transnational University Limburg
School of Information Technology

"A 2.5D Modelling and Animation Framework Supporting Computer Assisted Traditional Animation"

Dissertation submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy: Computer Science
transnational University Limburg
June 21, 2004

Fabian DI FIORE

Supervisor: Prof. dr. F. Van Reeth

2004

transnationale Universiteit Limburg
School voor Informatietechnologie

"Een 2.5D Modelleer- en Animatieraamwerk ter Ondersteuning van Computer Geassisteerde Traditionele Animatie"

Proefschrift voorgelegd tot het behalen van de graad van
Doctor in de Wetenschappen, richting Informatica
aan de transnationale Universiteit Limburg
21 juni 2004

Fabian DI FIORE

Promotor: Prof. dr. F. Van Reeth

2004

"If we knew what it was we were doing, it would not be called research, would it?"

Albert Einstein

Acknowledgements

This dissertation would not have been possible without the contributions and support of many people. I consider myself fortunate to have worked with a number of truly remarkable people.

First and foremost, I wish to express my sincere gratitude to my supervisor prof. dr. Frank Van Reeth for his inexhaustible support, guidance, compassion, encouragement, and collaboration. Frank's endless creativity and vision have had a huge impact on how I approach the exciting world of computer assisted traditional animation.

I have learned much from all my other colleagues, who also became good friends throughout the years. I especially want to thank Koen Beets, Johan Claes, Joan De Boeck, Tom De Weyer, Marc Flerackers, Jan Fransens, Erik Hubo, Tom Mertens, Marc Ramaekers, Chris Raymaekers, Tom Van Laerhoven, William Van Haevre, and Gert Vansichem.

Without the help of the graphical and creative input of, especially Joan Cabot and Xemi Morales, and also Paul Akkermans, Liesbeth Beckers, Bjorn Geuns, and Luis Gutiérrez, our publications would have been much less lively.

I am very grateful to the EDM staff, in particular prof. dr. Eddy Flerackers, managing director, and also prof. dr. Frank Van Reeth, prof. dr. Philippe Bekaert, prof. dr. Karin Coninx, prof. dr. Wim Lamotte, and Peter Vandoren, for making it a very supportive and instructive place to work.

Many thanks go also to Johan Claes for the many valuable paper reviews, to Simi Varghese, and to ANDROME NV (ANDROME NV 04) for freely putting available to us their CreaToon® plugin SDK (CreaToon 04).

Furthermore, I want to thank the many reviewers of the different conferences for helping improving the quality; and the institutions funding our research, especially the EFRD (European Fund for Regional Development) and the Flemish Government. This funding further allowed me to travel to international conferences and personally meet other researchers.

Finally, I thank my parents, my family, and particularly my brilliant, Ruth, for their support, love and encouragement during this time of hard work.

Abstract

Traditionally, 2D animation production has been a labour-intensive artisan process of building up sequences of drawn images by hand which, when shown one after the other at a fixed rate, resemble a movement. Most work and hence time is spent on drawing, inking and colouring the individual animated characters for each of the frames.

Existing computer assisted animation already has a huge impact on traditional animation. However, the most time-consuming and labour-intensive aspect — in-betweening — is not well enough computerised that it could be (completely) performed by the computer and thus replacing traditional animation. Moreover, there is also a legitimate concern that the extensive computerisation will remove much of the creative aspects of the process.

This dissertation introduces a new method for automatic in-betweening in computer assisted traditional animation. The solution is based on novel 2.5D modelling and animation techniques within the context of a multi-level approach, starting with basic 2D drawing primitives at level 0, over explicit 2.5D modelling structures at level 1 and inclusion of 3D information at level 2, to high-level tools at level 3. Similar to the 3D animation process, our method also distinguishes clearly between a modelling and an animation phase. Next to automatically calculating in-between frames, we particularly focus on reintroducing necessary 3D information — which is only present in the animator's mind — in order to preserve volumes and proportions, and to prevent temporal aliasing. We also present new techniques and tools to draw, manipulate and animate (stylised brush) strokes, in order to preserve the natural way of drawing and editing, and to give the animator the same freedom of exaggeration to create animations as s/he is bearing in mind. Furthermore, we describe a novel approach to design artistic and believable trees in a cartoon-like style, and an extension of this approach to the more turbulent movements of a series of gaseous phenomena.

We believe that the provided solutions are easy to use, and empower a much quicker cartoon production without hampering the artists' creativity.

Contents

Chapter 1

Introduction

"Animation isn't an illusion of life. It is life."

Chuck Jones

Traditional animation is an art form. It is the process of creating a sequence of drawn images which, when shown one after the other at a fixed rate, resembles a lifelike movement. When we speak about 2D animation in the context of this dissertation, we refer to animations of which the background, objects (buildings, cars, ...) and characters (persons, animals, ...) are hand-drawn. The movements and orientations of the characters in the drawn world resemble real life (e.g., a squirrel moving behind or in front of a tree) but the characters themselves do not mimic reality exactly: animators do not want to reproduce real life but they tend to create animations that are recognisable by people. At the same time the animation can be playful, it can be a caricature or any other kind of artistic expression. Figure 1.1 shows some images of the type of characters we would like to animate. For example, in figure 1.1(a) the animation artist aims to express the cuteness of the dog by focussing the attention on the engaging smile. Also notice that the legs of the dog are not drawn anatomically correct, though they still are recognisable.

Besides animating living characters, we're also interested in background objects and natural phenomena such as trees, fire and smoke (figure 1.2).

Traditionally, 2D animation production has been a labour-intensive artisan process of building up animated sequences by hand. Most work and hence time is spent on drawing, inking and colouring of the individual animated characters for each of the frames. We refer the reader interested in a more detailed overview of hand-drawn animation production to (Whitaker 81; Blair 94; Patterson 94; Nettleship 95; Griffin 01; Williams 01).

Figure 1.1: Some typical animation characters. a) A cute traditional cartoon animation dog. b) A hairy cat (drawn in detail). c) A stylised character. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

Figure 1.2: Some scenery elements. a) Cartoon trees. b) Fire and smoke.

Within the boundaries of this dissertation, it is our goal to eliminate these time-consuming aspects of traditional animation, especially the repeated drawing of all characters for each frame. Furthermore, it is also our goal to give the animator the same freedom of exaggeration s/he is bearing in mind to create animations such as shown in figures 1.1 and 1.2.

The main contributions of our research in the field of computer assisted traditional animation are:

- a *multi-level 2.5D modelling and animation framework*;
- a system that assists the animator with an *automatic in-betweening* solution;
- the introduction of *approximate 3D objects* which help retaining volumes and proportions, and which ensure frame–to–frame coherence;
- a *multi-level sketching tool* that resembles the 'pencil-and-paper' approach which helps retaining the natural way of drawing and editing,

and offers additional functionality such as rapidly creating approximate 3D models and deforming animation objects;

- the introduction of *facial emotion channels* to assist a graphical artist throughout the creation of traditional facial animation;

- new techniques and tools to draw, manipulate and *animate stylised brush strokes* in order to avoid temporal aliasing artefacts such as brushes popping up in successive frames;

- a novel approach to design *artistic and believable trees* in a cartoon-like style, which can be rendered by an animated camera to produce a convincing 3D-like experience;

- and an extension of this tool to create *stylised animations of gaseous phenomena*.

The provided solutions should be especially beneficial in the production of traditional animation feature films and series.

In chapter 2, we start by giving an overview of 2D animation. We consider traditional animation from an artist's point of view, as well as existing computer assisted traditional animation techniques. Our attention particularly focuses on the existing limitations and fallacies. The chapter ends with our view about how traditional animation should be computerised in order to offer surplus values to the animator.

To solve these problems, we introduce a new method for automatic inbetweening in computer assisted traditional animation (chapter 3). The solution is based on novel 2.5D modelling and animation techniques within the context of a multi-level approach, starting with basic 2D drawing primitives (curves) at level 0, over explicit 2.5D modelling structures at level 1 and inclusion of 3D information by means of skeletons at level 2, to high-level tools (e.g., a deformation tool and possibly other tools for supporting specific purposes such as facial expression) at level 3. The underlying methodologies are explained and implementation results are presented.

So far, animators still have to make great efforts when interacting with existing computer assisted traditional animation software. Most drawings are built up from curves which only can be manipulated by pointing, clicking and dragging of control points which make up the curves. Therefore, we present a sketching tool (chapter 4) that assists the animator throughout multiple stages of the animation process. This tool helps retaining the natural way of drawing and editing, and supports additional higher-level functionality — not available in traditional animation — such as an easy-to-use free-form deformation tool.

In chapter 5 we focus on employing approximate 3D input models as a means to incorporate necessary 3D information. These approximate 3D models help retaining volumes and proportions, and ensure frame–to–frame coherence.

Chapter 6 concentrates on eliminating the time-consuming process of drawing all the emotions of a character, which have to be seen from different viewpoints. To establish these goals, we introduce the concept of *facial emotion channels*, of which each represents a facial part expressing an emotion. Furthermore, we present a novel approach through which an emotionally meaningful 2D facial expression from one point of view can be created from a reference expression in another point of view.

In chapter 7 we introduce new techniques and tools to draw, manipulate and animate stylised brush strokes in computer assisted animation production. We particularly focus on facilitating the tedious process of drawing and managing the numerous brushes that are painted on top of each other, and on avoiding temporal aliasing artefacts such as brushes popping up in successive frames.

A novel approach to design artistic and believable trees in a cartoon-like style, which can be rendered by an animated camera producing a convincing 3D-like experience, is explained in chapter 8.

Chapter 9 extends this approach to more turbulently moving animation, more particularly, stylised animations of gaseous phenomena with a strong emphasis on user control.

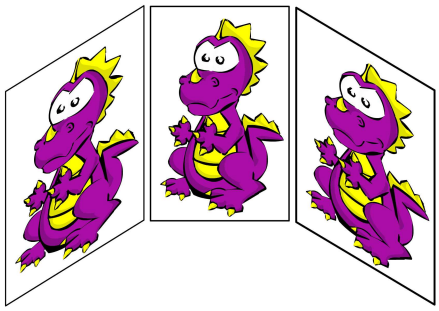
In chapter 10, we discuss future directions to extend our research, which is an ongoing process.

This is followed by our general conclusions in chapter 11.

This dissertation ends with a bibliography, lists of figures and listings, and a Dutch summary.

Chapter 2

Overview of 2D Animation



"Start by doing what's necessary, then what's possible and suddenly you are doing the impossible."

Saint Francis of Assisi

Contents

In this chapter, we give an overview of 2D animation. We consider traditional animation from an artist's point of view as well as existing computer assisted traditional animation techniques. At the end, we'll discuss our view about how animators *should* benefit from computerised traditional animation.

2.1 Traditional Animation from an Artist's Point of View

Broadly speaking, traditional animation is defined as a technique in which the illusion of movement is created by depicting a series of individual drawings on successive frames. Unlike live action, where the camera is running continuously, each frame of an animation film is shot one by one. Moreover, as characters are separated into several layers, each single frame might consist of numerous layers stacked up together.

The drawing process itself is done in three phases: (i) lead animators draw the most significant images, which are referred to as *extreme frames* or poses, containing the major features of the action; (ii) assistant animators produce *key frames* between the extreme frames, hence detailing the desired animation action; while (iii) less experienced animators (in-betweeners) are responsible for creating all the remaining *in-between frames* of the animation, resulting in a smooth sequence of drawings. Drawing these frames is known as 'in-betweening' or 'tweening', as animators are drawing the frames that go between the key frames.

We refer to (Blair 94; Patterson 94; Williams 01) for readers interested in an in-depth explanation, or to appendix A for a concise summary.

In the following sections we focus on two important aspects: timing and in-betweening.

2.1.1 The Importance of Faking and Timing

In traditional animation (Whitaker 81; Blair 94; Williams 01), realistic behaviour is not always required, but there's a need for fake, yet very impressive or dramatic effects. It is the animator's job to synthesise movements and to apply just the right amount of creative exaggeration to make the movements look natural within the cartoon medium.

Consequently, although the motion needs to look plausible, high-quality animation uses caricature methods such as squash and stretch, anticipation and exaggeration, slow in and out, and appeal in a non-physically motivated way to convey story, mood, and character; and to direct the audience's attention (Whitaker 81; Lasseter 87; Barzel 97).

Of course, the amount and type of caricature varies — from object to object, scene to scene, pose to pose, and frame to frame — to meet the needs of the moment.

2.1.2 The Importance of In-betweening

In this section we give a brief outline on creating in-betweens in traditional animation.

In-betweening is the process of generating all the frames of a motion sequence given its start and end frames. Basically the in-between technique works as follows: the (lead or assistant) animator specifies two key drawings of the object, one as it should appear in frame F_i and another as it should appear in frame F_j, $j > i$ (e.g., figure 2.1). Next, the in-betweener creates additional drawings between them to determine the appearance of the object in frames F_{i+1} to F_{j-1}.

Figure 2.1: Examples of key frames drawn by a lead animator. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

Usually the animator specifies the development of the movement between key frames in terms of the trajectories of specific reference points on the key frames. The trajectories indicate how those parts of the first key frame get to their final position in the second key frame. They are sometimes associated with marks or knots indicating the position of the reference points in the in-between frames, the number of knots being the same as the number of in-between drawings required.

The in-betweener has to fill in the rest of each in-between drawing, relying on general knowledge of the way in which the object depicted by the drawings is supposed to move. This is an artisanal process involving among other things timing charts, backlights, flipping, shifting, and rolling through the drawings.

This is explained into detail by Jon Hooper and Michel Gagne who give a step by step explanation of the traditional in-betweening proces (Hooper 04).

2.2 Computer Assisted Animation

The term "Computer Assisted Animation" suggests that computers bring something new to the traditional way of animating. Already in the 1960s xerography, a non-computer technology, replaced one stage in the traditional process: it eliminated the need for hand-tracing drawings to cells.

One decade later, two computer graphics techniques became well enough established and understood that they could start to be applied to the traditional animation process. Firstly, two-dimensional image processing and frame-buffer techniques became of sufficient quality so drawings or even live-action film could be digitised, manipulated on the computer, and then output directly to film. This presented the possibility for the latter stages of the animation process, such as ink-and-paint and camera work to be computerised. Secondly, the ability of the computer to render and animate rigid polygonal bodies from any camera angle became well-established.

This offered the potential for certain types of static scenery backgrounds to be completely generated on the computer, as well as certain types of foreground objects such as cars, spaceships, etc. These could either be output as pen-plotted line drawings for photographing on a traditional animation camera stand, or composited directly with digitised drawing on a computer paint system.

However, the most time-consuming and labour-intensive aspect — in-betweening — is not well enough computerised so it can start replacing traditional animation techniques. Furthermore, there is also a legitimate concern that the extensive computerisation will remove much of the creative aspects of the process.

In the following subsections we first give a brief overview of the current state-of-the-art of computer assisted animation: we consider the same important issues talked about in section 2.1, as well as the field of non-photorealistic rendering. Then, we end this chapter with our view (section 2.3) about how traditional animation should be computerised, while not tempering the animator's creativity.

2.2.1 Timing in Computer Assisted Animation

In a production environment, it is essential that animators can easily adjust and edit pose and timing with per-frame accuracy, to respond to the needs and feedback of the director or client. Dynamic simulation often turns out to be inappropriate — the control methods don't directly lend themselves to the non-physical motion and quick per-pose and per-frame editing that we aim for

as well as interaction speed becomes an issue.

As it turns out, *key frame animation systems* do suit these tasks quite well (Barzel 97). Of course, the key frame approach still requires talented animators: the models do not automatically generate animation; they simply provide shape controls for an animator to work with.

2.2.2 In-betweening in Computer Assisted Animation

In-betweening in Computer Assisted Animation (CAA) is a technique for calculating values that change systematically during an animation. Generally, it is known over which (key) frames in an animation an output value will change. So, in-betweening involves taking two (key) shapes (S_{start} and S_{end}) and calculating the in-between shapes, S_{inbetw_i}, which transform the shape from S_{start} to S_{end}. In fact, it is simply moving a point (or a series of points) from its initial position to a final position.

In CAA, there are essentially two types of in-betweening systems (Yu 99): *shape-based* and *skeleton-based*.

Shape-based In-betweening

Shape-based in-betweening techniques focus on *determining correspondences* between shapes (e.g., polygons) which consist of a different number of points (e.g., vertices).

Already in the early seventies, Burtnyk and Wein reported on the use of computers in the generation of key framed animation (Burtnyk 71). They proposed the most simple shape interpolation: linear interpolation. Linear interpolation calculates for each given start point, p_1, and end point, p_2, t intermediate points, p_i, at equal intervals along a straight line between p_1 and p_2.

Catmull (Catmull 78a) was among the first to discuss the issues underlying computer-assisted animation, indicating that the main problem is to be found in the lack of explicit 3D information in 2D hand-drawn cartoon pictures, making in particular the problem of in-betweening a hard task. Even for the 'simple' cases in which animation is parallel to the drawing canvas, no straightforward solutions exist(ed). For example, calculating in-betweens using a linear interpolation of Cartesian coordinates does not preserve shapes or proportions, as shown in figure 2.2.

Some of the problems with in-betweening have later been addressed by Reeves, utilising moving point constraints (Reeves 81). A moving point (MP) is a curve in space and time which constrains both the trajectory and dynamics (i.e. path and speed) of a point on the animated object. The animator can

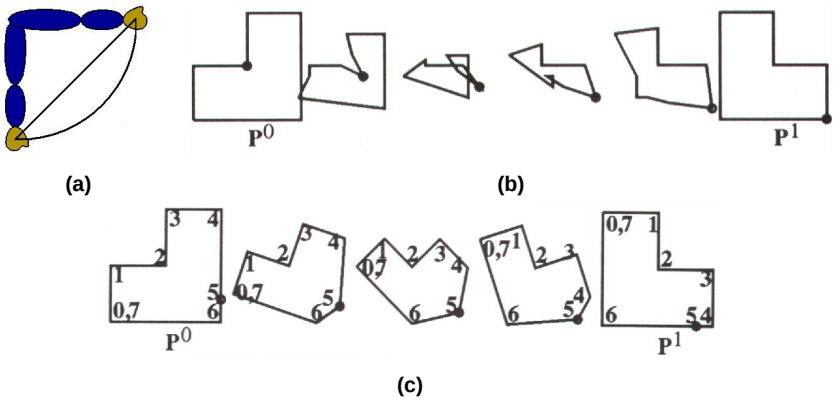


Figure 2.2: a) Rotating arm with the shoulder as pivot point. Using linear interpolation causes the hand to follow the path of the straight line instead of following the curved circle segment path. b) While interpolating, angles can converge to zero resulting in self-intersections. Copyright © 1992 ACM, Inc. (Sederberg 92). c) Fixed angles are not preserved during the interpolation process, leading to unwanted results. Copyright © 1992 ACM, Inc. (Sederberg 92).

specify as many MPs as necessary. Points on the key frames not directly constrained by a MP are constrained by a smooth blending of their neighbouring MPs. A major issue of Reeves's method is that a large number of curves are required to control complex image pairs.

Yu and Sederberg et al. propose the use of linear interpolation of local polar coordinates (PCLI) to circumvent the shrinking problem inherent in linear interpolation (Yu 90; Sederberg 93).

Other work on shape blending is presented in (Sederberg 92), (Goldstein 95) and (Ranjian 96). Sederberg et al. employ a special purpose interpolation scheme which interpolates the length of the edges of the polygon and the angles between them. Goldstein and Gotsman use a multi-resolution representation, in which the surface is represented at different levels of detail with the lower-resolution representation being smoother than the higher resolution version. At the lowest resolution the polygon is convex. Their approach leads to pleasing results for objects such as stick figures, where the relative thickness of the object parts has to be conserved during the morphing animation. Ranjan and Fournier use a representation of objects as union of circles to define a distance between two objects and to base a method to interpolate between the two.

The general problem of interpolating between two 2D shapes has also (and

still is) studied in the morphing community (Wolberg 98; Van den Bergh 02).

More recently, (Kort 02) presented a method for computer aided in-betweening. The content of each key drawing is analysed and classified into strokes, chains of strokes and relations that hold among them. Rules decide what parts of different drawings may be matched. Finally, generated animation paths between corresponding strokes determine the resulting in-betweens.

Skeleton-based In-betweening

The problems which are inherent in linear interpolation (figure 2.2) are mostly due to the lack of a central structure. After all, all vertices are interpolated independent of each other. In contrast, skeleton-based in-betweening techniques redefine all points (i.e. vertices) of the shapes (i.e. polygons) relative to other points or a central point, resulting in a skeletal representation. As a result, the shape is no longer defined as a set of individual vertices.

Through skeleton control, a significant increase in the capability for controlling motion dynamics in key frame animation is achieved. This technique allows an animator to develop a complex motion sequence by animating a stick figure representation of an image (Burtnyk 76). This control sequence is then used to drive an image sequence through the same movement. The simplicity of the stick figure image encourages a high level of interaction during the design stage. Its compatibility with the basic key frame animation technique permits skeleton control to be applied selectively to only those components of a composite image sequence that require enhancement.

Shapira and Rappoport (Shapira 95), use a star-skeleton representation. In this representation, the polygonal shapes are split into several star-shaped polygons, which are represented by the edge points and an extra star-point that is used to connect the different star-shaped polygons. Vertex positions are defined by (i) the distance to the star-point, and (ii) the angle between the reference direction (X-axis) of the star-point and the vector starting from the star-point to the vertex position itself. Since the vertices first are sorted by their angle, linear interpolation (which does not affect the sorting of the vertices) can be used without worrying about self-intersections. Figure 2.3 indicates the difference between shape-based and skeleton-based in-betweening.

In 1996, Yu et al. proposed a technique for object deformation using quaternions (Yu 96b). Their technique can be regarded as a 3D extension of PCLI (Sederberg 93) in which scaling and rotation operations are implemented using quaternions.

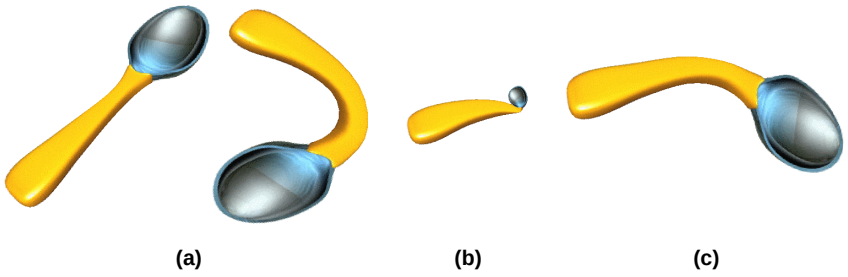


Figure 2.3: a) Two extreme frames of a spoon. b) Shape-based in-betweening (in this case linear interpolation) does not preserve the volume and introduces self-intersections. c) Skeleton-based in-betweening (in this case star-skeleton interpolation) does preserve the spoon's volume in spite of the rotation and the deformation.

2.2.3 Toon Rendering

We conclude this section by mentioning that 'Toon rendering', a subcategory in the non-photorealistic rendering (NPR) domain, also offers solutions to automating traditional animation.

The research about non-photorealistic rendering is diverging in multiple directions. Some people concentrate on merely improving the quality of 2D stand-alone drawings (Salisbury 96; Schlechtweg 96). Others incorporate real 3D input and render everything in more artistic ways (Winkenbach 94; Claes 97). One step further is animating this 3D non-photorealistic view of the world. Barbara Meier describes how 3D painted drawings can be animated (Meier 96). Markosian (Markosian 97) and Masuch (Masuch 98) successfully demonstrate the possibilities of rendering and animating 3D line drawings.

Philippe Decaudin was the first to specifically look at 3D cartoon rendering. In his Ph.D. dissertation (Decaudin 96) he extensively describes the particular problems and shows possible solutions, however in a far from real-time implementation. A much speedier approach is shown by (Lake 00), who is juggling with on-the-fly calculations of texture coordinates in a one-dimensional texture. Claes et al. used an approach similar to theirs to generate cartoon-style images and animations, and succeeded in speeding up the process while at the same time improving the quality of the images (Claes 01).

So, starting from 3D geometrical models, NPR techniques can generate possibly stylised cartoon renderings, depicting outlines with the correct distortions and occlusions.

However, two main drawbacks can be identified as far as traditional cartoon animation is concerned: (i) the approaches require extensive modelling and

animation of 3D characters and objects; and (ii) the final results are known to render the underlying 3D geometry 'too' accurate. In traditional animation, animators do not mimic reality exactly; instead they like to exaggerate it, putting emphasis on specific expressive details that cannot exist in the real 3D world.

For example, consider the case of showing the relative position of eyes on a head. Figure 2.4 shows images, obtained by using NPR techniques, of a cartoon man's head face-on (a) and sideways (b), and the same views as an animator is likely to draw it (c, d). We see that the side view (b) is geometrically correct (only the right eye is shown), contrary to (d) where the eyes are not drawn anatomically correct. However, the eyes in (d) might be more effective at expressing the sense of action that the animator would like to emphasise.

Recently, research has been done to bring 2D aspects into 3D animation. Rademacher presented a view-dependent model in which a 3D model changes shape based on the direction it is viewed from (Rademacher 99). The model consists of a base model and a description of the model's exact shape (key deformations) as seen from specific key viewpoints. Li et al. allow a traditional animator to modify frames in the rendered animation by redrawing the key features such as silhouette curves (Li 03). These changes are then integrated into the animation. These approaches, however, are not yet employable in actual practice.

An in-depth overview of published work in the NPR domain can be found in (Gooch 01; Strothotte 02; Reynolds 04).

2.3 Discussion: Computerising Traditional Animation

In the previous sections we discussed traditional animation as well as computer assisted (traditional) animation. We particularly focussed on the artistic and labour-intensive aspects: timing, faking and in-betweening.

There is a consensus among animators that existing computer assisted animation indeed has a huge impact on the time-consuming aspects of traditional animation. This is mainly due to the use of key framing systems (section 2.2.1) together with in-betweening techniques (section 2.2.2), and the availability of commercial packages (Animo 04; Moho 04).

However, in-betweening in traditional animation, is not just interpolating between key drawings. When drawing the in-betweens, animators utilise

- their background knowledge of the physical rules of the world;

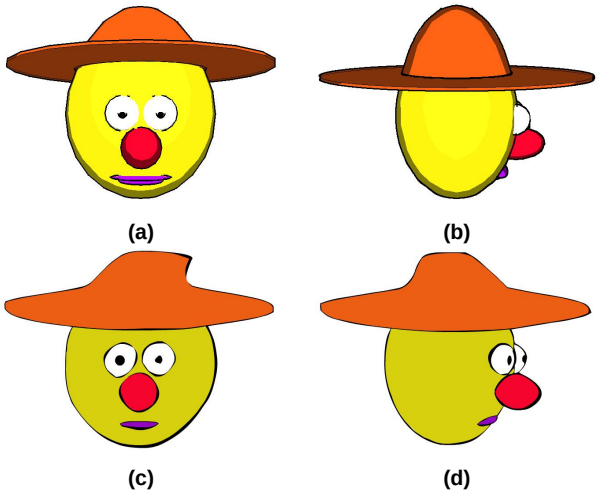


Figure 2.4: These pictures show (a) a man's head face-on and (b) a side-view obtained by NPR techniques and the same views (c, d) as an animator is likely to draw it.

- their expert knowledge of when to bend or ignore these rules, and
- the emotions they intend to evoke by the animation.

Moreover, there is a (legitimate) concern that the computer will remove much of the creative aspects of the process.

Therefore, we believe that studying and falling back on the traditional animation production process is very important for the development of new computer animation techniques. Moreover, to take benefit of the almost one century rich history of traditional animation, computer techniques can/should be very similar to and modelled after the traditional ones.

So, in our view, computer assisted animation *should* at least yield following advantages in order to persuade traditional animators to make the shift to CAA:

1. Savings in cost due to the significant reduction in the time required to generate good-quality in-betweens.
2. The computer should present the possibility for the animator to immediately see the finished results of his/her work in real-time. Whenever this would be impossible, an adequate approximation should be provided.
3. Any number of in-betweens may be created for a pair of key frames without any extra work.

4. The in-betweener should be dynamic enough to handle complex problems such as rotations toward and away from the camera. Of course, the in-betweening system has no understanding of what the object really is (an exception here is the artificial intelligence approach of (Kahn 77)), and hence correspondences may be indicated by the animator.

5. After colouring the first extreme frame, the in-betweener automatically colours the additional extreme frames and resulting in-betweens.

6. Computer assisted animation should offer surplus values, with respect to traditional animation, such as sophisticated drawing tools.

7. The in-betweener should provide simple and intuitive tools for editing the motion and timing of the animation.

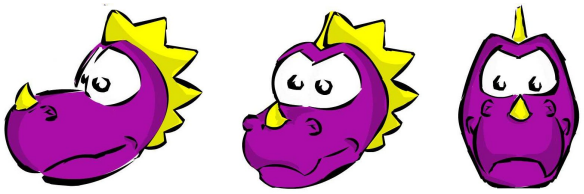
8. In-betweening should be more than "moving cut-outs". Hence, the in-betweener should combine highly intelligent shape interpolation with e.g. user defined arcs of motion to let an animator create in-betweens with 'life'.

In this dissertation, we take into account each of the above mentioned items throughout the following chapters. For example, items 1 to 4 and 7 are considered in chapter 3. Chapter 5 discusses amongst others item 5. Item 6 is tackled in chapters 4, 6 and 7, while item 8 is considered in chapters 8 and 9.

Of course, creating an animation involves more than this: e.g., squash and stretch, anticipation, follow through and overlapping action, slow in and out, exaggeration, appeal, ... We believe, however, this is entirely up to the animator and is regardless of the kind of animation (2D or 3D, traditional or computer assisted). It is important for the system to be flexible enough to allow this kind of control.

Chapter 3

2.5D Modelling and Animation Framework:
Automatic In-betweening in CAA



"Very few cartoons are broadcast live – it's a terrible strain on the animators' wrists."

David S. Cohen

Contents

This chapter introduces a new method for automatic in-betweening in computer assisted traditional animation[1]. The solution is based on novel 2.5D modelling and animation techniques within the context of a multi-level approach, starting with basic 2D drawing primitives (curves) at level 0, over explicit 2.5D modelling structures at level 1 and inclusion of 3D information by means of skeletons at level 2, to high-level deformation tools (and possibly other tools for supporting specific purposes such as facial expression) at level 3. The underlying methodologies are explained and implementation results are presented.

3.1 Introduction

When looking at the 3D computer animation pipeline (from a broad perspective), one can generally distinguish clearly between (i) a modelling stage, in which the 3D objects, characters and backgrounds are interactively modelled using 3D polygon meshes or curved surfaces, (ii) an animation stage, in which objects and characters are animated using a variety of animation tools, and (iii) a rendering stage, in which the final images of the animation sequence are being calculated frame by frame.

In traditional 2D animation (Blair 94; Patterson 94; Nettleship 95), the 'Ink and paint' process could somewhat be regarded as being the equivalent of the rendering stage in 3D animation, but the 'modelling' and 'animation' processes, however, are not explicitly present. They are combined into a single drawing process, which can be broken down into three sub-stages: (i) lead animators draw the most significant images, which are referred to as extreme frames or poses, containing the major features of the action; (ii) assistant animators produce key frames using the extreme frames, hence detailing the desired animation action; while (iii) less experienced animators are responsible for creating all the remaining in-between frames of the animation.

As will be clear from following subsections, we introduce a new method for automatic in-betweening in 2D. To this end, similar to the 3D animation process, we also distinguish clearly between a modelling and an animation phase.

This chapter is organised as follows. Section 3.2 summarises related approaches by which we gain a good insight in the problems inherent to automated in-betweening. Section 3.3 elaborates on our framework, while some

[1] An earlier version of this material was presented in (Di Fiore 01).

results are shown in section 3.4. We end this chapter with a discussion (section 3.5).

3.2 Related Approaches

Already in 1978, Catmull pointed out the issues underlying computer-assisted animation (Catmull 78a). The principle difficulty is that the animator's drawings are two dimensional projections of three dimensional characters as appearing in the animator's mind, making in particular the problem of in-betweening a hard task.

An early (1991) experimental 2D key frame animation system (INKWELL) is discussed in (Litwinowicz 91). INKWELL focuses on three goals: (i) to achieve an easy and intuitive user interface; (ii) to be able to produce character animation; and (iii) the flexible re-use of previously defined motion sequences. Litwinowicz succeeded in achieving these aims, but animations could suffer from occlusion problems. The author acknowledges this is due to the use of a strictly prioritised drawing order in his solution, but he did not indicate any further research to prevent this.

A few years later, Patterson et al. re-addressed the issues in the '78 paper of Catmull, confirming automating in-betweening to be the key problem in 2D animation since it has to happen consistently with our 3D intuition of how the drawings should change (Patterson 94). To be more precise, 2D lacks necessary 3D information. Essentially, this breaks down into two sub-problems: (i) how silhouette outlines change and (ii) how the various parts of the object occlude themselves.

According to Patterson et al., self-occlusion is the most serious obstacle to any purely 2D, or drawing-based, automatic in-betweening strategy. In many cases it can be dealt with simply by separating the objects into layers or a hierarchy display model (HDM), as is done in traditional animation (section A.4). For the same character, in theory many HDMs could be created in which each HDM consists of the same number of subparts, but has a different drawing order and different shape of the subparts (and hence silhouette). In fact, each HDM corresponds to a specific view of the character.

Consequently, a collection of HDMs for the same character forms the main set of entries of an electronic model sheet for that character and, in effect, substitutes for a true 3D model from which HDMs could otherwise be deduced. Using model sheets like this is usually referred to as *2.5D modelling* — not 2D but not really 3D either. In fact, a collection of HDMs essentially reintroduces some 3D information to the 2D drawings.

The work presented in this chapter focuses on detailing some of the issues addressed in (Patterson 94): (i) creating in-betweens and coping with (ii) self-occlusions and (iii) changing silhouettes. As Patterson et al. suggest, we do this by exploiting explicit 2.5D modelling of hierarchical models (HDMs) in a multi-level animation architecture. As will be clear from the following chapters, in this dissertation 2.xD modelling would be a more appropriate definition as the amount of necessary and, hence, reintroduced 3D information varies from animation to animation. However, for convenience, we'll use the term 2.5D modelling throughout this text.

3.3 Multi-Level 2.5D Modelling and Animation

Considering 2D animation from a technical standpoint, two distinctly different categories can be distinguished: (I) transformations in a plane parallel to the drawing canvas (the X-Y plane), such as rotations around the Z-axis and translations within a plane parallel to the X-Y plane, and (II) transformations outside the drawing canvas, especially all rotations around an axis different from the Z-axis.

The former category of transformations is relatively easy to deal with, whereas the latter is the main cause of all the trouble in automating the in-betweening process (i.e. the underlying sub-problems of silhouette changes and self-occlusion). It is in the latter type of animation where the 3D structure comes into play that is underlying the objects and characters in traditional animation (and which is present in the animator's — and viewer's — mind), but which is not present in the 2D drawings.

The 2D animation systems referenced in (Litwinowicz 91), (Coninx 96) and (Van Reeth 96) are only capable of animating characters and objects in category (I): they essentially narrow down to animating hierarchies of layered 2D sub-shapes within an overall layered 2D world space or a true 3D world space; all transformations take place in planes parallel to the drawing plane and the drawing order of the 2D sub-shapes is essentially fixed in time.

A true 2.5D animation system, however, should also be capable of animating characters in category (II). Hence, we present a multi-level 2.5D modelling and animation approach:

- level 0 holds the basic building primitives, being sets of attributed 2D curves (section 3.3.1),
- level 1 manages and processes explicit 2.5D modelling information (sections 3.3.2 and 3.3.3),
- level 2 incorporates 3D information by means of 3D skeletons (section 3.3.4), and

- level 3 is more open-ended and introduces higher level tools such as non-affine deformations (section 3.3.5).

As will become apparent from the subsequent subsections, our 2.5D methodology clearly distinguishes a modelling phase and an animation phase (hence following 3D computer animation in this respect).

3.3.1 Level 0: Basic Building Primitives

For research purposes, we decided to utilise subdivision curves to represent sub-objects at level 0 in our solution. We refer the reader to (Zorin 00) for an in-depth overview of the state of the art in subdivision technology. In our case we opted for subdivision curves with an additional support of normal interpolation and local tension control around control points. However, other curve types such as Bézier curves or NURBS (Oddy 92) could have been used just as well. This allows us to use only a limited number of control points to fully control a subject with an irregular outline (cf. figure 3.1).

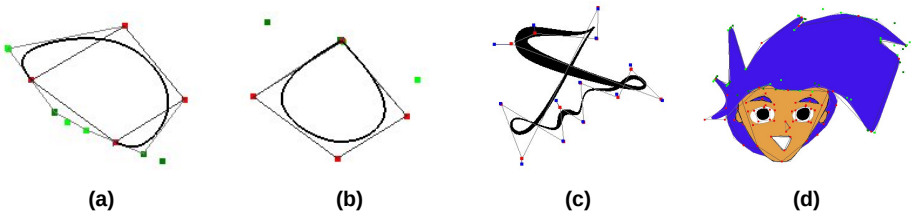


Figure 3.1: Basic building primitives. (a) shows a open subdivision curve that has two interpolating control points, (b) shows the ability of tension control, while (c) depicts a curve with varying line thickness. These kinds of curves are used to model the character of figure (d).

Each curve can be closed or opened, can be coloured (outlines as well as internally) and can have a varying line thickness.
We make use of these curves to build our HDMs. This is explained in next chapter.

3.3.2 Level 1: Explicit 2.5D Modelling Information

Level 1 is fundamental in the realisation of category (II) functionality. The basic notion can best be understood by observing the fact that especially rotations of (sub-objects) around the X- and Y-axis need special attention. Referring to the character depicted in figure 3.2(a), one can easily see that rotations around the Z-axis would not cause any problems: the outlines and

silhouettes of the character's head as well as the drawing order of the various sub-objects would remain fixed with respect to each other. When rotating the character around the vertical (Y-)axis (coming out of the drawing canvas), however, one can clearly notice (figure 3.2(b–c)) heavy changes in outlines and silhouettes as well as changes in drawing order (and when rotating further than indicated in figure 3.2(c), even more drawing order changes would be needed, as all the facial parts would disappear behind the head itself).

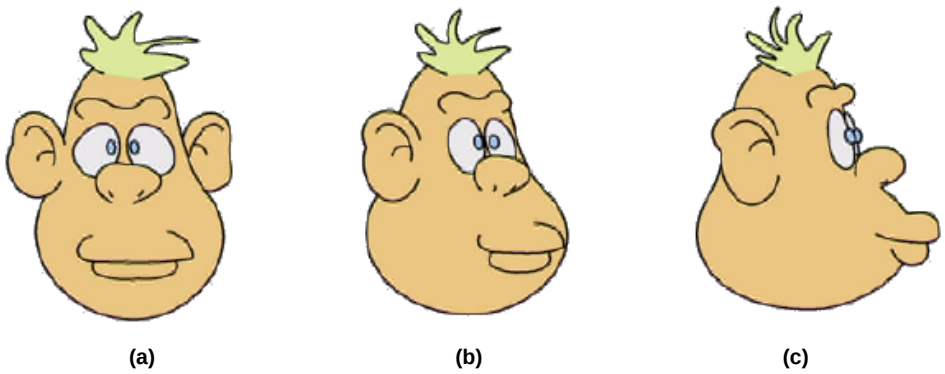


Figure 3.2: These pictures show (a) a man's head face-on, (b) a 45 degree-view, and c) a side view as an animator is likely to draw it. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

Hence, we propose a 2.5D modelling structure in which characters and objects are modelled as sets of depth ordered primitives (cf. section 3.3.1) with respect to the X-axis and Y-axis rotations. For each set of 'important' X-Y-rotations of the object / character relative to the virtual camera, the animator draws a hierarchical set of ordered curve primitives (i.e. HDM), functionally comparable to the extreme frames in traditional animation (cf. section 3.1), and which we therefore also call an 'extreme frame'. Likewise, the set of all these extreme poses is comparable to the model sheet database in traditional animation (cf. section A.1).

