Maic Masuch, Stefan Schlechtweg, Bert Schönwälder: "daLi! - Drawing Animated Lines!" . In: Oliver Deussen, Peter Lorenz (Eds.): Proceedings Simulation und Animation '97, S.87-96, SCS Europe, 1997.

daLi! — Drawing Animated Lines!

Maic Masuch, Stefan Schlechtweg, Bert Schönwälder

Department for Simulation and Graphics,
Otto-von-Guericke University of Magdeburg,
Universitätsplatz 2, D-39106 Magdeburg, Germany,

e-mail: {masuch|stefans|bert}@isg.cs.uni-magdeburg.de

Abstract

We present an animation system for the creation of non-photorealistic 3D animations. Our system daLi! is able to render images using a common 3D model as input. This model may be enriched with additional information concerning for instance hierarchy, structure and presentation style. The rendering is done analytically, which means the output is resolution independent. It results in a series of images that depict the animation using line-drawings. The implementation of daLi! was completely done in Smalltalk.

Keywords: *non-photorealistic rendering, non-photorealistic animation, computer-generated line-drawings, abstract images*

1 Introduction

This paper describes an object-oriented approach for creating non-photorealistic animations and an overview of the design and implementation of daLi!, a realization of these concepts. The paper is organized as follows: Section 2 gives the basic motivation for generating non-photorealistic images. The following Section 3 gives a brief overview of related work. Sections 4 and 5 focus on the two main parts, namely the rendering of still frames and the animation of those frames. The design of our object-oriented animation system is described in Section 6. The paper ends with some concluding remarks and visions for future work.

2 Non-photorealistic Imaging

Today's computer graphics research concentrates almost exclusively on photorealistic images, although traditionally-created drawings have an artistic quality that computer-generated images lack. The abstraction of a depicted scene can have essential advantages. The artist can simplify a picture by leaving out unnecessary, disturbing details and he can focus the viewer's attention on important features. He can also stress the importance of certain parts of a depicted scene through variation of the drawing style, e.g. less important regions may be painted with bright, fading lines, while relevant parts may be depicted with strong bold lines. The resulting image is still somehow realistic, but it may differ from

a photorealistic presentation in shape, color, texture, lights and shadows. Despite of—or due to—these deviations, non-photorealistic images are common in the fields of scientific illustration and classical art.
In order to enrich the means of (computer) graphical expression we are developing methods for rendering non-photorealistic images using a "sketchy style". Since we wish also to depict abstract information in an image we concentrate on line-drawings. They themselves are an abstract medium and so it is easy to code abstract information.
When creating non-photorealistic images on a computer it soon comes to mind to combine subsequent pictures into an animation that also benefits from abstraction and simplification. Due to the temporal nature of moving pictures the viewer always has less time to decode the presented information. Thus it is of utmost importance not to distract the viewer's attention with unnecessary detail.

In general, an animation system consists of two main parts:

- a rendering component which creates single images (frames)
- an animation component providing necessary scene information as input for creating subsequent images and combining the frames to an animation

Our system dali!, an application of these concepts, is described in Sections 4 and 5.

3 Related Work

In the last few years research in the field of non-photorealistic rendering has become a subject of greater interest within the computer graphics community.
WINKENBACH and SALESIN presented techniques for rendering pen-and-ink-style illustrations from a 3D model with the help of "prioritized stroke textures" [WS94]. LANSDOWN and SCHOFIELD [LS95] proposed a 2D painting system named *Piranesi* that operates on information from an underlying 3D model to create non-photorealistic pictures (which they call "expressive"). LEISTER [Lei94] described a rendering system that produced images resembling copper-plates using a modified ray-tracer. HSU and LEE [HL94] generated non-photorealistic images using "skeletal strokes" for drawing articulated lines in 2D.
All these approaches, however, mainly concentrate on creating single images. Of course there are animation systems like Softimage's TOONZ or TIC-TAC-TOON [FBC+95] that uses computer-aided vector-based sketching and painting techniques. But as these systems are simulating the traditional paper-based production process, they are restricted to two dimensions. As in classical animation, the third dimension is indicated by the use of transparent layers, i.e. a foreground layer for the main character and several background layers. This, to a certain extent, creates the illusion of three dimensions [TJ81]. The only system for rendering 3D animations in a painterly style is proposed by MEIER [Mei96]. Here colored particles that stick to the 3D model are used to simulate a hand-crafted look.

