Available online at www.sciencedirect.com
ScienceDirect
Computers & Graphics 30 (2006) 754–766
COMPUTERS & GRAPHICS
www.elsevier.com/locate/cag

Technical Section

G-strokes: A concept for simplifying line stylization

Tobias Isenberg[a,*], Angela Brennecke[b]

[a]Department of Computer Science, University of Calgary, Canada
[b]Department of Simulation and Graphics, Otto-von-Guericke University of Magdeburg, Germany

Abstract

In most previous NPR line rendering systems, geometric properties have been directly used to extract and stylize certain edges. However, this approach is bound to a tight stylization of strokes as the focus lies on the edge extraction. Styles are applied to the currently extracted edges, making it necessary to re-do certain computations whenever several different styles are to appear concurrently in the same rendition. Consequently, the generation of renditions is often constrained to one or two styles to keep computational cost low. To broaden the possibilities of generating highly expressive line drawings we introduce the concept of G-strokes. In contrast to the above-mentioned approach, we propose to keep all edges and to extract the geometric properties instead. According to these properties, one style could be applied to a particular set of edges and another style could be applied to another set of edges without having to extract the designated edges anew. This makes it easy to enrich the set of line stylization means, allowing more freedom and creativity for generating varied line drawings. We show a number of possible G-strokes using both simple and complex examples to demonstrate the power of our approach.

Keywords: Non-photorealistic rendering; Line rendering; Line stylization; Stroke pipeline; G-strokes; G-buffers

1. Introduction

In the past two decades, line rendering has been established as one of the major areas of research within the growing field of non-photorealistic rendering (NPR) [1,2]. Fueled by the development of a variety of silhouette extraction algorithms [3] as well as feature detection techniques (e.g., [4]), numerous methods for line and stroke-based rendering using a wide range of styles have been and are being conceived. In particular, the use of object-space edge extraction facilitates the further stylization and processing of these edges as *strokes* since they are available in analytic form.

Typically, the stylization process is implemented using a *stylization pipeline* within which strokes are processed. In general, a stylization pipeline comprises a sequence of pipeline elements. At each stage of the pipeline, data is either modified, added, or simply prepared for the next element in line. The sequential approach of this procedure, therefore, is appropriate for the analytic stroke stylization process where strokes are to be stylized in a number of steps [5–7]. The interconnection between the line drawing and its generation technique is crucial to the below-stated problem and the necessity of the G-strokes concept: the rendition's explanatory power depends on the stroke's topology and style which are in turn established and altered by the pipeline elements. Therefore, the pipeline elements as well as their combination have to be as flexible as possible to achieve the favored line drawings.

However, the more stylization elements are being created and added to the stylization pipeline, the more difficult the stylization process itself becomes. This is because a new element may introduce new data that not only has to be captured but also has to be processed. For instance, texture or line thickness parameters may be added to the coordinates and their indices of the current stroke set. Whenever the indexing of the strokes changes, the parameters also have to change. Consequently, all pipeline elements that already have been implemented need to be adapted as well in order to handle the new data. Only then can the old elements be used together with the new one.

*Corresponding author. Tel.: +1 403 210 9507; fax: +1 403 284 4707.
E-mail address: isenberg@cpsc.ucalgary.ca (T. Isenberg).

0097-8493/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2006.07.006

Likewise, every new element also has to ensure that it can handle the increasing number of already existing data sets. Therefore, a two-way dependency between the pipeline's elements and the processed stroke data exists. This makes the development of a comprehensive line stylization and rendering toolkit increasingly complex and difficult to manage.

Inspired by the groundbreaking work of Saito and Takahashi on *G-buffers* [8], we propose the concept of *G-strokes* as a solution to this problem. We regard all data added to the stroke (coordinates and indices) by a pipeline element as geometric stroke properties and call them *G-strokes*. These are maintained parallel to the underlying geometry. In this context, for example, the stroke's parameterization (e.g., for texturization) and visibility are geometric properties. The latter could be captured in a G-stroke telling the current stroke set which strokes are visible and which are not and could then be used to apply a certain style to the extracted edges (see Fig. 3).