Internally, the supporting data structure consists of a set of channels. Each channel narrows down to an infrastructure in which the animators can introduce 'world knowledge' into the various subparts of the characters to be animated. Two standard channels are always put into action: one to store the Y-rotations of the object and another to store the X-rotations. For each animation channel, the data structure is a straightforward list, indexed by a specific parameter (e.g., a rotation angle in the case of a channel representing X-rotations). Our system also allows the animator to introduce custom

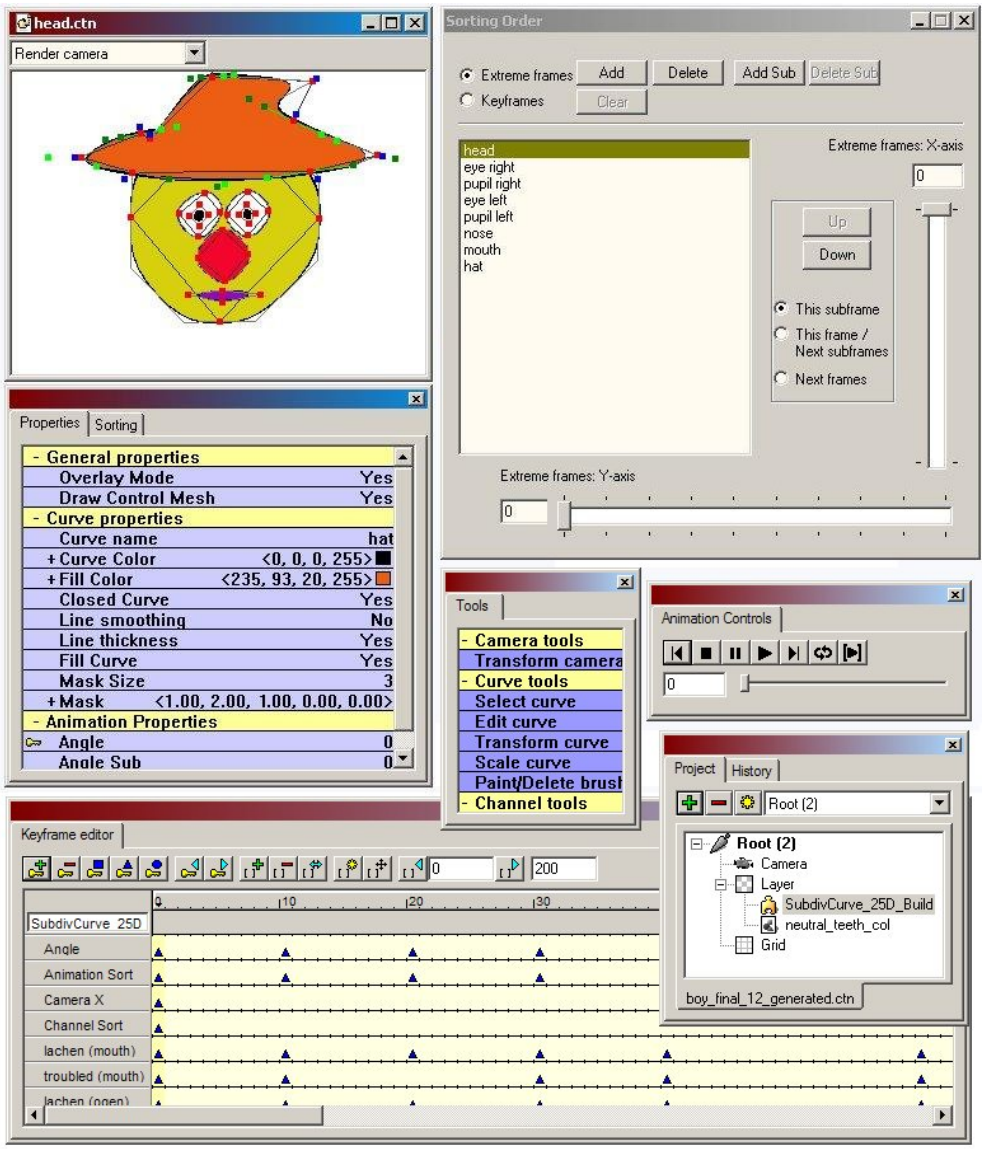


Figure 3.3: A snapshot of some UI components supporting level 1 functionality. Copyright © 2004 ANDROME NV (ANDROME NV 04).

channels, such as channels representing various expressions (e.g., chapter 6). Consequently, in addition to references to the underlying curve primitives, each extreme frame also holds an index parameter for the defined channels. Figure 3.3 shows the UI components supporting level 1 functionality. The rendering window shows a graphical rendering of the depth-ordered curves. For visualisation purposes the curves are shown with their control points. For each extreme frame, the 'Sorting Order' window depicts the drawing order of the sub-objects at issue. New extreme frames are entered by selecting new rotational angles (using the X-axis and Y-axis slider bars), drawing the respective curves of the sub-objects and setting their relative drawing order. In order to simplify the correspondence problem when interpolating between various instances of a particular curve, the curve drawing is currently implemented as a 'curve copy and re-edit'-tool.

Figure 3.4 shows four example extreme frames, which are created for a 2.5D house object. The number of extreme frames to be produced for a given 2.5D animation object/character obviously depends upon its complexity and the range of angles it traverses with respect to the animation camera. In this example case, the four extreme frames suffice to automatically generate each frame of an animation in which the virtual camera makes a vertical angle (relative to the house) between 0 and 30 degrees, and a relative horizontal angle between 0 and 45 degrees.

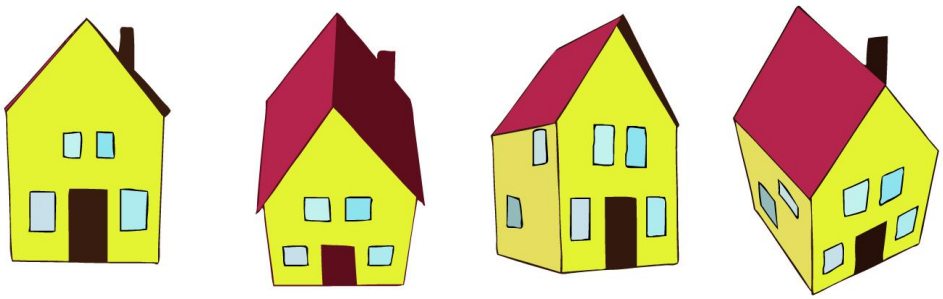


Figure 3.4: These are the four extreme frames created by the animator. In each case the same house is drawn but as if the virtual camera is rotated some degrees around the vertical (Y) or horizontal (X) axis. (a) $Y = 0$, $X = 0$; (b) $Y = 0$, $X = 45$; (c) $Y = 30$, $X = 0$; (d) $Y = 30$, $X = 45$.

For a representative background object or non-deformable animation object, about eight extreme frames are typically drawn to fully cover all rotational angles around the Y-axis, with some additional extreme frames for rotational

angles around the X-axis. This seems potentially to be a large number of extreme frames, but one should bear in mind that the traditional (hand drawing) animator not only has to deal with (=draw) these extreme frames, but also the key frames and the in-between frames, with little options to re-use material. In our automated approach the manual 'drawing' stays limited to modelling of the extreme frames.

3.3.3 Level 1: 2.5D Animation

Once the extreme frames are generated, it becomes possible to automatically render snapshots of the 2.5D objects within the range covered by the extreme frames. In order to create animated sequences, the animator specifies key frames in time (for one or more channels), e.g., for positioning and orienting a 2.5D object or for positioning and orienting the virtual camera in the (3D) animation world. Thus, key framing determines the timing of the animation sequence (figure 3.5).

Rendering such a key frame — and indeed also an in-between frame — narrows down technically to utilising the position and orientation of a 2.5D object to calculate the relative rotational angles (vertical and horizontal) with respect to the virtual camera. These angles are then used to search the surrounding extreme frames with respect to the horizontal and vertical angle. This can yield (i) two extreme frames, in case only extreme frames with relative orientations around a single axis (mostly the vertical) are present, (ii) four extreme frames in case multiple extreme frames with relative orientations around both axes are available, or (iii) three extreme frames in case extreme frames with relative orientations around a single axis (mostly the vertical) are specified aside a single additional extreme frame around the other axis. The control points of the drawing primitives (curves) in level 0, referenced by the selected extreme frames, are interpolated linearly but can also be interpolated using higher order interpolation schemes such as proposed in (Kochanek 84) and section 2.2.2.

The drawing order specified/stored in the selected extreme frames is utilised to determine the drawing order of the interpolated curves at issue.

The general problem of interpolating between two 2D shapes is not trivial and has also been (and still is) studied in the morphing community (Wolberg 98). The correspondence problem is very important in object-space morphing and some useful approaches may be found there. Sederberg and Greenwood interpolate the length of the edges of the polygon and the angles between them (Sederberg 92), while Alexa et al. use Delauney triangulated polygons (Alexa 00), and instead of the outline, transform the triangles of the resulting

mesh. Some researchers (Carmel 97) propose an algorithm to automatically establish the correspondence. Others (Shapira 95; Johan 00) on the contrary claim that the user can best specify the correspondence manually. We follow their advice and let the user specify the correspondence as the curve drawing is currently implemented as a 'curve copy and re-edit'-tool.

Figure 3.5 shows some key frames of an animation sequence automatically generated using the four extreme frames in figure 3.4 and the position of the virtual camera as specified in time be the animator. The key frames specify the timing of a virtual camera movement (with view point in the centre of the house) from sideways in front, upwards to up-front and then down again to end in front of the house.

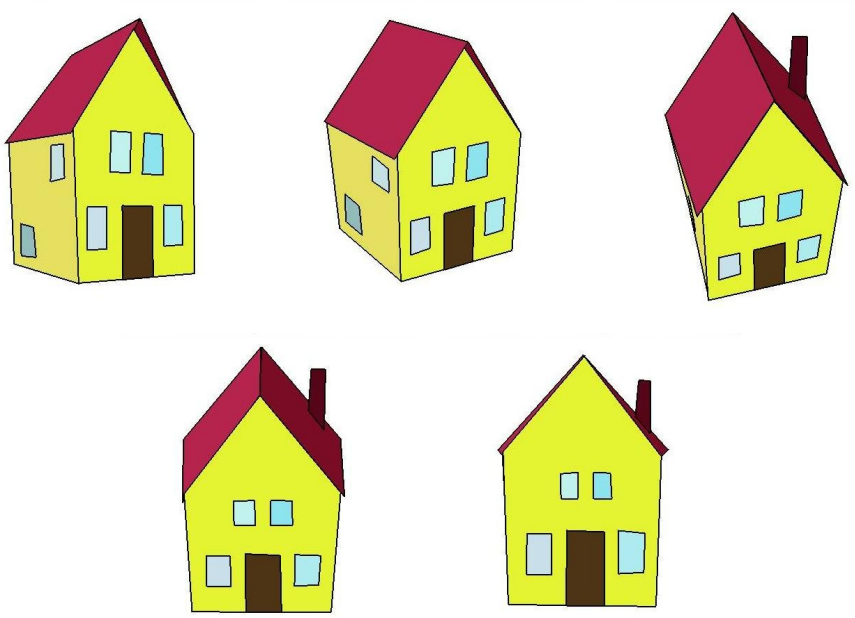


Figure 3.5: Key frames of an animation sequence using the extreme frames in figure 3.4.

Our approach of animation covers both *pose-to-pose* and *straight-ahead* animation (Blair 94). Drawing or setting up key poses and then drawing or creating in-between images is referred to as pose-to-pose animation. This is the basic computer key frame approach to animation and is excellent for tweaking, timing, and planning out the animation ahead of time. In our approach, pose-to-pose animation starts with developing/planning the extreme poses of the

characters in the 2.5D modelling phase, over specifying key frames in the 2.5D animation phase, to automatic in-betweening.

In straight-ahead animation the animator draws or sets up objects one frame at a time in sequential order until the sequence is complete. In this way there is one drawing or image per frame that the animator has setup. This approach tends to yield a more creative and fresh look but can be difficult to time correctly, and tweak. One example of 'straight ahead' animation is where the fingers of a hand tap impatiently on a desk. It is virtually impossible to 'key' and 'in-between' such an action, therefore it needs to be animated frame-by-frame. In our approach straight-ahead animation is supported in the 2.5D modelling phase by drawing extreme frames using 'curve copy and re-edit' functionality, and level 3 functionality such as the deformation tool described in chapter 7.

3.3.4 Level 2: Incorporating 3D Information

For animating scenery/décor elements (such as houses) and non-deformable animated objects only undergoing affine transformations (such as cars or airplanes), level 1 functionality could suffice. However, for animating lively characters consisting of many sub-parts and protruding limbs, additional support — 'more 3D' — is needed. In our solution this corresponds to level 2 functionality.

On this level, the notion of 3D skeletons, often encountered in 3D computer animation, is introduced. A skeleton is a hierarchical structure consisting of connected bones. Each sub-object of a given 2.5D animation character (modelled as curves introduced at level 0) that is capable of being animated with respect to the other sub-objects of the character at issue, is attributed to a specific bone in the skeleton (we have an experimental set-up in which the attribution can be weighted across two connected bones, but this is not relevant to the main line of the story here). Technically, this implies that the control points of the underlying curves are defined in a local coordinate system, which is placed on the bone. By manipulating the bones of the skeleton, the attached curves of the corresponding sub-object will be affected. The basic categorisation introduced in section 3.3 is still valid in level 2: category (I) transformations are relatively easy, whereas category (II) transformations cause problems.

Indeed, rotation of a bone in a plane parallel to the drawing canvas (i.e. around the Z-axis) implies a similar rotation of the attributed control points — and hence curves — of the sub-object at issue. Category (II) transformations are handled broadly analogous to the level 1 solution, by exploiting a 2.5D

modelling structure in which extreme frames are defined. Two differences exist as compared to the level 1 approach: firstly, the modelling of curves at extreme frames does no longer need the explicit setting of X-axis and Y-axis angles, as these are now handled implicitly by orienting the bones in 3D space; and secondly, the explicit manual ordering of sub-objects is also no longer needed, as this ordering is done automatically using the 3D positional information of the bones to which the curves are attributed. In case the automatic ordering should not produce the desired result, e.g., due to coinciding or crossing (when the right hand moves behind the hip in figure 3.6) bones or when the animator wants some dramatic effect not adhering the bone-order, some additional book-keeping in the 2.5D data structure is used to fall back to the normal level 1 approach. Figure 3.6 shows a typical skeleton supporting composition and manipulation of a 2.5D animation character.

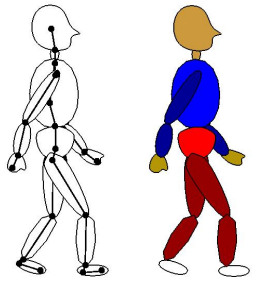


Figure 3.6: (a) shows a typical 3D skeleton supporting the composition and manipulation of a 2.5D animation character, whereas (b) depicts the same animation character in 'filled' mode using a sorting order derived automatically from the 3D skeleton bones.

The approach underlying the 2.5D animation creation in level 2 is also similar to the solution provided on level 1, and narrows down to

- modelling of the 3D skeleton and attributing sub-objects of the 2.5D animation character to a specific bone;
- specification of key frames by the animator, now by orienting skeleton bones in time;
- rendering of the key frames (and in-between frames), which again calculates relative rotational angles with respect to the virtual camera in order to find the appropriate extreme frames; and
- interpolating the curves at issue, now drawing them in an order directly derived from the 3D positional data of the bones.

Obviously, key frames could also be specified by exploiting inverse kinematics (Welman 93), other physics based approaches or incorporating existing 3D geometric models (e.g., chapter 8).

3.3.5 Level 3: High-Level Tools

Level 3 offers the opportunity to include in the proposed solution high-level tools on top of the level 2 (and also levels 1 and 0) functionality. Examples of these tools are high-level deformation tools and tools for specific purposes, such as facial animation. Here, we present a category (I) manipulation tool that works on control points across various sub-objects. Before manipulating the curves at issue, a group of control points is selected by drawing a selection lasso or rectangle over the part of the character one wants to change, similar as one would use in a bitmap editor (cf. figure 3.7(a)).

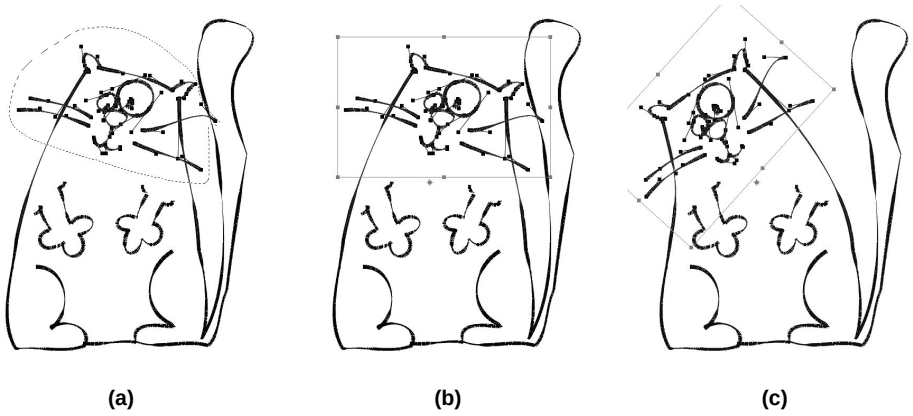


Figure 3.7: High-level transformation tool. a–b) Selecting. c) Transforming.

The control points within the selection will be marked as a group. The control points in figure 3.7(b) are shown to visualise the group for illustrative purposes. Like in bitmap editors, a tool for transforming a selection of the drawing can be defined. The transformation tool encapsulates a pivot point, a translate tool, a scale tool and a rotate tool. The advantage of transforming a group that consists of control points instead of a selection of pixels is that the curves will stay connected (cf. figure 3.7(c)).

When using the pivot point as an anchor for the group we can interconnect several groups. This allows the user to create a hierarchical model of a curve drawing based on the natural anatomy of the character. Such a model can be

used to control the movement of a cartoon character in a more suitable way with the aid of forward and inverse kinematics tools (this implies the direct use of the underlying 3D skeleton defined in level 2).

3.4 Results

In this section we show some results of our proposed solution.

The current implementation offers real-time displaying and editing of the results, maximising the comfort of the animator who wishes to adapt the animation to his/her artistic needs.

Figure 3.8 shows some snapshots of an animation of a running person, depicting the in-betweening process, including changing silhouettes and self-occlusions. Figure 3.9 shows some snapshots of an animation exploiting the

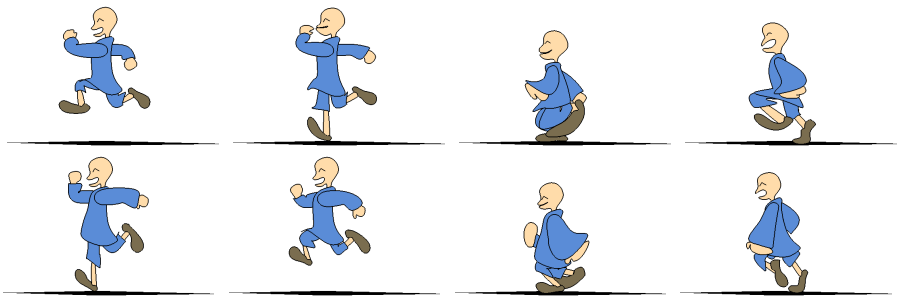


Figure 3.8: Some snapshots of an animation of a running person, depicting the in-betweening process. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

data presented in figure 3.4, acting as a representative of 2.5D animation of a typical background/décor element for which level 1 functionality suffices. Figure 3.11 shows some animation snapshots of an airplane, acting as a representative of 2.5D animation of non-deformable animated objects only undergoing affine transformations. Notice that the objects can be fairly irregular and may even contain holes. Also for this type of object, level 1 functionality suffices. We used the extreme frames shown in figure 3.10 for generating the sequence.

Figure 3.12 depicts the motion of a 2.5D rotating head using the extreme poses of figure 3.2.

Figure 3.13 consists of three images depicting some extreme frames of an animated hand, modelled as a skeleton based sub-object in level 2. Snapshots

Figure 3.9: Snapshots of an animation of a typical background/décor element.

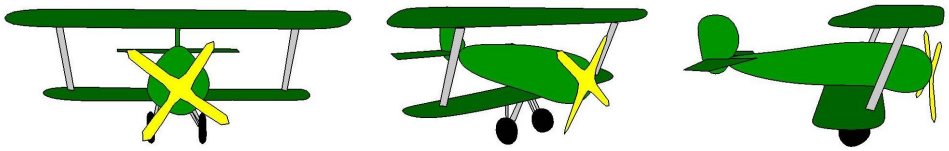


Figure 3.10: These pictures show the extreme frames of a 2.5D airplane.

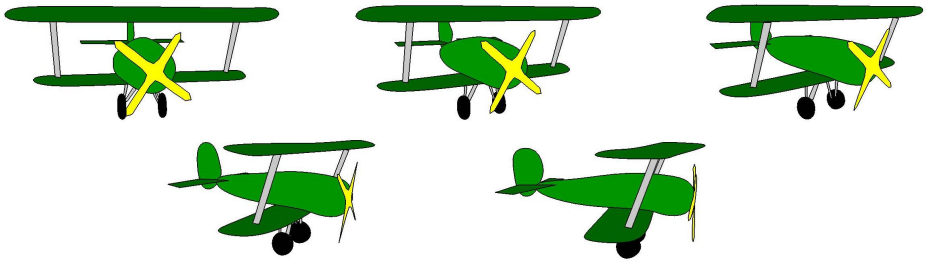


Figure 3.11: Snapshots of an animation of a typical animation object undergoing only affine transformations.

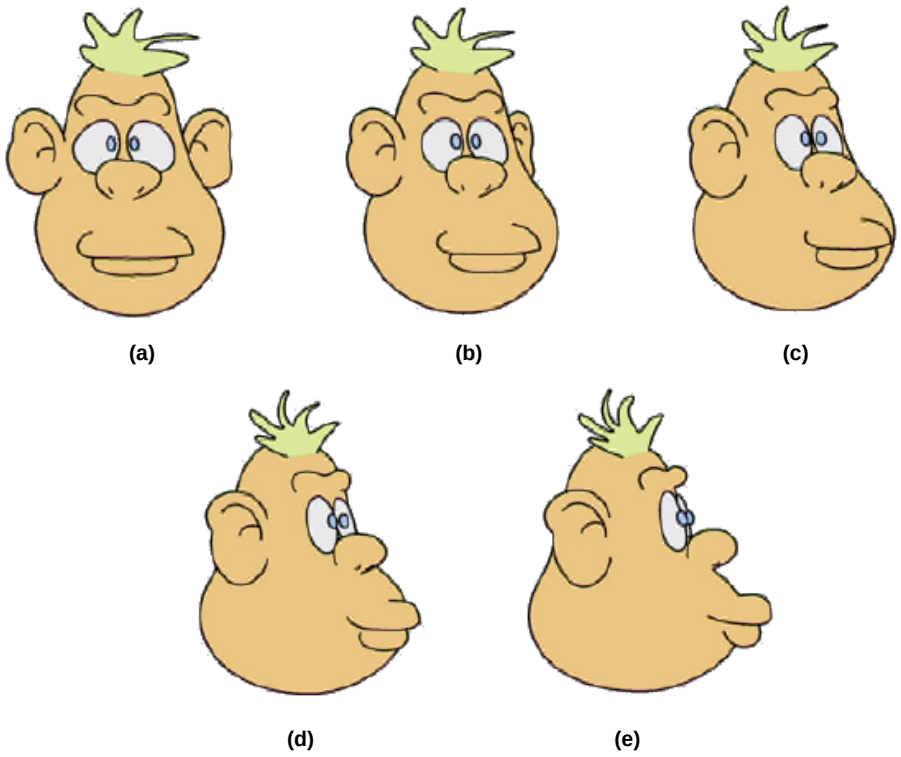


Figure 3.12: Mixture of extreme and in-between frames of a rotating head. a, c, e) Extreme frames. b, d) In-between frames. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

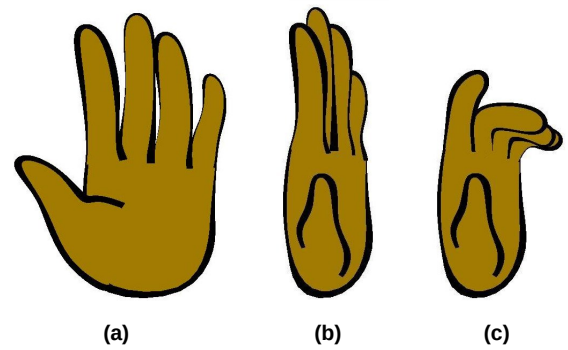


Figure 3.13: Extreme frames in a skeleton-based hand animation: (a) shows the hand at the start of the rotation while (b) depicts the end of it. (c) shows the same hand as after the rotation but of which all fingers are bend down somewhat.

of an animation sequence exploiting these extreme frames are shown in figure 3.14.

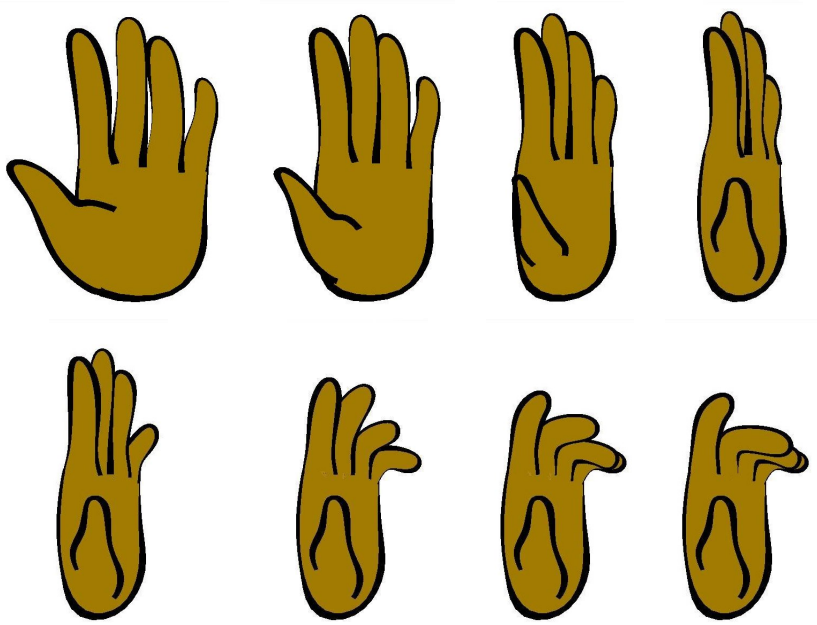


Figure 3.14: These pictures show some snapshots of an animation sequence generated from the extreme frames in figure 3.13.

We finish this section by stating that an animator can change at any time during a production the ordering and the shape of the underlying curves, which will implicitly generate a new extreme frame for the 2.5D animation object/character.

3.5 Discussion

In this chapter, we presented a novel approach to eliminate the time-consuming aspects of traditional animation, especially the individual drawing of all the characters in all separate frames, and at the same time retaining the freedom of an animator to express his/her artistic needs. We accomplished this by moving towards a computer-assisted animation process, in which automatic in-betweening is realised by means of a multi-level 2.5D modelling and animation solution. The underlying concepts and principles have been detailed and illustrative 2.5D animation examples have been given.

Further improvements of our framework include the introduction of new

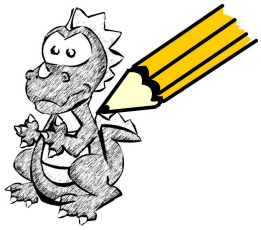
techniques and tools for drawing and manipulating stylised curves. It is also our goal to simplify the current way of modelling the basic building primitives (curves) of a character at level 0 of our solution. That is, we want to prevent the animator from explicit controlling the curves. This is handled in next chapter.

Another interesting level 2 extension is the incorporation of approximate (=heavily simplified) 3D models aside the 3D skeletons to support 2.5D model construction at level 0 and 1. This should aid animators amongst others in keeping the 'volume' of their 2D characters when animating them. This is discussed in chapter 5.

Further research on level 3 of our proposed solution, especially the inclusion of high-level tools for modelling and animation, is described in chapters 6, 8 and 9.

Chapter 4

"Pencil-and-Paper" Animation:
A Multi-level Sketching Tool



"Well, back to the old drawing board."

Peter Arno

Contents

In most existing vector drawing systems, the drawing primitives can only be manipulated by pointing, clicking and dragging of control points. In this chapter, we present a multi-level sketching tool that assists the animator throughout multiple stages of the animation process[1]. This tool helps retaining the natural way of drawing and editing, and is easily extendable to offer additional functionality — not available when using traditional drawing tools — such as performing high-level operations (e.g., a free-form deformation tool).

4.1 Introduction and Motivation

The conventional means for interacting with curve drawing and editing tools is through "point-click-and-drag"-procedures. Hence, creating and especially editing curve strokes is a tedious task, even for an experienced computer user.

Consider for example figure 4.1 in which an animation character is shown without(a) and with(b) all the control points of the curves that make up the character. It is obvious to see that due to the amount of control points an animator easily will get lost trying to figure out which control point to select in order to manipulate (a part of) a particular curve. Even when the animator succeeds in selecting the right control point, s/he finds it hard to find out how to manipulate the control point in order to achieve the desired result s/he is bearing in mind.

As a result, most animators find it difficult to make the shift from traditional animation towards computer assisted animation.

For this reason and also because computer animation is not only about drawing, we present a multi-level sketching tool that is designed to assist an animator throughout *various stages* of the animation process. That way it is our aim to provide the animator with an intuitive tool that can be used as if using the "pencil-and-paper" approach. We primarily focus on its use as a tool for creating, editing and transforming strokes which make up the drawn objects, but also on allowing additional functionality supporting animation.

The next section gives an overview of related work, while the following sections elaborate on our approach and present results. We end with our conclusions.

[1] An early version of this chapter was presented in (Di Fiore 02b).

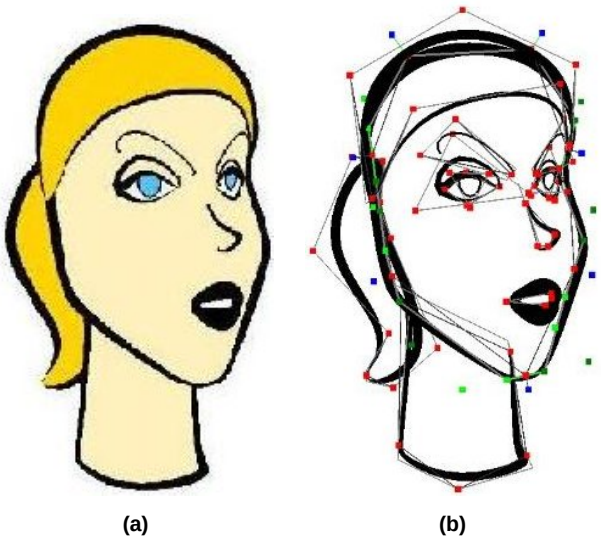


Figure 4.1: a) Example of an animation character. b) The character of (a) shown with all the control points of the underlying curves. Copyright © 2004 ANIMANTE (ANIMANTE 04).

4.2 Existing Sketching Tools

This section elaborates on existing sketching tools employed to (i) enhance the animator's freedom or (ii) to introduce new sophisticated tools.

4.2.1 The Animator's Freedom of Drawing

In recent years, free-form strokes have become quite popular as a means to draw 2D free-form objects.

Hsu et al. presented an animation system for producing 2D animations using skeletal strokes as the basic primitive (Hsu 93; Hsu 94). In their work, skeletal strokes can be regarded as a realisation of the brush and stroke metaphor using arbitrary pictures and its deformations as ink. Since any arbitrary picture can be defined to be a skeletal stroke, a huge variation in styles is possible. However, the animator still has to cope with a lot of additional work — providing input images and deforming images — in comparison with traditional animation.

More recently, Vansichem et al. proposed an approach for drawing and manipulating stylised curves without the need to explicitly manipulate (i.e. point, click and drag) control points of underlying splines (Vansichem 01).

Their approach generates curves on-the-fly by processing (pressure sensitive) stylus data at real-time. This simplifies the interaction drastically because the animator can exploit direct manipulation tools on the curves themselves.

We use a similar technique as described by (Vansichem 01) as this comes very close to the sketching method that traditional animators tend to use.

4.2.2 Sophisticated Tools

Free-form strokes are also being used for rapidly designing 3D free-form objects. Igarashi et al. present a gesture-based sketching interface (TEDDY) to quickly and easily design 3D free-form models out of 2D sketched strokes (Igarashi 99). In their system the designer first interactively draws 2D free-form strokes depicting a 2D silhouette. A 3D polygonal object is automatically constructed by inflating the region surrounding the silhouette, making wide areas fat and narrow ones thin. Oliver Bimber describes a system that extends this sketching interface by supporting dynamic gesture recognition (Bimber 99). A particular grammar extracts the required information on-the-fly.

As the features of our sketching tool include performing free-form deformations of objects, we briefly highlight some related work in the field. In 1986, Sederberg & Parry presented a technique for deforming solid 3D geometric models in a free-form manner (Sederberg 86). First, a local coordinate system is imposed on a parallelepiped region. Next, they impose a grid of control points on the parallelepiped region. That way the deformation is specified by movements of the control points. In fact, mathematically, the deformation function is defined in terms of a tensor product trivariate Bernstein polynomial in which the coefficients of the Bernstein polynomial are actually the control points of the grid.

Coquillart et al. proposed an interactive technique for animating deformable objects (Coquillart 91). They provide an alternative to the metamorphosis method that often is used to create intermediary shapes in order to make a smooth transition between two key shapes. This technique, *animated free-form deformation*, relies on the free-form deformation technique described in (Sederberg 86).

Our work also builds further on some core ideas of Igarashi et al. and Sederberg et al. such as the use of free-form strokes to draw the outlines, and an easy to use free-form deformation tool.

4.3 Implementation of a Multi-level Tool

In this section, we introduce a multi-level sketching tool as a helpful means assisting the animator throughout various stages of the animation process. We show how this tool can be used to (i) create and manipulate strokes which make up the final drawing, and (ii) perform high-level operations such as intuitively deforming animation characters. Furthermore, we'll give some directions regarding extending its functionality.

4.3.1 Free-form Strokes

When drawing on paper, artists have an almost unlimited drawing freedom. They can easily draw all kinds of stylish curves just by using a pencil, they can dynamically alter the width and the curvature of the curves and moreover; when they are not satisfied with the result, they can make (simple) corrections just by drawing new curves along or on top of the "wrong" ones.

Figure 4.2 shows a sketched drawing of a dog as a traditional animator is likely to draw it.

Our basic aim in this context is to support the animator in the creation and manipulation of the curves representing animation characters, without having to take recourse to "point-click-and-drag" metaphors.

Therefore, we introduce a sketching tool which allows for intuitive drawing of cartoon figures.

The creation of a curve is done interactively by sampling a (potentially pressure sensitive) stylus along the trail of the stroke. In order to allow for real-time high-level manipulations on the sketches, the individual pixels that make up the sketch are not used. Instead, internally a high-level representation is created using cubic Bézier curves, as is illustrated in figure 4.3.

A curve is created as follows. While sampling a (pressure sensitive)[2] stylus we simultaneously perform an iterative curve fitting technique based on least-squared-error estimation. Existing curve drawing solutions mostly take recourse to a "batch" curve fitting solution, in which the fitting is done after the stroke drawing is finished, whereas our fitting is done on-the-fly while the curve is being sketched.

Manipulations on the splines are actually performed on their control points, but this happens transparently for the user. The control points are never shown in the user interface. Figure 4.4 shows the dog of figure 4.2 drawn with our sketching tool.

[2]Depending on the exerted pressure, strokes are drawn brighter or darker.

Figure 4.2: A sketched drawing of a dog as an animator is likely to draw it. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

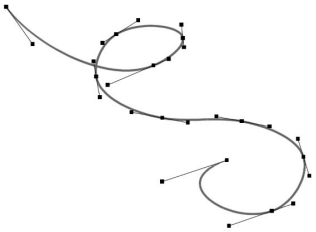


Figure 4.3: Sketch created with our sketching tool (shown with control points).

Aside the creation and editing of free-form strokes, we also include support for performing affine transformations upon (selections of) strokes.

Notice that the animator does not have to worry about picking, clicking and dragging control points associated with the underlying curve primitives. That way we preserve the same freedom of drawing an animator has when using the "pencil-and-paper" approach.

Figure 4.4: The same dog as in figure 4.2 drawn with our sketching tool. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

4.3.2 Support for High-level Operations

In the previous sections we showed how our sketching tool provides an effective means to assist the animator throughout the drawing stage when creating artwork.

In this section we explain the use of the same sketching tool as a means to perform high-level operations that are not available when working in the

traditional way. For demonstration purposes, we consider deforming in a free-form manner 2D objects consisting out of free-form strokes. So, in this context, it is our aim to indicate that an animator can deform objects without having to fall back on complicated deformation programs that are not intuitive to use at all.

Other possible high-level operations include:

- the easy and rapid construction of geometric models (e.g., *approximate 3D objects* discussed in chapter 5)
- support for sophisticated drawing tools (e.g., paint and air brush described in chapter 7)

As it was the case for free-form strokes, we differentiate between a lower level representation which is hidden for the user and which basically performs the actual deformation, and a higher level representation which allows the animator to do real-time intuitive manipulations.

The use of our technique is similar to the *polyline deformation* approach (Burtnyk 76; Parent 01). Firstly, the user sketches a *(source stroke)* over (part of) the object. This represents the current state of the object. Secondly, a *(deformation stroke)* is drawn, which can be seen as a deformation of the source stroke. The latter indicates the desired deformation of the object. This is illustrated in figure 4.5.

A major drawback of the standard polyline deformation method is that it lends itself only to globally deforming (mainly serpentine) objects. Therefore, in order to support both local and global deformations, we also put an evenly spaced 2D grid of control points G_{ij} on the object corresponding to the free-form deformation grid described in (Sederberg 86). Consequently, our system does not change the object's shape directly, instead, the 2D grid is deformed which in turn deforms the underlying object.

Our deformation algorithm is as follows (listing 4.1). The system first divides both strokes into N segments and maps these onto each other. For each segment s_n of the input stroke we also store the transformation t_n which is defined by transforming segment s_n of the source stroke to the corresponding segment d_n of the deformation stroke. Next, each grid point is attributed to the nearest segment of the source stroke. Then, for each segment s_n, all assigned grid points undergo the stored transformation t_n. Finally, for each control point of the underlying curves of the object, we calculate its deformed position by applying Sederberg's free-form deformation method using the changed grid (Sederberg 86). As a result the object gets deformed. All of this is hidden and happens transparently for the user.

Local deformations are supported by restricting the polyline deformation to the nearest 2D grid points only (e.g., within a distance D) (figure 4.6), whereas global deformations are achieved by subjecting the whole 2D grid to the polyline deformation (figure 4.7).

```
Deform Object( strokes s, d; grid G_ij; object O )
create N segments, s_n and d_n, n : 1..N
for each source segment s_n, do
  store transformation, t_n, to d_n
end for
for each grid point g_ij, do
  search for nearest source segment, s_n
  if (distance <= D) then
    attribute g_ij to s_n
end for
for each source segment s_n, do
  for each attributed grid point g_ij, do
    apply t_n to g_ij
  end for
end for
FFD( G_ij, O )
```

Listing 4.1: Overview of our deformation algorithm.

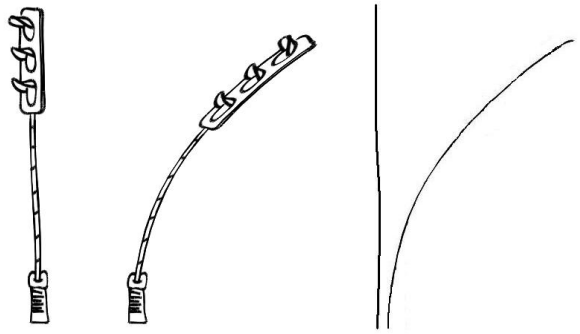


Figure 4.5: a) Original lamppost. b) Deformed lamppost. c) Source and deformation stroke.

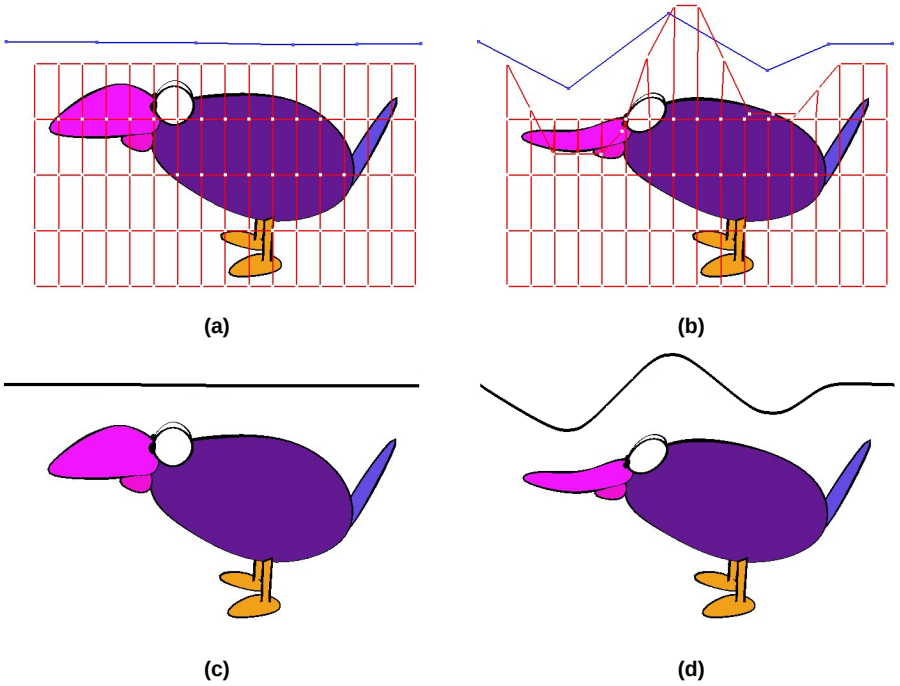


Figure 4.6: Example of local deformation. a–b) Original and deformed bird shown with deformation grid. c–d) Original and deformed bird as shown to the user.