4 Still-Frame Generation

The generation of one animation frame can be compared with the classical rendering process for a single image. The input to the render engine is an enriched model consisting of:

- the 3D geometry data in polygonal representation
- additional information concerning:
  - the model structure (relations between the scene objects which are in some way relevant for the depiction of the scene/animation)
  - attributes of the scene objects, for instance material properties or special information for choosing the correct presentation style

All information regarding the position of objects, cameras and lights—which are based on the object movement data—are received from the animation engine (see Section 5) and converted into additional object attributes. So far we have been investigating the rendering of polygonal models. Due to the modular design (see Section 6), dali! can be extended to handle other geometrical representations as well.
Once given the input data, it is processed in a way similar to a standard rendering pipeline. Since we use polygonal models, this common approach can be chosen. This means that the geometry is transformed and projected and the hidden surface removal is performed.
For the process of hidden line (and surface) removal we use an extension of an algorithm developed by SECHREST and GREENBERG in 1981 [SG81]. This 3D sweep-line approach uses coherence information between neighboring faces in an object, thus it is very efficient. A disadvantage, however, is that this algorithm can only handle non-penetrating models and not all models can be modified to this demand. We extended this algorithm to meet the special requirements encountered when creating line-drawings. For example it is important to know for each line whether it disappears under a face. Also it is desirable to categorize the lines into those forming the contour of an object and those depicting the internal structure. This additional information makes it possible to choose an appropriate presentation style.
The output for each object consists of a list of its currently visible faces and lines, with each line carrying certain attributes about its type (contour, smoothing, etc.). Since a clearly drawn contour is of great importance for the depiction of an object, the single contour lines are combined into a set of control vertices which can be interpolated by parametric curves. Besides the lines generated from direct calculations on the model's edges, hatching lines can be created using 3D surface information.

The lines are finally drawn using *line qstyles*. These styles describe for each line the

- position and course as a spline and
- its appearance in terms of width and brightness variation over the length.

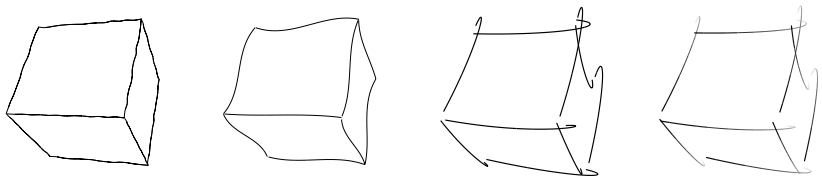

Figure 1: A cube drawn with different line styles

The description of the path depicting an edge results from the rendering process and is given attributes which contain information about the brightness of the line resulting from lighting calculations. The style is given by the user and may indicate, for instance, material properties. For drawing the lines, each path is combined with the corresponding style resulting in a curve with possibly changing width and brightness over the length. These curves are sent to the output module, creating a vector oriented representation of the rendered frame. As this representation is resolution independent, we do not have to care about aliasing problems, these are relegated to the output device. This output device converts the image into the desired output format as described in Section 6.

5 Animating the Frames

The animation engine has to perform two main tasks:

- determine the object movements according to the given keyframe data
- supply the render engine with new scene information

One part of the input for the animation engine is the 3D geometry model and the corresponding additional information. The other part is a set of keyframes that describe the movement of the objects and general animation settings. An object can be of any 3D shape, light or camera. In order to generate a sequence of images for each object its transformations are calculated. For the given keyframes, this calculation can be done directly, whereas for the intermediate frames the object movements have to be interpolated. To perform this task daLi is using *Hermite* interpolation [WW92].
During the creation process of the animation, particular attention must be paid to the appearance of the lines. Since they are drawn using line styles—and these styles may vary randomly—the frame-to-frame consistency of subsequent images is slightly disturbed. This "pulsing" or "waving" of lines gives the animation a hand-crafted appearance. The coherence between frames depends on the chosen line style. If a "wild" style has been selected for the representation of an object (this means the curve has a certain degree of freedom to differ from the line segment it represents) the object's contour and inner lines show a

wide range of variation. Choosing a calm style with only very small permitted deviations leads to a smooth representation. It is intended to extend the parametrization of line styles in order to control the randomness of the form and appearance of displayed lines.

6 System Design

We use VisualWorks, a Smalltalk environment, to develop dali. This enables us to build a modular system, in which components can be tested and even used independently. The animation engine receives different types of data:

- The 3D geometry of scene objects (including information about lights and cameras),
- additional information (hierarchy, importance, etc.) and
- the animation data which are separated into
  - animation settings (number of frames etc.) and
  - the movement of all objects described with keyframes.

