TicTacToon: A Paperless System for Professional 2D Animation

6 authors, including:

Jean-Daniel Fekete

National Institute for Research in Computer …

185 PUBLICATIONS   **3,652** CITATIONS

SEE PROFILE

Erick Bizouarn

Alcatel Lucent

5 PUBLICATIONS   **74** CITATIONS

SEE PROFILE

Thierry Galas

GE Healthcare

6 PUBLICATIONS   **146** CITATIONS

SEE PROFILE

TicTacToon: A Paperless System for Professional 2D Animation

Jean-Daniel Fekete*   Érick Bizouarn   Éric Cournarie   Thierry Galas   Frédéric Taillefer

2001 S.A.
2, rue de la Renaissance, F92184 ANTONY Cedex

*LRI, CNRS URA 410, Bâtiment 490
Université de Paris-Sud, F91405 ORSAY Cedex

Abstract

TicTacToon is a system for professional 2D animation studios that replaces the traditional paper-based production process. TicTac-Toon is the first animation system to use vector-based sketching and painting: it uses an original method to transform a pen trajectory with varying pressure into a stroke of varying thickness, in real-time. TicTacToon provides resolution independence, a virtually infinite number of layers, the ability to dynamically manage perspective and sophisticated support for reuse of drawings. Other innovations include replacement of the rostrum model with a 3D model and integration into the overall 2D animation production process.

TicTacToon is in daily use by 2D animation studios for a wide range of productions, from commercials to television series and even a feature film. The user interface enables professionals to sketch and draw as they do on paper. Over 100 professional animators have used the system over a period of two years and most need less than an hour before beginning productive work. TicTac-Toon eliminates most tedious tasks and frees professional animators for more creative work.

Keywords: 2D animation, vector-based sketching, cel animation

1 Introduction

The field of professional 2D animation has not profited much from advances in computer-assisted animation. Most professional studios still animate by hand, using a process that has changed little since the 1950's. In striking contrast to related fields such as commercials, art and 3D animation, 2D animation studios use computers in a supporting rather than a central role. Walt Disney Feature Animation [28] is the exception; they have been using a computer-assisted system since 1987. The key issues identified by Edwin Catmull [9] 17 years ago remain issues today.

Why are computer graphics tools so difficult to apply to 2D animation? It is not enough to simply solve technical problems. The studio must also be convinced. Today's creation process is essentially a production line, in which a studio of 50 to 300 people work together to produce tens of thousands of drawings for a single feature film or television episode. Everyone has a specified role and follows detailed procedures to move from one stage to the next. Any

attempt to computerize the traditional set of tasks must take into account overhead costs in both time and quality.

This paper begins by describing the animation process, to illustrate the problems faced by 2D animation studios. We then examine other solutions, partial or global, that have already been proposed. We then describe TicTacToon and evaluate it based on how well it addresses specific technical issues, handles user interface concerns and fits within the social organization of an animation studio.

1.1 The Traditional Animation Process

Figure 1 shows the traditional animation process. Most steps involve an *exposure sheet*, which lists all the frames in a scene. Each line includes the phoneme pronounced by each character and the order and position in which the camera will shoot the figures and the background.

Each scene requires a set of stages, of which only the background can be painted in parallel with the animation stages (from key-frame to paint). The stages include:

Story Board: Splits script into *scenes* with dialog and music.

Sound Track: Records dialog and music in prototype form.

Sound Detection: Fills the dialog column of an *exposure sheet*.

Layout: Manages the drawing of backgrounds and main character positions, with specifications for camera movement and other animation characteristics.

Background Painting: Paints the background according to the layout.

Key Frame Animation: Draws extreme positions of characters as specified by the layout. Provides instructions for the *in-betweeners*.

In-Betweening: Draws the missing frames according to the key-frame animator's instructions.

Cleaning: Cleans up the drawings to achieve final quality of the strokes.

Paint: Photocopies the clean drawings onto acetate celluloid (cels) and paints zones with water color.

Check: Verifies animation and backgrounds according to the layout and approves for shooting.

Record: Shoots frame-by-frame on film or video, using a *rostrum* (explained later).

2   State of the Art

Robertson's [27] survey of commercial computer graphics systems identifies two main types of animation systems: *Ink and Paint* and *Automated In-Betweening*. In both cases, all artwork is drawn on paper and later digitized and managed by the computer after the cleaning stage (Figures 1(a) and 1(b)).

2.1   Ink and Paint Systems

Ink and Paint systems perform the following steps, starting from scanned images of each animated character:

- Remove noise from images.

- Close gaps between strokes to prepare for paint.

- Paint using seed-fill.

- Compose images of characters and backgrounds, applying zoom, rotation and pans as in a *rostrum.*

- Record on film or video.

2.2   Automated In-Betweening Systems

Automated In-Betweening systems begin with a scanned key frame and perform the following steps:

- Clean and vectorize, sometimes using semi-automatic methods.

- Match pairs of drawings or match drawings to a template (see below).

- Paint, if colors are not part of the template.

- Interpolate in-betweens according to an exposure sheet.

- Render and compose images of characters and backgrounds, applying zoom, rotation and pans as in a *rostrum.*

- Record on film or video.

- Put in database for reuse.

2.3   Technical Issues

Catmull [9] and, more recently, Durand [10] discuss several technical issues to address in order to provide an effective system. We review some of them here.

2.3.1   Input of drawings

All commercial systems involve scanning drawings. Ink and Paint systems — like PEGS [34] and Toonz [21] — must scan every drawing whereas Automated In-Betweening systems — like Animo [7] — need only scan the key drawings. Catmull mentions tablets as a possible solution, but Durand argues that, "as sophisticated as they may be at this time, [they] could not offer the same versatility as traditional tools."