4.4 Discussion

In this chapter, we presented the use of a multi-level sketching tool as a powerful tool to assist the animator throughout various stages of the drawing process when creating an animation.

In most existing vector drawing systems, the drawing primitives can only be manipulated by “point-click-and-drag”-procedures. This is a very difficult and tedious process and feels anything but natural. Together with the fact that these systems don’t offer a surplus value, it causes animators not to make the shift from traditional to computer assisted animation.

Our multi-level sketching tool can be used to overcome these problems. We successively described how to create drawings by using free-form strokes, and how to support high-level operations such as free-form deformations.

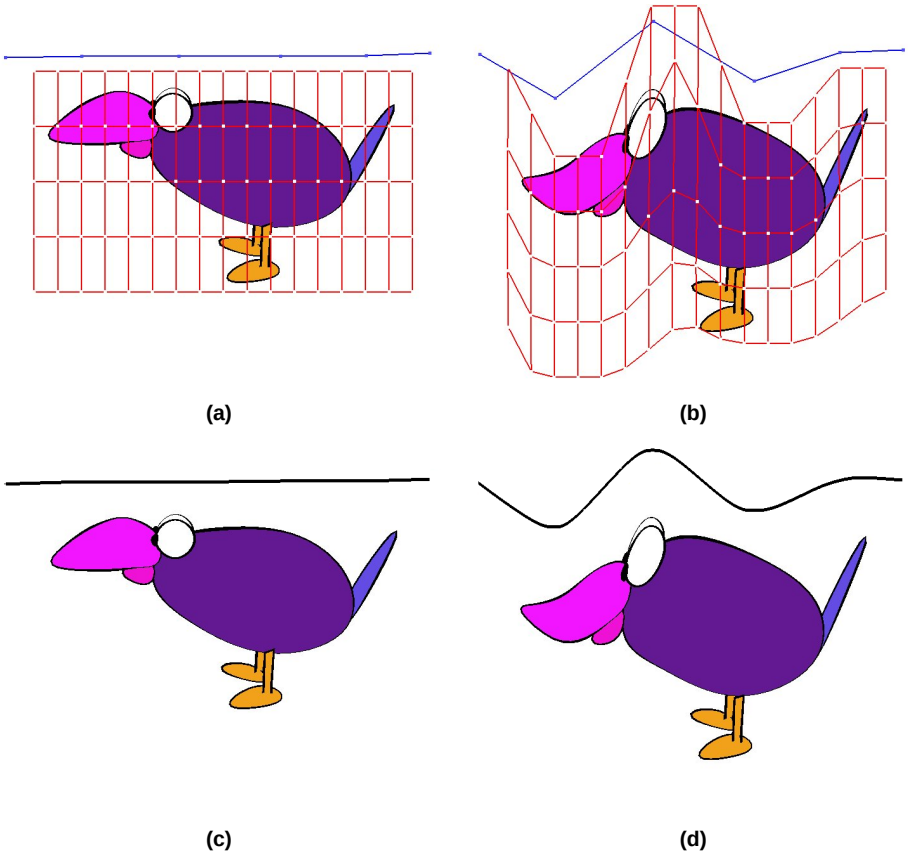
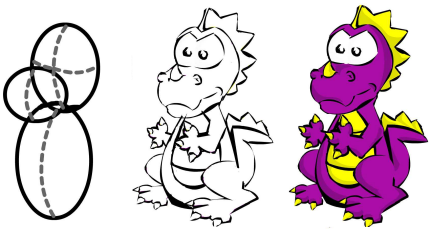


Figure 4.7: Example of global deformation. a–b) Original and deformed bird shown with deformation grid. c–d) Original and deformed bird as shown to the user.

Chapter 5

Enrichment of Traditional Computer Assisted Animation:

Employing Approximate 3D Models



"In character, in manner, in style, in all things, the supreme excellence is simplicity."

Henry Wadsworth Longfellow

Contents

This chapter focusses on how traditional CAA can be enriched in order to assist the animator during the 2.5D modelling phase[1]. We present a novel tool for traditional animation, based on approximate 3D models, which helps retaining volumes and proportions, and ensures frame-to-frame coherence when creating 2D animations.

5.1 Introduction

When bringing hand-drawn characters to life animators rely on world knowledge of the items they are creating. Without this knowledge (which is often partial or approximate), creating a satisfactory animation would be a nearly impossible task.

For example, when animating a walking dog, the relative sizes between the pieces which make up the animal should be retained (see figures 5.1(a–b)). More difficulties arise when the animator chooses to replace the simpler cartoon style of figure 5.1(a) to the more painterly style of figure 5.1(c). To retain the frame-to-frame coherence, the applied painted strokes may not suddenly appear and disappear, nor move or deform with respect to the object. Without such coherence, the *temporal aliasing* would make the final animation hard to enjoy.

Existing software to assist traditional animation lacks the 3D representation needed to tackle this kind of shortcomings. Therefore, we introduce a novel approach in which *approximate 3D models* are used to guide the animator throughout various stages of the animation process. We focus on its use as a tool for (i) depicting and retaining the volume and overall shape of the objects which make up the scene, (ii) rapidly inking the outlines by tracing silhouettes and marker lines of the objects, and (iii) preventing animations suffering from temporal aliasing.

This chapter is organised as follows. Section 5.2 gives an overview of related work and indicates the differences with our new approach. The central theme of this chapter, creation and use of approximate 3D objects, is elaborated in section 5.3, while section 5.4 presents the obtained results. We end with our conclusions (section 5.5).

[1] An earlier version of this material was presented in (Di Fiore 02a).

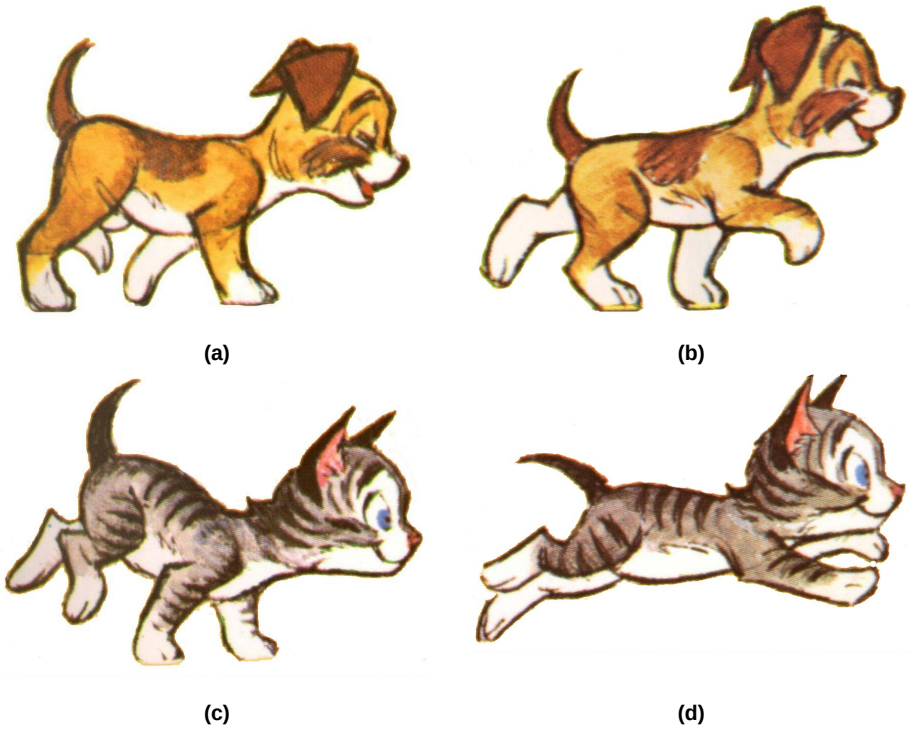


Figure 5.1: a–b) Two different key poses of a dog. In both versions the volume is the same. c–d) Two different poses of a cat. The stripes of the cat in figure (d) have to correspond with the stripes in figure (c) in order to avoid a "noisy" animation. Copyright © 1994 Preston Blair (Blair 94).

5.2 Painterly Rendering and Similar NPR Techniques to Enrich 2D Animation

In 1996, Barbara Meier presented an interesting solution to retain the overall volume of objects and to render animations in a specific artistic style without the need to draw each frame by hand and without loosing frame-to-frame coherence (Meier 96). She introduced methods to obtain a painterly render style starting from 3D geometrical objects. 2D brush stroke attributes are obtained from reference pictures and particles attached to the 3D model define the brush stroke locations. Meier managed to eliminate the "shower door" effect that disturbs many other NPR approaches to obtain a good frame-to-frame coherence. This approach leads to very impressive results for rigid objects. However, it requires extensive modelling and animation if it should

be applied to fully animated characters. In practice, also for rigid objects, a lot of hard work is involved to build up the models and to obtain and properly assign all necessary brushes.

Kowalski et al. presented a method to render a 3D scene in a stylised manner (Kowalski 99). They did this to suggest the complexity of the scene without explicitly representing it. Their method exploits procedural stroke based textures, *graftals*, to render the scene. These *graftals* place geometric elements procedurally into the scene to produce effects including fur, grass and trees. By using a difference image algorithm, the *graftals* are distributed over the 3D surfaces to achieve a desired screen-space density. Because these *graftals* stick to the 3D surfaces, this approach is suitable for interframe coherence. However, the drawbacks include that for each frame the *graftals* are regenerated and that each new *graftal* texture requires an additional procedural implementation.

Lee Markosian extended the work of Kowalski and presented a new framework addressing some of the above mentioned issues (Markosian 00). The framework also introduced the concept of *tufts* which manage the multiresolution behaviour of *graftals* according to specifications of the scene designer. This new framework addresses many of the problems encountered with the previous one but is considerably slower for complex scenes, and animators still have to create textual files manually to define looks and behaviours for *graftals*.

More recently (Isenberg 02; Kalnins 03), Isenberg et al. and Kalnins et al. describe ways to render stylised silhouettes of animated 3D models with temporal coherence. Li et al. allow a traditional animator to redraw the silhouette outlines in order to stylise the motion (Li 03).

A major issue in these systems based so much on full 3D models, is that they also generate a very pronounced 3D look, which we want to avoid. We want to allow the animator to change outlines in specific key frames and adapt them to the feeling and effects to help him reach his/her artistic goals. Furthermore, many details are extremely hard to construct in 3D, while it is much simpler to design very convincing look-alikes in 2D. For example, ask a designer to construct a fancy staircase in 3D or make an animation of a walking dinosaur and watch another artist draw a much more fancy 2D version during the time needed to start up the designer's favourite 3D software. This difference between 2D and 3D modelling is even more apparent when subtle animation effects (artistic expressions, caricatures, ...) are involved.

We refer the interested reader to (Gooch 01; Strothotte 02; Reynolds 04) who provide an extensive overview of published work, covering these and many other techniques employed in the non-photorealistic rendering domain.

5.3 Approximate 3D Models

In this section, we introduce approximate 3D models as a helpful tool assisting the animator throughout various stages of the drawing process. We successively show how these 3D objects can be used to (i) depict the volume of the objects for each extreme frame, (ii) form an idea of where the outlines should be drawn and how they look like, and (iii) providing for frame-to-frame coherence when rendering the animation in for example a painterly style.

5.3.1 Basic Shape Development

When animating, one of the very tedious factors to consider is to maintain the proportions of all the objects throughout the entire animation process. That is, at the very beginning the animator has to start with the development of the basic shape of the objects before going on with the development of features and other details (such as movement expressions).

In traditional animation (Blair 94; Patterson 94), when drawing the objects in different poses and actions (i.e. drawing the extreme frames), animators first make reference drawings (e.g., figure 5.2(a)), which contain proportion guidelines, of the separate objects. In a second step, the animator refers to these proportion guidelines to create very rough "sketches" of the extreme poses of the objects (figure 5.2(b)).

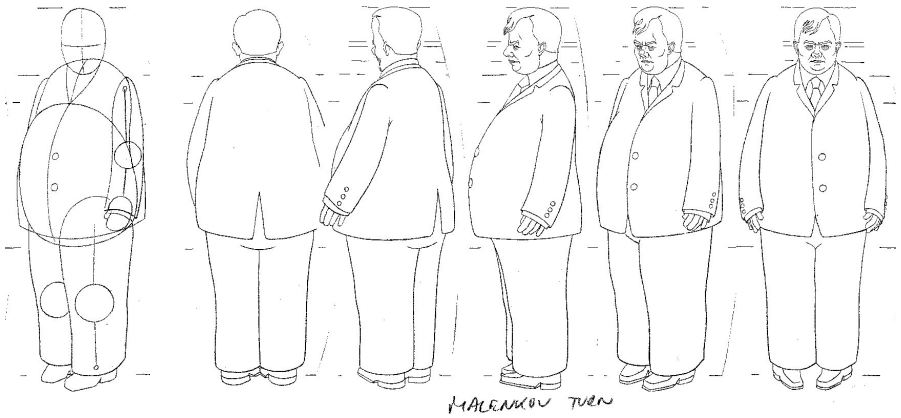


Figure 5.2: a) Proportion guidelines of a character. b) Rough "sketches" of the extreme poses of the same character. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

Usually, the reference drawings are only constructed from circular and

rounded "3D-ish" base forms (see figure 5.2(a)). The reason for using rounded forms instead of other shaped forms is because of their simplicity, and they are well understood when several animators work together on the same project.

It is clear that this traditional procedure is a cumbersome task and that any mistake causes major consequences for the whole animation. Only very experienced animators are natural at performing this task.

Therefore, we introduce the use of approximate 3D models to easily and rapidly create basic shapes of objects without the need to refer to reference drawings.

As simple 3D objects are utilised in our efforts to enrich computer assisted animation, we briefly highlight some related work for rapidly creating such objects.

Igarashi et al. present a gesture-based sketching interface (TEDDY) to quickly and easily design free-form models (Igarashi 99). In their system, the user first interactively draws a 2D silhouette consisting of several 2D free-form strokes. A 3D polygonal object is automatically constructed by inflating the region surrounding the silhouette, making wide areas fat and narrow areas thin. Bimber extends this sketching interface with support for dynamic gesture recognition (Bimber 99). A particular grammar enables on-the-fly extraction of the required information.

Our work shares parts of their core ideas, such as the use of free-form strokes to draw the outlines and working with simplified 3D models.

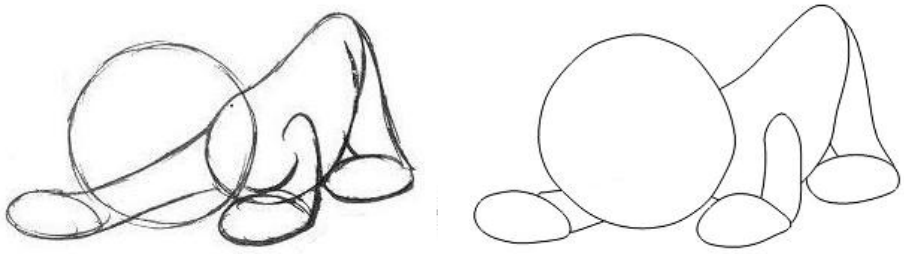


Figure 5.3: Sketched reference drawing. a) Copyright © 1994 Preston Blair (Blair 94). b) The same reference drawing as shown in (a) by using approximate 3D objects.

First, the user sketches (and possibly modifies) 2D circular and rounded forms as if drawing on paper. We use our sketching tool (chapter 4) to create

and alter these 2D objects, so the drawings consist out of a collection of free-form strokes.

Our system then interprets these circular and rounded forms to automatically construct a 3D polygonal object. This approach is considerably more simplified as compared to the methods described in (Igarashi 99; Bimber 99), but it is sufficient to support the easy and rapid construction of the plain approximate shapes that traditional animators tend to use.

For example, drawing an ellipse generates an elliptical 3D object. As you can see in figure 5.3(b), we only show the silhouettes (Claes 01) of the 3D objects as they are much more similar to the 2D rounded and circular forms than, for example, a shaded version of the 3D object.

We also included support for modifying the 3D objects and performing affine transformations upon them. That way, whenever the animator wants to create new extreme poses (extreme frames), s/he just has to perform basic transformations on a copy of the 3D objects without having to construct them again.

As one can see, the animator does not have to worry about retaining the volume of the objects. Also, the time spent on creating extreme frames is reduced drastically as the animator does not have to start over again every time s/he creates a new extreme frame.

5.3.2 Inking the Outlines

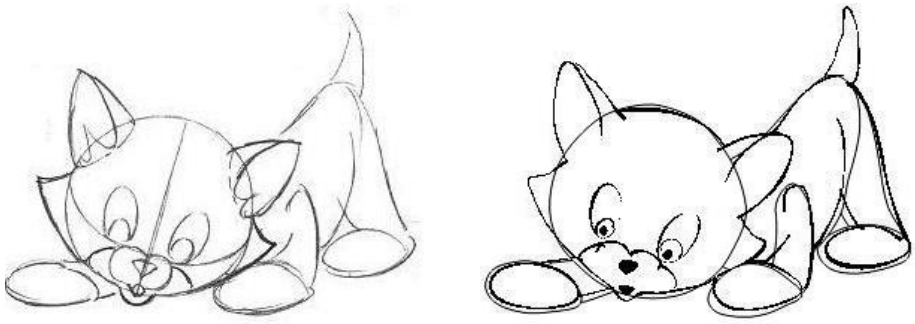
Once the basic shapes of the extreme frames are created, features and other details can be added. Typically, cartoon characters can be characterised by putting emphasis on the outlines (silhouette of the character).

As we already only render the silhouettes of the approximate 3D objects, these are very helpful to the animator.

First, the animator orients the approximate 3D model to a desired orientation and then draws the outlines (based on the silhouette of the 3D objects) by hand for a particular extreme frame. We implemented this as a "silhouette copy and re-edit tool". In a following step, the inking of the other extreme frames is done. This is very easy for the reason that the animator only has to reposition the 3D object and alter the copied outlines, instead of creating them from scratch.

Because the animator refers to the silhouette outlines of the approximate 3D models s/he is assured that the proportion of the characters is preserved.

Figure 5.4(a) shows the version an animator is likely to draw next to our results (figure 5.4(b)).



(a) (b)

Figure 5.4: a) Copyright © 1994 Preston Blair (Blair 94). b) The same extreme frame as shown in (a) by using our system.

5.3.3 Rendering the Animation

In this section we explain the use of the same 3D objects as an underlying aid to render objects for example in a painterly style while ensuring a smooth animation that does not suffer from temporal aliasing.

When looking at the traditional animation process, animators have to paint by hand every single frame of the animation (Patterson 94; Griffin 01). Here we can identify two problems: (i) it is a very time intensive process, for example to create an animation which lasts 25 minutes using 15 frames per second requires 22,500 frames to be drawn, and (ii) colouring the objects in a non-uniform way, such as painterly rendering by using brush strokes, soon results in an animation that suffers from *temporal aliasing* because it is too hard to avoid that brush strokes randomly change or suddenly pop up with each frame. This lack of frame-to-frame coherence is due to the animator's difficulty to estimate the position of the same brush stroke in different extreme frames.

Once the extreme frames are created (basic shapes and outlines) the animator can start the drawing process. We provided the option to choose between displaying only the currently selected extreme frame or displaying (and working on) all extreme frames in a multi-window approach. If the user chooses the second option, all extreme frames are shown in separate windows and any modifications (such as painting strokes) made to one of these are immediately reflected in the others.
The painting process itself is as described in listing 5.1.

In our application, brush strokes are RGBA textured polygons which we created using an external drawing program. That way, the animator has full control over the shape, size, density, texture, ... that make up the brush.

```
While modelling
select underlying curve primitive
adjust brush stroke parameters
for each stroke gestured by the animator, do
  collect 2D screen positions
  for each 2D screen position, do
    transform screen position to 3D object space
    create particle
    for all extreme frames, do
      calculate 3D position of particle
      transform 3D position to screen space
      store position
    end for
  end for
end for
```

```
During animation (at run-time)
for each frame in time, do
  generate in-between frame
  for each sorted primitive, do
    draw primitive
    for each associated particle, do
      orient particle orthogonal to view vector
      draw brush stroke into paint buffer
    end for
  end for
end for
```

Listing 5.1: The painting process of our system.

Regarding the painting process, first of all, the animator has to select one of the underlying curve primitives to which brush strokes are to be attributed. Hence, the underlying drawing order of the curve primitives specified in the extreme frames is utilised to determine the drawing order of brush strokes. This nicely solves the problem of self-occlusions (see section 3.2).

Painting itself occurs by gesturing strokes (chapter 4) upon the currently selected curve primitive. At this moment, the animator does not see the underlying 3D models but only the silhouettes. This is done to keep the painting process similar to the traditional painting procedure.

At the same time (when gesturing the strokes), our system transforms the current position in 2D screen space to the object space of the corresponding underlying 3D surface. This delivers us a set of 3D points (which lie on the surface of the underlying 3D object).

Then, with each of the points we associate a particle which stores the cur-

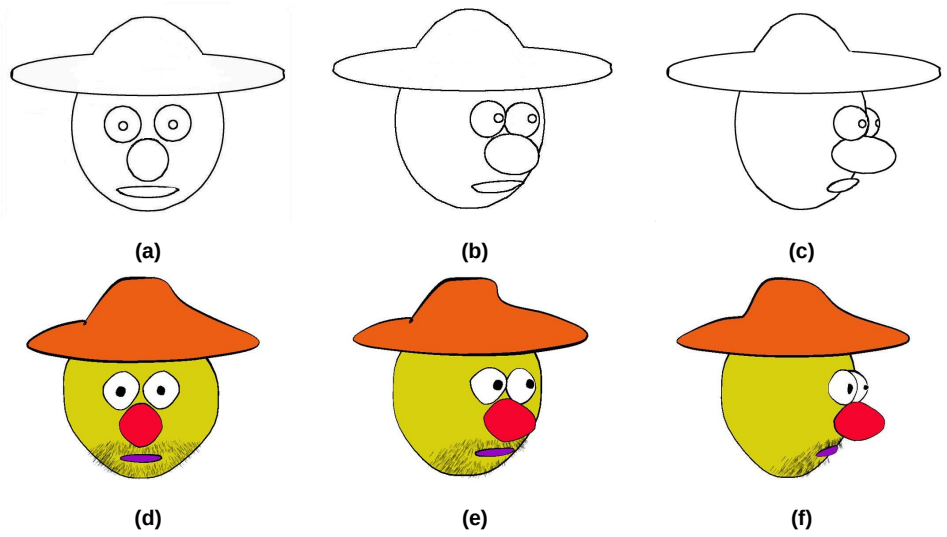


Figure 5.5: a–c) Three extreme positions of a person's head constructed with circular and rounded forms. d–f) The same three extreme frames, after inking the outlines, filling the primitives and painting some brush strokes.

rent selected brush, its position and orientation. Finally, for each extreme frame we calculate the 3D position of the particles by exploiting the underlying 3D surfaces. These 3D positions are then transformed to 2D screen space: our system can thus handle the brush strokes in the same way as the underlying drawing primitives (curves) and so we can exploit the automatic in-betweening method introduced in chapter 3. Note that, by inspecting the normal of the underlying surface, particles that should be occluded in 3D can easily be detected and thus marked as invisible for a particular frame.

The advantages of this method are: (i) for the extreme frames, the particles are positioned automatically by exploiting the underlying 3D surface and thus whenever the animator switches to another extreme frame the same particles are shown, transformed according to the transformation of the underlying 3D object, which guarantees frame-to-frame coherence, and (ii) for the in-between frames, we can turn to 2D, thereby fully utilising automatic in-betweening.

Upon drawing, the particles are always oriented to lie in the plane orthogonal to the view vector. That way we prevent the animation to suffer from perspective distortions, which we want to avoid in 2D animation. Finally, we utilise the drawing order of the underlying curve primitives to determine the drawing order of the brush strokes.

Figures 5.5(d–f) show three painted extreme frames of a person's head.

These are based on the 3D representations in figures 5.5(a–c). We only applied some brush strokes to figure 5.5(d), depicting the hairs of an untidy beard and moustache. As you can see in figures 5.5(e–f), the same brush strokes are displayed automatically at where we would expect them to appear: e.g., the moustache is still displayed under the nose and above the mouth. Notice that the character's hat in figures 5.5(d–f) is not geometrically correct with respect to the 3D representation and that some of the brush strokes (hairs) may appear outside the inked outlines. This is intentionally done by the animator to express the careless style of the character.

We conclude this section by stating that the use of approximate 3D models is an effective means to provide for frame-to-frame coherence because objects are painted just once and the corresponding brush strokes are positioned automatically for all frames.

5.4 Results

In this section we present some results.

Figure 5.6 shows some snapshots of an animation exploiting the extreme frames presented in figure 5.5.

In figure 5.8 we show some animation snapshots of an airplane. These are computed starting from the extreme frames of figure 5.7.

The extreme frames (without the outlines drawn) of an exotic bird are shown in figure 5.9. Note that we did not use curves to draw the wing of the bird. Instead it is completely painted upon the body. Figure 5.10 shows some snapshots of an animation of the bird.

Figure 5.11 consists of images depicting some extreme frames of an animated moon which is painted in a pointillism-like painting style. Snapshots of an animation sequence exploiting these extreme frames are shown in figure 5.12.

5.5 Discussion

In this chapter, we presented the use of approximate 3D models as a powerful tool to assist the animator throughout various stages of the drawing process when creating an animation.

Existing computer animation software either employs full 3D input models or uses purely 2D approaches. In the former, 3D models are rendered and animated in non-photorealistic styles but require extensive 3D modelling, and it is difficult to achieve lively or subtle movements. The latter lacks the 3D

representation which is present in the animator's mind and therefore only an experienced animator easily succeeds in retaining the volume of objects and preventing temporal aliasing.

Our approach demonstrates how approximate 3D objects can be used to overcome these problems. We successively described how to retain proportions, how to rapidly ink silhouettes and how to provide for frame-to-frame coherence when rendering in a painterly style.

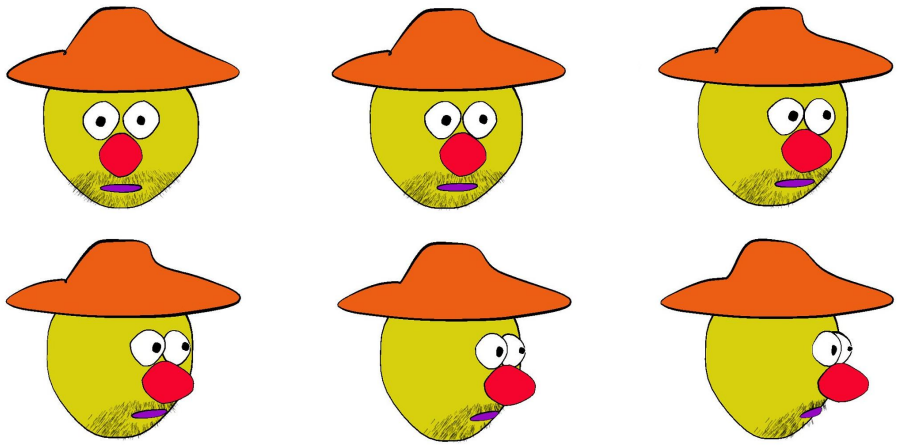


Figure 5.6: Some snapshots of an animation sequence generated from the extreme frames in figure 5.5.

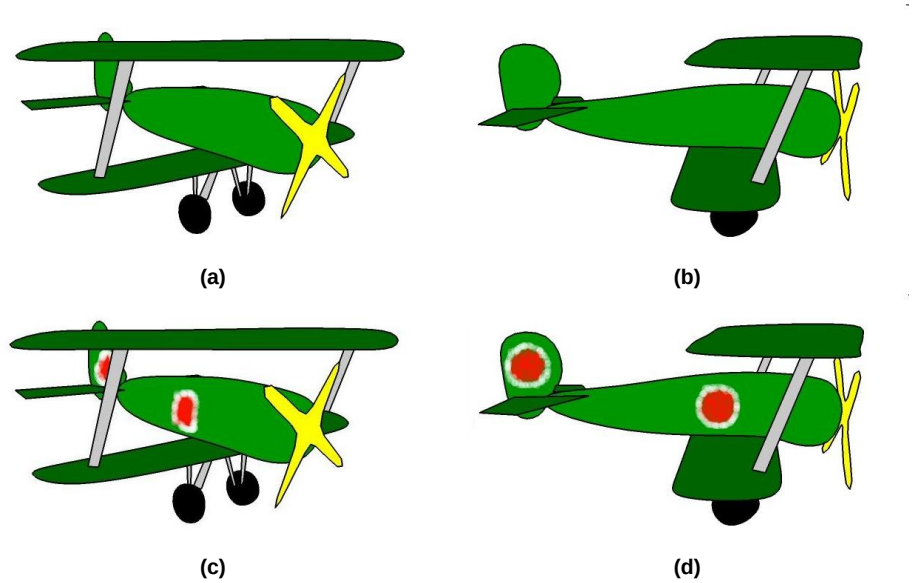


Figure 5.7: a–b) Some extreme frames, with inked outlines and filled primitives, of an airplane. c–d) The same extreme frames after having painted the object with brush strokes.

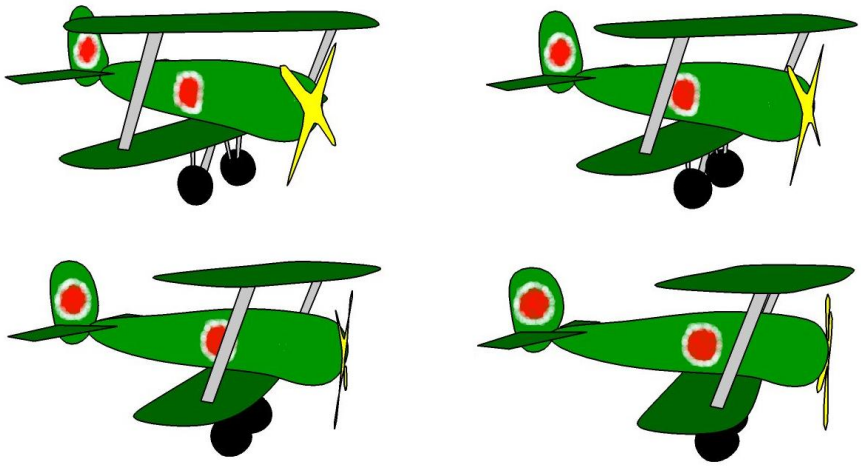


Figure 5.8: Snapshots of an animation of the plane shown in figure 5.7.

Figure 5.9: Painted extreme frames, without inked outlines, of an exotic bird.

Figure 5.10: Some snapshots of an animation of the exotic bird depicted in figure 5.9.

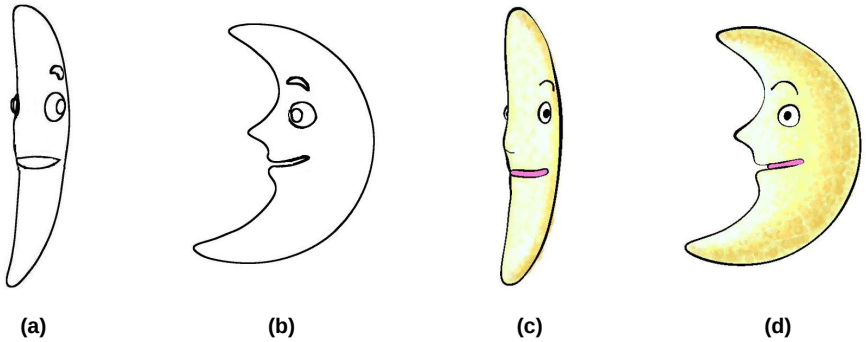


Figure 5.11: a–b) Some extreme frames of a moon. c–d) The same extreme frames after having painted the moon with coloured brush strokes to achieve a pointillism-like painting style.

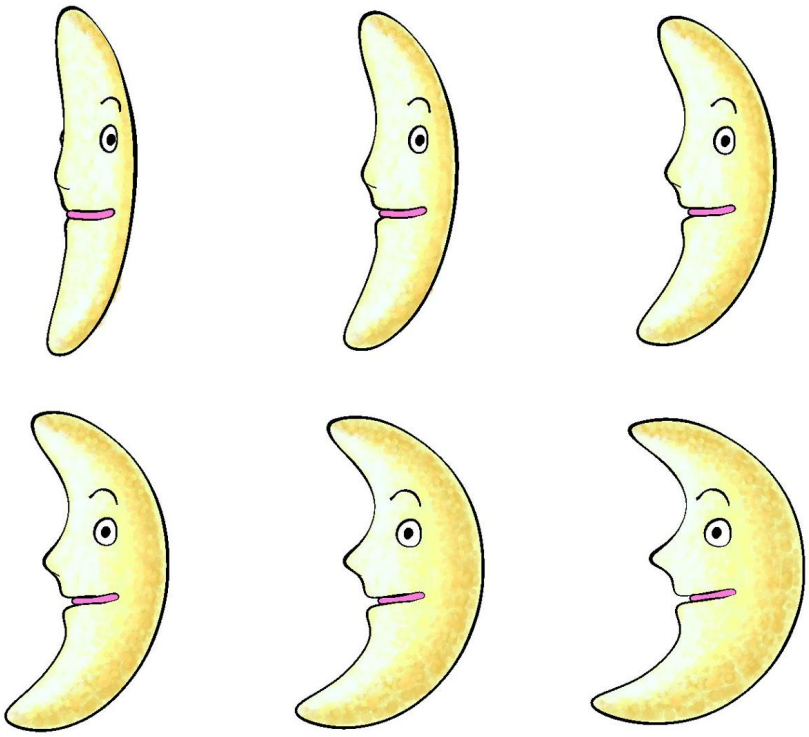
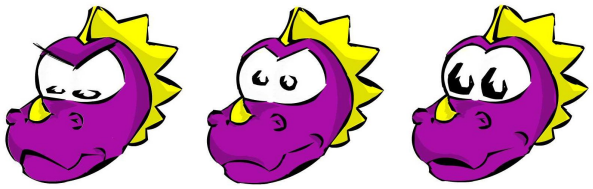


Figure 5.12: These pictures show some snapshots of an animation sequence generated from the extreme frames in figure 5.11.

Chapter 6

Facial Animation:
Mimicking 3D Transformations of Emotional Stylised Animation with Minimal 2D Input



"All cartoon characters and fables must be exaggeration, caricatures. It is the very nature of fantasy and fable."

Walt Disney

Contents

In this chapter we introduce a novel approach to assist a graphical artist throughout the creation of traditional facial animation[1]. We focus on eliminating the time-consuming process of drawing all the emotions of a character, which have to be portrayed from different viewpoints. Furthermore, we aim at preserving the animator's freedom of expressing the artistic style s/he is bearing in mind.

To establish these goals, we introduce the concept of *facial emotion channels*, of which each represents a facial part expressing an emotion. Furthermore, we present a novel approach through which an emotionally meaningful 2D facial expression from one point of view can be created from a reference expression from another point of view. The provided solution is easy to use and empowers a much quicker cartoon production, without hampering the animation artists' creativity.

6.1 Introduction

Traditional animators speak from experience when they say that animating the face is one of the most challenging and rewarding tasks. Under normal circumstances, people immediately can tell by the first look on someone's face under which emotional state the person finds himself/herself.

Although facial expressions of humans are limited by anatomical constraints, some still manage to pull dozens of faces of which each conveys an emotion. Cartoon characters, on the other hand, lack these constraints and hence are capable of making countless faces of which a substantial part is beyond realism. Drawing all emotions for all characters is without doubt a labour-intensive process. Certainly when the characters have to be seen from multiple viewpoints, most time is spent on drawing facial expressions.

To solve the mentioned problems, we first introduce the concept of *facial emotion channels* (FECs), of which each can be seen as the representation of a particular facial part expressing a specific emotion. Furthermore, we present a novel approach through which an emotionally meaningful 2D facial expression from one point of view can be created from a reference expression in another point of view. The provided solution is easy to use and empowers a much quicker cartoon production while we do not limit the animation artists in their creativity.

[1] An earlier version of this material was presented in (Di Fiore 03b).

Figure 6.1: A wolf staring at the camera. The orientation of the ears, the tiny pupils and the small mouth indicate he is surprised.

Figure 6.1 shows an image of the type and look of facial expressions we would like to create and animate. As can be seen, the animator does not mimic reality exactly: emphasis is put on specific expressive details that do not exist in the real 3D world. In this example the animator wishes to express the astonishment of the wolf by focusing on the orientation of the ears, the tiny pupils and the small mouth.

This chapter is organised as follows. Section 6.2 describes previous work in the field and indicates the differences with our philosophy. In section 6.3 we first introduce the term *facial emotion channels* (FECs), then the automatic generation of these FECs is elaborated on. Section 6.4 provides clarifying examples. Finally, we end with our conclusions (section 6.5).

6.2 Existing Facial Animation

In this section we look at and evaluate existing techniques present in realistic facial animation systems, 2D animation systems and systems that exploit 3D models.

6.2.1 Towards Realism

Starting with (Parke 72), a lot of research has been carried out into the field of realistic facial modelling and animation.

For the modelling part this has led to the development of interesting techniques including (i) *physics based muscle modelling* (Platt 81) which mathematically describes the properties and behaviour of human skin, bone and muscle systems, (ii) the use of *free-form deformations* (Kalra 92) where the muscles are embedded in an imaginary flexible control volume that can be manipulated by displacing control points and whose deformations are reflected

onto the muscle, and (iii) the use of *spline muscle models* such as subdivision surfaces (Catmull 78b) to support smooth and flexible deformations and to model sharp creases on a surface or discontinuities between surfaces.

For the animation part, the difficulties in creating life-like character animations led to performance driven approaches. These approaches include (i) animation using *motion tracking* (Li 01) where markers on one's face are continuously tracked and deliver (precise) motion data which can be used for driving specific animation systems, and (ii) animation employing motion *retargetting* (Gleicher 98) where the animated motion from one character is adapted to another.

We refer the interested reader to (Noh 98) for an extended survey and classification (taxonomy) of facial modelling and animation methods.

As can be seen, there's a wide range of tools and techniques available which aim at creating realistic (facial) animations. Since we are interested in creating lively cartoon animations starting from 2D, we limit this discussion to research that employs some of the discussed techniques (whether adapted or not) to produce 2D animations.

Rose et al. present an inverse-kinematics methodology which exploits the interpolation of *example*-based motions and positions (Rose III 01). The key issue of their system is to allow an artist's influence to play a major role in ensuring that the system always generates plausible results. Starting from a small number of example motions and positions, an infinite number of interpolated motions between and around these examples are generated at high frame rates. This methodology is highly focused on positioning articulated figures and therefore does not lend itself to the generation of facial animation.

Bregler et al. use capturing and retargetting techniques to track the motion from traditionally animated cartoons and retarget it onto new 2D drawings (Bregler 02). By using animation as the source, similar new animations can be generated. This approach leads to very impressive results, but unfortunately some drawbacks prevent it to be used extensively. The retargetting process is very dependent on a good choice of the source and target key-shapes which one has to select and draw manually. The animator has to watch carefully that the chosen key-shapes cover the entire cartoon space (the entire range of possible poses). Furthermore, the creation of the target-key shapes — these shapes replace the source key-shapes — is a very tedious task since each source key-shape requires a target key-shape to be drawn manually.

Fidaleo et al. present a facial animation framework based on a set of Co-articulation Regions (CR) for the control of 2D animated characters (Fidaleo 02). CR's are parameterised by muscle actuations and are abstracted to high-level descriptions of facial expression. In practice, video footage (of an actor) is

analysed and used to control expressive gestures. The major advantage of this system consists of the independence between the actor and the controlled object. This is maintained by mapping through a single set of controlled parameters. This implies that each new character requires a neutral face frame *and* an explicit reconstruction sample for each CR state to be defined. Furthermore, the current system is not capable of handling animated characters (for example, a person who's looking around).

The described techniques are promising and deliver very appealing results. However, major issues are present in the systems which are heavily based on realistic input. In the animation stage, they don't offer much freedom of exaggeration and other artistic modifications. In addition, the modelling stage involves a lot of tedious and cumbersome work for the animator.

The first issue arises out of working with real motion data: real motion data generates a very realistic look, but in fact that's what we want to escape from when creating cartoon animations. What we need is some kind of *unrealistic* (exaggerated, caricatural, ...) motion data. But that is impossible when working with real actors unless the data is tweaked, which unfortunately is a hard task.

