

References

ATI TECHNOLOGIES INC. 2005. *Alpha to coverage*, Radeon SDK Mar. 2006.

LOOP, C., AND BLINN, J. 2005. Resolution independent curve rendering using programmable graphics hardware. In *Proceedings of SIGGRAPH 2005*, 1000-1010.

NVIDIA INC. 2004. *Antialiasing with Transparency*, http://developer.nvidia.com/object/transparency_aa.html.