Toonz and PEGS scan drawings at a multiple of the final resolution to avoid jaggies. PEGS can optionally micro-vectorize scanned animations. The Animo system provides tools for automatic tracing from the scanned key drawing. In all cases, an operator is required for checking the results.
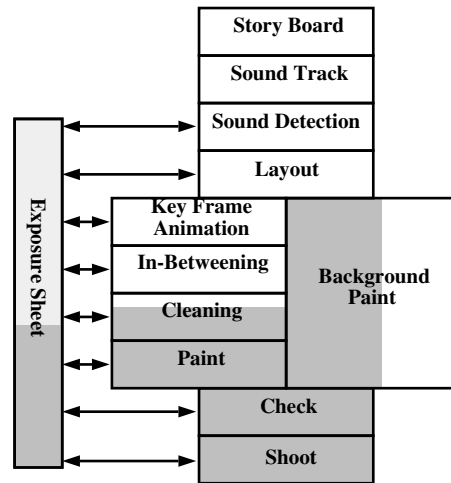
(a) The Ink and Paint Process
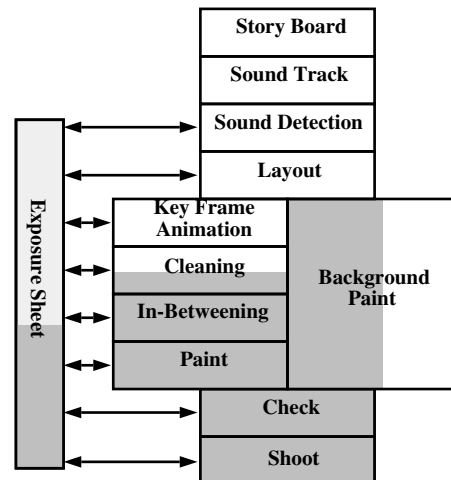
(b) The Automated In-Betweening Process
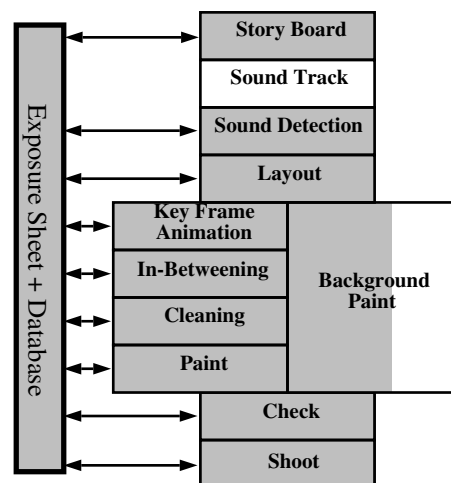
(c) The TicTacToon Process

Figure 1: Work flow of the different stages in animation Work done on computers is marked with a dark background

2.3.2 Automated In-Betweening

The two main classes of Automated In-Betweening systems are based on templates [6, 7] or explicit correspondence [31]. In template-based systems, a designer creates a template for each character. Animators must then restrict the character's movements according to the template. An operator must specify the correspondence between each key frame and the template. Explicit correspondence systems use two key drawings and generate in-betweens on a curve-by-curve basis. An operator is also required to specify the correspondence between the curves on the two key drawings. Sometimes, a zero-length curve must be created when a new detail appears or disappears between them.

2.3.3 Animation Painting

Ink and Paint systems use an area flooding algorithm [19]. All systems optimize the painting when successive drawings of a character contain zones that are mostly aligned and have the same color.

Automated In-Betweening systems associate graphic attributes with zones during the specification of correspondence.

2.3.4 Construction of Image Layers

Ink and Paint systems only manipulate pixel images. They support transforms to the geometry (pan, zoom, rotate), as well as to the color intensity, as described in [35]. In addition, all the commercial systems can apply special effects — specified in the software exposure sheet — to layers (color transforms, blur, transparency, etc.)

Vector-based systems use a variant of the scan-line algorithm [13, pages 92–99] to transform their graphical structure into a pixel image. For graphical attributes, flat colors and constant transparency are simple to implement. Automated In-Betweening systems such as [7] can also use textures and shading on animated characters since they can maintain space coherence.

2.3.5 Composition of Image Layers

Ink and Paint systems compose the layers using the alpha-blending arithmetic [24]. Vector-based systems can choose to compose layers at the pixel level or manage the composition during scan-conversion [8]. Commercial products provide no information about this point.

For recording, all current computer-assisted systems use the rostrum model to specify the way images are composed, positioned and turned relative to the camera. A rostrum includes a camera, a set of movable transparent trays holding the cels and a background area (Figure 2). The camera axis is always perpendicular to the layers of cels, but can be panned, moved forward or back, and zoomed in or out. Cels can also be moved or rotated on the trays.

The rostrum model makes managing an animation's perspective difficult. The animator must use traditional techniques for drawing perspective and translate it into the physical movement of the different layers of the rostrum. The computer system can only check if the calculation was correct and help to fix errors. Although Levoy [18] has proposed using a 3D editor, no commercial system has implemented one.

2.3.6 Other Computer Tools

Studios are not hostile to computer systems. They already use computer tools to assist them in stages such as storyboarding, sound detection, and layout. However, the data they produce must be re-entered into the process by hand.

Storyboards can be fine-tuned with an editing system (e.g. [2]) by mixing rough images with a sound track. The resulting storyboard must then be re-entered by hand.
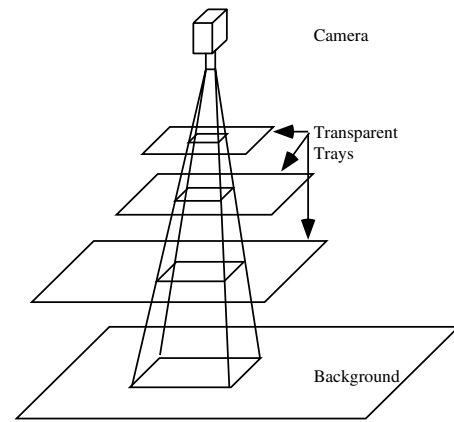

Figure 2: A Rostrum.

Sound Detection can be performed automatically by transforming the sound track into a list of phonemes which are then be transcribed by hand onto an exposure sheet.

Complicated layouts can be created with a 3D program, live action or rotoscoping. However, these cannot be integrated: the layout is printed on paper and used as in a traditional layout.

Animation can be checked with pencil tests. The animators record several drawings on a computer and play the animation in real time. Each drawing must be input with a scanner or camera and a subset of the exposure sheet must be typed in. Although computers have been available at this stage for some time, they are not always efficient. Animators must usually wait for the equipment, so they tend to draw more in-betweens than necessary. They later remove them, after checking the action on the pencil test machine.

3 Animation with TicTacToon

TicTacToon is designed to support a paperless 2D animation production line. To be successful, we had to solve a number of technical problems, provide a user interface that enhances the existing skills of professional animators and take into account the existing organizational structure of today's animation studios. This section describes the overall system and how it addresses each issue.

3.1 System Overview

TicTacToon structures the stages of the production line into a set of tasks. A workstation is assigned to a specific stage and individual tasks are performed within rooms [3] listed below:

- Lobby
- Storyboard/Database
- Layout
- Exposure Sheet
- Sketching/Animation
- Paint
- Render/Record
- Scan
- Electronic Mail

Figure 1(c) shows that TicTacToon supports all stages of production, except sound track recording. Major features include:

- a paperless process to avoid changing media,

- resolution independence to use real perspective and to enable reuse of drawings at any zoom level,

- a user interface that replicates the tools used by animators on paper, and

- a 3D model to replace the rostrum model.