Secondly, our philosophy behind creating animations is not of the kind that an animator has to engage himself/herself in placing markers on someone's face, specifying feature points on images or objects, drawing all extreme poses to cover cartoon space or as an alternative doing extensive 3D modelling, ... and in the end still ending up with an animation that does not resemble what s/he was bearing in mind.

To summarise, techniques and methods to create realistic facial animation are very advanced and promising but are neither in the animation nor in the modelling stage suitable for computer assisted traditional facial animation.

6.2.2 Sticking to 2D

In 1996, Thórisson described a dedicated facial animation system, *'ToonFace'*, that uses a simple scheme for generating facial animation (Thórisson 96). In this system, a face gets divided into seven main features, each with a specific number of control points of which the position can be fixed, move in one dimension or move in two dimensions. Drawing is done by filled two-dimensional polygons which can have an arbitrary number of vertices. These polygons are differentiated between *free polygons* which cannot be animated, *feature-attached polygons* which are associated with a whole feature and inherit its movements, and *point-attached polygons* which only change when a specific control point is being moved. The author succeeded in attaining her

goal, that is to take a simpler, more artistic approach. However, one almost always ends up with similar animations. The use of fixed regions consisting of predefined control points which at their turn have definite degrees of freedom involves a lot of limitations which cannot be overcome by the animator. Other major drawbacks include that the characters always have to look towards the camera, and that creating new extreme poses involves a lot of work since all control points have to be manipulated by hand.

Recently, various systems have been developed for animating virtual actors, talking heads, virtual announcers, etc. Ruttkay et al. discuss *'CharToon'* which is an interactive system to design and animate 2D cartoon faces (Ruttkay 00). The architecture consists of following three components: (i) the *face editor* (modeler) with which a face can be built up from pre-cooked components, (ii) an *animation editor* to define time curves in order to animate the components of the face, and (iii) the *player* which generates the frames of the animation. More recently (Ruttkay 01), the system is extended to let the animator define and impose constraints on control points in order to ensure a desired animation. The use of *'CharToon'* is similar to *'ToonFace'* but it solves also some of its problems by means of the variable amount of control points and the introduction of skeleton-animated basic components. Despite its wide range of potential applications (faces on the web, games for kids, ...) the system is too specific for creating professional cartoon animations. Two major drawbacks compared to our approach are that transformations outside the drawing plane are not supported, and that all key frames still have to be created manually.

We conclude that 2D facial animation systems are much easier to use than realistic approaches but unfortunately do suffer from too many constraints by which the animator's artistic expression is substantially limited.

6.2.3 Towards 3D

This section discusses systems which *turn* to 3D information in order to create appealing 2D animations.

Recently popular, non-photorealistic rendering (NPR) (Reynolds 04) techniques (in particular, 'Toon Rendering') are used to automatically generate stylised cartoon renderings. Starting from 3D geometrical models, NPR techniques can generate possibly stylised cartoon renderings depicting outlines with the correct distortions and occlusions. Despite the automatic generation, it requires heavy modelling and animation of 3D objects and in any case the results suffer from being too '3D-ish' since the underlying 3D geometry is rendered too accurate.

Rademacher presents a view-dependent model in which a 3D model changes shape based on the direction it is viewed from (Rademacher 99). The model consists of a base model and a description of the model's exact shape (key deformations) as seen from specific key viewpoints. Given an arbitrary viewpoint the deformations are blended to generate a new, view-specific 3D model. This way, the artistic contributions (key viewpoints) of the animator are always reflected in the generated view-specific model. However, the animator still has to construct the base model and its deformations for each key viewpoint which is undoubtedly time consuming. Moreover, this approach is directed to a type of high-end animation where a 3D look is of prime importance, whereas we want to create animations with 2D details that are too elaborate (or sometimes even impossible) to model in 3D.

6.3 Mimicking Facial Animation with Minimal 2D Input

Our approach can be situated in the taxonomy of (Noh 98) under geometry manipulations which exploit interpolation and parametrisation, and is implemented as a level 1 and 3 extension of our framework (chapter 3).

In the following subsections we'll explain the creation of facial expressions (level 1 extension) by means of facial emotion channels, and the automatic generation of facial extreme frames (level 3 extension).

6.3.1 FECs: Facial Emotion Channels

In this section we introduce the concept of *facial emotion channels* (FEC) which can be seen as the building blocks of any facial expression. Suppose the animator wants to create an animation of a character whose emotions vary from being happy to feeling sad. Clearly, these emotions will be reflected in the character's face. Figures 6.2(a–c) show some extreme emotions that can occur in this character's cartoon space. Note that for example when the wolf is happy, this is expressed by almost all facial parts, such as the engaging smile, staring eyes and a raise of the eyebrows.

Instead of modelling a 'complete' happy face at once, we model every individual part (face outlines, mouth, nose, eyes, eyebrows, ...) of the face independent of the others. That is, for each individual part, we model one neutral version (which depicts no emotion at all) and a set of emotional versions (one version for each emotion that has to be supported). Looking again at figures 6.2(a–c), for the mouth this can be seen as modelling a neutral, a

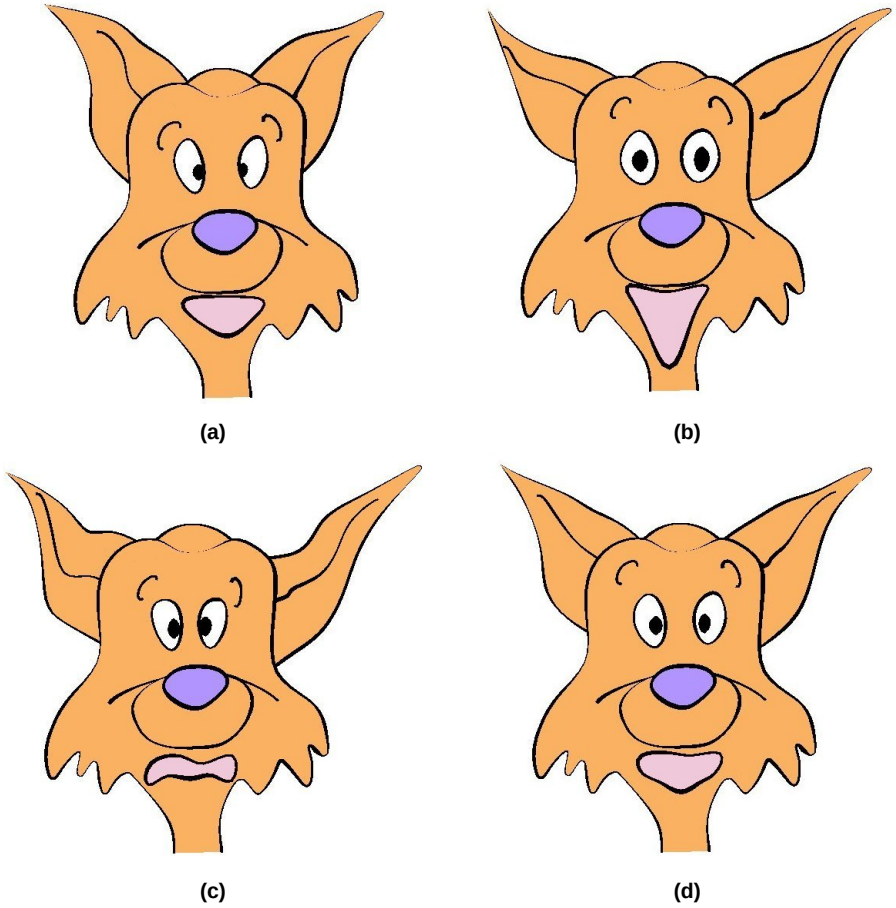


Figure 6.2: Facial expressions of a wolf depicting different emotions: a) no emotion (neutral), b) laughing, c) troubled and d) mixed emotions (semi-laughing and semi-troubled).

smiling and a troubled mouth. The other facial parts are modelled in a similar way.

Concerning the animation phase, the animator only has to specify key frames in time by entering parameters using the same methods as described in chapter 3. Afterwards, the automatic in-betweening method comes into play and generates the desired animation.

This gives the animator the opportunity to create countless different facial expressions without having to model each expression manually, contrary to earlier systems (Thórisson 96; Ruttkay 00) where facial expressions are modelled as one entity and hence are fixed. This is illustrated in figure 6.2(d) which shows the same wolf but this time with 'mixed emotions': the facial expression is built up from a user-defined interpolation of a happy and a troubled face (semi-laughing and semi-troubled). Note that no modelling took place to create this expression, only some easy understandable parameters had to be chosen by the animator.

One can argue that the modelling of all emotions for every facial part involves a lot of work. It is obvious that the different wanted emotions have to be modelled at least once because that's the only way an animator can convey the artistic images s/he is bearing in mind. Section 6.3.2 presents a solution for the case when facial expressions have to be seen from multiple viewpoints.

For a particular viewpoint, each facial part expressing an emotion is called *facial emotion channel* (FEC)[2]. In practice, the animator decides himself/herself how many FECs to create and what each FEC should depict. Hence, it is perfectly possible that the animator creates only one FEC to represent both the eyes and the eyebrows instead of creating two FECs where the first one corresponds to the eyes and the second one to the eyebrows.

Also, a FEC does not necessarily consist of only one emotional version. In fact, a FEC can be any N-tuple of versions depicting the same emotion. Each emotional version is identified with a percentage (0% – 100%) indicating the degree of emotion. A zero percentage represents no emotion, 100% maps to the extreme degree of emotion whereas the in-between percentages correspond to emotional versions with an intermediate degree. These degrees of emotion are functionally comparable to the emotion vectors described in (Prashant 03).

This is particularly very useful when the animator wants another behaviour than obtained with our automatically in-betweening approach. Suppose we have a FEC depicting a smiling mouth. If the channel consists of only two emotional versions of the mouth (e.g., 0% and 100%), then a 50% smiling

[2]Throughout this text, we use the terms *facial emotion channel*, FEC or channel to denote the same concept.

mouth would be the average of the neutral (0%) and extreme (100%) version. However, if the animator for some artistic reasons is not satisfied with this result, s/he can create the intermediate version (in our case 50%) himself/herself, which then has priority over the generated version.

Finally, the user also has the option to combine subsets of FECs into abstract groups, called *emotion groups*. That way, often used facial expressions can be stored and used in a much more intuitive way by only selecting the appropriate *emotion group*. This functionality is also present in other systems, like (Thórisson 96) and (Ruttkay 03).

To summarise, the concept of *facial emotion channels* enables the animator to easily and intuitively create countless different facial expressions without having to model each one by hand.

6.3.2 Automatic Generation of Facial Extreme Frames

In this section we show how an emotionally meaningful 2D facial expression from one point of view can be created from a reference expression from another point of view.

In the previous section, we introduced the concept of *facial emotion channels*. Essentially, FECs can be seen as the representation of a particular facial part depicting a specific emotion.

Now, recall that in order to achieve convincing 3D-like animations, our system requires the character to be modelled as seen from different viewpoints — about eight viewpoints suffice to fully cover all rotational angles around the upstanding axis. Consequently, each FEC needs to be remodelled for every viewpoint. As a result, the number of FECs to be modelled increases proportional to the number of viewpoints, and therefore also the time spent doing modelling.

We introduce a novel approach which aims at minimising this labour-intensive process. The ideal case would be to develop a tool that meets following requirements; (i) to automatically generate *all* FECs for all viewpoints, and (ii) to take into account the animator's artistic input.

Obviously, in order to properly convey the artistic feelings the animator is bearing in mind (what does a happy face look like?), at least for one viewpoint all FECs should be defined.

Figure 6.3 shows some facial expressions of a young boy. Besides the neutral expression (a), we recognise a happy, a troubled and a surprised face (b). (In fact, we have a neutral, a happy, a troubled and a surprised FEC for each facial part). Now, when multiple viewpoints need to be supported, the only manual intervention of the animator is *showing* the system how the

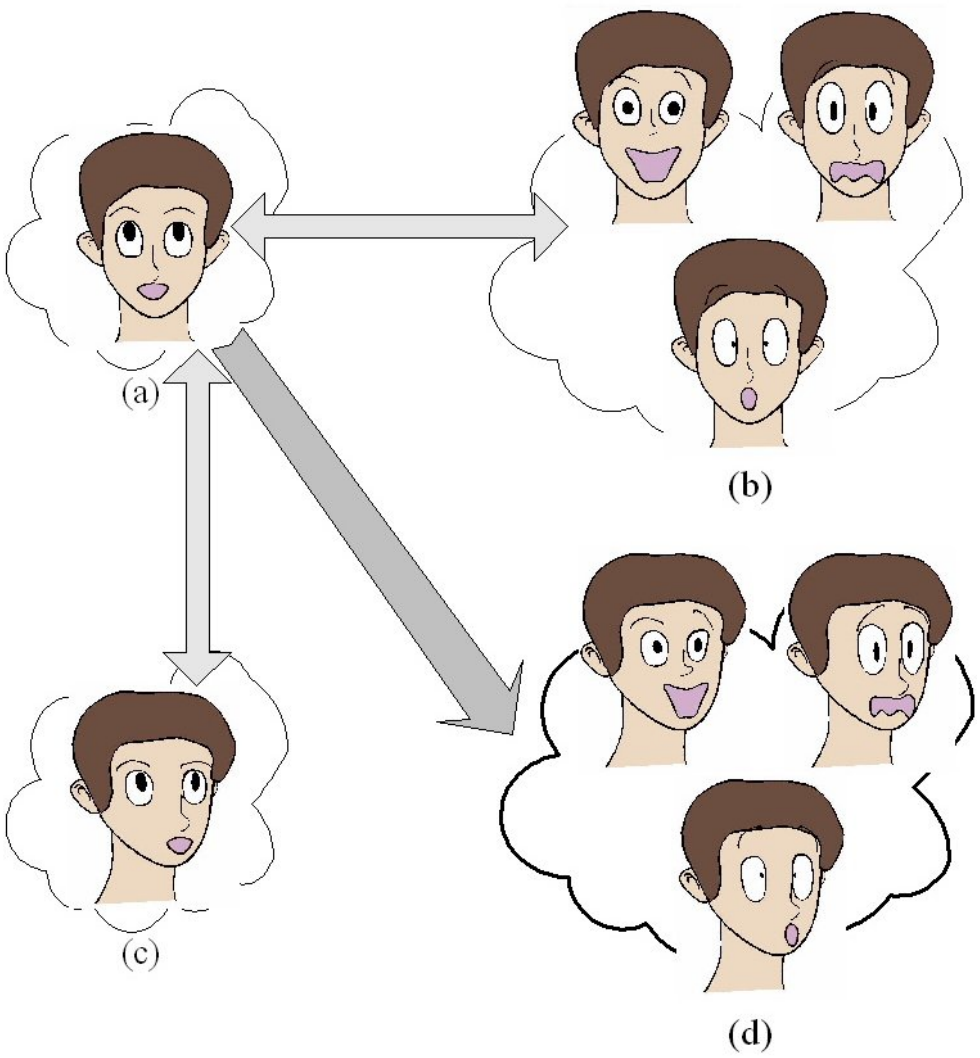


Figure 6.3: Overview chart of the automatic generation of facial extreme frames. a–b) Neutral and emotional versions modelled by the animator. c) Hint of the animator for a new viewpoint. d) Automatically generated facial expressions.

neutral FECs look like for the other viewpoints. This can easily be done by altering a duplicate of the original neutral FEC or by using techniques described in (Flerackers 02) and (Kort 02). After that, the system comes into play and automatically generates all other FECs for each viewpoint. Figure 6.3(c) shows the *hint* of the animator, whereas the images of figure 6.3(d) are the automatically generated expressions. We refer to section 6.3.3 for an in-depth description of our approach.

Note that already for a trivial scene (e.g., five emotions: laughing, troubled, surprised, crying, sad; and 4 viewpoints) twenty modelling interventions can be eliminated, and this for each facial part.

Moreover, as can be deduced from the pictures of figure 6.3(d) the original artistic input of the animator is preserved throughout the automatic generation of the FECs. Nevertheless, the animator has the possibility to alter the generated results at any time.

In this section we introduced a novel approach that proves to be very helpful in assisting the animator in creating FECs. As a result, the time-consuming aspects of modelling FECs have been minimised, by the automatic generation of most FECs, while respecting the artistic input provided by the animator.

6.3.3 Implementation

Consider again the example in section 6.3.2 for creating an animation of a young boy. We explain our algorithm on the basis of a single emotion e_j, $j : 1..J$, for one particular facial part F_p, $p : 1..P$, since each pair of an emotion and a facial part gets individually streamed to the algorithm, independently of the others.

The user always starts to model the FECs as seen from a particular viewpoint $vp_k, k : 1..K$, in our case we start with a front facing viewpoint vp_0. Suppose we have two FECs for facial part F_p; one to represent neutral emotion e_n, $F_p^{e_n,vp_0}$, and another, $F_p^{e_j,vp_0}$, to depict the emotional version of F_p. This is illustrated in figures 6.3(a–b).

Since each FEC is a collection of ordered primitives (in our case curves), we write FEC $F_p^{e_n,vp_0}$ as the set of curves $c_l^{e_n,vp_0}$, $l : 1..L$. In a similar way, the emotional FEC $F_p^{e_j,vp_0}$ is built up from the curves $c_l^{e_j,vp_0}$.

At its turn, each curve $c_l^{e_j,vp_0}$ consists of I control points $p_{l,i}^{e_j,vp_0}$, $i : 1..I$. Regarding the fact that each emotional version can be represented as a modified version of the neutral version, each emotional curve also can be seen as a modified version of its neutral equal[3].

[3](Kort 02) describes a cost function based approach to determine the correct matching

In our case these modifications actually are transformations and hence for each control point $p_{l,i}^{e_j,vp_0}$ of the emotional curve $c_l^{e_j,vp_0}$ we store a weighted matrix $M_{l,i}^{e_j}$ which describes the transformation of the i^{th} control point, $p_{l,i}^{e_n,vp_0}$, of the neutral curve, $c_l^{e_n,vp_0}$, to itself.

That way, we express $p_{l,i}^{e_j,vp_0}$ by following equation:

$$p_{l,i}^{e_j,vp_0} = M_{l,i}^{e_j} \cdot p_{l,i}^{e_n,vp_0} \tag{6.1}$$

Now, in order to automatically create FECs for the other viewpoints vp_k, $k : 1..K$, the user is asked to help the system by doing a demonstration. The user starts with the earlier modelled neutral FEC, $F_p^{e_n,vp_0}$, and transforms it (by using a set of predefined tools or by using techniques described in (Flerackers 02) and (Kort 02)) into a new version which represents the same neutral version but corresponding to viewpoint vp_k. We refer to the newly created version as $F_p^{e_n,vp_k}$. This is illustrated in figure 6.3(c).

For each control point $p_{l,i}^{e_n,vp_k}$ of the curve $c_l^{e_n,vp_k}$ we store a weighted matrix $M_{l,i}^{vp_k}$ which describes the 'viewpoint' transformation of the i^{th} control point, $p_{l,i}^{e_n,vp_0}$, of the curve, $c_l^{e_n,vp_0}$, to this point.

Consequently, we can express $p_{l,i}^{e_n,vp_k}$ by following equation:

$$p_{l,i}^{e_n,vp_k} = M_{l,i}^{vp_k} \cdot p_{l,i}^{e_n,vp_0} \tag{6.2}$$

At this point, the automatic generation comes into play. In order to find the emotional versions of $F_p^{e_n,vp_k}$ we need to find all curves $c_l^{e_j,vp_k}$. In practice, it narrows down to calculating the control points $p_{l,i}^{e_j,vp_k}$, which are in fact the emotional counterparts of $p_{l,i}^{e_n,vp_k}$.

Analogous to equation 6.1 we express $p_{l,i}^{e_j,vp_k}$ as follows:

$$p_{l,i}^{e_j,vp_k} = M_{l,i}^{e_j} \cdot p_{l,i}^{e_n,vp_k} \tag{6.3}$$

Using equation 6.2 we get:

$$\begin{aligned} p_{l,i}^{e_j,vp_k} &= M_{l,i}^{e_j} \cdot (M_{l,i}^{vp_k} \cdot p_{l,i}^{e_n,vp_0}) \\ &= (M_{l,i}^{e_j} \cdot M_{l,i}^{vp_k}) \cdot p_{l,i}^{e_n,vp_0} \end{aligned}$$

The weighted matrices $M_{l,i}^{e_j}$ and $M_{l,i}^{vp_k}$ are diagonal matrices[4] and so commutative under multiplication:

$$p_{l,i}^{e_j,vp_k} = (M_{l,i}^{vp_k} \cdot M_{l,i}^{e_j}) \cdot p_{l,i}^{e_n,vp_0} \tag{6.4}$$

$$= M_{l,i}^{vp_k} \cdot (M_{l,i}^{e_j} \cdot p_{l,i}^{e_n,vp_0})$$

$$= M_{l,i}^{vp_k} \cdot p_{l,i}^{e_j,vp_0} \tag{6.5}$$

of curves.

[4]Each matrix describes the transformation from one control point to another.

To summarise, the emotional control points $p_{l,i}^{e_j,vp_k}$ can be seen as (i) an emotional version of the neutral control points of viewpoint vp_k (equation 6.3), or (ii) a viewpoint vp_k specific version of the emotional control points of viewpoint vp_0 (equation 6.5), or (iii) the result of a combined operation on the neutral control points of viewpoint vp_0 (equation 6.4).

Although each solution delivers exactly the same results, we use the third solution since it expresses the new control point as the result of an operation on its neutral counterpart of the starting viewpoint.

So, from now on we calculate each unknown $p_{l,i}^{e_j,vp_k}$ by following equation:

$$p_{l,i}^{e_j,vp_k} = M_{l,i}^{e_j,vp_k} \cdot p_{l,i}^{e_n,vp_0} \tag{6.6}$$

The automatically generated expressions are pictured in figure 6.3(d).

6.4 Results

We have used this approach on some examples. For demonstration reasons, we chose to animate only certain elements of the face (mouth, nose, eyes, eyebrows). Also, the following examples only address motion from head-on to profile. However, our approach is as much suitable when the camera is tilted or when the character turns away from the camera. For example, when the character also has to look up or down, it suffices to create only the new neutral FECs which represent rotations around the horizontal axis. All other emotional FECs will be generated automatically in a similar way as described in section 6.3.2.

Figure 6.4 shows some snapshots of facial expressions of a girl seen from different viewpoints. The different facial parts that we modelled are the eyes, the pupils and the mouth. Figures 6.4(a–d) show the animator's artistic input expressions. We recognise the neutral (a), laughing (b), troubled (c) and surprised (d) expression.

In order to generate emotional versions for the other viewpoints, two new neutral versions (e and i) are provided (one for each viewpoint). This results in the generation of respectively expressions (f–h) and expressions (j–l).

Notice that the outlines of the mouth in figure 6.4(g) cross the outlines of the face. This is due to the fact that the proportions of the facial parts are intentionally not geometrically correct with respect to 3D.

Figure 6.5 consists of images depicting some facial expressions of a young boy. We separately modelled the eyes, the pupils, the eyebrows, the nose and the mouth. Figures (a–d) are the source expressions as modelled by the animator whereas figures 6.5(f–h) show the generated expressions.

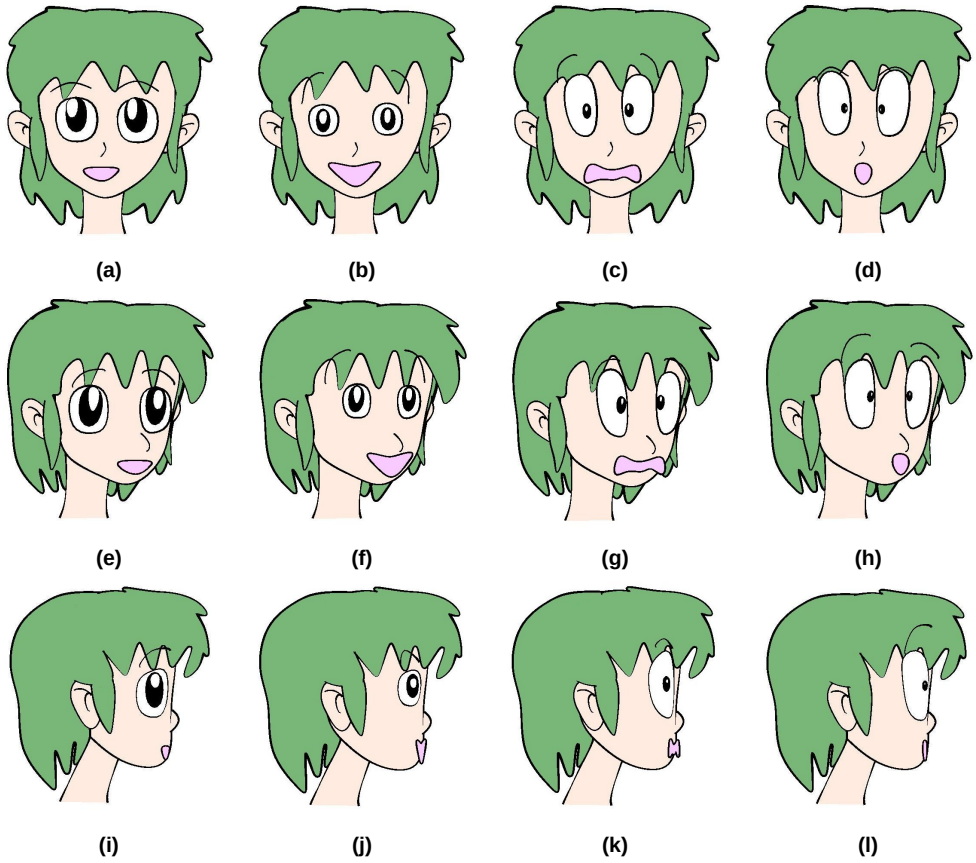


Figure 6.4: Facial expressions of a girl. a) Neutral emotion. b–d) Facial expressions depicting emotions. e, i) Viewpoint specific neutral versions which in combination with (a–d) lead to the generation of respectively (f–h) and (j–l).

As can be derived from our examples, the drawing style from the input images is successively preserved throughout the automatic generation process. This clearly demonstrates the benefits of our concept.

6.5 Discussion

In this chapter, we presented a novel approach to assist the animator throughout the time-consuming process of traditional facial animation, especially the individual drawing of all the emotions of a character which has to be seen from different viewpoints. At the same time we need to preserve the freedom of an

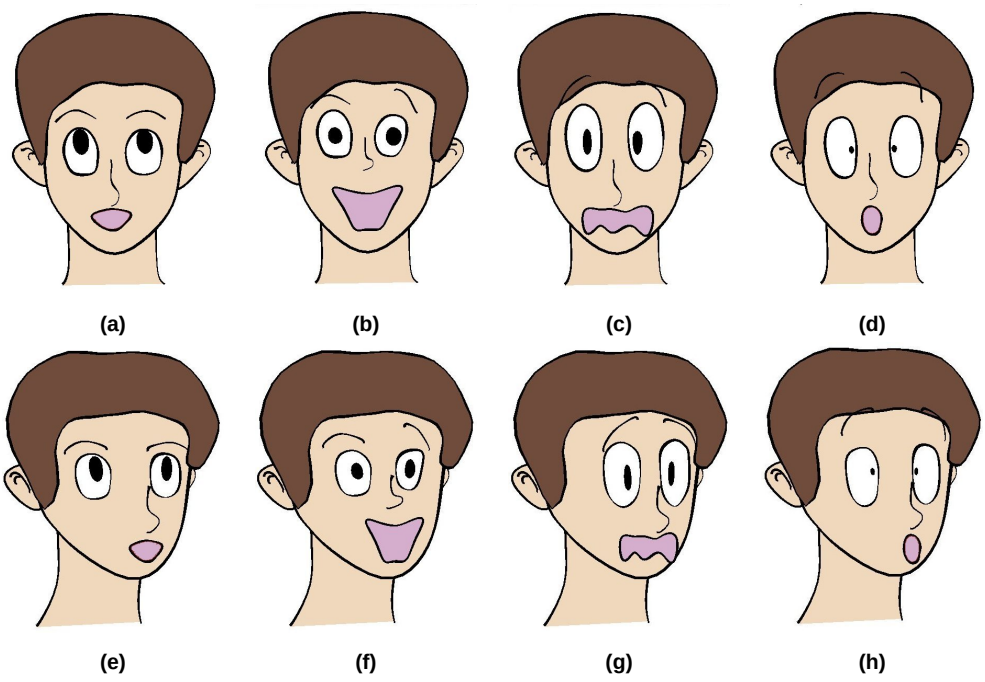


Figure 6.5: Facial expressions of a boy. a) Neutral version. b–d) Source expressions. e) Viewpoint specific neutral version. f–h) Generated expressions.

animator to express the artistic style s/he is bearing in mind.

We accomplished our goals by first introducing the concept of *facial emotion channels* (FECs), of which each can be seen as the representation of a particular facial part expressing a specific emotion. Instead of modelling a complete face at once, each individual facial part is modelled separately. Hence, creating countless facial expressions is feasible without having to model all of them. Furthermore, we provided a novel approach (and according examples) through which most FECs can be generated automatically. Starting with a minimal input of the animator (all FECs for *one* viewpoint and a neutral FEC for each other viewpoint) *all* emotional FECs for all other viewpoints are automatically generated. As a result, the time-consuming process of creating all FECs is reduced to a minimum while we still preserve the artistic style provided by the animator.

Chapter 7

Stylised Animation:
Modelling in 2D Enabling Fluid Stylised Animation

"A good style must, first of all, be clear. It must... be appropriate."

Aristotle

The work described in this chapter has been developed in context of the European research project IST-2001-37116 'CUSTODIEV' of which part of the objective is to explore new stylistic models of not-quite-photorealism in a sliding scale between realism and stylism (CUSTODIEV 04).

Contents

This chapter introduces new techniques and tools to draw, manipulate and animate stylised brush strokes in computer assisted animation production[1]. We focus on eliminating the tedious process of drawing and managing the numerous brushes that are painted on top of each other, and on avoiding temporal aliasing artefacts such as brushes popping up in successive frames. Moreover, we also aim at giving the animator the same kind of painting tools as if painting in the traditional way. Aside this, we present some higher-level tools that enable the animator to locally and globally control user selected parts of the drawings. This simplifies the interaction drastically.

7.1 Introduction

When looking at existing developments, most systems only support the creation of drawings that are made out of curves. For cartoon animations this is relatively easy, since these characters typically are built up just from silhouette lines which are coloured and filled in a uniform way.

Difficulties arise when the animator chooses to replace the simpler cartoon style to a more painterly style such as the one depicted in figure 7.1. This kind of artistic drawings usually consists of numerous strokes that are painted one over the other. Therefore, when animating these drawings it is difficult for the animator to manage that much brush strokes in a proper and intuitive way. In addition, the applied painted strokes may not suddenly appear and disappear during the animation, nor move or deform with respect to the object. Without such frame-to-frame coherence, the temporal aliasing makes the animation hard to enjoy.

This chapter is organised as follows. Section 7.2 describes previous work in the field and indicates the differences with our philosophy. In section 7.3 we elaborate on our approach. First, we elucidate the modelling and animation stages. Then, the creation, (high-level) manipulation and animation of single brush strokes as well as painted characters is explained. Section 7.4 provides clarifying results, while we end with our conclusions (section 7.5).

[1] An earlier version of this material was presented in (Di Fiore 03c).

Figure 7.1: Animation of a palm tree blown by the wind. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

7.2 Existing Stylised Techniques

In this section we dwell on techniques starting from pure 2D drawings and some approaches found in the painterly rendering and non-photorealistic rendering (NPR) domain.

7.2.1 Pure 2D Approaches

Over the past years, a lot of work has been reported upon creating 2D stylised paintings. The majority of the publications concentrate on how to simulate real brush styles like pencils, airbrushes, watercolour, etc.

In 1990, Haeberli demonstrated an interactive painting system for quickly producing a painted representation of a still image (Haeberli 90). More recently, a new algorithm was presented by Hertzmann for producing paintings from images (Hertzmann 98). Brush stroke sizes are selected to convey the level of detail present in the source image using a multi-scale algorithm. Direction normals, stroke curvature and other parameters describe a space of rendering styles that can be created and modified by artists and graphic designers. The painting process itself is built up in a series of layers and so generates convincing results. Nowadays, a lot of commercial packages provide a wide variety of painterly image filters and brushing tools based on these ideas. However, a major drawback of these techniques is that they are only suitable for creating still images.

We refer the interested reader to (Gooch 01) for an overview of how to simulate artistic media such as brushes and other painting tools.

We can conclude that 2D painting systems are easy and intuitive to use and the way of working very much resembles the traditional way of working. However, these techniques do not take into account the succession of images

and so can only be applied to create still images. In order to create stylised animations, new techniques will be required to maintain temporal coherence for each brush stroke.

7.2.2 Starting from 3D: Non-Photorealistic Rendering Techniques

Barbara Meier's solution to render animations in a specific artistic style starting from 3D geometrical objects eliminates the 'shower door' effect (strokes seem to be disconnected from the objects they represent and float around) that disturbs many other approaches to obtain a good frame-to-frame coherence (Meier 96). However, a lot of hard work is involved to accurately grasp the appearance an artist is imagining: models have to be built up and all necessary brushes need to be assigned properly.

Kaplan et al. discuss an algorithm for rendering subdivision surface models in a variety of artistic styles using an interactively editable particle system (Kaplan 00). The algorithm is suitable for modelling artistic techniques explicitly by the user, or automatically by the system. Frame-to-frame coherence is maintained due to the use of particles to represent hand drawn strokes. However, a lot of user interventions are needed since the editing of strokes involves the individual editing of each of the corresponding particles.

More recently, Kalnins et al. presented a system that lets an animator interactively embellish a 3D model with stylised strokes (Kalnins 02). In fact, strokes are drawn over the model from one or many viewpoints. The user has a wide variety of brush styles to choose from and so is given a high degree of control over the stylisation. Nevertheless, the results suffer from being either too cold or too '3D-ish'.

In chapter 5 we described a painterly rendering method using approximate 3D models. This technique produces nice results and preserves frame-to-frame coherence when 'decorating' 2D animation with brushes. However, it is not intended to create fully stylised animations from scratch.

To sum up, starting from 3D models has the advantage of automatically generating stylised renderings and/or animations but at the cost of heavy modelling and animating, and generating a very 3D look.

7.2.3 Painterly Rendering Techniques

In 2000 (Hertzmann 00), Hertzmann presented a method for painterly video processing. Basically, successive frames of animation are painted over, applying paint only in regions where the source video is changing. Optionally, brush strokes may be warped between frames using computed or procedural optical

flow. This method produces animation with a novel visual style distinct from previously demonstrated algorithms. However, it gives the subjective impression of rather a 'living' painting that is continually being painted over, than a fluent animation that is made in a particular painted style.

More recently, the same author described a system that uses a relaxation algorithm combined with search heuristics to produce painted animations starting from images or video (Hertzmann 01). The user keeps control over the process by varying the relative weights of energy terms. The energy function in turn yields an economical style and produces greater temporal coherence than the previous technique. A drawback of this method is that the energy function is very difficult to optimise, which therefore limits the number of animation styles.

To summarise, painterly rendering techniques are promising and deliver quite aesthetic results. On the other hand, from an artistic standpoint a lot of constraints are imposed on the traditional animator such as the automatic generation of brush strokes and the restricted number of available animation styles. Furthermore, in these approaches animation is only possible when a video or a sequence of images as input is available.

7.3 Implementation of Fluid Stylised Animation

In traditional 2D animation (Blair 94; Williams 01), the animator can easily draw all kinds of stylised brushes on paper, and s/he can dynamically alter the width and the curvature of the painted strokes. Moreover, when the animator is not satisfied with the result, s/he can make (simple) corrections just by drawing or painting over new strokes along or on top of the 'wrong' ones.

To this end, we use the same sketching method as explained in chapter 4, but replace the uniform outlines by stylised brushes.

However, a major difference with pure 2D vector drawings is that painted animation usually consists of much more strokes. Therefore, the emphasis of this work is on substantially improving on these techniques by managing these numerous brushes in a proper and intuitive way so that artists do not need to worry about mistakes being made while editing, or brushes suddenly popping up during the animation.

Following subsections successively describe the modelling and manipulation of brushes, the animation of brushed characters and some higher-level tools to manipulate drawings.

7.3.1 Modelling/Drawing and Manipulating Individual Brushes

Our basic aim in this context is to support the animator in the creation and manipulation of the strokes representing animation characters. Also, we want to provide a user interface that is much more friendly than the standard 'point-click-and-drag' metaphors. Therefore, we extend our multi-level sketching tool (chapter 4) that mimics the sketching method used by artists when sketching strokes with a pencil on paper: splines themselves are not drawn as solid lines, instead the animator can choose from a wide range of stylised brush tools including a solid brush tool (figure 7.2(a)), a paintbrush tool (figure 7.2(b)) and an airbrush tool (figure 7.2(c)).

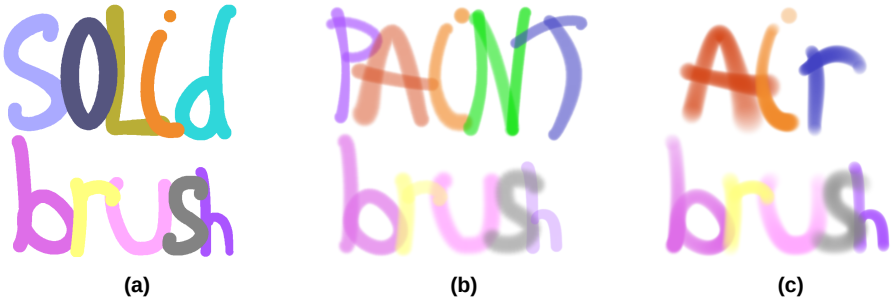


Figure 7.2: Different kinds of brush tools. a) Solid brushes. b) Paintbrushes. c) Airbrushes.

Figure 7.3 depicts a flower modelled with our system using different kinds of brushes.

As can be seen in figure 7.4 the animator's process of drawing characters and objects very much resembles traditional painting: first, a basic (rough) layer is drawn; in the following steps, several layers are drawn upon each other to refine the current result.

Aside the creation and editing of free-form strokes, the same affine transformations upon (selections of) strokes as described in chapter 4 are supported as well.

Brushes: Implementation

Figure 7.5 depicts the pipeline of the paintbrush tool. The tool creates three objects: (i) a colour layer with (ii) a mask, and (iii) a temporary bitmap layer. Depending on the user input from the canvas, the tool renders the brush into the mask (using $dst = max(dst, src)$). The circular brushes have following parameters: radius, opacity, and softness. The mask is a grayscale image used

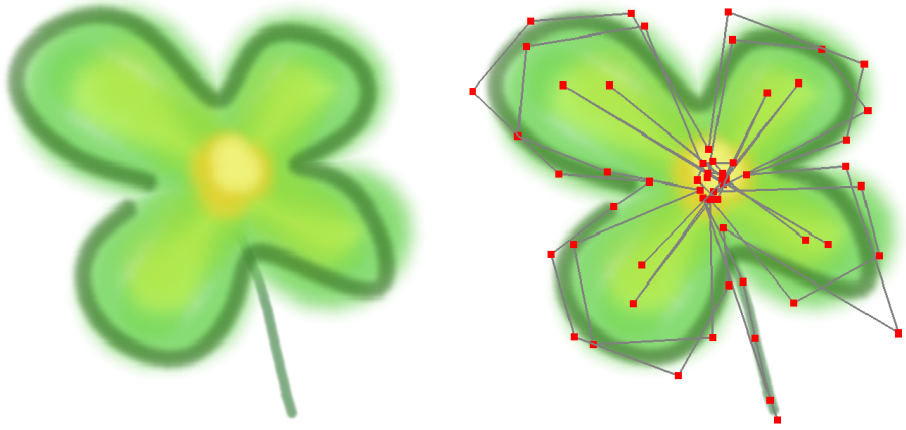


Figure 7.3: a) Picture of a flower created with our system. b) The same flower, depicted with all underlying control points. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

to mask painting. Then, using this mask, the colour layer is rendered into the working layer. Rendering the colour just copies the colour into the destination layer. The result is that a stroke covers the image with the chosen opacity, even when the same place is passed over multiple times (figure 7.2(b)).

The airbrush tool works in a similar way (figure 7.6), however, brushes can overlap each other and will create more opaque areas (figure 7.2(c)). An airbrush not only releases paint on movement, but at regular intervals while the stylus is down.

To render a stroke on the image, a spline is made out of the points gathered from user input. On this spline at fixed distances (depends on the brush settings, default 25% of the diameter), a brush is drawn. The spline however doesn't cross pixels necessarily through the middle. If we would just round to pixels and render the brush, we would get artefacts. To avoid this, brushes should be rendered with sub-pixel accuracy. Therefore, we create not one, but 16 brush-masks, by selecting the correct one depending on which sector of the pixel the spline crosses. We achieve accuracy to 1/4 of a pixel.