The animation system can be split into two parts, namely the animation engine and the render engine (as shown in figure 2):

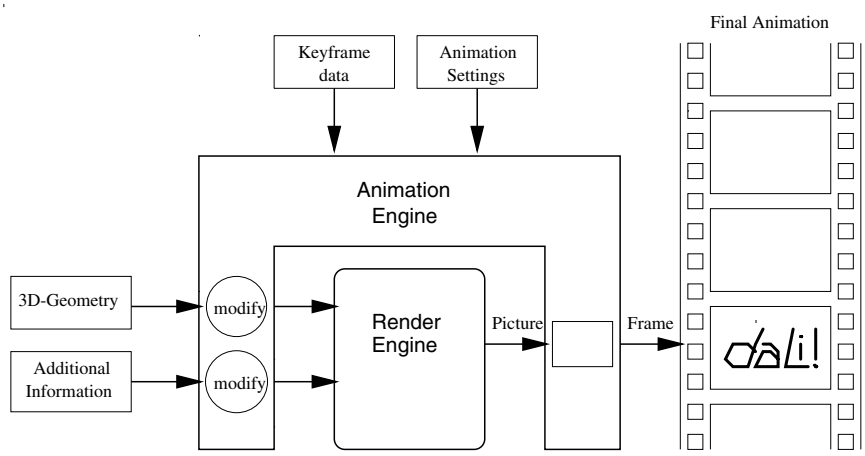

Figure 2: Overview of the animation system dali.

Because it is unnecessary to develop yet another modelling system, we simply use 3D Studio to set up an animation. 3D Studio is a high-end animation system for personal computers provided by Autodesk. Besides its native file format 3DS, it can import other formats (e.g. DXF) and therefore provides an open interface to other modelling systems. After setting up the scene—constructing all objects, setting up cameras and lightsources—the motion paths and object transformations are modelled in the 3D Studio keyframer. Finally

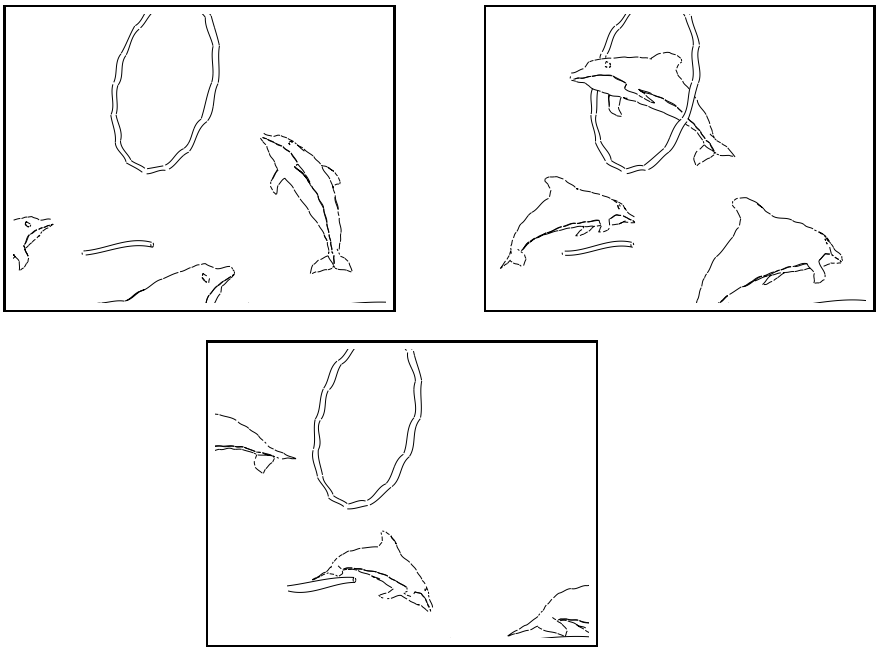

Figure 3: Jumping dolphins: Three frames taken from an animation

a 3DS file is written. This data (referred to in figure 2 as keyframe data) can be read and interpreted by a 3DS parser, which is part of the animation engine. The animation engine can therefore be seen as a filter that provides the render engine with the necessary animation data combined with the additional information. This information consists of the number of frames and settings concerning the line styles. The scene is rendered and an analytical description (technically speaking: a collection of lines) is passed back to the animation engine, which paints the vector oriented frame representation on an abstract image. After that a special output device writes subsequent frames into a predefined directory. Currently dali! supports *PostScript* as resolution independent output format and *TIFF*, *BMP*, *GIF* and *AnimatedGIF* as resolution dependent output formats. Finally, the series of images can be combined into an animation using standard MPEG or Quicktime encoders.