These features eliminate many tedious tasks and increase productivity. TicTacToon also provides innovative solutions to the following problems:

- drawing directly on a computer system,

- painting on a vector-based system, and

- providing an environment acceptable to traditional animators without too much training.

TicTacToon runs on Digital Alpha AXP workstations using the X Window System [29] and a modified version of the InterViews toolkit [20, 11]. TicTacToon does not require any special graphic hardware; it interacts with the digitizer using the X Input Extension protocol [12].

3.2   Technical Issues

The design of TicTacToon poses several technical problems. This section describes how TicTacToon handles sketching, painting and rendering. Sketching is vector-based and painting uses planar maps and a gap-closing technique.

3.2.1   Vector-based sketching

Vector-based sketching poses two main problems: finding fast-enough algorithms and making a user interface suited to the animator's work. Three steps are involved: providing graphical feedback as the user draws, vectorizing the curve when the pen goes up, and redrawing the curve in place of the traced feedback. Users should not notice the last two steps.

The tablet sends an event every time the pen's state is modified. The state contains a pressure level, ranging from 64 to 512, depending on the digitizer, and X and Y coordinates, with a precision of 0.002 inches. The surface is 18"x12". Current digitizers can send up to 200 events per second if the user moves quickly enough.

TicTacToon uses the brush/stroke model described in [26]. A *brush* can be any convex shape and is defined by a Bézier path [1]. A *stroke* is a line drawn with a brush; the width is modulated by the amount of pressure applied to the pen. Pen pressure is normalized between 0 (no pressure) and 1 (full pressure). The brush shape is scaled according to the pen pressure and a linear modulation function. This function is defined by two scaling factors for pressures 0 and 1. If the factors are equal, the result is a fixed-width pen. In general, though, the scale factors are 0 for pressure 0 and 1 for pressure 1.

To compute the strokes, we use a variant of a fast curve fitting algorithm developed by Thierry Pudet. Compared to Plass [23], Schneider [30] and more recently Gonczarowski [16], this algorithm tries to optimize the fitting time rather than the number of Bézier segments that fit a set of sampled points.

Least Squares Curve Fitting   Previous curve fitting algorithms start with the samples, a tolerance parameter and perform the following steps:

1. Filter the samples to reduce noise.

2. Find corners or other singular points.

3. Define an initial parameterization for the samples.

4. Perform a least-squares curve fitting.

5. If the distance between the curve and the samples greatly exceeds the tolerance, split the samples in two parts, apply step 3 to both parts and connect the resulting segments.

6. If the distance between the curve and the samples exceeds the tolerance, change the parameterization and restart at 4.

7. Otherwise, output the segment.

In step 5, two more operations should be performed when splitting the samples: the splitting point should be chosen carefully and the resulting fitted curves should connect smoothly, i.e. the derivative at the splitting point should be evaluated and the least-squares fit should be constrained to maintain the direction of the derivatives at the connecting ends.

Step 5 is the bottleneck for this algorithm. Splitting the samples at the point of maximum distance to the computed curve, as in [30, 16], is computationally expensive, since the distance must be computed for every sample. Yet, this step, together with step 6, is essential for minimizing the number of segments.

Pudet avoids steps 6 and thus avoids computing the distance from the fitted curve to each sample point. Step 5 becomes:

5. If the distance between the curve and *any sample point* exceeds the tolerance, split the samples in two parts *at the middle*, apply step 3 to both parts and connect the resulting segments.

6. Otherwise, output the segment.

If the algorithm were used to fit cubic Bézier segments, it would produce too many small segments. Instead, Pudet fits quintic Bézier segments, which have more freedom than cubic and tend to fit more samples without re-parameterization.

Variable Width Stroke   Pudet's algorithm performs a second curve fitting for the outline of the stroke. Our variation improves feedback by eliminating this second fitting. We recompute the curve's envelope when it is redrawn and cache it to avoid further recomputation. We draw the envelope of each curve by generating a single polygon for the whole outline of the curve's stroke. We have optimized the most common brush shapes, e.g., circles and ellipses.

In order to make pressure information independent of the stroke's geometry, we define a *stroke profile* that maps a number between 0 and 1 (the normalized curvilinear index or NCI) to the pressure at that point. For example, the NCI for a point halfway along the stroke would be .5. This technique speeds computation of the envelope polygon and lets the stroke trajectory be modified while still keeping pressure information.

Another problem is noise: even the best digitizers suffer from both electronic interference and mechanical vibration. We apply a simple low pass filter to the samples. The fitting in itself filters the data, enabling the system to track the original gesture more accurately.

In summary, curve fitting transforms the input samples into a list of quintic Bézier segments and a stroke profile. Benefits include resolution independence (i.e., smooth curves at all zoom levels), compact representation of the pen's trajectory and filtering of the original stroke.

We have tested the vector-based sketching technique with professional animators for two years. This software requires a workstation with a fast FPU to achieve good performance; we used Digital AXP machines. The results have been very promising. The only problem reported involved performance when sketching and simultaneously running another program that loads the machine.
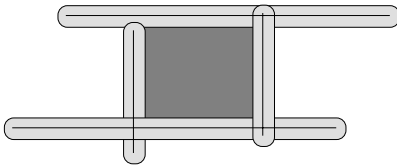
Figure 3: A zone looks closed but is actually open at the upper left corner.

We can envision a pathological case: drawing a very long stroke without releasing the pen. However, we found that this rarely occurs in practice. Animators only do it when cleaning a drawing and they still take more time to start a new stroke than for TicTacToon to draw the previous one.

3.2.2 Painting with planar maps and gap closing

A drawing consists of a list of strokes. In order to paint it, we must define a set of zones from these strokes, i.e. compute the topology of the drawing. We use the MapKernel library [14] to compute the planar topology defined by a set of Bézier paths. This topology is incrementally modified when adding or removing edges and also supports hit detection. The set of zones is extracted from the planar map as a new set of Bézier paths.

To compute the planar map, TicTacToon uses the central path rather than the outlines of each stroke, which halves the planar map's size. This allows later modifications of the brush strokes, e.g., when reusing a character. However, using central paths causes a problem: a zone that appears closed may actually be open. Even if the edges of the strokes intersect, the central paths may not (see Figure 3).