When blending colours, the following formula is usually used: $dst = a * src + (1 - a) * dst$. However sometimes this gives an unwanted result. For example, if we blend the bright colours cyan (0, 255, 255) with red (255, 0, 0) using an alpha value of 128, the result is a dark colour grey (128, 128, 128). To get a better result, the opacity has to be used as gamma instead of alpha.

Figure 7.4: The process of drawing characters and objects: first, a basic (rough) layer is drawn; in the following steps, several layers are drawn upon each other to refine the current result. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

Furthermore, because of the high fill rate, the real-time requirement for the drawing process cannot be achieved without graphics acceleration. Consequently, we render the mask directly into a texture using a *pbuffer*. Furthermore, we control the quality by adapting texture and brush resolutions, and texture filtering.

The current implementation can be easily extended with features like taking into account paper structure, or painting tools like water colours.

7.3.2 Animating Brushes

In order to create stylised animation, we can fall back on our 2.5D framework. So, the basic 2D drawing primitives at level 0 are in fact the on-the-fly created strokes. Once the animator has created the extreme frames, s/he only has to specify key frames in time by entering parameters. Afterwards, the automatic in-betweening method comes into play and generates the desired animation.

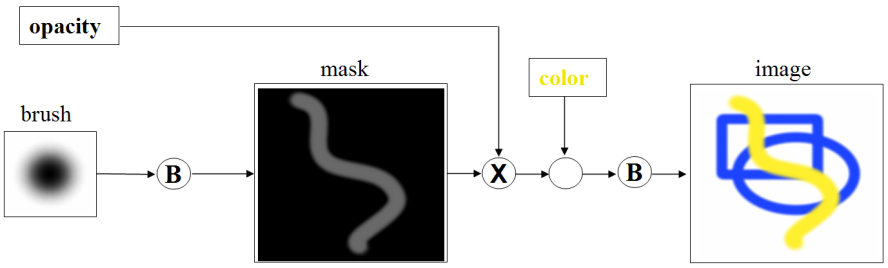


Figure 7.5: Pipeline of the paintbrush tool.

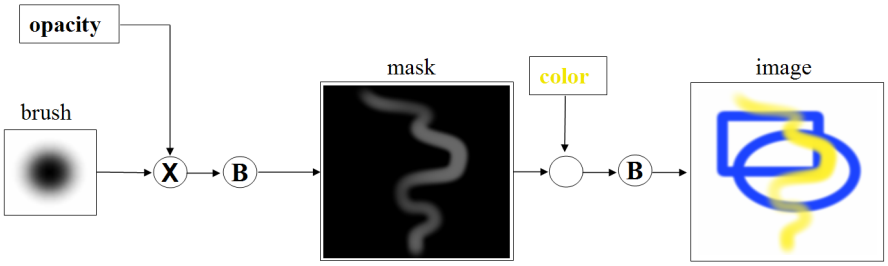


Figure 7.6: Pipeline of the airbrush tool.

As a result, convincing 3D-like animations starting from pure 2D painted drawings can be made.

Figure 7.7 shows some extreme frames and an in-between frame of an animation of a butterfly.

This chapter builds further on this framework as follows. The basic 2D drawing primitives at level 0 are in fact the on-the-fly created cubic Bézier curves of previous section whereas the extreme frames are sets of modelled brush strokes that make up an object.

7.3.3 Manipulating a Drawing

The techniques described in the previous section are all modelling and animation tools that work locally on a single brush stroke. In this section, we will introduce some tools that enable the animator to manipulate the drawings on a higher level.

We successively describe a grouping tool, transformation tools, a deformation tool and a hierarchical structure. Each of these tools can operate both on the whole drawing as well as on a user selected part of the drawing.

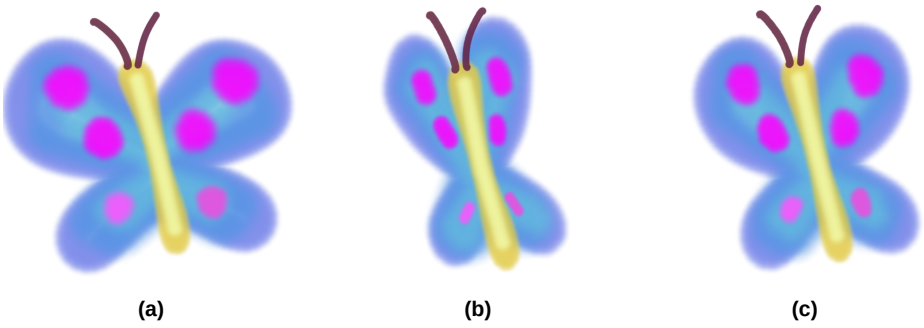


Figure 7.7: a–b) Extreme frames of an animation of a butterfly. c) Generated in-between frame. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

Grouping

The grouping tool acts as the starting point of all higher-level tools that manipulate the drawing. Basically, this tool groups together some of the control points that make up the brushes, contrary to other applications that simply group the selected pixels. As will be clear from the following subsections, our approach is preferable.

Since the underlying mathematical representation of the brush strokes needs to be hidden for the user, grouping is done by drawing a selection lasso over the part of the drawing one wants to change. After selecting, the control points within the lasso will be marked as a group and can be manipulated as will be explained in following sections.

Figures 7.8(a–b)) show the grouping tool in action. For visualisation purposes we depict the tool without (a) and with (b) the selected control points.

We also provided the functionality to save and restore groups for reuse in the current or other key-frames. Furthermore, control points can be restricted to be exclusively part of only one group.

Transformation

We also included support for performing transformations on parts of the drawn character. Existing applications transform drawings on a per-pixel basis which results in artefacts because the transformed parts are cut out and then pasted at a new position. In our case the transformation tools (translate, rotate, scale, ...) only affect the control points selected by the grouping tool and so the animator has local control over the drawing while preserving the continuity

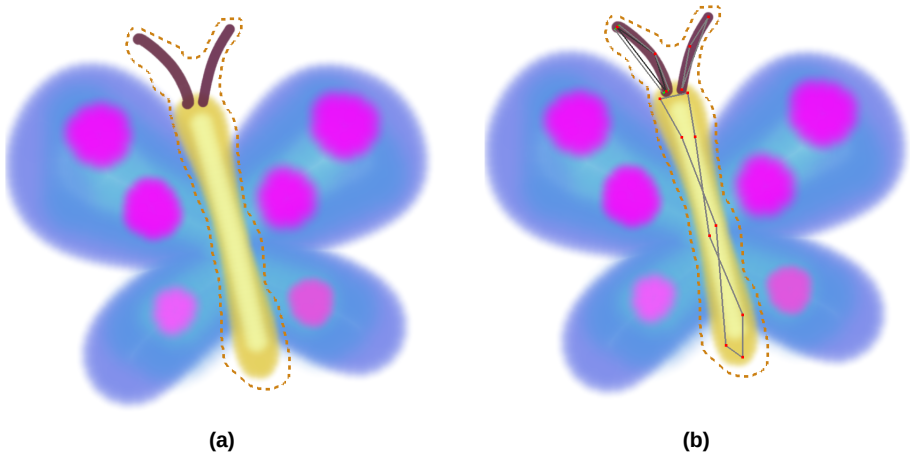


Figure 7.8: Grouping tool in action. a) Upon selecting (no control points are shown). b) Upon selecting (only the relevant (i.e. currently selected) control points are displayed). Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

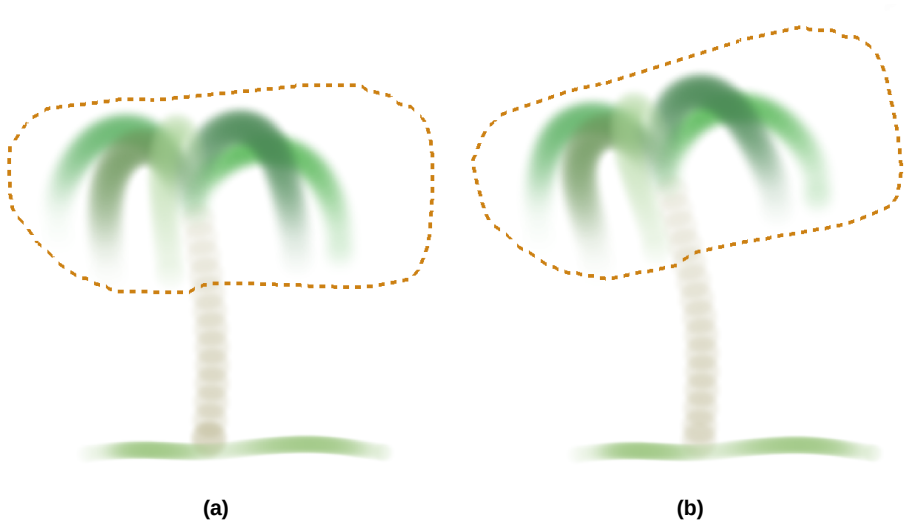


Figure 7.9: Transformation tool in action. a) Untransformed palm tree. b) The same palm tree after rotating a selected part of the image. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

and connectivity of brushes. This is illustrated in figure 7.9. As one can see, rotating the leafs of the palm tree does not create a hole between the rotated and fixed part. Instead, the brushes that make out the trunk of the palm tree stay connected to the leafs and are even lengthened in order to preserve the continuity.

Deformations

In this section we explain the use of a free-form deformation tool as a means to easily and rapidly manipulate (deform) a selected part of the character in a free-form manner. Existing free-form deformation applications usually impose an evenly spaced N-dimensional grid of deformation control points $G_{d_1,..,d_N}$ on the character. Then, the user has to manipulate these deformation control points in order to deform the character. For the 2D case this deformation grid always is rectangular.

This way of deforming imposes major constraints on the animator's way of working. Manipulating the deformation control points is not intuitive at all since it is hard to estimate how the movement of a single control point will affect the character. Furthermore, because of the rectangular shape of the deformation grid, one only can deform rectangular parts of the object and so it is difficult to have local control. The most important issue, however, is that existing free-form deformation applications are pixel based and do not take the underlying control curves into account. As a result, connectivity and continuity of the brushes may be lost.

For the second issue, in our approach, the grouping tool is used to select the part of the character to be deformed. Hence, all control points belonging to the selected group are marked as deformable. Then, our system calculates the bounding box surrounding the deformable control points. The resulting bounding box now acts as an deformation grid of which each dimension d gets divided in X_d cells which is a user adjustable parameter. The advantage of this approach is that only the selected part of the image (enclosed by the grouping tool) is eligible for deformation, despite the rectangular shape of the deformation grid. As a result, the animator is assured of having local control. Also, taking into account the new positions (due to the deformation) of the control points all affected brushed are regenerated and therefore the connectivity and continuity are preserved.

Regarding the first issue, in chapter 4 we presented a sketching tool that enables the animator to intuitively perform free-form deformations just by drawing two strokes: one that represents the initial state of the object, and a second one that indicates how the object should be deformed. The same

method can be employed to manipulate our previously generated deformation grid.

As a result, the combination of the grouping tool and generation of a deformation grid together with the use of an intuitive tool makes the direct manipulation and visualisation of the brush's control points and deformation grid superfluous: all of this is hidden and happens transparently for the user.

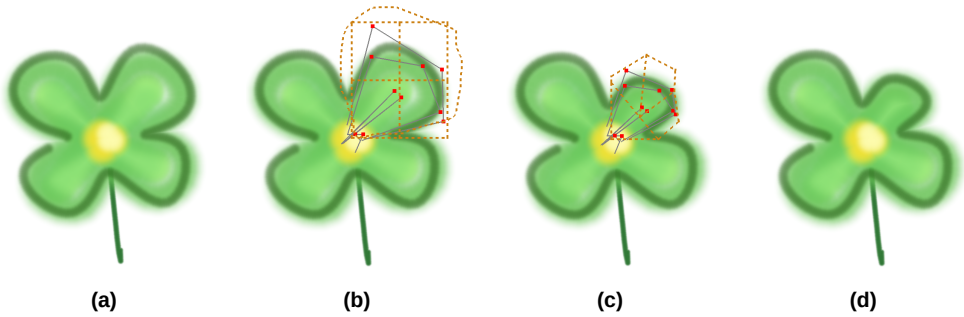


Figure 7.10: Free-form deformation tool. a) Before deformation. b) After selecting a part to be locally deformed (we also displayed the lasso tool, the control points that will be affected by deformations, as well as the deformation grid). b) After deformation (with deformation grid and control points shown). d) After deformation. Note that the deformation has only affected the selected part of the image while preserving the connectivity and continuity of the brushes. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

Figure 7.10 illustrates our free-form deformation tool. For illustrative purposes we also depicted the selected control points as well as the generated deformation grid. As one can see in figure 7.10(c) our deformation tool has only affected the selected part of the image (flower's leaflet) while preserving the connectivity and continuity of the brushes.

Figure 7.11 gives an example sequence of extreme frames created using our free-form deformation tool.

Hierarchical Structures

We can interconnect several groups of brushes by defining an anchor point for each group. This allows the user to create a hierarchical model of a character drawing. Such a model can be used for animating lively characters. Obviously, this kind of control can also be used in combination with forward and inverse kinematic tools or other physics based approaches to control the movement of a character in a more physical way.

Figure 7.11: a) Original extreme frame. b–c) New extreme frames after deforming the original image. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

7.4 Results

Figures 7.12 and 7.13 depict an overview of the method of working. Figure 7.12 indicates the process of creation/refinement, while figure 7.13 shows a mixture of extreme and in-between frames.

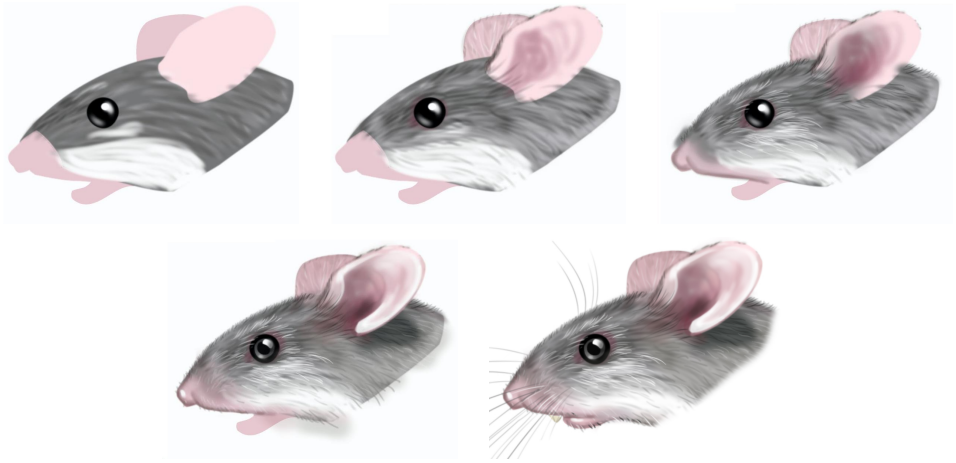


Figure 7.12: Overview of the painting process. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

Figure 7.14 depicts some snapshots of a moving head. For this animation, a straight-ahead approach was taken: only extreme frame was modelled from scratch, the others were made using the deformation tool.
The pictures in figure 7.15 display a rotation out of the drawing canvas.
Through paint and air brushing, our system also easily lends itself to different

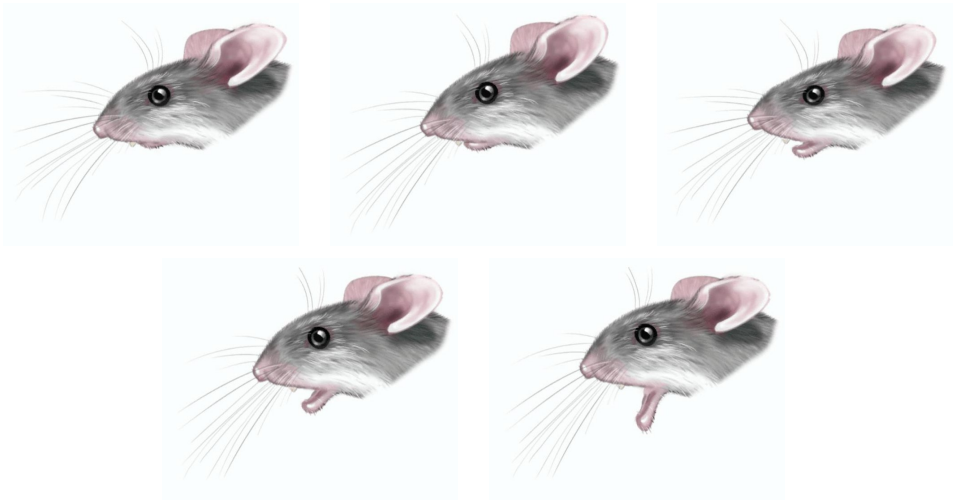


Figure 7.13: Mixture of extreme and in-between frames. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

kind of styles. This is depicted in figure 7.16.

Finally, we end this section with an image sequence taken from a small movie, created using our technique. Figure 7.17 shows the animator's preparation of the excerpt displayed in figure 7.18.

7.5 Discussion

In this chapter, we presented a novel approach to assist the animator throughout the time-consuming process of traditional painted animation. To be more precise, the creation, manipulation, managing and animation of the numerous strokes that are painted one over the other to make up the characters.

The provided solution is intuitive to use and empowers the production of painted animation while not hampering the animation artists' creativity.

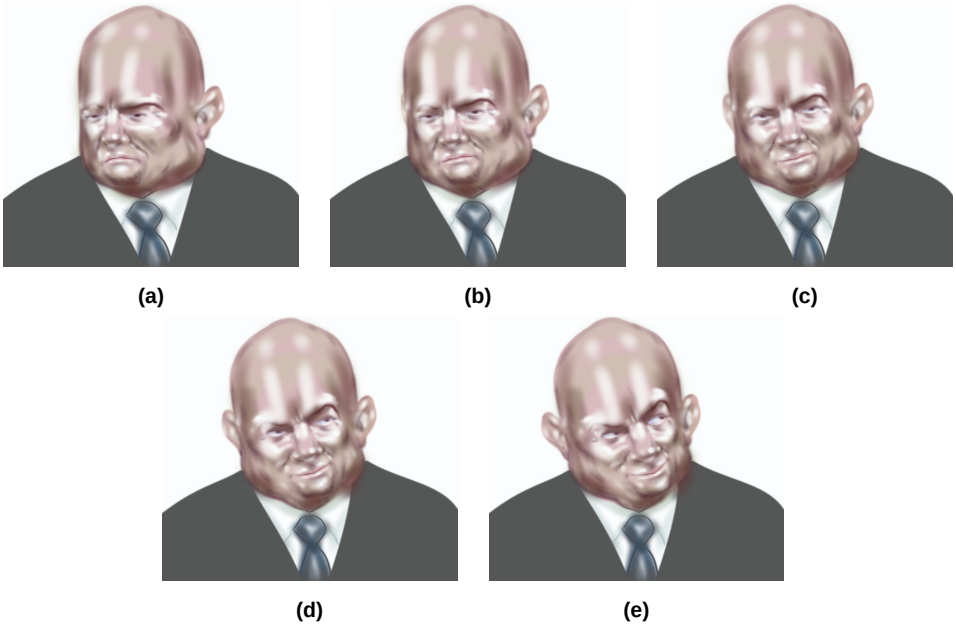


Figure 7.14: Stylised animation of a man turning his head. a, c, e) Extreme frames. b, d) In-between frames. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

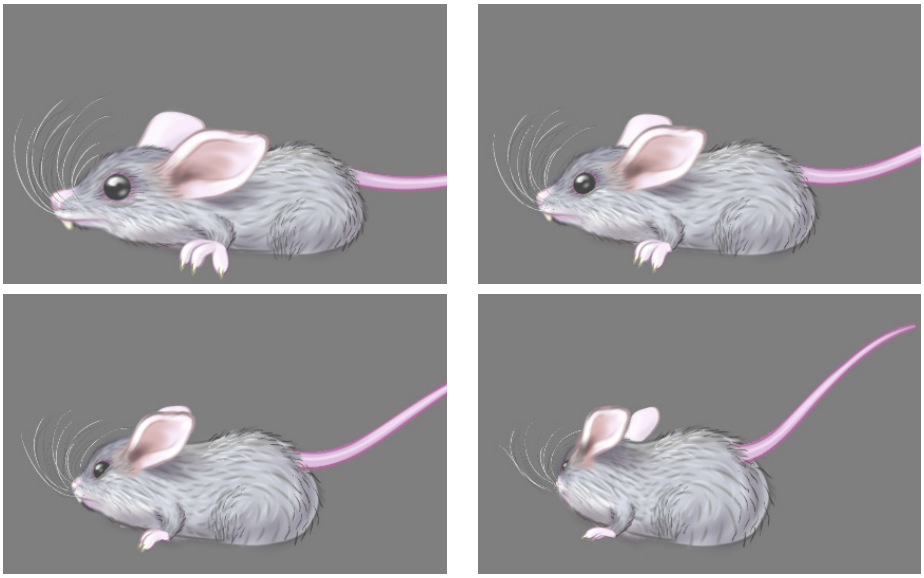


Figure 7.15: Stylised animation of a mouse rotating out of the drawing plane.

Figure 7.16: Illustration of different styles. a) Cartoon style. b) Washed out drawing. c) Airbrushed. d) Manga character. e) Pen drawing. f) Charcoal drawing. Copyright © 2004 ANIMANTE/Xemi Morales/Ricardo Puertas (ANIMANTE 04).

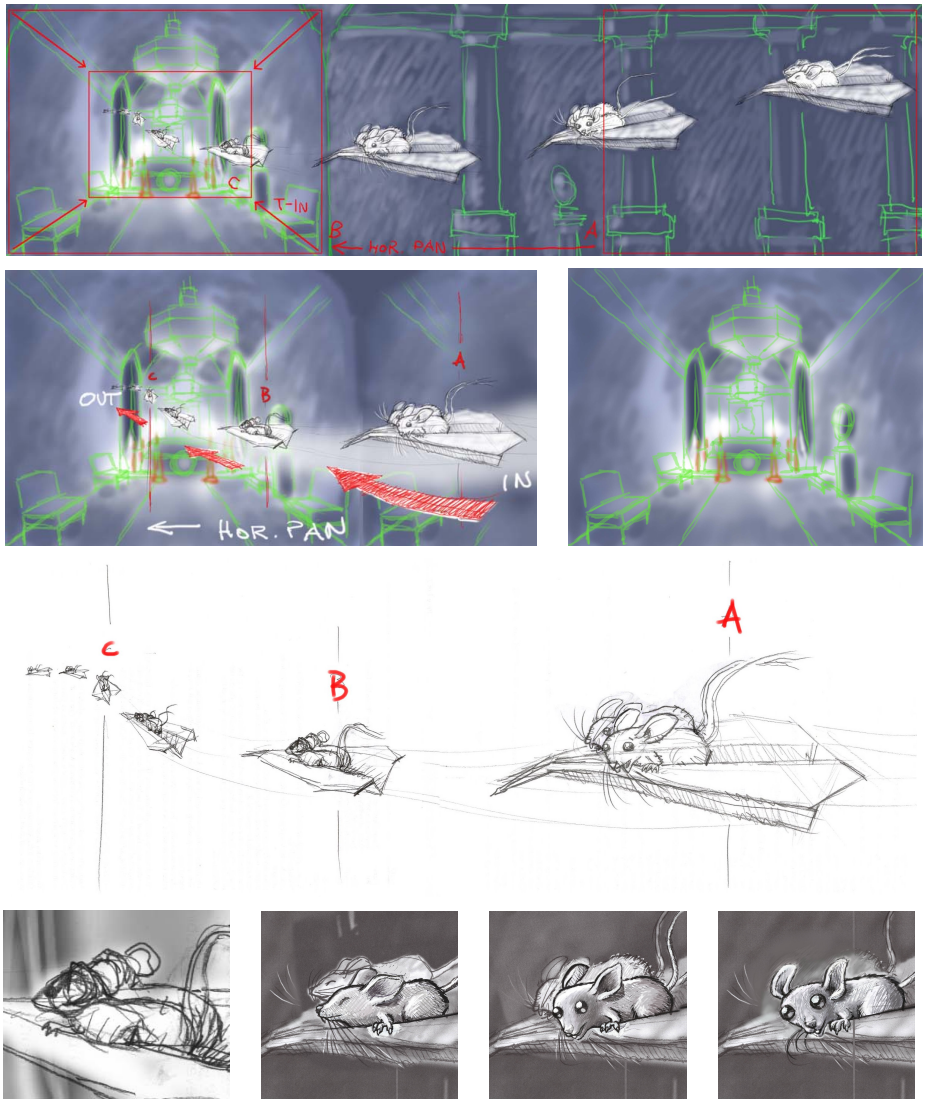


Figure 7.17: Animator's preparation of the movie showed in figure 7.18. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

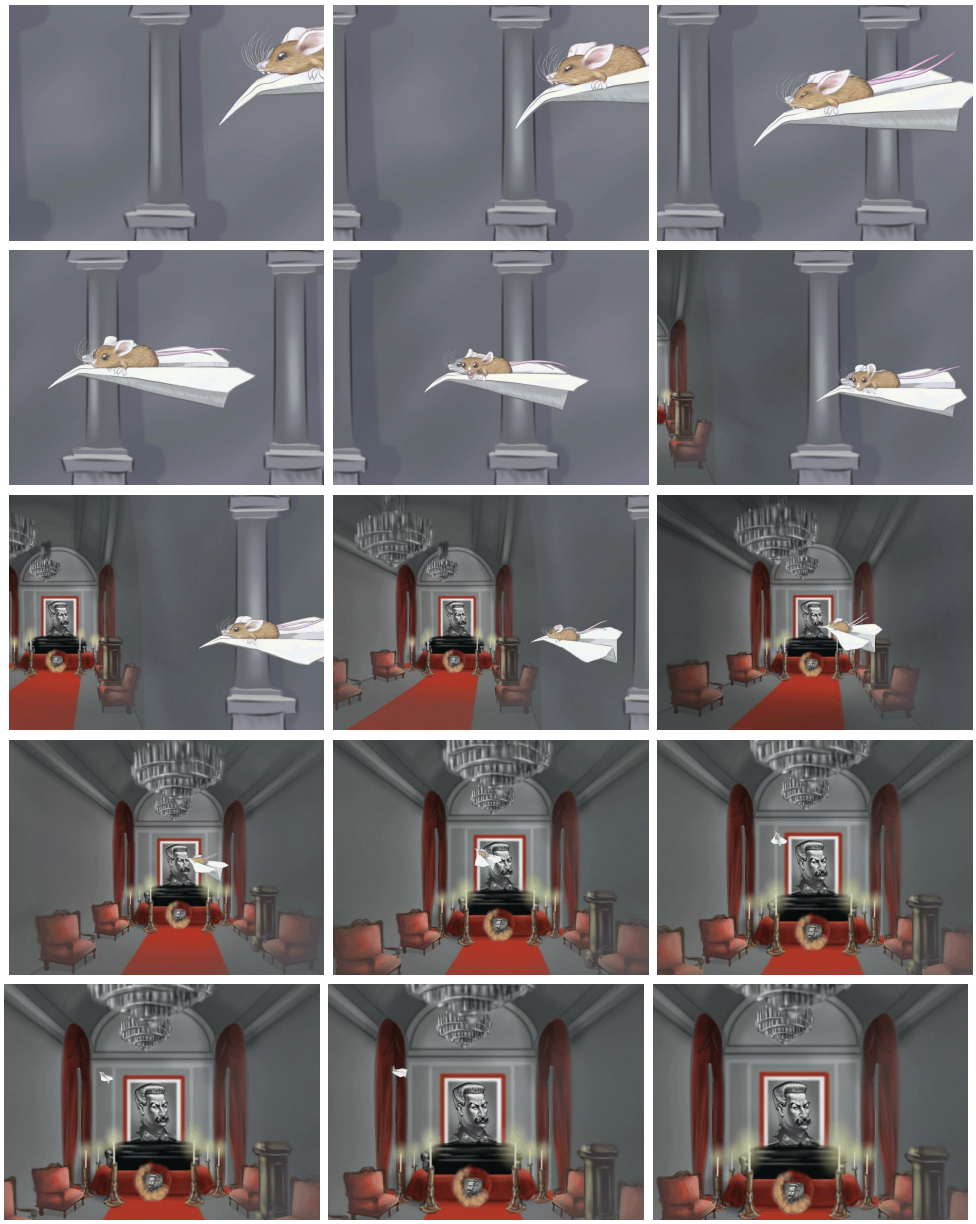
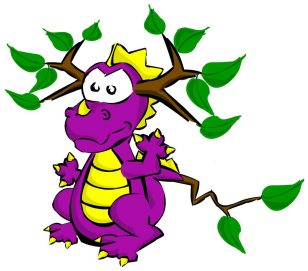


Figure 7.18: Excerpt of a stylised 2D movie. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

Chapter 8

Procedural Animation:
Natural Phenomena



"You can't just let nature run wild."

Walt Disney

Contents

In this chapter, we present a novel approach to design artistic and believable trees in a cartoon-like style, which can be rendered by an animated camera to produce a convincing 3D-like experience[1]. While existing computer assisted 2D animation is able to generate the desired effects, it still depends too much on the creation of many hand drawn images and it has difficulties to avoid artefacts popping up in successive frames. On the other hand, existing approaches fully depending on 3D geometries give little artistic freedom.

In order to provide good solutions to these difficulties, we present a hybrid approach, combining the advantages of both 2D and 3D approaches. From an underlying 3D geometry we get the necessary information to obtain an acceptable level of 3D behaviour and a good frame-to-frame coherence, whereas 2D artistic input is employed to obtain any desired 'look', both of the rendering and of the animation.

8.1 Introduction

During the last years, the use of CGI elements in the production of cartoon animations has become a very common thing. Our work deals with the category of CGI effects which classifies the CGI standards as "things that are too numerous or tedious to draw". In our case, the animation of cartoon-like trees. Figure 8.1 shows some snapshots of the kind of animations we are striving after.

Consider for example figure 8.1. From an animator's point of view two technical issues arise. The first issue concerns the animator's ability to picture in mind the objects to be drawn. For characters this is relatively easy, since s/he can rely on his/her ready knowledge. However, this is not the case for trees, due to their complex, recursive structure (what does a tree look like viewed from aside or behind?). The second issue deals with the numerous branches and leaves constituting a tree. For example, when walking around a tree, the branches and leaves may not suddenly appear and disappear, nor move or deform in a way that is not conform to the current animation flow. Without such coherence, the temporal aliasing makes the final animation hard to enjoy.

To tackle these issues, while preserving the animator's freedom, we present a hybrid approach that benefits from existing 2D and 3D approaches. On the

[1] An earlier version of this material was presented in (Di Fiore 03a).

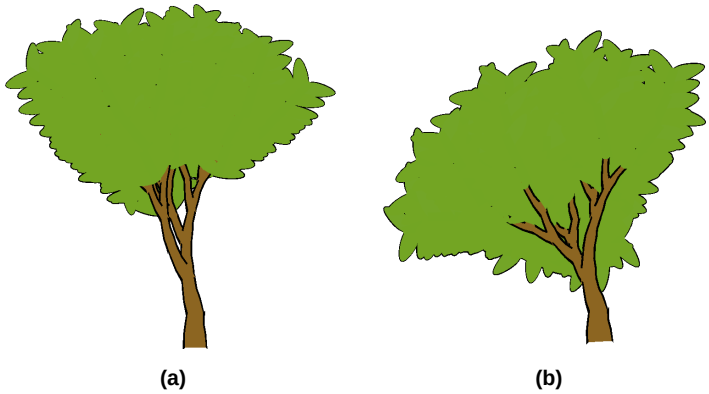


Figure 8.1: Some snapshots of an animated tree. a) Regular view. b) View from below.

one hand, 3D geometrical models are exploited to extract 3D information from, which is necessary for providing frame-to-frame coherence, while in order to preserve the animator's freedom of creativity 2.5D modelling and animation techniques are used.

This chapter is organised as follows. Section 8.2 gives an overview of differences of existing approaches. Section 8.3 elucidates the central theme of our paper, rendering and animating cartoon-like trees. Section 8.5 provides clarifying results. We end with our conclusions (section 8.6).

8.2 Difficulties of Existing Approaches

In this section we dwell on techniques starting from pure 2D drawings and some approaches found in the non-photorealistic domain exploiting 3D geometrical models.

8.2.1 Pure 2D Approaches

For Walt Disney's feature animation *'Mulan'* (Guaglione 98), CGI was used to animate Bamboo plants. Each plant is composed of several 2D layers and this for certain camera angles. Painterly shaders are used to simulate brush strokes in the graduated tones on the leaves. The advantage of this approach lies in the fact that the animator has full control over the final style. However, a major drawback is that the artist still has to paint the plant manually for all camera angles. As a result, it is up to the animator to know how plants

look like from different views, and consequently to provide for frame-to-frame coherence.

In 2000, Cohen et al. described an interactive system, *'Harold'*, that enables animators to create 3D-like animations starting from drawing only 2D objects on billboards (Cohen 00). In order to create a convincing 3D-like animation all planar strokes are reoriented in a view-dependent way as the camera moves through the world. *'Harold'* is very suitable for rapidly creating expressive and visually rich 3D worlds. However, as all billboards are always rotated to face the viewer, it is not suitable for asymmetric objects, such as trees, because their look has to change with every new camera angle.

We can conclude that 2D systems are easy to use and deliver good-looking results. However the drawbacks are that they either still involve a lot of work or suffer from too many constraints.

8.2.2 Starting from 3D: Non-photorealistic Rendering Techniques

The painterly rendering algorithm presented by Barbara Meier leads to very impressive results for rigid objects, but requires extensive modelling and animation if it were applied to fully animated trees (Meier 96).

Also the work of Kowalski et al. (later extended by Markosian (Markosian 00)) is suitable for preventing temporal aliasing (Kowalski 99). However, the drawbacks include that animators still have to create script files manually to define looks and behaviours.

More recently, Deussen & Strothotte presented a method of rendering pen-and-ink illustrations of trees automatically (Deussen 00). First, they start with detailed 3D tree models consisting of a tree skeleton and leaves. Then, a computer-generated pen-and-ink illustration is achieved by applying existing NPR-techniques (Gooch 01; Strothotte 02) to the tree skeleton and by drawing the leaves using abstract drawing primitives. This approach enables animators to generate illustrations with different drawing styles and levels of abstraction. Yet, with respect to the creation of cartoon animations, two important issues arise. Firstly, the animator's contribution to the rendering of the tree skeleton is limited since standard NPR-techniques have to be used. Secondly, the used 3D models already contain foliage data and that way the artist's creativity is constrained by the overall shape of the 3D model. Furthermore, coloured versions need also to be developed for the purpose of cartoon animation.

8.3 Modelling Cartoon Trees

From the previous section we learned that both 2D and 3D approaches have their own difficulties. In order to provide good solutions to these differences, we present a hybrid approach combining the advantages of 2D as well as 3D approaches.

This fits perfectly in our 2.5D animation framework: the incorporation of 3D information will be implemented as level 2 functionality while providing explicit 2D modelling information can be handled in level 1.

8.3.1 Incorporating 3D Information

When creating animated characters, it makes sense to let an animator hand-draw the extreme views of the whole character from scratch. That is because an animator relies on his/her ready knowledge of the 3D character s/he is about to draw. The same idea counts when one is animating more complex characters (e.g., a walking dog) where the difficulty lies in retaining the proportions and overall volume of the character. Here the animator can be guided by means of rapidly created approximate 3D models (chapter 5)

However, since trees are far more complex than everyday characters these approximate 3D models are unsuitable. This is due to the complex, recursive structure of trees which is extremely hard to model starting from only 2D rounded forms.

For those reasons, the underlying models in our approach are *realistic* 3D geometries of trees. That way we can incorporate necessary 3D information in order to preserve the overall shape, as well as to ensure frame-to-frame coherence. This is further explained in section 8.4.1.

In traditional animation (Blair 94) an artist always draws the outlines of the tree skeleton but never draws individual leaves. Instead, leaves are grouped together per branch or per tree. For that reason, we use simple tree models consisting only of a tree skeleton. As will be clear from section 8.4.3 this information is also sufficient to draw the foliage.

For research purposes, the underlying models are realistic 3D geometries generated by L-systems (Prusinkiewicz 90) (figure 8.3), but other approaches (e.g., (Lintermann 99)) for generating the necessary 3D information can be used as well.

The Representation of Plants using L-systems

An L-system is a parallel string rewriting system that constructs a generally more complex string of characters from a less complex string by using produc-

tion rules. A production rule is defined as follows:

$$label : lc < pred > rc : cond \rightarrow succ : prob$$

label: a label to mark the production rule.
lc: (optional) a left context after which the predecessor must follow to make the rule applicable.
pred: the predecessor, the character from the original string that is to be replaced by a more complex sequence of characters.
rc: (optional) a right context in front of which the predecessor must be positioned to make the rule applicable.
cond: (optional) a condition which must evaluate to true, based on parameters from *lc*, *pred* or *rc*, to make the rule applicable.
succ: the successor, a string of characters to replace the predecessor.
prob: a stochastic value, indicating the probability for this rule to be selected when several rules are applicable.

Starting with an axiom (a begin sequence) the L-system replaces each of the characters (in parallel) when using these production rules. Each of these symbols needs to get a proper visual interpretation, to obtain a visual model. A well known technique to do this is called "Turtle Interpretation" (Prusinkiewicz 90). In this case the string of characters acts as a sequence of changes, applicable to the state of a drawing tool (a pen, the "turtle", ...). Each symbol represents a specific change to the state. At any time, the state is characterised by a position p and three mutually perpendicular orientation vectors H, U and L, indicating the heading, the up direction and the direction to the left. Symbols like $+$, $-$, &, $\wedge$, \ and / rotate the state (see figure 8.2). Other symbols can move the state to a different position, for instance when drawing a branch. More information on how to create and use L-systems can be found in (Prusinkiewicz 90; Prusinkiewicz 94; Prusinkiewicz 96; Prusinkiewicz 97).

Měch and Prusinkiewicz introduced a variant of L-systems, called "open" L-systems (Měch 96), which contain a mechanism for incorporating "external" data in production rules. Special symbols are introduced in the production rules. After rewriting a sequence, the symbols are interpreted and replaced by associated parameter values such as lengths and rotation angles. This way, specific alterations can be made to the model to take into account the requirements of the environment.

We used open L-systems as generated by Van Haevre et al. (Van Haevre 03; Van Haevre 04). Figure 8.3 shows some input models that we use.

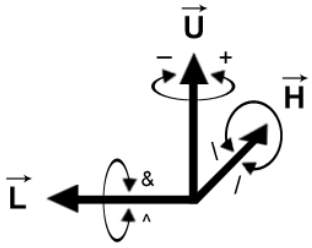


Figure 8.2: The state of the drawing tool needed for graphical interpretation of a character sequence is characterised by a position p and three mutually perpendicular orientation vectors H (heading), U (up) and L (left). $+$, $-$, $\&$, $\wedge$, $\backslash$ and $/$ rotate the state.

Figure 8.3: 3D rendered images of our input models.

8.3.2 Explicit 2.5D Modelling Information

In this section we show how we preserve the animator's freedom to express the artistic style s/he is bearing in mind.

One of the other purposes of the approximate 3D models in chapter 5 is to ink the outlines of the desired object rapidly by tracing silhouette lines and marker lines. This works well when animating characters since these can usually be built from multiple approximate 3D objects. That way, tracing all outlines of all approximate objects soon results in a 2D layered model which perfectly fits in our 2.5D methodology.

Nevertheless, letting the animator create a layered model is nearly impossible (which of all the branches comes first or last or in-between?).

So instead of modelling the whole tree at once, we have the basic parts of a tree (branches and foliage) drawn by the animator independent of the input model. The animator has only to draw one single branch and one piece

of foliage. The more branches and foliage are drawn, the more variation is achieved. In this way, the animator creates a kind of repository which does not depend on the underlying model and thus is highly reusable.

Figures 8.4 and 8.5 show some extreme frames of foliage and a branch as the animator bears them in mind. Notice that both the branch and foliage are geometrically incorrect when compared to real life. However, it is exemplified clearly that it is entirely up to the animator to choose the artistic style for every component of the final animation.

Figure 8.4: Some extreme frames of foliage.

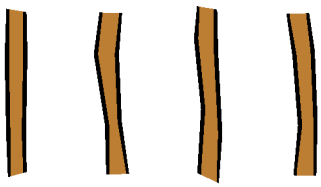


Figure 8.5: Some extreme frames of a branch.

Several different kinds of foliage are depicted in figure 8.6.