One can think of a scene as a collection of objects. These objects may have common and distinctive features. It is possible to model common features only once and allow them to be inherited by child objects. So, for instance, every scene object—no matter if it is a 3D geometry object, a camera or a lightsource—can be moved. Consequently this movement can be inherited by child objects. Thus the mechanisms implemented in the animation engine can be applied to every object in the scene.

Once having agreed on a common interface and protocol, dali! can be extended very easily just by developing new object-specific code for a certain action. An application for this might be the extension of dali! to a multi-style animation system which creates line drawn

animations as well as painterly or more realistic looking animations.
The use of object-oriented techniques offers a lot of advantages especially in the graphics field. Although the rendering time is a critical factor when using such an environment, we discovered that the advantages of an object-oriented development far outweigh the disadvantages. The rendering time strongly benefits from the implementation of the very efficient SECHREST/GREENBERG algorithm.
The images at the end give an example of an animation created with dalí. They show some frames out of a small 200 frame animation inspired by the logo of dalí.

7 Future Work

Although dalí has produced some fine results, there is still a lot of undiscovered country ahead. For instance, we would like to implement various line-drawing styles for texturing different surfaces. In addition, as the complete rendering is done analytically, the generation of shadows raises some new problems, that could easily be solved in a photorealistic context. This will be one of our tasks for the near future.
After coping with the more technically structured problems mentioned above, we aim to develop new techniques for the representation of many objects, e.g. leaves of a tree or clouds. In the past artists have developed some powerful techniques to depict these kinds of objects, e.g. painting a tree, with some characteristic strokes rather than painting every single leaf.
Finally, we think it can be promising to experiment with changes of object importance (visualized by the use of different line-drawing techniques) over time.

8 Conclusion

The animation of non-photorealistic computer graphics is a complete new field of graphical research. Therefore neither conventional photorealistic rendering techniques nor 2D animation techniques can be applied without essential modifications. The animation system dalí, as presented here, allows the user to create non-photorealistic animations in the style of line-drawings. Different line styles can be applied to an object or to parts of an object. The use of 3D Studio as an animation tool allows the user to benefit from the advantages of a high-end animation system. An object-oriented approach was chosen to implement the system. This choice lead to major improvements in the development as well as to a clear structure of the modules. dalí can therefore be easily extended.
With the inclusion of abstraction techniques it is possible to create animations which can visualize a lot more than those rendered photorealistically.

Acknowledgements

The authors wish to thank all their colleagues in Magdeburg for their support and helpful discussions; Tobias Isenberg for the implementation of the output module and Lars Schumann for his work on the line styles.

References

[FBC+95] J. Fekete, É. Bizouarn, É. Cournarie, T. Galas, and F. Taillefer. TicTacToon: A Paperless System for Professional 2d Animation. In *Computer Graphics*, Annual Conference Series, pages 79–89. ACM SIGGRAPH, ACM Press, 1995.

[HL94] S.C. Hsu and I.H.H. Lee. Drawing and Animation Using Skeletal Strokes. In *Computer Graphics*, Annual Conference Series, pages 109–118. ACM SIGGRAPH, ACM Press, 1994.

[Lei94] W. Leister. Computer Generated Copper Plates. *Computer Graphics Forum*, 13(1):69–77, 1994.

[LS95] J. Lansdown and S. Schofield. Expressive Rendering: A Review of Nonphotorealistic Techniques. *IEEE Computer Graphics and Applications*, 15(3):29–37, May 1995.

[Mei96] B. J. Meier. Painterly Rendering for Animation. In *Proceedings of SIGGRAPH'96 (New Orleans, Louisiana, August 4–9, 1996)*, Annual Conference Series, pages 477–484. ACM SIGGRAPH, ACM Press, 1996.

[SG81] S. Sechrest and D. P. Greenberg. A Visible Polygon Reconstruction Algorithm. In *Proceedings of SIGGRAPH'81 (Dallas, Texas, August 3–7, 1981)*, Annual Conference Series, pages 17—27. ACM SIGGRAPH, ACM Press, 1981.

[TJ81] F. Thomas and O. Johnston. *The Illusion of Life: Disney Animation*. Hyperion, 1981.

[WS94] G. Winkenbach and D. H. Salesin. Computer-Generated Pen-and-Ink Illustration. In *Proceedings of SIGGRAPH'94 (Orlando, Florida, July 24–29, 1994)*, Annual Conference Series, pages 91—100. ACM SIGGRAPH, ACM Press, 1994.

[WW92] A. Watt and M. Watt. *Advanced Animation and Rendering Techniques*. Addison-Wesley, 1992.

dal

dal

dal

dal

dal

dal