Automatic In-betweening in Computer Assisted Animation by Exploiting 2.5D Modelling Techniques

Fabian Di Fiore, Philip Schaeken, Koen Elens, Frank Van Reeth

Expertise Centre for Digital Media - Limburg University Centre
Wetenschapspark 2, B-3590 Diepenbeek
Belgium
E-mail: {fabian.difiore, philip.schaeken, koen.elens, frank.vanreeth}@luc.ac.be

Abstract

This paper 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 (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 elucidated.

Keywords: 2.5D modelling, 2.5D animation, character animation, real-time animation, interpolation, in-betweening, computer-assisted animation.

1. Introduction

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 paper, we refer to animations where 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 recognizable 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 shows some images of the type of characters we would like to animate. In this example the animator aims to express the cuteness of the dog by focussing the attention on the engaging smile. Notice also that although the legs of the dog are not drawn anatomically correct, though they still are recognizable!

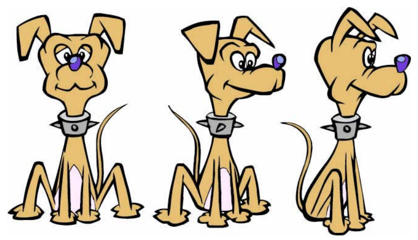


Figure 1. A cute traditional animation dog.

Traditionally, 2D animation production has been a labour-intensive artisan process of building up animated sequences by hand. [2, 13, 15] give the interested reader an overview of how a hand-drawn animation is prepared. Most work and hence time is spent on drawing, inking and colouring of the individual animated characters for each of the frames.

Within the boundaries of our study, it is our goal to eliminate these time-consuming aspects of traditional animation, especially the repeated drawing of all characters in all frames. Furthermore, it is also our goal to give the animator the same freedom of exaggeration to create animations such as shown in figure 1. We establish our goals by developing a computer animation process that assists the animator with an automatic in-betweening solution. The approach is based on a new 2.5D modelling and animation technique, which we implemented as a multi-level software architecture. The provided solution should be especially beneficial in the production of traditional animation feature films and series.

The paper is organised as follows. Section 2 describes the related work in the field. Section 3 puts 2.5D modelling and animation in its context and details our multi-level approach. We show some results in section 4 while conclusions and topics for future research are given in section 5.

2. Related work

Already in the early seventies, [3] reported on the use of computers in the generation of keyframed animation. Catmull [5] 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.

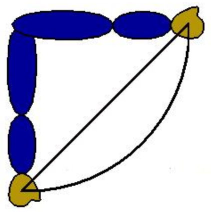


Figure 2. Rotating arm with the shoulder as pivot point. Using linear interpolation causes the hand to follow the path of the straight line. In that case, during the animation, the arm will shrink at the beginning and enlarge towards the end of the animation. It is obvious that the hand has to follow the curved circle segment path during animation in order to preserve the original shape and proportions of the arm.

Some of the problems with in-betweening have later been addressed by [17], utilising moving point constraints. [27] proposes the use of linear interpolation of polar coordinates to circumvent the abovementioned shrinking problem.

[12] discusses an experimental 2.5D keyframe animation system, focussing 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. He succeeded in achieving his aims, but animations could suffer from occlusion problems. Litwinowicz acknowledges that 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.

[15] re-addresses the issues in the '78 paper of Catmull, confirming automating in-betweening to be the key problem in 2D animation, which essentially breaks down into two sub-problems: how silhouette outlines change and how the various parts of the object occlude themselves. The use of (i) polar coordinate interpolation, (ii) appropriate continuity control and (iii) a 2.5D hierarchy display model (HDM) are identified as key components to effective in-betweening. The work presented in this paper focuses on detailing some of the issues addressed in [15], by exploiting explicit 2.5D modelling of hierarchical models in a multi-level animation architecture.

Our previously reported work on exploiting 2.5D animation techniques focuses on positioning and animating hierarchical 2D cut-out animation characters [22] respectively free form characters [6] in a 3D world space.

Over the last few years various commercial software packages have been developed for assisting the animator in the production of traditional cartoon animation. [9] provides a good overview on the available systems, including some case studies.

We conclude this section by mentioning that 'Toon rendering', a subcategory in the non-photorealistic rendering (NPR) domain, also offers solutions to the occlusion problem and the changing silhouette problem. 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 3 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 emphasize.

An in-depth overview of published work in the NPR domain can be found at [18].

3. Modelling and animation in 2.5D

3.1. Traditional 2D animation versus 3D computer animation

When looking to 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 with 3D polygon meshes or curved surfaces, (ii) an

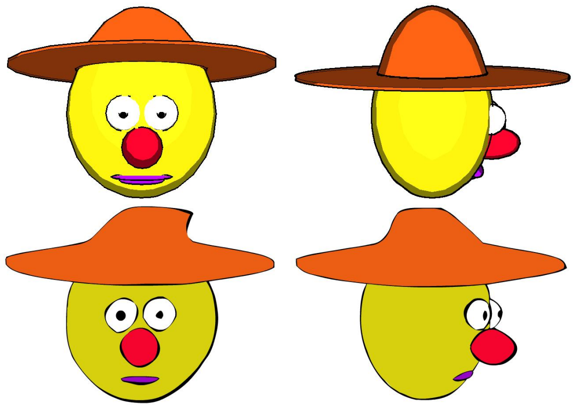


Figure 3. These pictures show (a) a man's head face-on and (b) a side-view obtained by NPR techniques and the same views (c, d) as an animator is likely to draw it.

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 [2, 13, 15], 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) main 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 are responsible for creating all the remaining in-between frames of the animation.