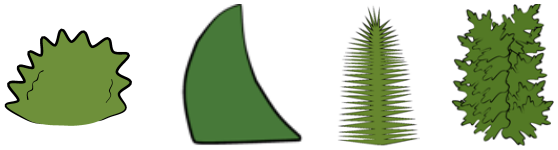


Figure 8.6: Some examples of foliage. a) Deciduous. b) Pine. c) Spruce. d) Oak.

8.4 Animating Cartoon Trees

In this section we show how animated cartoon trees can be created starting with a realistic 3D model and a repository of extreme frames of branches and foliage.

8.4.1 Overview of the Animation Process

This section gives an overview of the animation process for which the pseudo code is depicted in listing 8.1.

Preprocess(model m)	**Draw**(frame f)
//build hierarchy tree of branches	**for** each branch, b^{3D}, **do**
for each encountered branch, b^{3D}, **do**	calculate average depth
assign random 2D branch, b^{2D}	**end for**
if (level l >= FOLIAGE) **then**	sort branches by depth
assign random 2D foliage, f^{2D}	**for** each sorted branch, b^{3D}, **do**
end for	**DrawBranch**(b^{3D}) on top
end for	**DrawFoliage**(b^{3D}) on top, **if** present
	end for

Listing 8.1: The drawing process of a tree.

The pre-processing step, which is executed only once per session, parses the geometry model and builds a hierarchy tree of branches. During the reading process, for each branch b^{3D}, the animator's repository is traversed for a random 2D branch, b^{2D}, and random 2D foliage, f^{2D}. From now on, the 2D branch, b^{2D}, is attributed to its 3D counterpart. This is fundamental in providing frame-to-frame coherence. The same happens for f^{2D} if the current level in the hierarchy tree is eligible for having foliage. This is controlled by a user defined parameter that defines the lowest level in the tree on which foliage is present.

For each frame of the actual animation, we first of all determine the current drawing order of all branches and foliage since we are working with 2D objects which contain no depth information. This drawing order can directly be derived from the 3D positional data of each b^{3D}. Finally, the branches and foliage are drawn one after another, to begin with the back ones.

The artist can configure a level of detail, depending on the particular effect s/he wants, to achieve a specific animation. At the lowest level, only the outlines of the global foliage are shown while at the highest level outlines are shown for the most prominent parts of the tree. The different results are discussed in section 8.5. The drawing of branches and foliage itself is explained in sections 8.4.2 and 8.4.3.

8.4.2 Drawing the Branches

Listing 8.2 shows the algorithm for drawing a single branch.

Consider an arbitrary branch, b^{3D} of our 3D tree model at frame f of the animation.

DrawBranch(branch b^{3D})
calculate orientation b^{3D}
generate in-between 2D branch, b^{2D}
align b^{2D} parallel to upstanding axis
scale width of b^{2D} according to age
scale height of b^{2D} to screen length of b^{3D}
position b^{2D} in screen space
connect b^{2D} to its parent

Listing 8.2: The drawing process of a branch.

Given the coordinates of its position and orientation relative to its parent, we first calculate its orientation in world space. That is, the absolute rotations around the vertical (Y) and horizontal (X) axis are retrieved. This information is then streamed to our in-betweening algorithm which creates a 2D branch, b^{2D} (e.g., figure 8.7(a)).

At this stage, our system demands that the generated branch is aligned with the upstanding axis. The top and the bottom of the branch need to be flattened as well. That way, the remainder of the algorithm can be fulfilled in a more simple way and guarantees better preservation of the animator's style. In order to align a branch with the upstanding axis, the first step is carried out by the animator during the modelling stage. As can be seen in figure 8.5, branches are modelled vertically. However, animators can make mistakes and consequently the in-betweening process can introduce some deviations. As a final step, our system refines the alignment by rotating b^{2D} around the Z-axis so that the virtual line, which connects the upper and the lower midpoints of the branch, is parallel to the upstanding axis. To flatten the branch, a simple operation suffices which involves only translations of the control points along the curve they belong to. Figure 8.7 shows a generated in-between branch b^{2D} (a) and its aligned version (b). Note that aligning the branch does not hamper the animator's style since it involves only affine transformations.

When looking at real trees, one notices the different widths of the branches: older branches are thicker than younger ones. So in order to create a believable, attractive tree, we scale the width of each b^{2D} according to b^{3D}'s level l in the hierarchy tree. This level corresponds to the age a_l of a real branch. We attribute age a_l to the start of b^{2D} and a_{l-1} to its end. Since each age a_l is associated with a fixed width w_l we are assured of creating a tree becoming narrower when approaching the top. For the scaling of the width itself we can take advantage of the aligned branch since now we just have to rotate the two vertical curves of the branch around the Z-axis in order to fulfill following

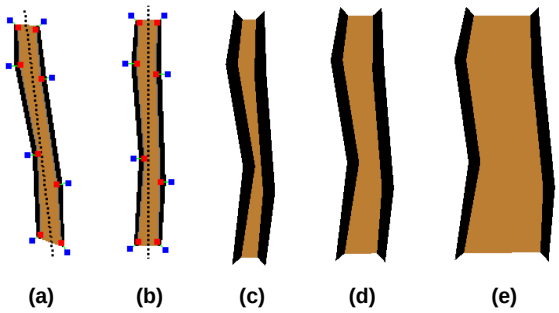


Figure 8.7: a) A generated in-between branch. b) After alignment. c–e) At different ages.

requirements:

$$D_x(lower\ left\ point,\ lower\ right\ point) = w_l$$
$$D_x(upper\ left\ point,\ upper\ right\ point) = w_{l-1}$$

Figures 8.7(c–e) show our branch at different ages. Note that the animator's style is preserved since only rotations around the Z-axis are performed.

Next, we scale b^{2D}'s height so it equals the length of b^{3D} in screen space. After that, we give it the same direction as b^{3D} and position it into the right place. This again requires only affine translations and Z-rotations.

Finally, in order to have a smooth transition from a parent branch to its child branches, we have to connect them in a way which is visually aesthetic. For parent branches with only one child it suffices to connect the lower outer points of the child branch to the upper outer points of its parent. For parent branches with two (or more) children, we first search the outermost children in screen space. Then, the lower left point of the most left child is connected to the upper left point of the parent and the lower right point of the most right child to the upper right point of the parent.

8.4.3 Drawing the Foliage

The drawing process of foliage (listing 8.3) is a simplified version of the previous algorithm.

Consider an arbitrary branch, b^{3D}, whose level in the hierarchy tree is eligible for having foliage. We first calculate its orientation in world space. This information is then streamed to our in-betweening algorithm in order to create 2D foliage, f^{2D}. Next, we scale f^{2D} so it's height equals the length of b^{3D} in screen space. After that, we give it the same direction as b^{3D} and

DrawFoliage(branch b^{3D})
calculate orientation of current 3D branch, b^{3D} generate in-between foliage, f^{2D} scale f^{2D} according to screen length of b^{3D} position f^{2D} in screen space

Listing 8.3: The drawing process of foliage.

position it into the right place. This requires only affine translations and Z-rotations.

8.5 Results

Figure 8.8 shows some snapshots of an animation of a bare tree. In figures 8.9, 8.10 and 8.12 the same tree is shown, this time with foliage. In figures 8.9 and 8.10 we rendered only the outlines of the global foliage which results in a traditional *'cartoon-ish'* tree whereas drawing the outlines of the foliage per branch causes the tree to look completely different (figure 8.12).
We used the oak model (figure 8.3(c)) to render the images of figure 8.13.

Figure 8.11 consists of images depicting different stylized renderings of the tree in figure 8.3(b). In (a) a less detailed foliage object is used compared to (b). As a consequence, the first image resembles much more a pine whereas the second one portrays a spruce. The effect of the animator's artistic input is even more apparent when looking at (c). Here the animator drew a detailed foliage object while letting the system render the outlines of each foliage object.

The current performance (Pentium III 600 MHz, GeForce 256 DDR) of the rendering process is shown in figure 8.14. The frame rate is significantly lower when rendering the foliage. This is due to the use of subdivision curves to represent every foliage object. This subdivision process occurs per frame for each in-between object and results in concave polygons, consisting of many vertices. These polygons, in turn, then have to be tesselated to ensure correct colouring.

Each example took an unexperienced user only few minutes to model the branches and foliage.

8.6 Discussion

In this chapter, we presented a novel approach to create artistic and believable cartoon-tree animations. Existing computer animation software either employs full 3D models or uses purely 2D approaches. On the one hand,

highly detailed 3D models are rendered in traditional non-photorealistic styles and leave no room for the animator to express his/her artistic feelings. On the other hand, purely 2D approaches require many repeated drawings which is very time-consuming. Furthermore, the numerous branches and leaves often cause the final animation to suffer from temporal aliasing.

We showed how a hybrid technique that takes advantage of the benefits of both 2D and 3D approaches, can be employed to overcome these problems. We successively described how to provide for frame-to-frame coherence by using 3D objects and how to retain the artist's freedom of drawing by exploiting 2.5D modelling and animation techniques.

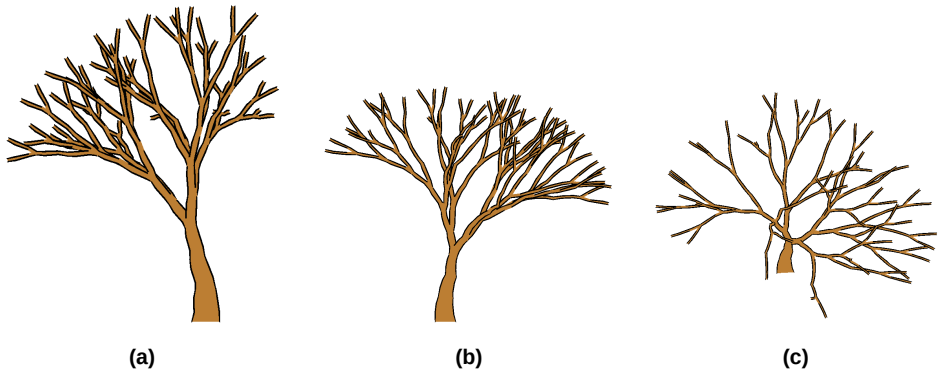


Figure 8.8: Snapshots of an animation of the bare tree depicted in figure 8.3(a). a) Regular view. b) Side view. c) View from above.

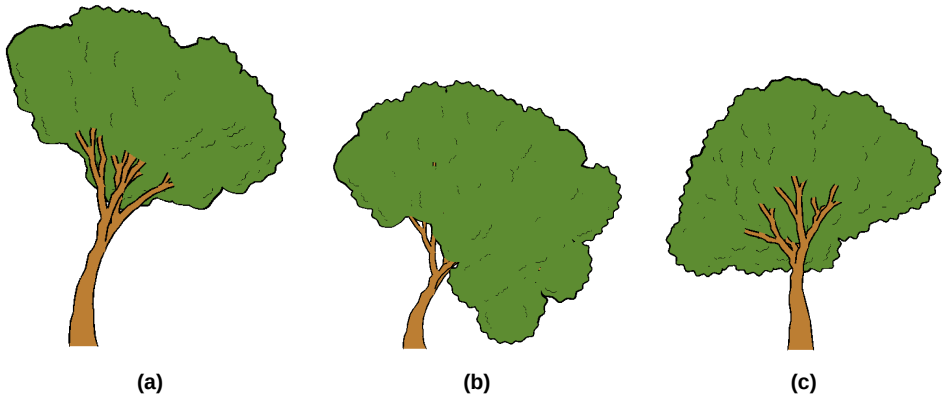


Figure 8.9: These images show the tree in figure 8.8 full of leaves. Only the silhouette of the global foliage is drawn. a) Regular view. b) View from above. c) View from below.

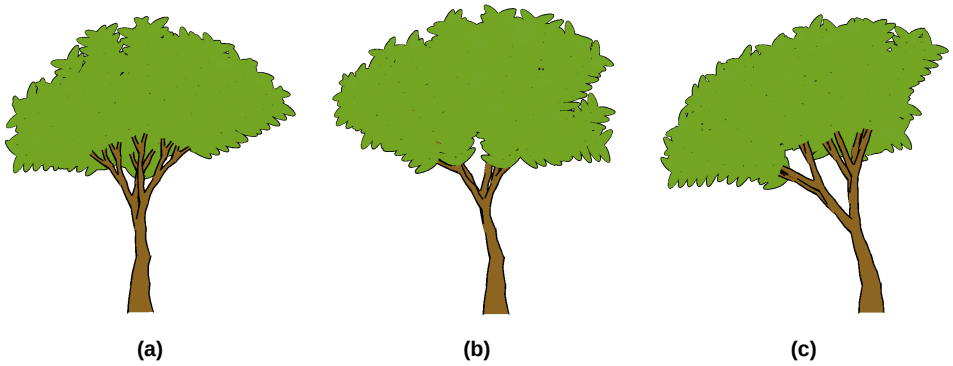


Figure 8.10: A deciduous tree covered with many little leaves. a) Regular view. b) Side view. c) View from behind.

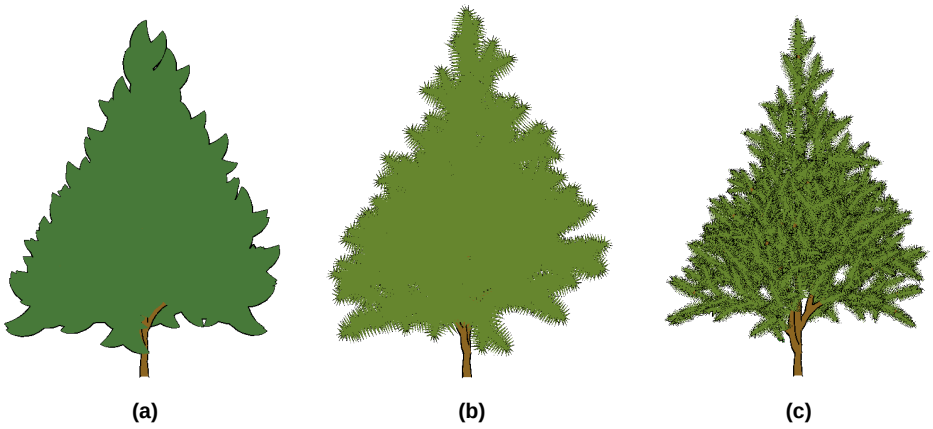


Figure 8.11: These pictures depict different stylised renderings of the tree in figure 8.3(b). a) A pine (less detailed foliage). b) A spruce (detailed foliage). c) Highly detailed spruce (foliage with pine-needles, all outlines drawn).

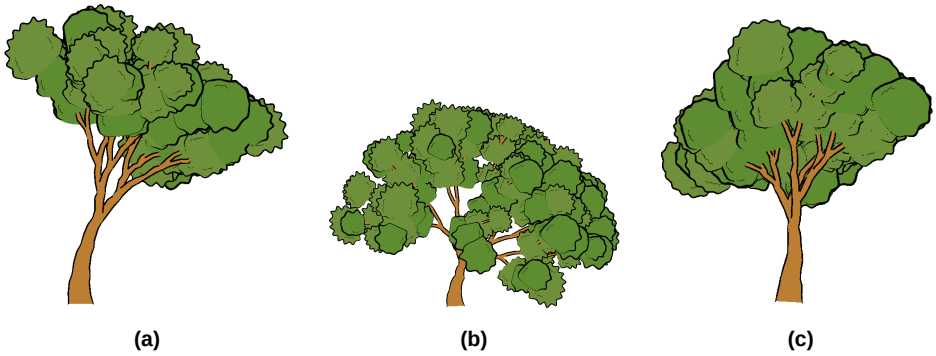


Figure 8.12: The same tree as in figure 8.9. The silhouette of each local foliage is drawn which results in a savannah tree. a) Regular view. b) View from above. c) View from below.

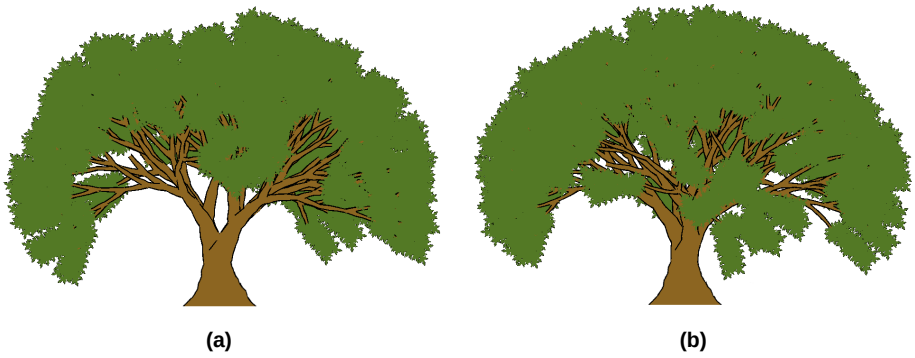


Figure 8.13: An old oak tree. a) Regular view. b) Side view.

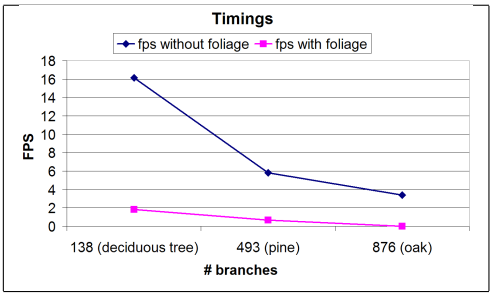
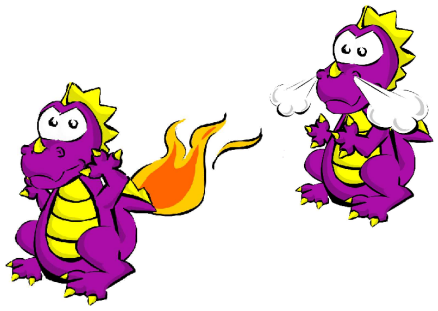


Figure 8.14: Performance overview.

Chapter 9

Procedural Animation:
Gaseous Phenomena



"We will either find a way, or make one."

Hannibal

Contents

In this chapter, we present a novel approach to create stylised animations of gaseous phenomena[1]. Existing computer-assisted systems employ computational physics-based approaches which generate 3D effects. Unfortunately, these effects usually don't look like traditional animation, nor can the user freely design the behaviour of the animation.

Our approach combines benefits from existing 2D and 3D approaches. Extending the application of particle systems, the particle paths vary both with time and with the 3D viewing direction. Rendering features such as smoothly varying common outlines and speedlines help to preserve the animator's artistic style. The incorporation into our structured 2D modelling and animation environment enables a stylised animation exhibiting a convincing frame-to-frame coherence and allows 3D camera movement without aliasing artefacts.

The provided solution demonstrates how an animator can remain in full control of a stylised process for effects animation and how one framework is suitable for a wide range of effects.

9.1 Introduction

Traditionally, stylised animations exhibit objects and characters with a hand-drawn look and resemble real life without trying to be an exact copy of reality. However, characters and objects are only parts of stylised animation; animated effects such as water, fire and smoke add realism — in a stylised manner — and make the whole more vivid.

The production of high-end feature films has sufficient resources enabling dedicated programmers and animators to work closely together in an elaborate process of trial and error to achieve outstanding effects. In our research, we look for solutions to be used in smaller-scale productions where animators have to find their way more independently. Aside the traditional animation effects, we also target stylised effects for virtual worlds.

Nowadays, for our targeted kind of productions gaseous phenomena are either drawn frame by frame (usually in short loops) or one has to revert to 3D computer generated effects and place these effects as layers over the main scene (Witting 99). Our goals are quite different, as we want to offer animators full control over every aspect of a lively animation and to integrate these effects seamlessly into the rest of their artwork.

To establish these goals, we present a hybrid approach that combines ex-

[1] An earlier version of this work is described in (Di Fiore 04).

isting 2D and 3D techniques. On the one hand, necessary 3D information is incorporated by letting the animator draw the animation paths and acceleration curves. This is necessary to provide for frame-to-frame coherence. On the other hand, structured 2D modelling and animation techniques are used to preserve the animator's creative freedom and to create convincing 3D-alike animations starting from pure 2D information.

This chapter is organised as follows. We start with an overview of related work on approaches involving explicit user control, as well as approaches exploiting physics based fluid dynamics. Then, the central theme of our paper, rendering and animating stylised gaseous phenomena, is elucidated. We end with some clarifying results and our conclusions.

9.2 Previous Work

In this section we review existing techniques that are currently used to create gaseous animations and indicate the differences with our philosophy. We start with techniques that aim at putting the animator in charge of the animation process, over physically-based realistic animations, to approaches that simulate realistic behaviour.

9.2.1 User Controlled Behaviour

In stylised animation (and many other cases), realistic behaviour is not always desired, but there's a need for fake, yet very impressive or dramatic effects (Barzel 97).

An appealing procedural paradigm for stylised animation of gaseous phenomena is due to Jinhui Yu (Yu 94; Yu 96a; Yu 99). He constructs cartoon effects by simulating the hand-drawing process (relying on reverse-engineering animation practices) through synthetic, computational means. Although this system emulates the stylish appearance and look and feel of traditional animation, little freedom is preserved for the animator: all cartoon effects are bound by fixed procedures with no possibility to influence the behaviour. Consider for example some smoke puffs going up and dissolving in the air. In this system, the trails of smoke follow a predefined sinusoidal function while each smoke puff is a fixed model of arcs with variations in position and size.

For the animated feature film *'The Prince of Egypt'* Patrick Witting presented a computational fluid dynamics system to produce smoke, water, and other effects (Witting 99). Images or animation sequences are used to initialise temperature fields which cause dynamic buoyancy-driven vortices to

evolve. Initially the animation is driven by the animators, since they are responsible for supplying these images. A major drawback, however, is that this freedom only counts at the start of the animation as the animator is limited to set up the system (i.e. supplying one or more initial images) after which physically accurate equations — time-dependent compressible Navier-Stokes — take over. As a result, the animator only has control at the start of the animation while the rest of the animation is guaranteed to be 'too' realistic. Also, this approach is less suited for a stylised cartoon look, as the sharp outlines get smoothed out by the diffusion process.

For another feature film, Lamorlette and Foster published a system to control the behaviour of physically based fire (Lamorlette 02). They also argue that physics-based simulations are not well fit for an artistic environment, as such simulations are too slow and difficult to control. Hence, they developed a fire animation system with eight stages which is built up from single flames modelled on a natural diffusion flame. This tool is effective because many stages can either be directly controlled by an animator or driven by a physics-based mode. However, the focus of this tool is on just one particular gaseous phenomenon and involves a quite elaborate workflow.

Another type of animator controlled behaviour was described by Treuille et al., who started from user-specified key frames to direct smoke animations (Treuille 03). There, the animator specifies smoke density and velocity key frames while an optimisation process determines the appropriate wind forces needed to satisfy the constraints. In this approach, the animator has a lot of control; however, this is heavily related to the amount of key frames used. Between every two key frames a set of parameterised forces comes into play to produce a realistic simulation that best matches the animator's goals. Consequently, the only guarantee the animator has is that the animation will pass 'through' the defined key frames, but it is unpredictable between two consecutive key frames.

9.2.2 Physics-based Realistic Behaviour

Computational fluid dynamics can create convincing dynamic gaseous simulations. The Navier-Stokes equations are an adequate model for fluid flow. This set of differential equations governs the motion of a fluid, expressing conservation of mass, linear momentum and energy for general motions.

Foster and Metaxas were among the first to produce nice results of animated gasses and fluids using the Navier-Stokes equations (Foster 97). Jos Stam improved on their results and proposed an implicit, semi-Lagrangian technique for updating the grids, resulting in much simpler computations that

were guaranteed to be stable (Stam 99). These algorithms were extended by many researchers, for example by Fedkiw et al., optimising the techniques for simulating smoke (Fedkiw 01).

Very recently, Selle et al. described a system to render smoke in a cartoon style (Selle 04). They displayed particles generated by a dynamic fluid flow simulation (Fedkiw 01), using depth differences in the image buffer to calculate the position of cartoon outlines, analogous to Deussen en Strothotte's tree rendering algorithm (Deussen 00). They introduced rotated and stretched particles, which indicate the flow better and obtained good-looking results of smoke colliding with a ceiling. However, their method still suffers from various drawbacks: (i) aliasing artefacts can not be avoided (due to an aliased depth buffer), (ii) it is difficult to control varying outline thicknesses, and (iii) viewpoint changes loose frame-to-frame coherence. In our solution, these are not present.

The physics-based approaches referenced above can produce very appealing and realistic results. However, from a stylised animation point of view, an important drawback is that the user has little influence on the behaviour of the animation. One may manipulate some initial parameters (such as viscosity, location, quantity, ...) but it is extremely hard to predict if and how the animation is altered by these changes. Moreover, the high computation and memory costs make these techniques less appropriate for an interactive trial-and-error approach of animators trying to find their best shot.

9.2.3 Simulated Realistic Behaviour

In contrast to taking resort to computational fluid dynamics, a lot of approaches use simpler and computationally less expensive methods.

For instance, King et al. presented a technique to animate amorphous materials on graphics hardware with dedicated texture memory (King 99). Object dynamics are achieved using procedural animation techniques while texture cycling is used to create local and global dynamics.

A lot of other approaches focus on just one particular gaseous phenomenon, often coupled with obliged movements. Dobashi et al. propose a simple, efficient method for animation of clouds (Dobashi 00). Cloud evolution is simulated using a cellular automaton while the dynamics are expressed by several transition rules. Perlin and Neyret extended *Perlin Noise* so that shaders that make use of it can be animated over time to produce flow textures with a swirling quality (Perlin 01). Recently, Raghavachary and Benitez used several existing commercial packages to construct a painterly wall of fire for the animated feature film *'Spirit – Stallion of the Cimmaron'* (Raghavachary 02).

Contrary to fluid dynamics, simple physics-based or heuristic approaches offer the user a certain degree of control and can be rendered quickly. However, often changes to the implementation have to be made in order to really control the animation or to increase the variety of stylised effects. Eventually, one ends up with a collection of autonomous dedicated systems.

To summarise, physics-based algorithms and their parameters are very hard to control and hence can realise only a crude approximation of the fake behaviour the animator tries to achieve. Procedural approaches, on the other hand, deliver the look and feel of traditional animation, but at the expense of the animator's freedom as the fake behaviour is strongly bound by fixed procedures.

9.3 Our Approach

The novelty of our approach lies again in the way how we combine benefits from 2D and 3D techniques. Necessary 3D information — required for frame-to-frame coherence and a correct drawing order — is incorporated by having the animator draw time and view-dependent 2D animation paths and acceleration curves. On the other hand, explicit 2D modelling and animation is used to concentrate both on user control and on flicker-free stylised rendering.

9.3.1 Modelling Gaseous Phenomena

Incorporating 3D Information: Animation Flow

Since many years, particle systems have been employed successfully to get flexible simulations/animations representing gaseous phenomena (Reeves 83). Particles or particle clusters are being moved through the animation space according to certain physics-based laws and directly rendered to the screen. Recently, Ilmonen and Kontkanen defined a *second order particle system* as an extension to the classical particle system in which, besides the visual particles, the particle sources and force generators are subject to the forces as well (Ilmonen 03).

In our approach we employ a variant of a *second order particle system* to represent gaseous phenomena. The physics-based laws being responsible for the motion of the particles are replaced by time and view dependent drawn versions of the animation path (i.e. trails particles should follow) and drawn curves representing the particles' acceleration of speed. This way, the animator is able to create more aesthetically pleasing animations in an easy and rapid way.

For creating an animation path, the animator just draws one or more strokes indicating the flow of particles. In order to achieve convincing 3D-like animations, time and view-dependent versions of the path can be modelled. All these different versions can be regarded as extreme frames (Blair 94) and will be used by our in-betweening method in the animation phase. Figure 9.1 depicts an example of a time-dependent animation path drawn by a user. As one can see, animation paths can take any free-form shape the animator is bearing in mind. The boundaries of this fat free-form shape actually reflect the boundaries and movement of the trails that the visual components are allowed to follow.

Each 2D animation path gets 'inflated' to a 3D trajectory. We use an adaptation of the gesture-based sketching interface (TEDDY) presented by Igarashi et al. in which 3D polygonal objects are constructed by inflating the region surrounding the silhouette, making wide areas fat and narrow ones thin (Igarashi 99). In our approach, we start with two drawn boundary paths in 2D. For each particle, first a path is chosen that is some (randomly generated) percentage halfway between these two drawn boundary paths. Then, this path is also translated in the direction perpendicular to the drawing plane, creating a 3D path. This perpendicular translation is given by another random value, in such a way that the volume containing all paths will form a 3D volume similar to Igarashi's volumes.

So, the trajectories the particles will follow are entirely defined by the animator in a familiar way. Via pure 2D input, necessary 3D information is generated, which helps preserve the overall shape and to ensure frame-to-frame coherence.

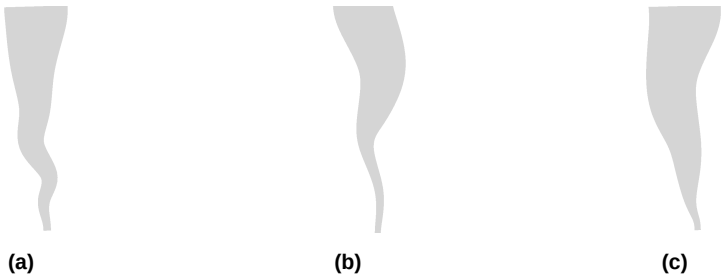


Figure 9.1: Examples of a time-dependent animation path drawn by the animator. a) Time t_0. b) Time t_1. c) Time t_2.

In addition, acceleration curves define how the particles are accelerated on their way along the animation path. This allows the animator better to control the animation and also permits all kinds of natural *and* 'unconventional'

behaving animations which otherwise are too cumbersome or even impossible using existing approaches. Moreover, as the system gives instantaneous visual feedback, the user can 'polish up' the animation at runtime. The acceleration curves attributed to the animation paths of figure 9.1 are shown in figure 9.2.

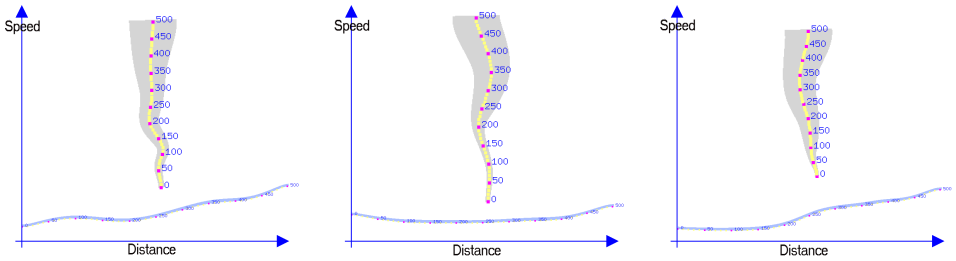


Figure 9.2: Examples of a time-dependent acceleration curve specified by the animator. These curves are attributed to the animation paths of figure 9.1. a) Time t_0. b) Time t_1. c) Time t_2.

Finally, some random 'deviation parameters' are assigned to each particle, including speed deviation, orientation deviation and mass' influence. These 'deviation parameters' prevent the particles' flow from looking artificial. As a result, particles can for example diverge from the trails bounded by the animation path.

Structured 2D Modelling Information: Visual Components

Instead of modelling the amorphous structure at once, we have the basic visual components (flames, drops, puffs, 'speedlines', ...) drawn by the animator independent of the animation path. In this way, the animator draws a stylised repository of the required basic gaseous components — including view-dependent versions — which does not depend on the underlying model and is highly reusable.

Figure 9.3 shows some extreme frames (i.e. combination of viewpoint and moment in time) of gaseous components. Notice that neither of the depicted elements is modelled true-to-nature. Moreover, depending on the extreme frame, the same component can change shape, colour, size, opaqueness, ...

9.3.2 Animating Gaseous Phenomena

We now explain how gaseous phenomena can be animated in a stylised way using 3D information and a repository of gaseous components.

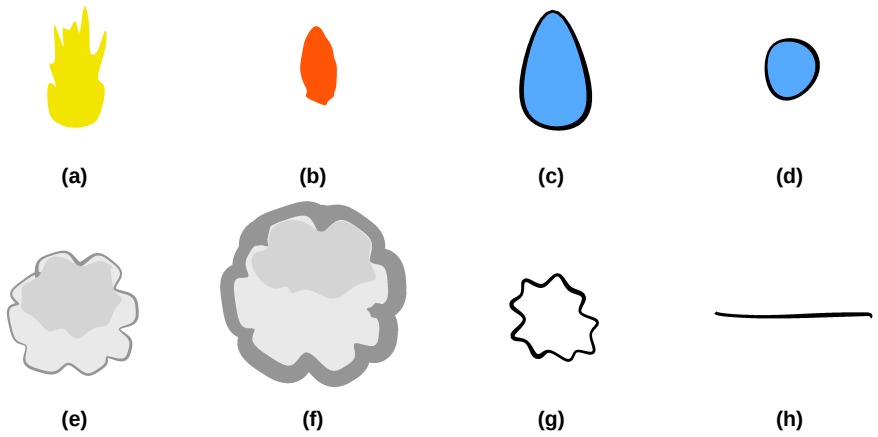


Figure 9.3: Some extreme frames of gaseous components. Note that for illustrative purposes all components are enlarged. a–b) The same flame at different ages. c–d) Drop of water as seen from different viewpoints. e–f) Cloud of smoke at different moments in time. g) Foam. h) Speed line indicating the animation flow.

An overview of the rendering process is given in listing 9.1.

Draw(frame f)
for each gaseous phenomenon, g_i, **do**
  inflate $ap_{g_i}^{2D}$ to a 3D trajectory
  read key frame information
  generate in-between 3D trajectory, $ap_{g_i}^{3D}$
  generate in-between 2D acceleration curve, $ac_{g_i}^{2D}$
  Process Particles(particles $p_{g_i}^{3D}$)
end for
sort all particles $p_{g_1..g_G}^{3D}$
Draw Particles($p_{g_1..g_G}^{3D}$)

Listing 9.1: Overview of the drawing process.

Consider an arbitrary gaseous phenomenon g_i at frame f.

As explained in previous section, before the animation, each 2D animation path first gets 'inflated' to a 3D trajectory.

During the animation, for each frame, we first parse the view-dependent and time-dependent key frame information specified by the animator. Typically, this information consists of a desired orientation, position and age of the animation path. This information then gets streamed to our in-betweening algorithm which creates an in-between 3D trajectory, $ap_{g_i}^{3D}$, and an in-between

2D acceleration curve, $ac_{g_i}^{2D}$.

Next, 3D particles, p^{3D}, are emitted into these trajectories (listing 9.2(top)). N new particles are emitted to replace the ones that 'finished' the animation. This is a user-controlled option that can be used to ensure a continuous fluent animation. For each new particle p_n^{3D}, a random 2D component p_n^{2D} is then taken from the animator's repository. From now on this structured 2D component is attributed to its 3D counterpart for its life span. After all new particles have been added to the current pool of particles, the pool gets updated. The process of updating each particle p_j^{3D} basically resembles the conventional way (Reeves 83) but in which user-defined animation paths and acceleration curves are used instead of physics-based or heuristic methods. Furthermore, we also store the depth, orientation and position for later use.

Once all particles have been processed for all animation paths, we need to determine the current drawing order of all visual components since we are working with 2D objects which contain no explicit depth information. This drawing order can directly be derived from the depth data that we stored for each p_j^{3D} in the processing step.

Next, for each particle, starting from the back to the front, an in-between version of the corresponding 2D gaseous component is calculated (listing 9.2 (bottom)). Take for example particle p_j^{3D}. Given the coordinates of its position and orientation in world space, the absolute rotations in screen space around the vertical (Y) and horizontal (X) axis are calculated. Also, the current point in time is determined. This information (current viewpoint and moment in time) is then streamed to our in-betweening algorithm which creates an in-between 2D particle, p_j^{2D}, that actually reflects the animator's artistic style. After that, we put it into the right place in screen space. This requires only affine translations and Z-rotations. As a result, depending on the viewpoint and moment in time, each p_j^{2D} will change shape, position, orientation, colour, size, opaqueness, ... Finally, the particle/component is drawn on top of the others. As a result, we achieve an animation that does not suffer from temporal aliasing, while reflecting the user's style.

Artists can configure a level of detail, depending on the particular effect they want to achieve for a specific animation. At the lowest level, the outlines of each component are shown while at the highest level outlines are shown only for the most prominent parts. A common outline is achieved by first drawing all the outlines of all the particles that are more or less on the same depth, and afterwards drawing the interior surface of these particles, effectively only erasing the outlines where particles overlap. Note that the particles have been partially depth-sorted to allow a correct treatment of transparencies. This sorting can be achieved in $O(n)$ time, because particles in a user-definable

Process Particles(particles p^{3D})

emit N new particles
for each new particle, p_n^{3D}, **do**
  assign random 2D component, p_n^{2D}
  add p_n^{3D} to p^{3D}
end for
for each particle, p_j^{3D}, **do**
  evolve
  store depth
  store orientation and position
end for

Draw Particles(particles p^{3D})

for each particle, p_j^{3D}, **do**
  read orientation and position
  generate in-between 2D particle, p_j^{2D}
  orient and position p_j^{2D} in screen space
  draw p_j^{2D} on top
end for

Listing 9.2: Top) The processing procedure of particles. Bottom) The drawing process of particles.

depth range are considered together.

9.4 Examples

We used the components (and variations of them) displayed in figure 9.3 to create a set of example animations. These examples contain gaseous phenomena (which are structured 2D models) as well as individual components which in turn are part of a higher-level structured 2D model (i.e. the scene).

Figure 9.4 shows some snapshots of a house with a smoking chimney, taken at different viewing angles. In this example we used components, depicting smoke puffs, which gradually (depending on their age) become larger and fade out. During the animation, the smoke puffs remain individual puffs or merge into one another (using blending operations) (Whitaker 81).

The images depicted in figure 9.6 depict a closer look at the barbecue party shown in figure 9.5.

Some images depicting a garden hose are shown in figure 9.7. We also included the possibility to let procedural approaches (for example, physics-based) take over (part of) the animation. This choice is entirely up to the animator and is fully configurable using the GUI of our program. In this

example, the animator chose the bouncing effect of the water to be controlled by a physics-based procedure: once the particles bounce (i.e. arrive at the end of the trajectory) simple physics rules take over the animation. At the same time, the visual components depicting drops of water are replaced by foam. Note also that some particles have been marked as being 'speedlines': they're always drawn on top of the 'regular' particles. In this case they indicate the flow of the water.

Figure 9.8 illustrates a variety of styles that can be achieved with our approach. It depicts the word 'CARTOON' catching various artistic kinds of fire: C = traditional fire, A = cartoon fire with thick outlines, R = smokey fire, T = bluish fire, first O = cartoon fire with thick outlines, second O = smokey fire, N = traditional fire.

Each example took an unskilled animator only few minutes to model the animation paths and gaseous components. All examples run at an interactive frame rate on a commodity personal computer (Pentium IV 3.06 GHz, RADEON 9000).

9.5 Discussion and Future Work

In this chapter, we presented a novel approach to create stylised and believable animations of gaseous phenomena such as water, fire and smoke. We described how the animator draws view dependent and time dependent animation paths and acceleration curves which provide our system with the necessary 3D information to assure a frame-to-frame coherent animation. Furthermore, we showed how to retain the artist's freedom of creativity by exploiting structured 2D modelling and animation techniques: instead of modelling the amorphous structure (or all the particles) at once, we have the basic visual components (flames, drops, puffs, 'speedlines', ...) drawn by the animator independent of the animation path.

Consequently, our approach shows how an animator can remain in charge of the entire animation process and demonstrates how one mechanism supports a complete range of effects.

We believe this work is significant for its novel contribution to computer-assisted traditional animation since it bridges techniques that either employ purely 2D approaches which are labour intensive and cause temporal aliasing, or 3D approaches which severely limit the artist's contribution.

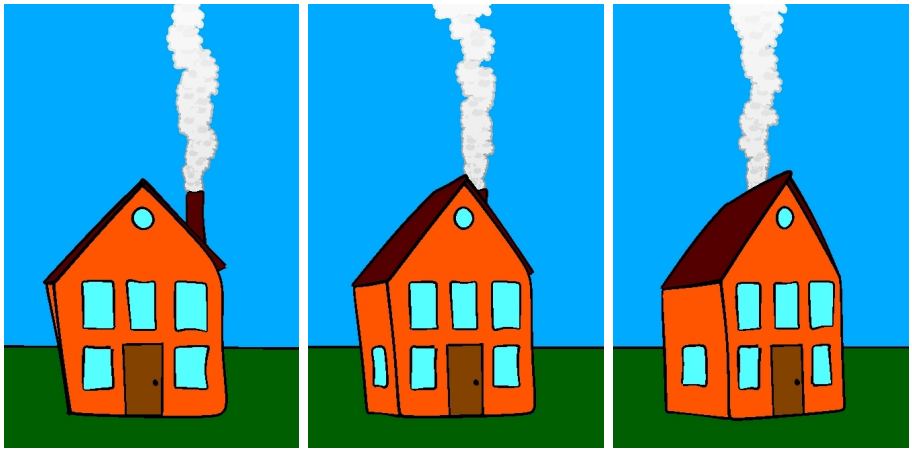


Figure 9.4: Snapshots of a house with a smoking chimney, taken at different viewing angles.