We use the algorithm we described in [15] that corrects this problem and finds the other small gaps that are always present in real drawings. Gaps are closed by adding invisible lines and changing the topology maintained by the MapKernel. This step takes several seconds and is usually run in the background to avoid making painters wait for the operation to finish. Once the gaps have been closed, a painter simply selects a color and points to the relevant zones. Painted zones are inserted at the beginning of the drawing's display list as one graphical element. This element consists of a grouped list of background zones, described as Bézier paths with a fill color. Since painted zones are rendered differently, as discussed below, this element is given a special tag.

3.2.3 Rendering

We use a painter's algorithm to render each graphic primitive into an image buffer which is then composed into the final image using alpha-blending arithmetic.

Our **graphical primitives** are very similar to PostScript [1], with the following differences:

- Instead of cubic Bézier paths, we support Bézier curves up to the 7th degree, though we only use quintics, cubics, quadratics and lines.

- Pixel images can have an alpha channel.

- Colors have an alpha component.

- We use a brush shape and a stroke profile to modulate the stroke width.

- Variations of attributes along and/or across the paths are supported, thus providing shading facilities (see Figure 5). We
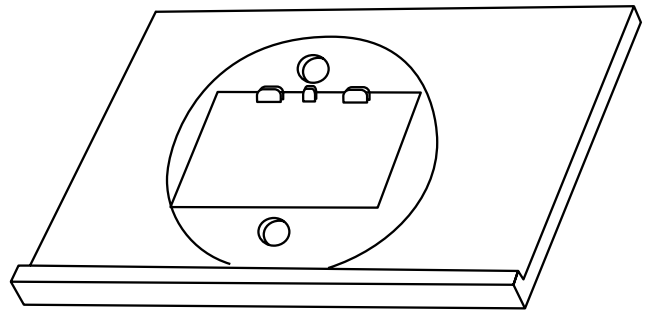

Figure 4: A Lightbox used by animators.

call them *annotations*. For now, they apply to color and transparency. An intuitive interface is available to generate such annotations.

The rendering algorithm proceeds by breaking down the strokes into elementary shaded polygons and scan-convert them.

First, the central Bézier path is discretized into line segments and normals are computed. Then, if the path only has a pressure profile, the stroke outline is generated and scan-converted. Otherwise, if there is an annotation along the path, attribute values are computed for each vertex of the discretized path and the stroke is split into 4-sided shaded polygons. When there is an annotation across the path, each polygon must be sliced into smaller pieces.

The polygons are then scan-converted into the image buffer. After the last polygon has been processed, the image buffer is merged with the final image.

The image buffer is used for applying special effects such as blur. It also helps to alleviate one current limitation of the alpha-blending arithmetic, which assumes that the area of a pixel covered by a polygon is randomly distributed over that pixel. Since the paths produced by the paint program are contiguous and never overlap, if two polygons overlap the same pixel, the total area covered is the sum of each area, which violates the assumption. Our algorithm renders all the components of such structured graphics into the same image buffer and merges pixel values by adding each color/alpha component (the "PLUS" operator of [24]). The image buffer is then overlaid on the final image, using the regular "OVER" operator.

3.3 User Interface Issues

Animators usually spend eight hours a day drawing, so it is critical to make the user interface both comfortable and easy to use. We observed animators at work in order to understand their needs and then iteratively designed the system, responding to their feedback about a series of prototypes.

This section describes how traditional animators work and then presents various aspects of TicTacToon's user interface, emphasizing tools that support sketching and the layout module.

Traditional animators work on a *Lightbox* (Figure 4). The disk is a sheet of translucent plastic that can be turned, using finger holes at the top and bottom. A strip light behind the disk lights up all layers of tracing paper at once.

Paper sheets are punched and inserted into peg bars, usually on top of the disk. The holes act as a reference and are used to accurately position the animations. Animators can work comfortably with this setup all day long.

3.3.1 Animation and Sketching Tools

TicTacToon's Animation Editor performs or enhances the following functions of a lightbox:

- stacking drawings,

- turning the light on or off, for stacks as well as individual drawings,

- changing the position of underlying drawings without actually modifying the drawings themselves,

- quickly flipping between successive sketches to find and check the best stroke position for a movement, and

- zooming, panning and turning the viewport.

TicTacToon provides a set of tools to support sketching, flipping among sketches and turning the viewpoint.

Sketching: Animators begin drawing in-betweens by stacking the initial and final key drawings. They put a new sheet of paper on the lightbox, turn it on, and sketch an in-between, according to the keyframe animator's instructions. They regularly check their drawings by turning off the light. The in-between animator must sometimes superimpose different parts of key drawings. For example, if a character jumps and his arm also moves, the movement of the arm may be seen more easily by superimposing the arms of the two key drawings.

Translated into TicTacToon actions, an animator creates a new drawing, drags the two key drawings, drops them into position, and begins sketching with the digitizer pen. Different parts of the key drawing can also be superimposed on the current drawing.

Flipping: Animators flip between sketches to check that their animations are correct. This requires manual dexterity, since they place the drawings between their fingers, and limits them to four or five drawings. They must also flip non-sequentially, since drawings are stacked as key 1, key 2 and in-betweens, rather than key 1, in-betweens and key 2.

TicTacToon can flip any number of drawings in any order. To create the impression of movement, it is important to switch between drawings in less than 5 milliseconds to maintain retinal persistency. We cannot guarantee this redraw speed since complex vector-based drawings can take an unbounded amount of time to draw. This is not a problem for most animators, but those who use a large number of strokes must check their animations with the pencil test module instead of flipping. Alternatively, we could cache the drawings as Pixmaps.

Turning the drawing: The human hand has mechanical constraints that limit its precision in some positions [33]. Animators can turn the disk to find the most comfortable and accurate drawing position. Most animators draw with the right hand and use the left for turning the drawing and turning the lightbox on and off.

TicTacToon also lets animators draw with one hand while looking at the drawing on the screen. Special function keys are assigned to the keyboard so that the other hand need not move to perform the following functions: undo/redo, flip up/flip down, turn the viewport, and reset the viewport.

Partial Editing of Strokes: We provide almost unlimited undo/redo. Animators make very few mistakes and would rather erase and redraw a stroke than twiddle with curve parts. We removed tools that enabled them to manipulate the Bézier curves because they were distracting and never used constructively. We may reconsider this decision if new paradigms for re-editing strokes prove successful (e.g., [4]).

Graphical feedback from the pen: We have experimented with several devices to provide feedback under the tracing pen. A ballpoint pen can be inserted into some cordless digitizers. If the animator places a sheet of paper on the digitizer, it is possible to draw with both the pen and the computer. However, animators do not like the feel of ball-point pens, which are too soft compared to their usual pencils. Moreover, they cannot "undo" a stroke on paper.