In contrast to Saito and Takahashi's G-buffers, G-strokes might need to be adapted during the stylization process since the underlying geometry or topology of the stroke may change. We demonstrate how this can be achieved and how the necessary programming work can be minimized, making it easy to add new pipeline elements without having to care for the existing data sets.

The remainder of this paper is structured as follows. In Section 2 we review related work with respect to the concept presented in this paper. Then, in Section 3 we discuss the problems arising from the previous handling of stylization pipelines and introduce our G-strokes concept to overcome them. In Section 4 we address implementation issues and design decisions we made to realize the concept. In Section 5 we present a number of case studies in order to illustrate the flexibility of a G-strokes-based stylization. In Section 6 we summarize our contribution and discuss directions for future work.

2. Related work

The field of NPR has diversified and grown considerably in recent years [1,2]. However, line rendering was one of the first issues to be discussed [8–10] and this topic continues to be one of the major areas of NPR (e.g., [3,4,7,11,12]). As one of the earliest and most important contributions for the area, Saito and Takahashi presented the G-buffer concept for enhancing the expressiveness of renditions [8]. In their paper, the authors describe how to extract additional data during the rendering process, store it in what they call G-buffers, and use it for computing NPR primitives. These primitives (silhouettes and feature lines) are then composited into the image to extend the comprehensibility of the shown objects. It is important to note that G-buffers use the same underlying topology as the rendition they were generated for, i.e., the $x \times y$ pixel matrix of the image. Thus, G-buffers form a stack of images, each recording a different property.

Although Saito and Takahashi used their G-buffers to store extracted linear features from 3D data, this happened entirely in image-space. Besides this pixel-based approach there are also two different approaches to extract edges and render strokes—hybrid methods and techniques in object-space [3]. In particular, the latter group is of interest for this paper as it offers a greater freedom in terms of line parametrization and further processing than hybrid and image-based techniques. In the area of object-space stroke generation, the concept of using line stylization pipelines has emerged. The pipeline's elements are used to extract significant edges from a model, concatenate them to strokes, stylize these strokes according to certain properties and parameters, and finally render them [6,7,12]. In particular, Grabli et al. discuss the process of line stylization in a pipeline in greater detail. Their main contribution to NPR line drawing is the separation of lines or edges from the attributes which guide the line stylization. This separation is achieved by collecting as much information on the scene as possible. The gathered data is categorized into 3D scene information (3D coordinates, normals, object IDs, etc.), auxiliary maps (local average depth, item buffer, etc.), the view map (a planar graph which is received by projecting the extracted feature edges into the view plane), and the current drawing (local stroke density). This data is then used to create style sheets for stylizing the lines extracted from a 3D model. These can now be arranged to model different NPR-pipelines. The individual style sheet modules operate on the 2D view map's edges and consist of selecting, chaining, splitting, and assigning attribute operation. Furthermore, several of these modules can be used simultaneously. In a final step, each resulting image layer is combined into a single image leading to a huge amount of different styles which can be used in one image. However, in contrast to our technique, they are bound to a sequential pipeline in a greater extent. Moreover, the information acquiring step is fairly complex and could be handled in an easier way.

3. G-strokes

In order to support the creative process of generating expressive line renditions, a wide variety of stylization elements and stroke properties need to be available to the artist. The realization of previous stylization pipelines hinders the creation of truly powerful line rendering systems. In the following, we present the concept of G-strokes that not only can overcome these limitations but also reduce both the amount of necessary coding for each new stylization element and the complexity of the resulting stroke pipeline.

3.1. A new stroke concept

A common way to represent an edge is to store the edge's segments as an indexed list with pointers to the actual coordinates of the edge's vertices, each segment being