Figure 9.5: Some snapshots of a barbecue rendered in a 'cartoon-ish' style. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

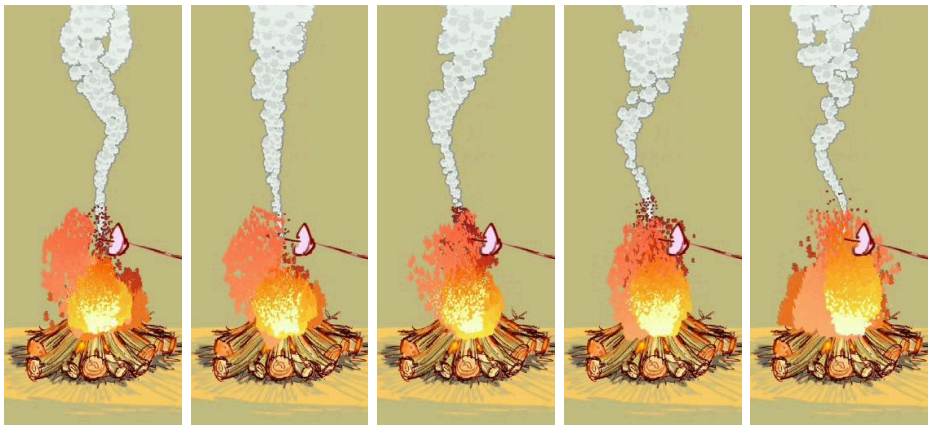


Figure 9.6: These pictures give a closer look at the animation depicted in figure 9.5. Copyright © 2004 ANIMANTE/Xemi Morales (ANIMANTE 04).

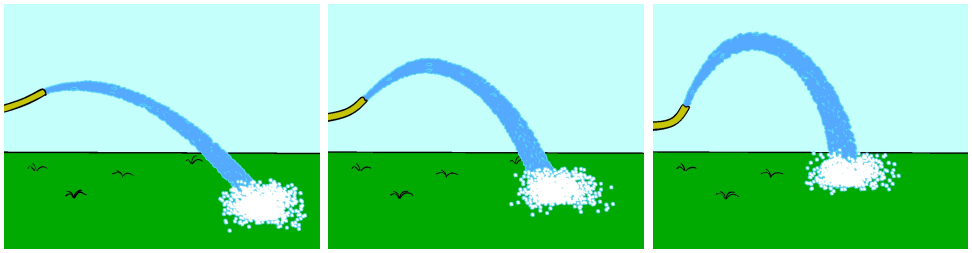


Figure 9.7: These pictures depict a stylised rendering of an animated garden hose. This example also demonstrates the use of 'speedlines' to indicate the flow of the water.

Figure 9.8: a) The word 'CARTOON' catching various artistic kinds of fire: C = traditional fire, A = cartoon fire with thick outlines, R = smokey fire, T = bluish fire, first O = cartoon fire (thick outlines), second O = smokey fire, N = traditional fire.

Chapter 10

Directions for Future Research

"When you're curious, you find lots of interesting things to do."

Walt Disney

Contents

This chapter provides a number of research topics that can be seen as direct extensions of the presented work. We are currently planning to investigate a number of these research topics.

10.1 Faking Dynamics of Clothes

The computer graphics society is striving for realism which is indistinguishable from reality. To this end, usually physics-based algorithms are used.

However, for character animation, animators (and also the viewers) prefer 'fake dynamics' rather than 'real dynamics'. The starting point of our interest is a publication by Ronen Barzel in which he describes the simulation of ropes

and springs using fake dynamics (Barzel 97). Barzel uses a key framed approach in which the shape of a 'flexible-body' model (ropes, strings) is defined by means of layered deformations (wave deformation, rest position, ...).

It is our goal to explore wether it is possible, and how to extend Barzel's approach to work as well with two (or higher) dimensional flexible models. Consequently, extending his approach to two dimensions would allow us, for example, to establish (faked) cloth simulation or hair rendering (Bando 03) for cartoon characters. Besides the obvious view-dependent channels, other ones could come into play such as channels representing the influence of external forces (gravity, wind, ...).

10.2 Constraining Facial Emotions

In chapter 6 we introduced a novel approach through which an emotionally meaningful 2D facial expression from one point of view can be created from a reference expression in another point of view.

We are interested in investigating the incorporation of geometrical constraints. For instance, the lack of these constraints can lead to situations like figure 6.4(g) (page 77) where in this case the mouth appears a little outside the girl's face. While at this moment the animator can adjust the results at any time, we are doing research on how to impose geometric constraints which for example avoid the mouth from appearing outside the face. Depending on the defined pay-off functions, this could then be corrected by for instance reorienting the mouth, scaling it up or down, etc.

10.3 (Realtime) Mapping of Facial Expressions to 2D Animation

Nowadays, there exist a lot of techniques in order to recognise facial expressions (Byoungwon 01; Li 01; Fidaleo 02). Most applications aim at knowing the emotional state of the user in order to facilitate the human-computer interaction. That way, facial expressions could provide a way to communicate basic information about needs and demands to computers.

However, we would like to examine how to integrate (existing) facial expression recognition software into our 2.5D animation framework. For example, real facial expressions (tracked by the recognition software) could be mapped directly to drawn 2.5D characters. As a result, the conventional user interface (GUI) can be replaced by, for instance, a(n) (live) actor.

10.4 Natural Phenomena

In chapter 8 we presented a novel approach to design artistic and believable trees in a cartoon-like style. In the future we want to explore the suitability of this technique to other kinds of vegetation. Furthermore, we want to extend the framework to also incorporate non-rigid animation, like movements influenced by the wind.

10.5 Stylisation of Animated Photographic Material

We plan on doing research on stylisation of animated photographic material. The idea is to start with photographs (e.g. of a person's head) taken simultaneously from various view points. Firstly, the animator specifies corresponding areas on the photographs. These areas will be treated as different layers making up the photographs. As a result, for each photograph a hierarchical display model (HDM) is constructed. Secondly, the animator stylises the specified areas using for instance standard drawing tools. Finally, our automatic in-betweening system comes into play exploiting all the HDMs and, hence, generating a stylised animation of the photographic material.

Chapter 11

Conclusions

"We shall not cease from exploration, and the end of all our exploring will be to arrive where we started and know that place for the first time."

T. S. Eliot

In this dissertation, we investigated how to eliminate the time-consuming aspects of traditional animation by shifting towards a computer assisted approach. By considering traditional animation — from an artist's point of view — as well as existing computer assisted traditional animation techniques, some key limitations and fallacies were identified. That is, the problem of the repeated drawing of all characters in all frames, and the lack of 3D information which only is present in the animator's mind. To this end, we particularly focussed on automatically calculating in-between frames, and reintroducing necessary 3D information. Furthermore, it was also our goal to give the animator the same freedom of exaggeration to create animations as s/he is bearing in mind.

First, we introduced a new method for automatic in-betweening in computer assisted traditional animation. The solution is based on novel 2.5D modelling and animation techniques within the context of a multi-level approach, starting with basic 2D drawing primitives (curves) at level 0, over explicit 2.5D modelling structures at level 1 and inclusion of 3D information by means of skeletons at level 2, to high-level tools (such as a deformation tool) at level 3.

In order to preserve the natural way of drawing and editing — instead of manipulating drawings by 'point-click-and-drag' procedures — we presented a multi-level sketching tool that assists the animator throughout multiple stages of the animation process, and that supports additional functionality such as deforming animation objects in an intuitive way.

Next, we focussed on employing approximate 3D input models as a means to incorporate necessary 3D information. These approximate 3D models help retaining volumes and proportions, and ensure frame–to–frame coherence.

We also elaborated on eliminating the time-consuming process of drawing all the emotions of a character, which have to be seen from different viewpoints. To establish these goals, we introduced the concept of *facial emotion channels*, of which each represents a facial part expressing an emotion. Furthermore, we presented a novel approach through which an emotionally meaningful 2D facial expression from one point of view can be created from a reference expression in another point of view.

In a following chapter we introduced new techniques and tools to draw, manipulate and animate stylised brush strokes in computer assisted animation production.

Next, a novel approach to design artistic and believable trees in a cartoon-like style, which can be rendered by an animated camera producing a convincing 3D-like experience, is explained.

We extended this approach to the more turbulent movement of a series of gaseous phenomena. Moreover, we supplied a powerful control allowing users to direct the animation to their artistic needs.

We believe that the provided solutions are easy to use, empower a much quicker cartoon production without hampering the animation artists' creativity, and, hence, should be especially beneficial in the production of traditional animation feature films and series.

Bibliography

[Alexa 00] M. Alexa, D. Cohen-Or & D. Levin. *As-Rigid-As-Possible Shape Interpolation.* In Proceedings of SIGGRAPH, pages 157–164. ACM, ACM Computer Graphics, 2000.

[ANDROME NV 04] ANDROME NV. World Wide Web, `http://www.androme.com/`, 2004.

[ANIMANTE 04] ANIMANTE. World Wide Web, `http://www.animante.com/`, 2004.

[Animo 04] Animo. *Cambridge Animation Systems.* World Wide Web, `http://www.animo.com/`, 2004.

[Bando 03] Yosuke Bando, Bing-Yu Chen & Tomoyuki Nishita. *Animating Hair with Loosely Connected Particles.* In Proceedings of Eurographics (EG2003), volume 22, pages 411–418. Computer Graphics Forum, 2003.

[Barzel 97] Ronen Barzel. *Faking Dynamics of Ropes and Springs.* IEEE Computer Graphics and Applications, vol. 17, pages 31–39, 1997.

[Bimber 99] Oliver Bimber. *Rudiments for a 3D Freehand Sketch Based Human-Computer Interface for Immersive Virtual Environments.* In Proceedings of Virtual Reality Software and Technology (VRST1999), pages 182–183. ACM, December 20–22 1999.

[Blair 94] Preston Blair. Cartoon animation. Walter Foster Publishing Inc., ISBN: 1–56010–084–2, 1994.

[Bregler 02] Christoph Bregler, Lorie Loeb, Erika Chuang & Hishi Deshpande. *Turning to the Masters: Motion Capturing Cartoons.* In Proceedings of SIGGRAPH, volume 21(3), pages 399–407. ACM, July 2002.

[Burtnyk 71] N. Burtnyk & M. Wein. *Computer-Generated Key-Frame Animation.* Journal of the Society Motion Picture and Television Engineers, vol. 8, no. 3, pages 149–153, 1971.

[Burtnyk 76] N. Burtnyk & M. Wein. *Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation.* Communications of the ACM, vol. 19, no. 10, pages 564–569, 1976.

[Byoungwon 01] Choe Byoungwon & Hyeongseok Ko. *Analysis and Synthesis of Facial Expressions with Hand-Generated Muscle Actuation Basis.* In Proceedings of Computer Animation (CA2001), pages 12–19, November 2001.

[Carmel 97] E. Carmel & D. Cohen-Or. *Warp-guided Object Space Morphing.* The Visual Computer, vol. 13, pages 465–478, 1997.

[Catmull 78a] E. Catmull. *The Problems of Computer-Assisted Animation.* In Proceedings of SIGGRAPH, volume 12(3), pages 348–353. ACM, August 1978.

[Catmull 78b] E. Catmull & J. Clark. *Recursively Generated B-spline Surfaces on Arbitrary Topological Meshes.* Computer-Aided Design, vol. 10, pages 350–355, September 1978.

[Claes 97] Johan Claes, Patrick Monsieurs, Frank Van Reeth & Eddy Flerackers. *Rendering Pen-Drawings of 3D Scenes on Network Computers.* In Journal of Winter School of Computer Graphics (WSCG1997), volume 1, pages 60–69, 1997.

[Claes 01] Johan Claes, Fabian Di Fiore, Gert Vansichem & Frank Van Reeth. *Fast 3D Cartoon Rendering with Improved Quality by Exploiting Graphics Hardware.* In Proceedings of Image and Vision Computing New Zealand (IVCNZ2001), pages 13–18. IVCNZ, November 2001.

[Cohen 00] Jonathan M. Cohen, John F. Hughes & Robert C. Zeleznik. *Harold: A World Made of Drawings.* NPAR2000: Symposium on Non-Photorealistic Animation and Rendering, pages 83–90, June 2000.

[Coninx 96] Karin Coninx, Frank Van Reeth & Eddy Flerackers. *Interactive Specification of $2\frac{1}{2}D$ Animation by Exploiting a Real-time 3D Rendering Architecture.* In Proceedings of Eurographics UK, London, pages 11–18, March 1996.

[Coquillart 91] Sabine Coquillart & Pierre Jancène. *Animated Free-Form Deformation: An Interactive Animation Technique.* In Proceedings of SIGGRAPH, pages 23–26. ACM, July 1991.

[CreaToon 04] CreaToon. *ANDROME NV.* World Wide Web, `http://www.creatoon.com/`, 2004.

[CUSTODIEV 04] CUSTODIEV. World Wide Web, `http://www.custodiev.org/`, 2004.

[Decaudin 96] Ph. Decaudin. *Modélisation par Fusion de Formes 3D pour la Synthèse d'Images - Rendu de Scènes 3D Imitant le Style "Dessin Animé".* PhD thesis, Université de Technologie de Compiegne (France), 1996.

[Deussen 00] Oliver Deussen & Thomas Strothotte. *Computer-Generated Pen-and-Ink Illustration of Trees.* In Proceedings of SIGGRAPH, pages 13–18. ACM, 2000.

[Di Fiore 01] Fabian Di Fiore, Philip Schaeken, Koen Elens & Frank Van Reeth. *Automatic In-betweening in Computer Assisted Animation by Exploiting 2.5D Modelling Techniques.* In Proceedings of Computer Animation (CA2001), pages 192–200, November 2001.

[Di Fiore 02a] Fabian Di Fiore & Frank Van Reeth. *Employing Approximate 3D Models to Enrich Traditional Computer Assisted Animation.* In Proceedings of Computer Animation (CA2002), pages 183–190, June 2002.

[Di Fiore 02b] Fabian Di Fiore & Frank Van Reeth. *A Multi-Level Sketching Tool for Pencil-and-Paper Animation.* In

Sketch Understanding: Papers from the 2002 American Association for Artificial Intelligence (AAAI2002) Spring Symposium. Technical Report SS–02–08, pages 32–36, March 2002.

[Di Fiore 03a] Fabian Di Fiore, William Van Haevre & Frank Van Reeth. *Rendering Artistic and Believable Trees for Cartoon Animation.* In Proceedings of Computer Graphics International (CGI2003), pages 144–151, July 2003.

[Di Fiore 03b] Fabian Di Fiore & Frank Van Reeth. *Mimicing 3D Transformations of Emotional Stylized Animation with Minimal 2D Input.* In Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia (GRAPHITE 2003), pages 21–28, February 2003.

[Di Fiore 03c] Fabian Di Fiore & Frank Van Reeth. *Modelling in 2D Enabling Fluid Stylised Animation.* In Proceedings of GraphiCon, International Conference on Computer Graphics & Vision, pages 124–130, September 2003.

[Di Fiore 04] Fabian Di Fiore, Johan Claes & Frank Van Reeth. A framework for user control on stylised animation of gaseous phenomena. Accepted for Computer Animation and Social Agents (CASA2004), July 2004.

[Dobashi 00] Yoshinori Dobashi, Kazufumi Kaneda, Hideo Yamashita, Tsuyoshi Okita & Tomoyuki Nishita. *A Simple, Efficient Method for Realistic Animation of Clouds.* In Proceedings of SIGGRAPH 2000, pages 19–28, 2000.

[Fedkiw 01] Ron Fedkiw, Jos Stam & Henrik Wan Jensen. *Visual Simulation of Smoke.* In Proceedings of SIGGRAPH, pages 23–30. ACM, 2001.

[Fidaleo 02] Douglas Fidaleo & Ulrich Neumann. *CoArt: Coarticulation Region Analysis for Control of 2D Characters.* In Proceedings of Computer Animation (CA2002), pages 17–22, June 2002.

[Flerackers 02] Chris Flerackers. New algorithms and techniques in 2.5d animation. Master's thesis, Transnationale Universiteit Limburg, June 2002.

[Foster 97] N. Foster & D. Metaxas. *Modeling the Motion of a Hot, Turbulent Gas.* In Proceedings of SIGGRAPH, pages 181–188, 1997.

[Gleicher 98] Michael Gleicher. *Retargetting motion to new characters.* In Proceedings of SIGGRAPH, pages 33–42. ACM, 1998.

[Goldstein 95] E. Goldstein & C. Gotsman. *Polygon Morphing Using a Multiresolution Representation.* In Proceedings of Graphics Interface (GI1995), pages 247–254, 1995.

[Gooch 01] Bruce Gooch & Amy Ashurst Gooch. Non-photorealistic rendering. A. K. Peters Ltd., ISBN: 1568811330, 2001.

[Griffin 01] H. Griffin. The animator's guide to 2d computer animation. Focal Press, ISBN: 0–240–51579–X, 2001.

[Guaglione 98] Eric Guaglione. *The Art of Disney's Mulan.* SIGGRAPH Course notes 39, July 1998.

[Haeberli 90] Paul Haeberli. *Paint By Numbers: Abstract Image Representations.* In Proceedings of SIGGRAPH, volume 24(4), pages 207–214. ACM, August 1990.

[Hertzmann 98] Aaron Hertzmann. *Painterly Rendering with Curved Brush Strokes of Multiple Sizes.* In Proceedings of SIGGRAPH, pages 453–460. ACM, 1998.

[Hertzmann 00] Aaron Hertzmann & Ken Perlin. *Painterly Rendering for Video and Interaction.* NPAR2000: Symposium on Non-Photorealistic Animation and Rendering, pages 7–12, June 2000.

[Hertzmann 01] Aaron Hertzmann. *Paint by relaxation.* In Proceedings of Computer Graphics International (CGI2001), pages 47–54, 2001.

[Hooper 04] Jon Hooper & Michel Gagne. *Ten Steps to a perfect In-between.* World Wide Web, `http://www.animationmeat.com/tenstepsmain.html`, 2004.

[Hsu 93] S. C. Hsu, I. H. H. Lee & N. E. Wiseman. *Skeletal Strokes*. In Proceedings of UIST1993 of the ACM SIGGRAPH & SIGCHI (Symposium on User Interface Software & Technology), pages 109–118. ACM, November 1993.

[Hsu 94] S. C. Hsu & I. H. H. Lee. *Drawing and Animation Using Skeletal Strokes*. In Proceedings of SIGGRAPH, pages 109–122. ACM, July 1994.

[Igarashi 99] Takeo Igarashi, Satoshi Matsuoka & Hidehiko Tanaka. *Teddy: A Sketching Interface for 3D Freeform Design*. In Proceedings of SIGGRAPH, pages 409–416. ACM, August 1999.

[Ilmonen 03] Tommi Ilmonen & Janne Kontkanen. *The Second Order Particle System*. In Journal of Winter School on Computer Graphics (WSCG2003), pages 240–246, 2003.

[Isenberg 02] Tobias Isenberg, Nick Halper & Thomas Strothotte. *Stylizing Silhouettes at Interactive Rates: From Silhouette Edges to Silhouette Strokes*. Proceedings of Eurographics (EG2002), vol. 21, no. 3, pages 249–258, September 2002.

[Johan 00] H. Johan, Y. Koiso & T. Nishita. *Morphing using Curves and Shape Interpolation Techniques*. In Proceedings of Pacific Graphics 2000, 8th Pacific Conference on Computer Graphics and Applications, pages 348–358, 2000.

[Kahn 77] K. Kahn & G. A. Gorry. *Mechanizing Temporal Knowledge*. Artificial Intelligence, vol. 9, pages 87–108, 1977.

[Kalnins 02] Robert D. Kalnins, Lee Markosian, Barbara J. Meier, Michael A. Kowalski, Joseph C. Lee, Philip L. Davidson, Matthew Webb, John F. Hughes & Adam Finkelstein. *WYSIWYG NPR: drawing strokes directly on 3D models*. In Proceedings of SIGGRAPH, pages 755–762. ACM, 2002.

[Kalnins 03] Robert D. Kalnins, Philip L. Davidson, Lee Markosian & Adam Finkelstein. *Coherent Stylized Silhouettes*. ACM

Transactions on Graphics, vol. 22, no. 3, pages 856–861, 2003.

[Kalra 92] P. Kalra, A. Mangili, Nadia Magnenat-Thalmann & Daniel Thalmann. *Simulation of Facial Muscle Actions Bases on Rational Free Form Deformations*. In Proceedings of Eurographics (EG1992), volume 11(3), pages 59–69, 1992.

[Kaplan 00] Matthew Kaplan, Bruce Gooch & Elaine Cohen. *Interactive Artistic Rendering*. NPAR2000: Symposium on Non-Photorealistic Animation and Rendering, pages 67–74, June 2000.

[King 99] Scott A. King, Roger A. Crawfis & Wayland Reid. *Fast Animation of Amorphous and Gaseous Phenomena*. In Proceedings of Volume Graphics (VG99), pages 333–346, March 1999.

[Kochanek 84] D. Kochanek & R. Bartels. *Interpolating Splines with Local Tension, Continuity and Bias Control*. In Proceedings of SIGGRAPH, volume 18(3), pages 33–41. ACM, July 1984.

[Kort 02] Alexander Kort. *Computer Aided Inbetweening*. NPAR2002: Symposium on Non-Photorealistic Animation and Rendering, pages 125–132, June 2002.

[Kowalski 99] Michael A. Kowalski, Lee Markosian, J. D. Northrup, Lubomir Bourdev, Ronen Barzel, Loring S. Holden & John F. Hughes. *Art-Based Rendering of Fur, Grass, and Trees*. In Proceedings of SIGGRAPH, pages 433–438. ACM, August 1999.

[Lake 00] Adam Lake, C. Marshall, M. Harris & M. Blackstein. *Stylized Rendering Techniques for Scalable Real-Time 3D Animation*. NPAR2000: Symposium on Non-Photorealistic Animation and Rendering, pages 13–20, June 2000.

[Lamorlette 02] Arnauld Lamorlette & Nick Foster. *Structural Modeling of Flames for a Production Environment*. In Proceedings of SIGGRAPH, pages 729–735, 2002.

[Lasseter 87] John Lasseter. *Principles of traditional animation applied to 3D computer animation.* In Proceedings of SIGGRAPH, volume 21, pages 35–44, 1987.

[Li 01] I-Chen Li, Jeng-Sheng Yeh & Ming Ouhyoung. *Realistic 3D Facial Animation Parameters fom Mirror-reflected Multi-view Video.* In Proceedings of Computer Animation (CA2001), pages 2–11, November 2001.

[Li 03] Yin Li, Michael Gleicher, Ying-Qing Xu & Heung-Yeung Shum. *Stylizing motion with drawings.* In Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pages 309–319. Eurographics Association, 2003.

[Lintermann 99] Bernd Lintermann & Oliver Deussen. *Interactive Modeling of Plants.* IEEE Computer Graphics and Applications, pages 2–11, 1999.

[Litwinowicz 91] Peter C. Litwinowicz. *INKWELL: A 2.5D Animation System.* Computer Graphics, vol. 25, no. 4, pages 113–122, 1991.

[Markosian 97] Lee Markosian, Michael A. Kowalski, S. Trychin, L. Bourdev, D. Goldstein & John F. Hughes. *Real-Time Non-photorealistic Rendering.* In Proceedings of SIGGRAPH, pages 415–420. ACM, August 1997.

[Markosian 00] Lee Markosian, Barbara J. Meier, Michael A. Kowalski, Loring S. Holden, J. D. Northrup & John F. Hughes. *Art-based Rendering with Continuous Levels of Detail.* NPAR2000: Symposium on Non-Photorealistic Animation and Rendering, pages 59–66, June 2000.

[Masuch 98] M. Masuch, L. Schuhmann & S. Schlechtweg. *Animating Frame-To-Frame-Coherent Line Drawings for Illustrated Purposes.* In Proceedings of Simulation und Visualisierung 1998, pages 101–112, 1998.

[Měch 96] Radomír Měch & Prusinkiewicz. Przemyslaw. *Visual Models of Plants Interacting with Their Environment.* In Proceedings of SIGGRAPH, pages 397–410, August 1996.

[Meier 96] Barbara J. Meier. *Painterly Rendering for Animation.* In Proceedings of SIGGRAPH, volume 25(4), pages 477–484. ACM, 1996.

[Moho 04] Moho. *Lost Marble.* World Wide Web, `http://www.lostmarble.com/`, 2004.

[Nettleship 95] T. Nettleship & P. Willis. *The Animachine Renderer.* Computer Graphics, pages 90–97, 1995.

[Noh 98] Junyong Noh & Ulrich Neumann. *A Survey of Facial Modeling and Animation Techniques.* Rapport technique, USC 99-705, 1998.

[Oddy 92] R. J. Oddy & P. J. Willis. *Rendering NURB Regions for 2D Animation.* Computer Graphics Forum, special issue EG, vol. 11, no. 3, pages C35–C44, 1992.

[Parent 01] Richard Parent. Computer animation: Algorithms and techniques. Morgan Kaufmann Publishers, ISBN: 1-55860-579-7, 2001.

[Parke 72] Frederic I. Parke. *Computer Generated Animation of Faces.* In Proceedings of ACM National Conference, pages 451–457, 1972.

[Patterson 94] J. W. Patterson & P. J. Willis. *Computer Assisted Animation: 2D or not 2D?* The Computer Journal, vol. 37, no. 10, pages 829–839, 1994.

[Perlin 01] Ken Perlin & Fabrice Neyret. *Flow Noise.* SIGGRAPH. Technical Sketches and Applications., page 187, August 2001.

[Platt 81] S. Platt & Norman Badler. *Animating facial expressions.* In Proceedings of SIGGRAPH, volume 15(3), pages 245–252. ACM, 1981.

[Prashant 03] Chopra Prashant & Joerg Meyer. *Modeling an Infinite Emotion Space for Expressionistic Cartoon Face Animation.* In Proceedings of the 6th IASTED International Conference on Computers, Graphics, and Imaging (CGIM2003), pages 13–18, August 2003.

[Prusinkiewicz 90] Przemyslaw Prusinkiewicz & Aristid Lindenmayer. The algorithmic beauty of plants. Springer-Verlag New York Inc., ISBN: 0–387–97297–8, 1990.

[Prusinkiewicz 94] Przemyslaw Prusinkiewicz & M. Hammel. *Language Restricted Iterated Functions, Koch Constructions and L-systems.* SIGGRAPH Course Notes, 1994.

[Prusinkiewicz 96] Przemyslaw Prusinkiewicz & Lila Kari. *Subapical Bracketed L-systems.* In Lecture Notes in Computer Science, Volume 1073, pages 550–564. Springer Verlag, 1996.

[Prusinkiewicz 97] Przemyslaw Prusinkiewicz, M. Hammel, J. Hanan & Radomír Měch. Plants to ecosystems. advances in computational life sciences, volume 1, chapitre L-systems: From the theory to visual models of plants, pages 1–27. CSIRO Publishing, ISBN: 0643059423, 1997.

[Rademacher 99] Paul Rademacher. *View-Dependent Geometry.* In Alyn Rockwood, editeur, Proceedings of SIGGRAPH, pages 439–446, Los Angeles, 1999. ACM, Addison Wesley Longman.

[Raghavachary 02] S. Raghavachary & F. Benitez. *Painterly Fire.* SIGGRAPH. Technical Sketches and Applications., page 225, 2002.

[Ranjian 96] V. Ranjian & A. Fournier. *Matching and interpolation of shapes using unions of circles.* Computer Graphics Forum, vol. 15, pages 129–142, 1996.

[Reeves 81] W. Reeves. *Inbetweening for Computer Animation Utilizing Moving Point Constraints.* Computer Graphics, vol. 15, no. 3, pages 263–269, 1981.

[Reeves 83] W. T. Reeves. *Particle Systems - A Technique for Modeling a Class of Fuzzy Objects.* In Proceedings of SIGGRAPH, pages 359–376, 1983.

[Reynolds 04] C. Reynolds. *Stylized depiction in computer graphics.* World Wide Web, `http://www.red3d.com/cwr/npr/`, 2004.

[Rose III 01] Charles F. Rose III, Peter-Pike J. Sloan & Michael F. Cohen. *Artist-Directed Inverse-Kinematics Using Radial Basis Function Interpolation.* In Proceedings of Eurographics (EG2001), volume 20(3), pages 239–250, 2001.

[Rubio 04] Kevin Rubio. *Star Wars Troops.* World Wide Web, `http://www.theforce.net/troops/`, 2004.

[Ruttkay 00] Zsófia Ruttkay & Han Noot. *Animated CharToon Faces.* NPAR2000: Symposium on Non-Photorealistic Animation and Rendering, pages 91–100, June 2000.

[Ruttkay 01] Zsófia Ruttkay & Han Noot. *FESINC: Facial Expression Sculpturing with INterval Constraints.* In Proceedings of Autonomous Agents 2001 (AA01). Workshop on Representing, Annotating and Evaluating Non-Verbal and Verbal Communicative Acts to Achieve Contextual Embodied Agents, May 2001.

[Ruttkay 03] Zsofia Ruttkay, Han Noot & Paul ten Hagen. *Emotion Disc and Emotion Squares: Tools to Explore the Facial Expression Space.* Computer Graphics Forum, vol. 22, no. 1, pages 49–49, 2003.

[Salisbury 96] M. Salisbury, C. Anderson, D. Lischinski & D. Salesin. *Scale-Dependent Reproduction of Pen-and-Ink-Illustration.* In Proceedings of SIGGRAPH, volume 30, pages 461–468. ACM, 1996.

[Schlechtweg 96] Stefan Schlechtweg & Thomas Strothotte. *Rendering Line-Drawings with Limited Resources.* In Proceedings of Graphicon, International Conference on Computer Graphics & Vision, volume 2, pages 131–137, 1996.

[Sederberg 86] Thomas W. Sederberg & Scott R. Parry. *Free-Form Deformation of Solid Geometric Models.* In Proceedings of SIGGRAPH, pages 151–160. ACM, August 1986.

[Sederberg 92] Thomas W. Sederberg & E. Greenwood. *A Physically Based Approach to 2D Shape Blending.* Computer Graphics, vol. 26, pages 25–34, 1992.

[Sederberg 93] Thomas W. Sederberg, P. Gao, G. Wang & H. Mu. *2D Shape Blending: an intrinsic solution to the vertex path problem.* Computer Graphics, vol. 27, pages 15–18, 1993.

[Selle 04] Andrew Selle & Alex Mohr. *Cartoon Rendering of Smoke Animations.* NPAR2004: Symposium on Non-Photorealistic Animation and Rendering, June 2004.

[Shapira 95] M. Shapira & A. Rappoport. *Shape Blending using a Star-Skeleton Representation.* IEEE Computer Graphics and Applications, vol. 15, pages 44–51, 1995.

[Stam 99] Jos Stam. *Stable fluids.* In Proceedings of SIGGRAPH, pages 121–128, 1999.

[Strothotte 02] Thomas Strothotte & Stefan Schlechtweg. Non-photorealistic computer graphics. modeling, rendering, and animation. Morgan Kaufmann Publishers, ISBN: 1-55860-787-0, 2002.

[Thórisson 96] Kristinn R. Thórisson. *ToonFace: A System for Creating and Animating Interactive Cartoon Faces.* Rapport technique, MIT Media Laboratory, Learning and Common Sense 96–01, April 1996.

[Treuille 03] Adrien Treuille, Antoine McNamara, Zoran Popovic & Jos Stam. *Keyframe Control of Smoke Simulations.* In Proceedings of SIGGRAPH, pages 716–723, 2003.

[Van den Bergh 02] Jan Van den Bergh, Fabian Di Fiore, Johan Claes & Frank Van Reeth. *Interactively Morphing Irregularly Shaped Images Employing Subdivision Techniques.* In Proceedings of 1st Ibero-American Symposium in Computer Graphics (SIACG2001), pages 315–321, July 2002.

[Van Haevre 03] William Van Haevre & Philippe Bekaert. *A Simple but Effective Algorithm to Model the Competition of Virtual Plants for Light and Space.* In Journal of Winter School on Computer Graphics (WSCG2003), pages 464–471, February 2003.

[Van Haevre 04] William Van Haevre, Fabian Di Fiore, Philippe Bekaert & Frank Van Reeth. *A Ray Density Estimation Approach to Take into Account Environment Illumination*

in Plant Growth Simulation. In Proceedings of Spring Conference on Computer Graphics (SCCG2004), pages 121–131, April 2004.

[Van Reeth 96] Frank Van Reeth. *Integrating $2\frac{1}{2}D$ Computer Animation Techniques for Supporting Traditional Animation.* In Proceedings of Computer Animation (CA1996), pages 118–125, 1996.

[Vansichem 01] Gert Vansichem, Emmanuel Wauters & Frank Van Reeth. *Real-Time Modeled Drawing And Manipulation Of Stylized Cartoon Characters.* In Proceedings of the IASTED International Conference on Computer Graphics and Imaging, pages 44–49, Honolulu, HI, USA, August 13–16 2001. IASTED.

[Welman 93] C. Welman. Inverse kinematics and geometric constraints for articulated figure manipulation. Master's thesis, Simon Fraser University, 1993.

[Whitaker 81] Harold Whitaker & John Halas. Timing for animation. Focal Press, ISBN: 0–240–51714–8, 1981.

[White 04] Tony White. World Wide Web, `http://www.tonywhite.net/`, 2004.

[Williams 01] Richard Williams. The animator's survival kit. Faber and Faber Limited, ISBN: 0–571–20228–4, 3 Queen Square London WC1N 3AU, 2001.

[Winkenbach 94] G. Winkenbach & D. Salesin. *Computer Generated Pen-and-Ink-Illustration.* In Proceedings of SIGGRAPH, pages 91–100. ACM, 1994.

[Witting 99] Patrick Witting. *Computational fluid dynamics in a traditional animation environment.* In Proceedings of SIGGRAPH, pages 129–136, 1999.

[Wolberg 98] G. Wolberg. *Image Morphing Survey.* The Visual Computer, vol. 14, no. 8/9, pages 360–372, 1998.

[Yu 90] Jinhui Yu. *Inbetweening for Computer Animation Using Polar Coordinates Linear Interpolation.* Research

Report 90/R23, Department of Computer Science, University of Glasgow, September 1990.

[Yu 94] Jinhui Yu. *A Hierarchical Flowing Water Model.* In Proceedings of the 7th National Conference on Imagery and Graphics, 1994.

[Yu 96a] Jinhui Yu & John W. Patterson. *A Fire Model for 2D Computer Animation.* In Proceedings of Computer Animation and Simulation, pages 49–60, 1996.

[Yu 96b] Jinhui Yu & John W. Patterson. *Object Deformation using Quaternions.* In Proceedings of Eurographics UK Chapter 14th Annual Conference, pages 75–88, 1996.

[Yu 99] Jinhui Yu. *Stylised Procedural Animation.* PhD thesis, University of Glasgow, 1999.

[Zorin 00] D. Zorin, P. Schröder, A. Levin, L. Kobbelt, W. Sweldens & T. DeRose. *Subdivison for Modelling and Animation.* SIGGRAPH Course notes 23, July 2000.

List of Figures

List of Listings

Appendix A

Overview of the Traditional Animation Process

"A great artist can paint a great picture on a small canvas."

Charles Dudley Warner

In this appendix we give a brief overview of the traditional animation process in general. We refer the interested reader to (Blair 94; Patterson 94; Williams 01) for a detailed overview.

A.1 Designing the Story(board)

Traditional animation begins with refining a *story* into a *storyboard*. A storyboard is a visual layout of events allowing the animators to plan the flow of the plot and the composition of the imagery (figure A.1). Next to the storyboard, model sheets are also prepared. These include the extreme poses of the characters to be drawn, from which intermediate attributes can be interpolated such as position, expression, colour, size, etc.

A.2 Preliminary Soundtrack

Before true animation begins, a *soundtrack* is recorded, so that the animation may be more precisely synchronised to the soundtrack. Given the slow, methodical manner in which traditional animation is produced, it is almost always easier to synchronise animation to a pre-existing soundtrack than it is to synchronise a soundtrack to pre-existing animation.

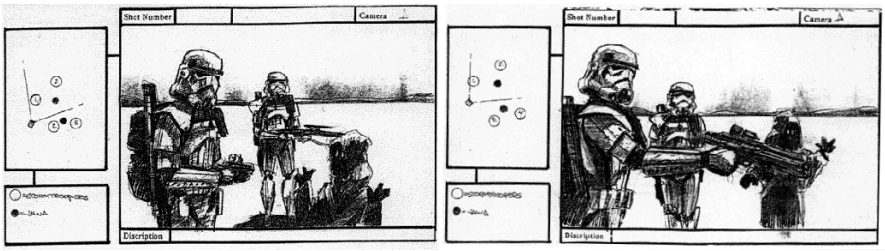


Figure A.1: Storyboard example. Copyright © 2004 Kevin Rubio (Rubio 04).

A.3 Animatic/Leica Test/Filmed Storyboard

An *animatic* (also called Leica reel or filmed storyboard) typically consists of pictures of the storyboard synchronised with the soundtrack. As there is no animation available yet, it is filmed from stills corresponding to the storyboard, but each still is held for as long as the corresponding sequence will take. The result gives an impression of the movie and allows the animators and directors to work out any script and timing issues that may exist with the current storyboard.

A.4 Scene Staging

Scene staging is the mapping out of each scene and refers to how to break down the scene into *layers*, including the separation of character components into layers (figure A.2). In this step, the animator actually *plans* the drawings to be used.

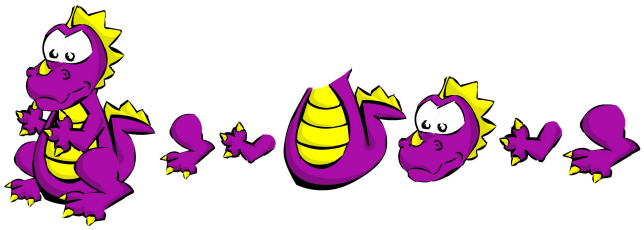


Figure A.2: Character and its separate layers.

A.5 Exposure Sheets

Unlike live action, where the camera is running continuously, each frame of an animation film is shot one by one. Moreover, each single frame might consist of numerous layers stacked up together. In order to keep track of which layers, and when, are needed for a particular frame, exposure sheets are used.

An *exposure sheet* (X-sheet) is basically a table which has a row for each frame (figure A.3). The columns show which layers are used, the camera positions, instructions for exposure etc. Once the soundtrack is recorded, it can be entered on the X-sheet as well.

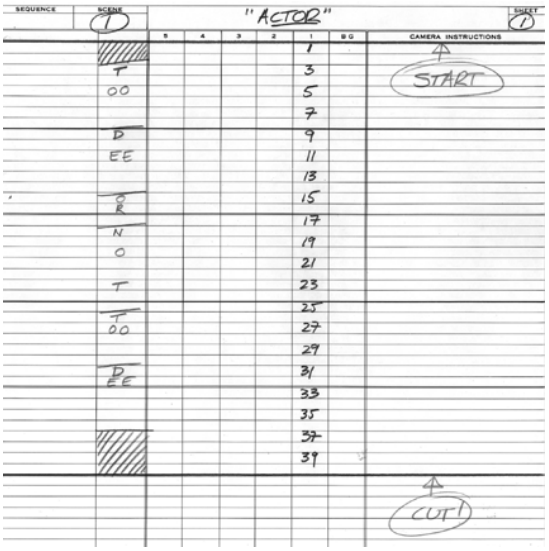


Figure A.3: Example of an exposure sheet. Copyright © 2004 Tony White (White 04).

A.6 Drawing and Pencil/Line Test

For each layer (excluding the backgrounds) of each frame, a line drawing (not coloured) is prepared in order to perform a pencil/line test. These drawings get mechanically aligned with the purpose of verifying that the movements are correct and that characters interact accurately.

A.7 Drawing the Cels

In this step, each layer (including the backgrounds) of each frame is transferred (and coloured) from paper to a thin, clear sheet of plastic called a 'cel'. First, the outlines are drawn by tracing with a pen or brush and ink. Next, the drawings are coloured with opaque acrylic paint.

The transparent quality of the cel allows for stacking up all cels on each other, as the cel of one character can be seen underneath the cel of another, and the opaque background will be seen beneath all of the cels.

A.8 Rostrum Camera

For each frame, cels are stacked up, illuminated and shot using a rostrum camera (figure A.4). The camera normally points vertically downwards and the artwork is placed horizontally.