We have also tried a cordless digitizer on an LCD screen. With current hardware, a distance of several millimeters separates the pen surface from the LCD screen. This produces a parallax error, similar to that observed with touch-sensitive screens [25]. It is difficult for animators to continue their strokes because they often miss the expected starting position.

Animators prefer to draw on a flat surface and look at the drawing on the screen. One animator gave a compelling argument as to why he preferred this approach. After three months of uninterrupted work using TicTacToon, he tried using paper again. He told us that he found it very annoying because his hand was always hiding some part of the drawing.

3.3.2 The Layout Module

The Layout supports the whole animation process. It provides two views: the exposure sheet and a 3D view (Figure 8). The exposure sheet presents the logical structure of an animation: which characters and backgrounds are present in a specific frame. The 3D view presents the graphical structure: the position of characters and backgrounds and the trajectory they follow.

Unlike traditional animation, which provides only a static specification of a scene, TicTacToon begins with a playable version of each scene, using rough drawings. A layout animator can reuse animations or backgrounds from a database and can check camera movements and synchronize them with character movements. A copy of the layout specification is handed off to subsequent stages, which replace rough drawings with new work. Animators can check their drawings at any time, to ensure that they conform to the layout and that the action works. The layout stage can also check that work has been done correctly.

Unlike the traditional rostrum model, TicTacToon positions animations in a 3D world, in a way similar to Levoy [18]. However, the camera axis is always kept perpendicular to the drawings. This model acts like a theater background designed to be seen from only one perspective. As in 3D systems, the characters and camera are assigned a general trajectory in 3D space. Since they can't turn around the X or Y axis, they use 2D instead of 3D general transforms. This model automatically maintains perspective and the correct stacking order.

The columns of a traditional exposure sheet relate directly to the levels of a rostrum. The columns in TicTacToon's exposure sheet contain one character or a character part. An animation can be seen as a list of successive drawings or grouped into a cycle, which is a repeatable list of consecutive drawings. For example, a walking character is usually defined with a cycle of 8 drawings. TicTacToon exposure sheets handle cycles as structured objects that can be placed on a trajectory and tuned precisely. Prototype walks can be stored for most characters and reused as the basis for all their walks. Television serials only define actions for a few specific angles: 0, 30°, 60°, 90° and the symmetricals. Storing reusable walks (and runs) saves time for both the layout and other animators.

The Layout stage can check complex camera movements on prototypes and precisely tune the movement of each character. Animators can modify prototype walks, as when a wounded character limps, and save time by reusing parts of the prototype walk.

As in traditional animation, story boards consist of a set of small sketches of important actions in a scene with an associated script, including dialog, music, and intentions. Figure 6 shows a Story Board designed with TicTacToon.

Traditional animation uses only eight mouth shapes (Figure 7). We have developed a mouth shape detection program that avoids using phoneme recognition and is speaker and language independent. The success rate is around 90%, with errors only at the transitions. Mouth shapes can be edited, played and corrected and the result is directly inserted in the sound column of the exposure sheet.

Figure 5: Rendering effects on strokes.
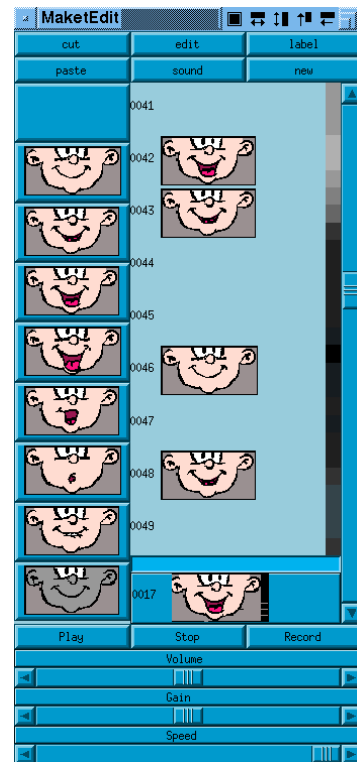
Figure 6: The story board module.
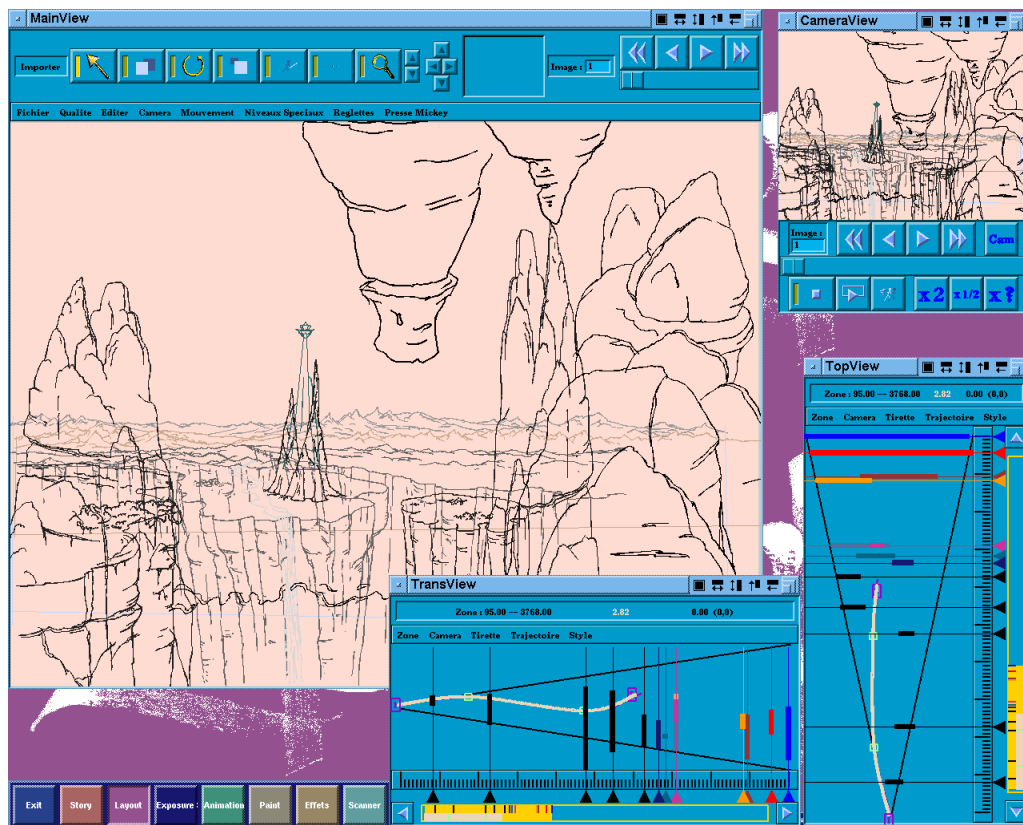
Figure 7: The Voice Detection module.

Figure 8: The Layout module showing a front view, top view and side view of a scene.

Over 100 professional animators in several studios have used TicTacToon over the past two years. So far, more than 90% were able to immediately switch from paper to TicTacToon. The unsuccessful animators included some who refused to try it, some who were afraid of computers in general and some who had difficulties with specific aspects of the user interface. Animators have been the strongest supporters of TicTacToon and some have convinced their studios to adopt it. This is in strong contradiction with Durand in [10].

We addressed an early criticism of the digitizer's pen and replaced it with a model that was both pressure sensitive and rigid. The other major criticism is the digitizer's surface which is too slippy for some and too soft for others. We find similar kinds of individual preferences about pencil hardness and paper quality. Although most animators adapt to the digitizer's surface after about a month, we are still investigating possible improvements.

3.4   Social organization of the work

To succeed, an animation system must provide more than useful technical features and a good user interface; it must also fit within the existing work context of an animation studio. Both formal and informal communication are essential for the smooth running of a studio. Since animators express themselves with cartoons as well as words, we provide a metamail compliant mailer [5] with TicTacToon. We receive mail messages from production sites all over the world, including formal (bug reports, request for information) and informal (jokes, tricks and caricatures), illustrating the same range of communication as would be found in a traditional animation studio.

TicTacToon supports a number of operations that help manage the production line. Animators exchange their work via exposure sheets and a distributed database. They also use electronic mail to notify the necessary people when their work is ready to be processed and get feedback about their own work. We do not want the studio staff to have to become system operators. TicTacToon does not require prior computer skills and we have hidden operating system commands such as copying directories and naming files.

TicTacToon is designed to enhance the animation process, not just raise productivity. As 3D production studios increase the scope of their work, we expect that they will require the same level of organizational support needed by today's 2D studios.

4   Discussion

Current tools to support animation have both advantages and disadvantages over traditional techniques. This section examines the advantages and disadvantages of two types of computer tools, Ink and Paint systems and Automated In-Betweening systems, and then discusses the relative advantages and disadvantages of TicTacToon.

4.1   Ink and Paint Systems

4.1.1   Advantages

Speed up painting: Hand painting is the most tedious task in traditional animation: using a computer can speed up the process by a factor of 5 [34]. The average speed is 100 drawings per day depending upon the number of zones per drawing and the number of colors being used. In traditional animation, a maximum of 40 drawings can be painted per day, with an average of 20.
Remove limits on the number of layers: Computers can also handle more than the five composition layers found in traditional animation, which is limited by the opacity of acetate celluloid.
Simplify shooting: A virtual rostrum is easier to use than a real rostrum.

Allow images from other sources: Highest quality backgrounds are still hand-painted and scanned in. However, images from other sources — like paint programs, rendering programs or digitized live action — can also be imported.

4.1.2   Disadvantages

Increase the cleaning time: Films require huge numbers of drawings, in the order of tens of thousands. Computer systems can add manual actions to traditional work and machine processing time can be slow. For example, Ink and Paint systems require cleaner drawings than traditional animation does. Animators must spend extra time cleaning to avoid having the computer spend more time filtering, which increases the cost of the hand-cleaning stage.
Consume large resources: Reuses, though possible, are only made by large studios. With the exception of backgrounds, most images are usually removed from disk after recording. Pixel images require a lot of storage, especially if the final format is 35mm film. The technology required to manage such images remains expensive and without a planned long term production (over two years), the extra storage and network cost is not guaranteed to be balanced by the increase in productivity. Some systems address this by vectorizing the images after the cleaning stage [34], most others don't [21, 22].
Add medium change costs: Finally, several marginal costs result from the transition between paper and computer:

- A special staff is required to scan and check drawings.

- The exposure sheet must be typed into the system.

- Information is not usually propagated back from the computer to the paper world. Since artists do not receive feedback, some problems recur.

The overhead of transferring between paper and computer accounts for a minimum of 10% for a well-organized process to 25% or more for less well-organized processes [17] in the final cost.

4.2   Automated In-Betweening Systems

4.2.1   Advantages

Reduce the number of hand-drawn in-betweens: Computer-assisted In-betweening is designed to reduce the number of hand-drawn in-betweens and increase reuse of animation. Once a drawing has been vectorized and the correspondence specified, an animation can be tuned precisely. Some actions can be reused at different tempos which also decreases the number of hand-drawn drawings.
Allow procedural rendering: Since zones and edges composing successive key drawings are matched, the interpolation process can automatically maintain space coherence between procedurally computed attributes used for rendering. For example, the checked texture of a character's shirt will be correctly animated when he moves his arms. Unlike with traditional animation, many special effects can be animated automatically.

Like Ink and Paint Systems, Automated In-Betweening Systems allow an unlimited number of layers, an easier shooting and access to images from other sources. Painting takes a small portion of the key frame matching process.

4.2.2   Disadvantages

Change the nature of in-betweening and limit its complexity: Crafting an animation with an Automated In-Betweening system is slow: We have found that inputing and tuning simple in-betweens take about the same time as drawing them by hand. Even with recent advances in algorithms, it remains difficult to match a drawing to a
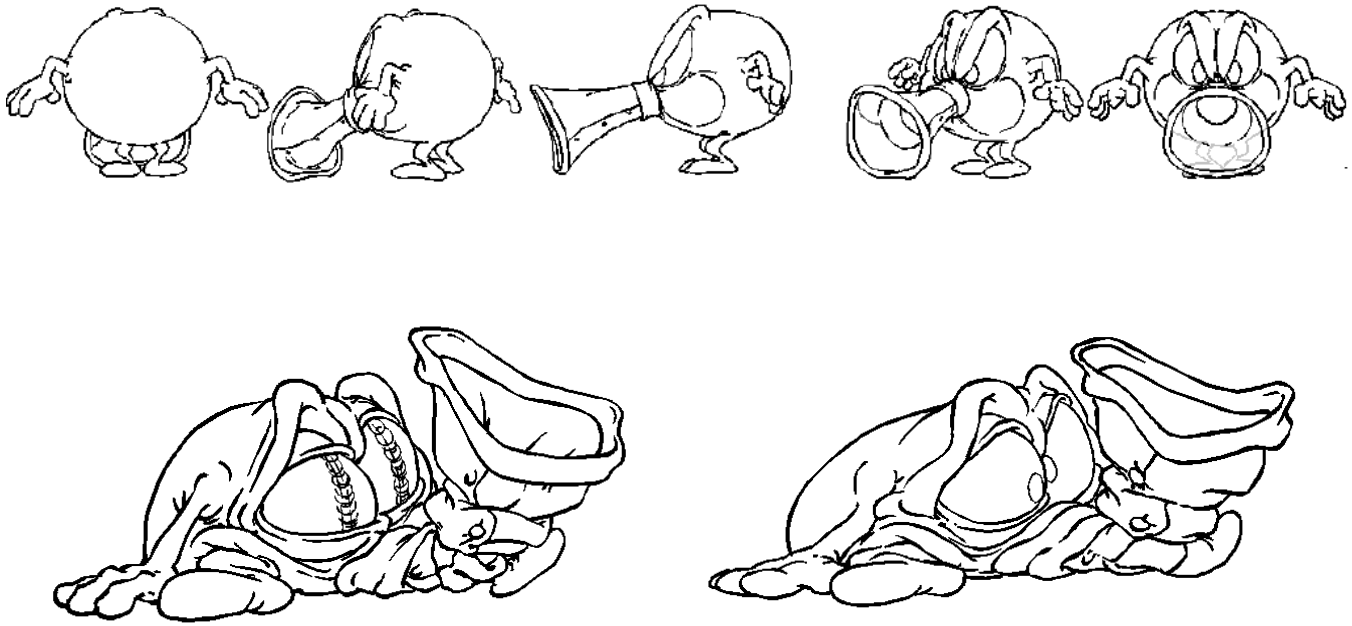
Figure 9: Model of the Klaxon character and two key drawings very distorted from the model.