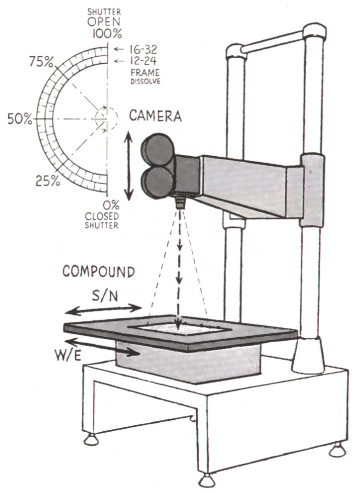


Figure A.4: Example of a rostrum camera. Copyright © 1994 Preston Blair (Blair 94).

A.9 Soundtrack Synchronisation

Already in step 2, a soundtrack is recorded, so that the animation may be more precisely synchronised to the soundtrack. However, in this last step, the final soundtrack is synchronised with, and added to, the movie.

Bijlage B

Samenvatting

"Het begin van alle wetenschap is de verwondering dat de dingen zijn zoals ze zijn."

Aristoteles

B.1 Inleiding

Alhoewel de productie van traditionele 2D animatie sterk geëvolueerd is sinds Disney in de jaren twintig van vorige eeuw zijn eerste stappen waagde, blijft het nog steeds een erg tijdrovend en arbeidsintensief proces (Whitaker 81; Blair 94; Patterson 94; Nettleship 95; Griffin 01; Williams 01). Het meeste werk en tijd wordt besteed aan het schetsen, tekenen en inkleuren van alle individuele figuren, en dit voor ieder afzonderlijk beeld van de animatie.

Het doel van deze thesis bestaat erin deze tijdrovende aspecten van traditionele animatie weg te werken door middel van adequate automatisering. Daarbij is het zeer belangrijk dat de animator een zo groot mogelijke vrijheid van expressie behoudt.

De volgende sectie geeft een overzicht van traditionele en geautomatiseerde animatie, en eindigt met een subsectie over hoe traditionele animatie volgens ons geautomatiseerd zou moeten worden. Alle daaropvolgende secties beschrijven onze specifieke bijdragen tot het automatiseringsproces.

B.2 Overzicht van 2D animatie

In hoofdstuk 2 werpen we een licht op hoe traditionele animatie geautomatiseerd kan worden. Daarvoor verdiepen we ons eerst in zowel traditionele animatie als geautomatiseerde traditionele animatie.

B.2.1 Traditionele animatie vanuit een artistiek standpunt

Traditionele animatie wordt gedefinieerd als een techniek waarin men door snelle opeenvolging van geleidelijk veranderende tekeningen beweging suggereert. De volgende sectie geeft een overzicht van het traditionele animatieproces, waarna we verder uitweiden over enkele belangrijke aspecten: het belang van tijd en dramatiek, en het tekenen van tussenliggende beelden.

Overzicht van het animatieproces

Volgende stappen geven een algemeen overzicht van het traditionele animatieproces (Blair 94; Patterson 94; Williams 01).

1. **Ontwikkeling van het *storyboard***
   Een *storyboard* is een visueel ontwerp waarop ruwweg de opeenvolgende situaties van het verhaal getekend zijn (figuur A.1, pagina 160). Op deze manier plant men de verhaallijn en compositie van de beelden. Naast het *storyboard* worden er onder andere ook *model sheets* aangemaakt waarop de uiterste poses van de karakters getekend worden.

2. **Voorlopige soundtrack**
   Meestal wordt er op voorhand een voorlopige soundtrack gemaakt zodat de animatie reeds tijdens de ontwikkeling met de geluidsband gesynchroniseerd kan worden.

3. ***Animatic*/Leica test**
   Een *animatic* (of gefilmd *storyboard*) is een film bestaande uit de stilstaande tekeningen van het *storyboard* en de soundtrack.

4. **Organisatie van de scène**
   In deze stap worden de scènes in lagen opgedeeld. Hetzelfde gebeurt voor de individuele figuren (figuur A.2, pagina 160). In feite plant men hier welke tekeningen gebruikt gaan worden.

5. ***Exposure Sheets***
   Animatiefilms worden niet continu gefilmd maar worden beeld per beeld gefotografeerd. Daarenboven is ieder beeldje typisch het resultaat van een aantal opeengestapelde lagen die tevens hergebruikt kunnen worden (zie stap 4). De registratie van welke lagen voor welke beelden gebruikt moeten worden, gebeurt aan de hand van een tabel. De rijen stellen de opeenvolgende beelden voor, de kolommen geven aan welke lagen gebruikt worden. Deze tabellen noemt men *exposure sheets* (figuur A.3, pagina 161).

6. **Tekenen en de lijntest**
   Voor iedere laag (uitgezonderd de achtergronden) van ieder beeld wordt een niet-gekleurde lijntekening gemaakt. Deze tekeningen worden vervolgens mechanisch op één lijn gebracht om het gedrag van de bewegingen na te gaan.

   Het tekenproces zelf gebeurt in drie fasen: (i) hoofdanimators zijn verantwoordelijk voor de meest belangrijke beelden, ook wel *extreme frames* genoemd, omdat ze de meest belangrijke kenmerken van mogelijke acties bevatten; (ii) assistenten maken vervolgens *key frames* en definiëren zo het tijdsverloop van de animatie; (iii) de minst ervaren animators maken tenslotte de tussenliggende beelden (*in-between frames*).

7. **Tekenen van de cellen**
   In deze stap worden alle lagen (inclusief de achtergronden) ingekleurd hertekend op een plastic vel (*cel*). Deze cellen zijn transparant en kunnen zo makkelijk op elkaar gestapeld worden om een figuur/scène te vormen.

8. **Rostrum camera**
   Voor ieder beeld van de animatie worden de cellen op elkaar gestapeld, belicht en vervolgens gefotografeerd met behulp van een *rostrum camera* (figuur A.4, pagina 162).

9. **Soundtrack synchronisatie**
   In deze laatste stap worden de soundtrack en de animatie gesynchroniseerd.

Het belang van tijd en dramatiek

Traditionele animatie (Whitaker 81; Blair 94; Williams 01) is dikwijls niet-conform de realiteit maar maakt graag gebruik van dramatische effecten. We denken bijvoorbeeld aan het pletten en uitrekken van karakters, anticipatie, overdrijving, versnelling, vertraging en andere niet-realistische manieren om een verhaal, gemoedstoestand of aard over te brengen naar het publiek (Whitaker 81; Lasseter 87; Barzel 97).

De verantwoordelijkheid ligt volledig bij de animator.

Tussenliggende beelden berekenen

Het uittekenen van de tussenliggende beelden is het meest tijdrovende aspect van het animatieproces. Assistenten maken referentietekeningen of *key frames* (figuur 2.1, pagina 7) van een object voor bijvoorbeeld beelden F_i en

$F_j, j > i$, waarna de gewone animators tussenliggende beelden tekenen om zo de beweging van het object te definiëren voor de beelden F_{i+1} tot F_{j-1}.

Dit is een artisanaal proces en omvat onder andere het gebruik van tijdtabellen en *backlights*, en het manueel omdraaien, verschuiven en doorbladeren van de tekeningen.

We verwijzen naar Jon Hooper en Michel Gagne voor een stap voor stap beschrijving van dit proces (Hooper 04).

B.2.2 Computer geassisteerde animatie

De term "computer geassisteerde animatie" geeft aan dat met behulp van automatisering traditionele animatie uitgebreid en verbeterd wordt. De laatste decennia is een deel van het traditionele animatieproces reeds geautomatiseerd: achtergronden die digitaal uitgetekend worden, het kopiëren van lijntekeningen naar cellen met behulp van een plotter, ...

Daarentegen is het meest tijdrovende en arbeidsintensieve aspect van traditionele animatie — tekenen van tussenliggende beelden — nog steeds niet voldoende geautomatiseerd.

De volgende subsecties beschrijven in het kort de huidige stand van zaken in CAA. Daarna bespreken we (in de volgende sectie) hoe traditionele animatie het best geautomatiseerd wordt.

Tijdscontrole

Het is belangrijk dat de animator de volledige controle heeft over de pose en timing van de karakters. Tot op heden gebruikt men hiervoor *key frame* animatiesystemen (Barzel 97).

Automatisering van tussenliggende beelden

Vertrekkende van 2D vormen kunnen we twee categorieën onderscheiden (Yu 99).

De eerste is *shape-based in-betweening* en concentreert zich voornamelijk op de overeenkomsten tussen vormen waarbij die vormen zelf meestal door polygonen voorgesteld worden. Lineaire interpolatie (Burtnyk 71) is daarvan de gemakkelijkste maar ook meest naïeve methode. De problemen inherent aan lineaire interpolatie worden beschreven in (Catmull 78a) (zie ook figuur 2.2, pagina 10). Doorheen de volgende jaren werden verschillende technieken ontwikkeld om deze problemen te vermijden (Reeves 81; Yu 90; Sederberg 93). Alternatieve methoden van *shape blending* worden beschreven in (Sederberg 92), (Goldstein 95), (Ranjian 96) en (Wolberg 98), en recent in (Kort 02).

De tweede categorie (*skeleton-based in-betweening*) gaat ervan uit dat de problemen van lineaire interpolatie te wijten zijn aan het ontbreken van een centrale structuur. Bijgevolg gaat men de punten van de vormen herdefiniëren ten opzichte van bijvoorbeeld een centraal punt. Hierdoor bekomt men een skeletachtige representatie van de vormen waardoor men tijdens het interpoleren betere resultaten bekomt (Burtnyk 76; Shapira 95; Yu 96b).

Een heel andere mogelijkheid vinden we in het gebied van niet-fotorealistische representatie (NPR), meer bepaald *toon rendering*. Hierbij vertrekt men van een 3D animatie, bestaande uit 3D geometrische figuren. Achteraf wordt automatisch een volledige 2D animatie gegenereerd met behulp van speciale NPR algoritmes (Reynolds 04; Gooch 01; Strothotte 02). Hieraan zijn echter twee zware nadelen verbonden: (i) het modelleren en animeren in 3D is arbeidsintensief en laat weinig variatie toe, en (ii) de gegenereerde animatie blijft er steeds '3D-achtig' uitzien (figuur 2.4, pagina 14).

B.2.3 Bespreking: hoe automatiseren

De vorige secties hebben aangetoond dat automatisering een zeer grote impact op de tijdrovende aspecten van animatie heeft. Dit is voornamelijk te danken aan het gebruik van *key frame* systemen en het automatisch berekenen van tussenliggende beelden.

Echter, het berekenen van tussenliggende beelden brengt meer met zich mee dan enkel interpoleren tussen twee vormen. De animator houdt bijvoorbeeld rekening met de achtergrondkennis en de voorstelling van de wereld, emoties die overgebracht moeten worden, ...

Daarenboven heerst er nog steeds een (gerechtvaardigde) ongerustheid dat automatisering de creativiteit van de animator aantast.

Wij zijn er van overtuigd dat computer geassisteerde animatie op traditionele animatie gebaseerd moet worden en op zijn minst volgende voordelen moet bieden.

1. reduceren van de kosten
2. de animator moet dadelijk resultaten zien van wijzigingen etc.
3. tussenliggende beelden moeten automatisch gegenereerd worden voor ieder paar *key frames*
4. de automatisering van tussenliggende beelden moet ook complexe situaties aankunnen zoals rotaties uit het tekenvlak
5. de inkleuring moet geautomatiseerd worden

6. ondersteuning van gesofisticeerde hulpmiddelen die niet aanwezig zijn in traditionele animatie

7. simpele en intuïtieve interface om de dynamiek en timing van de animatie te bepalen

Al de bovenstaande punten worden later in deze thesis verder uitgediept.

B.3 2.5D modelleer- en animatieraamwerk

In hoofdstuk 3 (pagina 17) hebben we een nieuwe methode geïntroduceerd om automatisch tussenliggende beelden te genereren.

We hebben ons voornamelijk gebaseerd op het werk van (Patterson 94) waarin de problemen inherent aan het automatisch uittekenen van tussenliggende beelden worden besproken: (i) hoe verandert de silhouet, en (ii) hoe overlappen de verschillende onderdelen van het object elkaar. In veel gevallen kunnen deze problemen worden tegengegaan door de objecten in lagen of een hiërarchische structuur (*hierarchy display model (HDM)*) te verdelen. Voor hetzelfde object kunnen meerdere HDM's worden gecreëerd waarbij iedere HDM uit dezelfde onderdelen bestaat maar eventueel met een andere vorm (en dus ook silhouet) en volgorde. Eigenlijk geeft iedere HDM het object weer, bekeken vanuit een specifiek gezichtspunt. Een collectie HDM's vervangt in feite een echt 3D model. We noemen dit *2.5D modelleren* omdat het noch 2D noch 3D is. Een collectie HDM's herintroduceert in feite essentiële 3D informatie in 2D tekeningen.

In dit hoofdstuk richten we ons voornamelijk op de problemen aangehaald door Patterson en co. We doen dit door gebruik te maken van expliciete 2.5D modellering van HDM's. In feite zou $2.xD$ modellering een betere benaming zijn aangezien de hoeveelheid nodige en bijgevolg geïntroduceerde 3D informatie varieert van animatie tot animatie. Gemakshalve zullen we in de rest van deze thesis de term 2.5D modellering hanteren.

B.3.1 2.5D modelleren

Technisch gezien, kunnen we twee categorieën onderscheiden in 2D animatie: (I) transformaties in een vlak parallel met het tekenvlak (X-Y vlak) zoals rotaties rond de Z-as en translaties binnen het X-Y vlak, en (II) transformaties buiten het tekenvlak, voornamelijk rotaties rond een as verschillende van de Z-as.

De eerste categorie van transformaties is relatief makkelijk af te handelen. Daarentegen is het in de laatste categorie dat noodzakelijke 3D informatie — die wel mentaal aanwezig is bij de animator en de kijker — ontbreekt.

We hebben ons gebaseerd op het 3D computer animatieproces waar men een onderscheid maakt tussen drie fases: (i) modellering, (ii) animatie, en (iii) weergave. In traditionele animatie, daarentegen, zijn modellering en animatie verweven.

Dit heeft geleid tot de ontwikkeling van een raamwerk, bestaande uit meerdere lagen. De gelaagde structuur begint op niveau 0 met 2D basisprimitieven, in ons geval *2D subdivision curves* (zie figuur 3.1, pagina 21).
Level 1 beheert en verwerkt expliciete 2.5D modelleerinformatie en is fundamenteel in het realiseren van transformaties buiten het tekenvlak. Objecten worden gemodelleerd als verzamelingen van primitieven (gesorteerd op diepte) met betrekking tot rotaties rond de X- en Y-assen. De animator tekent dus voor iedere verzameling van ‘belangrijke’ XY-rotaties (van het object ten opzichte van de virtuele camera) een verzameling van geordende primitieven. Dit kan men vergelijken met de *extreme frames* in traditionele animatie (Blair 94; Patterson 94) (zie figuur 3.2, pagina 22).
Niveau 2 biedt de mogelijkheid om extra 3D informatie te verzamelen door middel van 3D skeletten of benaderende 3D objecten.
Het laatste niveau, niveau 3, is bedoeld om meer geavanceerde functionaliteit te introduceren zoals een hulpmiddel om tekeningen te vervormen of methoden om gezichtsexpressies te realiseren.

B.3.2 2.5D animeren

In een eerste stap maakt de animator de extreme poses aan, bijvoorbeeld in figuur 3.4 (pagina 24). In een volgende stap creëert hij/zij door een minimale invoer een tijdslijn (de *key frames*) aan de hand van deze extreme beelden (figuur 3.5, pagina 26). In een laatste stap worden alle tussenliggende beelden automatisch gegenereerd (figuur 3.9, pagina 31).

Ook voor het volgordeprobleem hebben we een aantal methodes ontwikkeld, gaande van manueel de volgorde te bepalen tot meer automatische methodes waarbij gebruik gemaakt wordt skeletten.

Een overzicht van enkele resultaten wordt getoond vanaf pagina 30.

B.4 “Potlood-en-papier” animatie

Om de animator te overtuigen de stap van traditionele naar computer geassisteerde animatie te zetten, hebben we een *multi-level sketching tool* ontwikkeld

(hoofdstuk 4). In eerste instantie biedt die dezelfde functionaliteit aan zoals in traditionele animatie, namelijk een intuïtieve pen. Daarnaast biedt die extra functionaliteit aan die niet terug te vinden is in traditionele animatie.

B.4.1 Multi-level sketching tool

In de meeste bestaande applicaties worden de tekenprimitieven (meestal curves) niet alleen *intern* op een wiskundige manier voorgesteld, maar ook *extern* (het scherm), bijvoorbeeld in de vorm van controlepunten, normalen, ... (figuur 4.1, pagina 37). Deze applicaties eisen bijgevolg heel wat 'klik-en-sleep-met-de-muis'-werk van de animator.

Wij hebben een *sketching tool* ontwikkeld waardoor de gebruiker geen notie meer hoeft te hebben van de interne representatie van de basisprimitieven (Vansichem 01). De animator wordt dus enkel geconfronteerd met het uiteindelijke resultaat (figuur 4.4, pagina 41). Hierdoor is het mogelijk om met enkel natuurlijke bewegingen van de hand een tekening tot stand te brengen.

Deze *tool* biedt ook mogelijkheden die niet bestaan in traditionele animatie, zoals het vervormen van 2D getekende figuren op een zeer eenvoudige en intuïtieve manier. De animator schetst eerst een curve die het te vervormen object voorstelt. Vervolgens zullen wijzigingen aan deze curve gereflecteerd worden in het vervormen van het 2D object (figuur 4.7, pagina 45). Onderliggend, en dus verborgen voor de animator, maken we gebruik van een aangepast *free-form deformation* algoritme (Sederberg 86).

Andere gebruiksmogelijkheden zijn:

- snel en makkelijk aanmaken van geometrische figuren (bijvoorbeeld *benaderende 3D objecten* beschreven in hoofdstuk 5, pagina 47)
- de ondersteuning van gesofisticeerde tekenhulpmiddelen (zoals bijvoorbeeld de *paint-* en *airbrush* in hoofdstuk 7, pagina 79)

B.5 Verrijken van traditionele computer geassisteerde animatie

In hoofdstuk 5 toonden we verschillende meerwaardes ten opzichte van traditionele animatie door de introductie van 'benaderende 3D objecten' — objecten die het realistische model benaderen qua volume en opbouw.

De animator gebruikt deze 3D objecten, ten eerste, als leidraad om het volume en de vorm van de 2D getekende karakters te behouden gedurende de animatie. Ten tweede kan het silhouet van het 3D object als referentie dienen

om de contouren van het 2D karakter uit te tekenen. Ten derde kunnen we het driedimensionale uitbuiten om coherentie tussen opeenvolgende frames te verzekeren. Dit laatste kunnen we verduidelijken aan de hand van volgend voorbeeld. Wanneer de animator een gestreept karakter wil animeren moeten dezelfde strepen in de opeenvolgende frames telkens weerkeren en dit op posities conform met het realistische model (figuur 5.1, pagina 49). Indien dat niet zo is, zal de kijker een chaotisch bewegend beeld voor ogen krijgen. Het gebruik van 3D modellen kan daarbij helpen aangezien zo de positie en oriëntatie van de strepen voor ieder beeld automatisch berekend kan worden.

Voor de ontwikkeling van de 'benaderende 3D objecten' volstaat het dat de animator 2D circulaire vormen tekent. Vervolgens worden deze vormen 'opgeblazen' zodat we een 3D model bekomen (figuur 5.3, pagina 52). We gebruiken een aangepaste versie van het TEDDY algoritme (Igarashi 99) waarbij brede gebieden heel volumineus worden terwijl smalle gebieden dun blijven.

Om het behoud van coherentie tussen opeenvolgende frames aan te tonen, hebben we een interactieve versie van Meier's algoritme geïmplementeerd (Meier 96). De animator hoeft slechts één *extreme frame* bij te schilderen; de andere *extreme frames* worden automatisch aangepast (figuur 5.5, pagina 56).

Meer concrete resultaten worden getoond vanaf pagina 59.

B.6 Animatie van gelaatsuitdrukkingen

In hoofdstuk 6 hebben we ons verdiept in het vergemakkelijken van één van de meest uitdagende taken van een animator: een getekend karakter tot leven brengen door de animatie van diens gezicht.

B.6.1 Bestaande technieken

Reeds sinds enkele decennia voert men onderzoek naar de realistische voorstelling en animatie van gezichten (Parke 72; Gleicher 98; Li 01; Bregler 02; Fidaleo 02). We verwijzen de lezer naar (Noh 98) voor een uitgebreide taxonomie van bestaande realistische modelleer- en animatietechnieken van het gelaat.

De laatste jaren zijn er enkele applicaties ontwikkeld die voornamelijk gericht zijn op 2D animatie van gezichten (Thórisson 96; Ruttkay 00; Ruttkay 01). Deze applicaties hebben echter veel beperkingen; zo ondersteunen ze ondermeer geen rotaties uit het vlak.

B.6.2 Nabootsen van geanimeerde gelaatsuitdrukkingen met minimale 2D invoer

We hebben een methode ontwikkeld die een grafische artiest helpt doorheen de creatie en animatie van gezichtsexpressies. Het tekenen van alle emoties van een karakter is een zeer tijdrovend proces, vooral wanneer het karakter vanuit verschillende camerastandpunten moet worden getoond. Bijgevolg hebben we ons enerzijds geconcentreerd op de eliminatie van het tijdrovende aspect, en anderzijds op het behouden van de vrijheid die een animator heeft om zijn artistieke visie ten tonele te brengen.

Daarvoor hebben we eerst het concept van *facial emotion channels* (FECs) ingevoerd. Iedere FEC kan gezien worden als de representatie van een specifiek deel van het gezicht dat tegelijkertijd een bepaalde emotie uit. Het belangrijke daarbij is dat ieder gezichtsdeel apart wordt gemodelleerd, onafhankelijk van de andere delen. Dus een gezicht wordt niet meer als een geheel getekend maar als een verzameling van gezichtsdelen. Dit heeft als voordeel dat ontelbare verschillende gezichtsexpressies kunnen worden gecreëerd zonder alle afzonderlijke expressies manueel te tekenen (figuur 6.2, pagina 70).

Bovendien hebben we een methode ontwikkeld om de meeste FECs automatisch te laten genereren. We starten daarvoor met een minimale invoer van de animator, namelijk alle FECs bekeken vanuit een specifiek camerastandpunt tezamen met emotieloze FECs bekeken vanuit andere cameraposities. Ons programma zal dan alle FECs voor de andere camerastandpunten automatisch genereren, rekening houdend met de artistieke stijl van de animator (figuur 6.3, pagina 73).

Enkele resultaten zijn afgebeeld vanaf pagina 76.

B.7 Gestileerde animatie

In de vorige hoofdstukken concentreerde het onderzoek zich vooral op de cartoon stijl. In hoofdstuk 7 wordt dit uitgebreid naar andere meer complexe stijlen zoals de traditionele verfborstel, *airbrush*, ... Daarenboven wordt er onderzocht hoe deze stijl in een animatie kan worden gebruikt, aangezien dit traditioneel enkel voor stilstaande beelden wordt gebruikt.

B.7.1 Bestaande technieken

Wat 2D aanpak betreft, bestaan er verscheidene technieken om ofwel zelf gestileerde tekeningen te maken (Haeberli 90) of bestaande beelden stilistisch weer te geven (Hertzmann 98). Geen van deze technieken laat echter toe om ani-

maties te creëren. Recent heeft men wel methodes ontwikkeld om bestaande video's artistiek weer te geven (Hertzmann 00; Hertzmann 01).

Andere technieken vertrekken dan weer van 3D geometrische figuren waarop automatisch (Meier 96) of manueel (Kaplan 00; Kalnins 02) *particles* 'geplakt' worden die elk een penseelstreek voorstellen.

B.7.2 Implementatie van vloeiende gestileerde animatie

Ons onderzoek heeft geleid tot de ontwikkeling van verscheidene verfstijlen die de animator kan aanwenden (figuur 7.2, pagina 84). We onderscheiden (i) de *solid brush* waarbij een verflaag telkens de vorige overlapt, (ii) de *paint brush* welke een traditionele verfborstel is waarbij overlappende verflagen gemengd worden, en (iii) de *air brush* waarbij zowel de dikte als de opaciteit van de verf gewijzigd kunnen worden tijdens het verven.

Qua animatie hebben we vastgesteld dat er bijkomende hulpmiddelen nodig zijn die de traditionele animator niet voorhanden heeft. Zoals reeds aangehaald wordt de verfstijl in traditionele animatie enkel gebruikt om stilstaande beelden te maken. Deze stilstaande beelden zijn het proces van herhaaldelijk penseelstreken over elkaar te tekenen totdat het gewenste resultaat bereikt is. Met andere woorden, eenmaal het resultaat bereikt is, kan men nog onmogelijk achterhalen hoeveel penseelstreken er getekend zijn, waar de penseelstreken getekend zijn, uit welke kleur, opaciteit ze bestaan, ...

In ons systeem zorgen we er zelf voor dat deze penseelstreken bijgehouden worden zodat we steeds de juiste hoeveelheid, volgorde, invloed, ... ter beschikking hebben. Voor het animeren van deze stijl hebben we er dan ook voor gekozen om de verschillende extreme poses niet telkens opnieuw te laten tekenen door de animator. Naast de ondersteuning van selecties (figuur 7.8, pagina 89) en transformaties (figuur 7.9, pagina 89) hebben we ook een *free-form deformation tool* ontwikkeld die de animator toelaat een bepaald gebied van een tekening te selecteren (bijvoorbeeld de mond) en deze dan te vervormen naar het gewenste resultaat (bijvoorbeeld een lachende mond) (figuur 7.10, pagina 91 en figuur 7.11, pagina 92). Op deze manier hoeft de animator slechts één versie te modelleren terwijl de andere extreme poses door middel van het vervormingsinstrument kunnen gebeuren.

Dit garandeert de animator dat er doorheen de animatie evenveel penseelstreken gebruikt worden zodanig dat de animatie niet te kampen heeft met penseelstreken die plots te voorschijn komen of verdwijnen.

Vanaf pagina 92 worden verschillende resultaten getoond.

B.8 Natuurlijke fenomenen

De moeilijkheid in traditionele animatie bij het animeren van bomen bestaat erin dat bomen zulke complexe structuren zijn waardoor het voor de animator een haast onmogelijke opdracht is om deze te animeren. Denken we bijvoorbeeld maar aan de hoeveelheid takken waaruit een boom bestaat. Een goede animator zal weliswaar een mooie tekening van een boom in een bepaalde stijl kunnen maken, maar vraag hem eens om diezelfde boom te tekenen maar nu bekeken vanuit een andere hoek (bijvoorbeeld vanachter, langs boven, ...).

De enorme hoeveelheid takken belemmert ons om (i) de juiste hoeveelheid takken te weten waaruit de boom is opgebouwd, en (ii) een idee te vormen van de volgorde waarin deze takken getekend moeten worden. Wanneer deze informatie ontbreekt, krijg je animaties waarbij takken plots te voorschijn komen en weer verdwijnen en wat zo een flikkerend resultaat oplevert.

Onze aanpak (hoofdstuk 8) bestaat erin om de voordelen van 2D en 3D te combineren: we gebruiken 3D om de juiste positionele informatie van de takken en bladeren te weten terwijl we 2D gebruiken om de animator de nodige vrijheid qua tekenstijl te geven.

B.8.1 Bestaand werk

Voor de bespreking van bestaand werk rond de modellering en animatie van cartoon bomen maken we een onderscheid tussen 2D en 3D technieken.

Binnen de 2D grijpt men ofwel terug naar gelaagde structuren (Guaglione 98) die mooie resultaten opleveren maar zeer arbeidsintensief zijn, ofwel naar de meer geautomatiseerde methoden (Cohen 00) die makkelijk te gebruiken zijn maar de animator zeer veel beperkingen opleggen.

Driedimensionale technieken daarentegen zijn meestal gebaseerd op niet-fotorealistische technieken (NPR) waarbij men 3D geometrische figuren met behulp van NPR algoritmes artistiek gaat weergeven (Meier 96; Kowalski 99; Markosian 00; Deussen 00). Het nadeel van deze aanpak bestaat erin dat kleine details moeilijk te modelleren zijn en dat de uiteindelijke weergave er nog steeds 3D-achtig uitziet.

B.8.2 Onze aanpak

Het modelleren gebeurt als volgt. In ons systeem start de animator steeds met een kale 3D boom. Er zijn tal van systemen (al dan niet *open source*) en algoritmes ter beschikking die de meest realistische bomen creëren. Dit is een ideale start: we kunnen de nodige diepte-informatie en dus ook tekenvolgorde (betreffende de takken) te weten komen. De structuur van de gebruik-

te bomen wordt gegenereerd met behulp van L-systemen (Prusinkiewicz 90; Van Haevre 03; Van Haevre 04) (figuur 8.3, pagina 105).

Ons onderzoek heeft uitgewezen dat deze informatie voldoende is om een flikkervrije animatie te garanderen, m.a.w. een animatie met een goede tijdscoherentie.

We hebben de artistieke vrijheid van de animator behouden door hem/haar voorbeelden te laten tekenen van individuele takken en bladeren. De animator tekent een beperkt aantal takken die als voorbeeld zullen dienen, telkens aangevuld met enkele versies die weergeven hoe deze takken eruit moeten zien bekeken vanuit een welbepaalde hoek (figuur 8.7, pagina 109). Hetzelfde gebeurt voor het gebladerte (figuur 8.6, pagina 106).

Het animeren gebeurt als volgt. In een voorbereidende stap wordt aan iedere 3D tak een 2D getekend exemplaar toegekend. Vanaf een bepaalde diepte in de boom wordt er eveneens gebladerte toegekend. Voor ieder beeld van de animatie wordt eerst de tekenvolgorde van de takken en het gebladerte bepaald. Dan wordt er voor iedere tak een aangepaste 2D tak (en gebladerte) gecreëerd: hierbij wordt er rekening gehouden met de oriëntatie en leeftijd (figuur 8.7, pagina 109). Vervolgens worden alle 2D takken en gebladerte van achter naar voren getekend.

Het resultaat is dat een initiële kale 3D boom naar een 2D getekende boom (met gebladerte) wordt omgevormd die de stijl van de animator reflecteert. Meer nog, aangezien de animator meerdere versies heeft getekend (die eigenlijk verschillende zichten voorstellen) kunnen op deze manier vloeiende, niet-flikkerende animaties worden gevormd.

Enkele resultaten zijn weergegeven vanaf pagina 112.

B.9 Gasvormige fenomenen

Hoofdstuk 9 concentreert zich op de gestileerde animatie van gasvormige fenomenen zoals rook, vuur en zelfs water.

B.9.1 Bestaand werk

We kunnen de bestaande technieken in drie categoriëen onderverdelen.

De eerste categorie bestaat uit technieken die door de gebruiker kunnen worden gecontroleerd. Hierbij kan de animator zelf ofwel enkel de tekenstijl van de animatie aangeven (bijvoorbeeld in (Yu 94; Yu 96a; Yu 99) en (Lamorlette 02)), ofwel enkel het tijdsverloop bepalen (Witting 99; Treuille 03).

In de tweede categorie worden fysica-gebaseerde oplossingen gehanteerd die realisme nastreven (Foster 97; Stam 99; Fedkiw 01; Selle 04).

In de laatste categorie bevinden zich de technieken die met behulp van heuristieken realistisch uitziende resultaten genereren (King 99; Dobashi 00; Perlin 01; Raghavachary 02).

B.9.2 Onze aanpak

Onze methode bestaat erin dat we de voordelen van zowel 2D als 3D technieken combineren. We bekomen de nodige 3D informatie door de animator 2D animatiepaden en versnellingscurves — afhankelijk van de kijkrichting — te laten tekenen. Dit is nodig om een vloeiende opeenvolging van de beelden en een juiste tekenvolgorde te garanderen. Daarnaast blijft de artistieke stijl behouden dank zij het gebruik van gestructureerde 2D modelleer- en animatietechnieken.

Het modelleren gebeurt als volgt. De animator tekent eerst 2D animatiepaden die het verloop van de animatie aangeven (figuur 9.1, pagina 121). Om een 3D-achtige animatie te bekomen, kunnen ook andere versies — bijvoorbeeld afhankelijk van het gezichtspunt — worden aangemaakt. Deze paden worden later (tijdens de animatie) 'opgeblazen' (Igarashi 99) en vormen zo een 3D traject. In een nog latere stap van de animatie zullen 3D *particles* in omloop worden gebracht. Voor ieder animatiepad kan er eveneens een acceleratiecurve worden gedefinieerd (figuur 9.2, pagina 122). Deze curve bepaalt de versnelling van de *particles* afhankelijk van de tijd en de afgelegde weg.

In een volgende stap worden de individuele componenten zoals vlammen, waterdruppels, rookwolken, enz. getekend (figure 9.3, pagina 123). Hier kan men eveneens meerdere versies voorzien die elk met een bepaalde kijkrichting of tijdstip overeenkomen. Dit gebeurt onafhankelijk van de gedefinieerde animatiepaden. Aan iedere 3D *particle* wordt zo'n 2D getekende component toegekend.

Het animeren gebeurt voor ieder beeld van de animatie als volgt. Eerst wordt er uitgaande van de *key frame* informatie een aangepast 3D traject gegenereerd. Vervolgens worden de 3D *particles* verder in omloop gebracht. Uitgaande van de huidige *key frame* informatie wordt voor ieder *particle* een tussenliggende versie van de bijbehorende 2D component gecreëerd. Vervolgens worden alle 2D componenten gesorteerd waarna ze allemaal van achter naar voren uitgetekend worden.

Aangezien dit voor iedere frame van de animatie gebeurt, krijgen we een vlotte, niet-flikkerende animatie waarin de artistieke inbreng van de animatie gereflecteerd wordt.

Enkele voorbeelden hiervan worden getoond vanaf pagina 125.

B.10 Toekomstige uitbreidingen

In deze thesis hebben we de meerwaarde van automatisering ten opzichte van traditionele animatie besproken. Namelijk, animators blijven (in grote mate) gespaard van de tijdrovende aspecten inherent aan traditionele animatie. En dit zonder hun artistieke vrijheid te beknotten.

De graad en wijze van automatisering kan natuurlijk steeds verder worden uitgewerkt en verfijnd. In de volgende secties gaan we dieper in op enkele van de vele mogelijkheden.

B.10.1 Dynamica van kledij nabootsen

Gedurende vele jaren is het een van de hoofdbetrachtingen in de computer graphics onderzoeksgemeenschap geweest om via allerlei algoritmen het realisme in computeranimatie zo goed mogelijk de realiteit te laten benaderen; daarbij grijpt men voornamelijk naar fysica-gebaseerde simulaties.

Een publicatie van Ronen Barzel vormt het uitgangspunt van onze interesse (Barzel 97). Hij beschrijft de simulatie van koorden en veren aan de hand van nagebootste dynamica. Wij zouden hetzelfde principe willen toepassen op hogere dimensionale vormen. Een voorbeeld van een uitbreiding naar bijvoorbeeld twee dimensies zou ons moeten toelaten om kledij of haar (Bando 03) te simuleren voor cartoon figuren. Naast het kanaal dat verschillende standpunten van de camera vertegenwoordigt, zouden er andere aan bod kunnen komen. We denken daarbij aan kanalen om de invloed van externe krachten weer te geven, zoals zwaartekracht, wind, ...

B.10.2 Gelaatsuitdrukkingen geometrisch beperken

In hoofdstuk 6 hebben we getoond hoe we, vertrekkend van een bepaalde gelaatsuitdrukking vanuit een specifiek oogpunt, andere gelaatsuitdrukkingen vanuit andere oogpunten kunnen worden gegenereerd. Figuur 6.4(g) (pagina 77) toont echter dat er problemen kunnen optreden: een gedeelte van de mond van het meisje is buiten het gezicht te zien.

Op dit moment kunnen manuele interventies van de animator deze problemen oplossen. Wij willen echter bijkomende geometrische beperkingen opleggen waardoor, afhankelijk van de gedefinieerde *pay-off* functies, automatisch gepaste interventies uitgevoerd worden.

B.10.3 Gelaatsuitdrukkingen mappen op 2D animatie

Het huidig onderzoek rond het herkennen van gelaatsuitdrukkingen (in video of ander beeldmateriaal) is sterk gevorderd (Byoungwon 01; Li 01; Fidaleo 02). Het merendeel van de toepassingen heeft echter enkel betrekking op het herkennen van de emotionele toestand van de gebruiker om alzo de interactie tussen mens en machine te vergemakkelijken.

Wij zijn echter geïnteresseerd in de integratie van (bestaande) herkenningssoftware in ons 2.5D raamwerk. Gelaatsuitdrukkingen zouden in *real time* kunnen worden gedetecteerd (en geïnterpreteerd) waarna de herkende emoties op getekende 2D karakters worden overgebracht. Dit zou ons bijvoorbeeld toelaten om de conventionele gebruikersinterface (GUI) te vervangen door een (live) acteur.

B.10.4 Natuurlijke fenomenen

In hoofdstuk 8 hebben we een techniek gepresenteerd om artistieke bomen te modelleren en animeren in een cartoon-achtige stijl. In de nabije toekomst zouden we graag onderzoeken of dezelfde of een gelijkaardige techniek ook voor andere soorten begroeiing gebruikt kan worden.

Tevens zouden we het willen uitbreiden naar niet-starre animatie, zoals bewegingen ten gevolge van de kracht van de wind.

B.10.5 Gestileerde animatie van fotografisch materiaal

Het doel van dit onderwerp is om fotografisch materiaal op een gestileerde manier te animeren. Daarbij starten we met een reeks foto's (bijvoorbeeld van een hoofd) die tegelijkertijd maar vanuit verschillende standpunten getrokken zijn. In een eerste stap is het de bedoeling dat de animator de corresponderende gebieden op de foto's aanduidt. Deze gebieden beschouwen we als verschillende lagen. Dit heeft als gevolg dat in feite voor iedere foto een hiërarchisch model (HDM) geconstrueerd wordt. In een volgende stap kunnen de verschillende gebieden gestileerd worden met behulp van bijvoorbeeld bestaande tekenmiddelen. In een laatste stap gaan we de verschillende HDM's gebruiken om de tussenliggende beelden te genereren waardoor we een gestileerde animatie bekomen van het fotografisch materiaal.

B.11 Conclusie

In deze thesis hebben we onderzocht hoe de tijdrovende aspecten van traditionele animatie tot een minimum kunnen worden herleid. Daartoe hebben we

zowel traditionele als bestaande geautomatiseerde animatie bestudeerd, waarna we de meest belangrijke problemen geïdentificeerd hebben. Namelijk, het herhaald tekenen van alle figuren voor ieder beeldje van de animatie én het gebrek aan 3D informatie die enkel aanwezig is in het hoofd van de animator.

Om deze problemen op te lossen, hebben we ons voornamelijk geconcentreerd op het automatisch uittekenen van tussenliggende beelden en het herintroduceren van noodzakelijke 3D informatie. Verder was het eveneens ons doel om de animator dezelfde artistieke vrijheid te geven als hij/zij in gedachten heeft.

Eerst en vooral hebben we een nieuwe methode ontwikkeld om tussenliggende beelden automatisch te berekenen. Deze oplossing is gebaseerd op vernieuwende 2.5D modelleer- en animatietechnieken.

Om de animator te overtuigen de stap van traditionele naar computer geassisteerde animatie te zetten, hebben we een *multi-level sketching tool* ontwikkeld. In eerste instantie biedt die dezelfde functionaliteit aan zoals in traditionele animatie, namelijk een intuïtieve pen. Daarnaast worden er extra mogelijkheden aangeboden die niet terug te vinden zijn in traditionele animatie.

Verder hebben we de nadruk gelegd op het gebruik van 'benaderende 3D objecten' om zo het volume en de vorm van 2D getekende karakters te behouden gedurende de animatie, en om coherentie tussen opeenvolgende frames te verzekeren.

We hebben ook een methode ontwikkeld die een grafische artiest helpt doorheen de creatie en animatie van gezichtsexpressies, vooral wanneer het karakter vanuit verschillende camerastandpunten moet worden getoond.

In een volgend hoofdstuk hebben we de cartoon-stijl uitgebreid naar andere meer complexe stijlen zoals de traditionele verfborstel, *airbrush*, ... Daarenboven hebben we onderzocht hoe deze stijl in een animatie kan worden gebruikt, aangezien dit traditioneel enkel voor stilstaande beelden wordt gebruikt.

Vervolgens hebben we een nieuwe aanpak toegelicht om artistieke, geloofwaardige bomen in een cartoon-stijl te creëren.

Deze aanpak hebben we later uitgebreid naar de meer turbulente bewegingen van gasvormige fenomenen zoals rook en vuur.

We zijn ervan overtuigd dat de aangereikte oplossingen makkelijk te gebruiken zijn en de animator in staat stellen om veel sneller cartoon animaties te genereren zonder aan vrijheid te moeten inboeten.