template or two successive drawings to each other. Template-based systems require key drawings to conform to the model. (Figure 9 illustrates how much key frames can differ from the model.) Explicit correspondence systems must match each key drawing to the previous and next drawings. In both cases, the animation is restricted to fairly standard drawings; matching unusual images is very time-consuming.

Traditional animators use a special notation on the key frames to specify the rhythm of the animation. Automated In-Betweening systems make it difficult to control this rhythm, which lowers the quality of the animation. Generating automated in-betweens is time-consuming and transforms the animator's work from drawing images to removing and tuning the *slowing in* and *slowing out* of an action. Tuning in-betweens requires a cross of skills from a traditional animator and a computer graphist who can precisely manipulate Bézier curve handles. Few such animators exist.

Require a skill in modeling: Template-based systems require both 3D modeling as well as traditional animation skills. Crafting a good model is expensive and must be taken into account when estimating the cost of the production. Given this cost, these systems are only cost-effective for productions with few characters.

Does not provide as much reuse as expected: Animations are less reusable than might be expected. Reuse must be planned at the storyboard stage, which is not integrated with the computer part of the process. All these problems contribute to the perception by animation studios that computer assisted in-between programs produce poor quality animation.

Moreover, like Ink and Paint systems, they also add medium change costs.

4.3 TicTacToon

4.3.1 Advantages

Avoids medium change costs: With TicTacToon, all stages of the work are performed on computer (see Figure 10), with one workstation being allocated to a specific stage. This avoids the cost of transferring from paper to computer.

Offers resolution independence: Vector-based backgrounds provide both resolution independence and more importantly, an alpha channel. We also integrate pixel-based images, which can be made with a paint program or scanned in.

Allows reuses: TicTacToon animations are vector-based (Bézier-based) and require 20KB to 100KB per character. We provide a database system to manage and qualify animations.

Distributes information in a playable form: Each stage has access to all information, from the earliest written scenario to the database of reusable images. Animators can test scenes at any time, to clarify the author's intent. Paper need not be carried from place to place, risking loss or damage and animators can spend more time at their desks drawing.

Provides interactive tools to increase animator's productivity: TicTacToon provides a number of features that support high quality animation. Compared to Automated In-Betweening systems, animators spend more time on art work and can check their work interactively. The animations themselves can be much more complicated, sometimes containing hundreds of layers. Scenes can be played and checked before even starting character animation and background painting. As with traditional animation, TicTacToon permits varying width strokes.

Animators using TicTacToon can work up to 30% faster than with traditional animation. If we also include cut and paste and reuse of existing images, the savings are even greater.

TicTacToon share all the advantages of Ink and Paint systems.

Painting speed is roughly the same. By improving the rostrum model, it further simplifies the shooting of complex scenes, using hundreds of layers. Also, TicTacToon accepts scanned backgrounds and provides tools to manage them, including cut overlays (chroma key), assembly of background parts, and correct/adjust colorimetry. These tools are the most tedious to use and implement.

Pixel-based backgrounds can be made device-independent by scanning them at high resolution (we recommend 400 to 600 dpi). Although this requires more storage, the relative number of backgrounds is small (in the thousands) compared to the number of animated drawings.

4.3.2 Disadvantages

Changes the nature of layout: Some traditional layout animators reported problems using the layout program. They are more familiar with the rostrum model than the 3D model and have learned how to "cheat" with perspective. However, layout animators who work with special effects had no problems.

Does not allow fast enough flipping for some animators: Animators who use a large number of strokes can find the redraw latency too slow when flipping between animations. These animators must use the pencil test module to check their animation. Some animators would like an in-betweening module for simple animations (see the section on Future Work).

Does not change the nature of painting: Painting is still tedious, although we have tuned the user interface to be as simple as possible. The current interface is much faster, but keeps the painter so busy clicking the mouse to paint and using the keyboard to change the drawing that the job seems much more painful. The studios hire a non-skilled person to perform this task. We would like to avoid this gap in skill since it has important economic and social implications.

Limits the quality of vector-based backgrounds: Vector-based backgrounds does not yet achieve the level of quality of pixel based images made by the best paint programs or scanning.

5   Future work

We are enhancing TicTacToon in the following ways:

Speed up the painting process: We would like to handle cleaning and painting in the same stage. This would require automatically finding the color of a zone, either from a template or from a previously painted cel.

Support some procedural rendering: We already allow more graphical effects than used on traditional character animation. However, we are working on adding procedural rendering like computed texture for strokes and 3D effects.

Optimize rendering: Currently, a single frame can take from a few seconds to several minutes to compute. Many optimizations can be performed, including those described by Wallace [35] and Shantsis [32]. Others are more specific to vector-based drawings, such as caching rendered images for later reuse. However, since an image recorder takes 30 seconds to shoot a frame on a 35mm film, it is only important to increase rendering speed when images take more than 30 seconds.

Implement Computer-Assisted In-Betweening: Animators usually ask for assistance when in-betweening objects such as bouncing balls, falling snow, and air bubbles in water. We are working on a special module for this.

Connect with a 3D system: We are currently working on inferring 3D features from 2D drawings.

6   Conclusion

TicTacToon is a system for professional 2D animation studios that replaces the traditional paper-based production process. It provides a practical solution to the problem of converting from analog to digital 2D animation.

TicTacToon is the first animation system to use vector-based sketching and painting. Other innovations include replacing the rostrum model with a constrained 3D model and integration of the system into the overall 2D animation production process. TicTacToon eliminates most tedious tasks, freeing animators for more creative work. We believe that TicTacToon can be the first of a new generation of tools to support digital animation.

Acknowledgments

References

[1] Adobe Systems Incorporated. *PostScript Language Reference Manual*. Addison-Wesley, Reading, MA, USA, second edition, 1990.

[2] Adobe Systems Incorporated, 1585 Charleston Road, P. O. Box 7900, Mountain View, CA 94039-7900, USA, Tel: (415) 961-4400. *Adobe Premiere 1.0 User Guide*, 1993.

[3] D. Austin Henderson, Jr. and Stuart K. Card. Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention in a Window-Based Graphical User Interface. *TOGS*, 5(3):211–243, 1986.

[4] Thomas Baudel. A Mark-Based Interaction Paradigm for Free-Hand Drawing. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, Marina del Rey, 1994. ACM.

[5] Nathaniel S. Borenstein and Ned Freed. MIME (Multipurpose Internet Mail Extension): Mechanism for Specifying and Describing the Format of Internet Message Bodies. Request for Comments: 1341, June 1992.

[6] N. Burtnyk and M. Wein. Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation. *Communications of the ACM*, 19:564–569, 1976.

[7] Cambridge Animation. *The Animo System*.

[8] Edwin E. Catmull. A hidden-surface algorithm with anti-aliasing. *Computer Graphics (SIGGRAPH '78 Proceedings)*, 12(3):6–11, August 1978.

[9] Edwin E. Catmull. The problems of computer-assisted animation. *Computer Graphics (SIGGRAPH '78 Proceedings)*, 12(3):348–353, August 1978.

[10] Charles X. Durand. The "TOON" project: requirements for a computerized 2D animation system. *Computers and Graphics*, 15(2):285–293, 1991.

[11] Jean-Daniel Fekete. A Multi-Layer Graphic Model for Building Interactive Graphical Applications. In *Proceedings of Graphics Interface '92*, pages 294–300, May 1992.

[12] Paula Ferguson. The X11 Input Extension: Reference Pages. *The X Resource*, 4(1):195–270, December 1992.

[13] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Fundamentals of Interactive Computer Graphics*. Addison-Wesley Publishing Company, second edition, 1990.

[14] Michel Gangnet, Jean-Claude Hervé, Thierry Pudet, and Jean-Manuel Van Thong. Incremental Computation of Planar Maps. In Jeffrey Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 345–354, July 1989.

[15] Michel Gangnet, Jean-Manuel Van Thong, and Jean-Daniel Fekete. Automated Gap Closing for Freehand Drawing. [Technical Sketch] SIGGRAPH'94, Orlando, 1994.

[16] Jakob Gonczarowski. A Fast Approach to Auto-tracing (with Parametric Cubics). In Robert A. Morris and Jacques André, editors, *Raster Imaging and Digital Typography II— Papers from the second RIDT meeting, held in Boston, Oct. 14–16, 1991*, pages 1–15, New York, 1991. Cambridge University Press.

[17] Claude Huhardeaux. The Label 35 System. Personal Communication, 1989.

[18] Mark Levoy. A Color Animation System Based on the Multiplane Technique. In *Computer Graphics (SIGGRAPH '77 Proceedings)*, volume 11, pages 65–71, Summer 1977.

[19] Mark Levoy. Area Flooding Algorithms. In *Two-Dimensional Computer Animation, Course Notes 9 for SIGGRAPH 82*. ACM Press, New York, NY 10036, USA, July 1982.

[20] Mark A. Linton, John M. Vlissides, and Paul R. Calder. Composing user interfaces with InterViews. *IEEE Computer*, 22(2):8–22, February 1989.

[21] Microsoft Corporation. SoftImage Toonz Feature Summary. Part No. 098-58493, One Microsoft Way, Redmond, WA 98052-6399, 1995.

[22] Bill Perkins. The Creative Toonz System. Personal Communication, 1994.

[23] Michael Plass and Maureen Stone. Curve Fitting with Piecewise Parametric Cubics. *Computer Graphics (SIGGRAPH '83 Proceedings)*, 17(3):229–239, July 1983.

[24] Thomas Porter and Tom Duff. Compositing Digital Images. In Hank Christiansen, editor, *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 253–259, July 1984.

[25] Richard L. Potter, Linda J. Weldon, and Ben Shneiderman. Improving the Accuracy of Touch Screens: an Experimental Evaluation of Three Strategies. In *Proceedings of ACM CHI'88 Conference on Human Factors in Computing Systems*, pages 27–32, Washington, DC, 1988.

[26] Thierry Pudet. Real Time Fitting of Hand-Sketched Pressure Brushstrokes. In *Eurographics'94. Proceedings of the European Computer Graphics Conference and Exhibition*, Amsterdam, Netherlands, 1994. North-Holland.

[27] Barbara Robertson. Digital Toons. *Computer Graphics World*, pages 40–46, July 1994.

[28] Barbara Robertson. Disney Lets CAPS Out of the Bag. *Computer Graphics World*, pages 58–64, July 1994.

[29] Robert W. Scheifler and Jim Gettys. The X Window System. *ACM Transactions on Graphics*, 5(2):79–109, 1986.

[30] Philip J. Schneider. An Algorithm for Automatically Fitting Digitized Curves. In Andrew S. Glassner, editor, *Graphics Gems I*, pages 612–626, 797–807. Academic Press, 1990.

[31] Thomas W. Sederberg, Peisheng Gao, Guojin Wang, and Hong Mu. 2D Shape Blending: An Intrinsic Solution to the Vertex Path Problem. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 15–18, August 1993.

[32] Michael A. Shantsis. A Model for Efficient and Flexible Image Computing. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 147–154. ACM SIGGRAPH, ACM Press, July 1994.

[33] Peter Van Sommers. *Drawing and Cognition*. Cambridge University Press, Cambridge, 1984.

[34] Jean-Michel Spiner. *PEGS Technical Description V1.00*, 1994.

[35] Bruce A. Wallace. Merging and transformation of raster images for cartoon animation. *Computer Graphics (SIGGRAPH '81 Proceedings)*, 15(3):253–262, August 1981.

(a) Rough drawing.

(b) Clean drawing.

(c) The cleaned animation sequence.

(d) Painted drawing.

(e) Final scene.

Figure 10: A character at different stages in the TicTacToon system