3.2. Our solution: multi-level 2.5D modelling and animation

3.2.1. Introduction. Considering 2D animation from a technical standpoint, two distinctly different categories can be considered: (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 plane, 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 2.5D animation systems referenced in section 2, such as [6], [12] and [22] are 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 should also be capable of animating characters in category (II). Hence, we present a multi-level 2.5D modelling and animation approach:

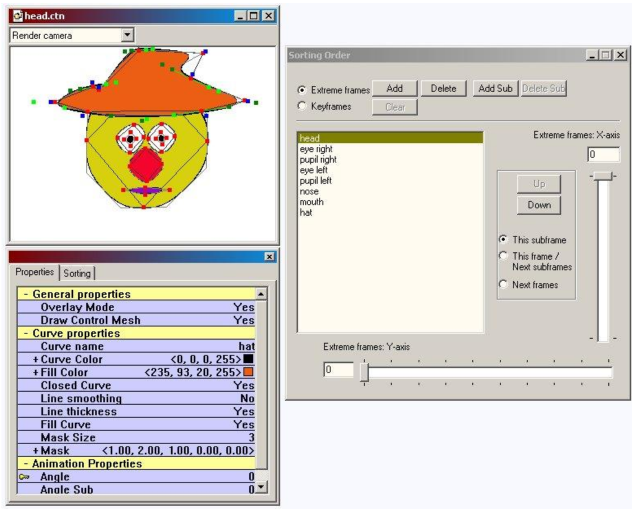
- level 0 holds the basic building primitives, being sets of attributed 2D curves (cf. section 3.3),
- level 1 manages and processes explicit 2.5D modelling information,
- level 2 incorporates 3D information by means of 3D skeletons, and
- level 3 is more open-ended and introduces higher level tools such as non-affine deformations, facial expressions, etc.

As will be clear 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).

3.2.2. 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 rotation of (sub-objects) around the x- and y-axis needs special attention. Referring back to the character depicted in figure 3(c), one can easily see that rotation 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, however, one can clearly notice (figure 3(d)) heavy changes in outlines and silhouettes as well as changes in drawing order (and when rotating further than indicated in figure 3(d), even more drawing order changes would be needed, as all the facial parts would disappear behind the head itself).

Hence, we propose a 2.5D modelling structure in which characters and objects are modelled as sets of depth ordered primitives (open or closed curves, cf. section 3.3) 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 set of ordered curve primitives, functionally comparable to the extreme frames in traditional animation (cf. section 3.1), and which we therefore also call an 'extreme frame'. Internally, for each 2.5D animation object/character, the supporting data structure is a straightforward list, indexed by two rotation angles. For each extreme frame, it holds all the references to the underlying curve primitives (each of which can have a meaningful name) as well as the drawing order of these primitives in this extreme frame.

Figure 4 shows the UI components supporting level 1 functionality. The rendering window shows a graphical rendering of the depth-ordered curves (including all the control points of the underlying curves - which can be switched of by hitting a toggle in the 'Properties' window). 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.



- General properties	
Overlay Mode	Yes
Draw Control Mesh	Yes
- Curve properties	
Curve name	hat
+ Curve Color	<0, 0, 0, 255> ■
+ Fill Color	<235, 93, 20, 255> 🟧
Closed Curve	Yes
Line smoothing	No
Line thickness	Yes
Fill Curve	Yes
Mask Size	3
+ Mask	<1.00, 2.00, 1.00, 0.00, 0.00>
- Animation Properties	
Angle	0
Angle Sub	0

Figure 4. A snapshot of some UI components supporting level 1 functionality.

Figure 5 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.

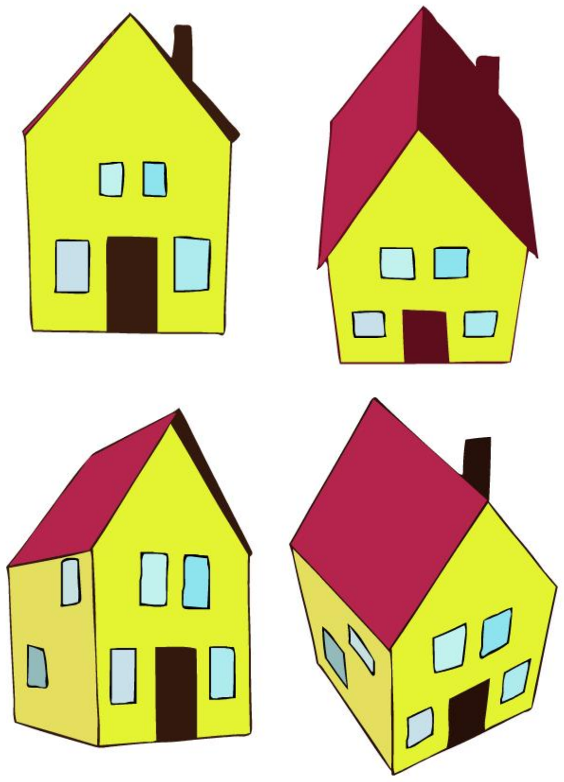


Figure 5. These are the four extreme frames created by the animator. In each case the same house is drawn but as if the virtual camera is rotated some degrees around the vertical (Y) or horizontal (X) axis. (a) Y=0, X=0; (b) Y=0, X=45; (c) Y=30, X=0; (d) Y=30, X=45.

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.

3.2.3. 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, 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.

Rendering such a key frame - and indeed also an in-between frame - narrows down technically to utilizing 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 [11].

The drawing order specified/stored in the selected extreme frames is utilized 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 been (and still is) studied in the morphing community [26]. The correspondence problem is very important in object-space morphing and some useful approaches may be found there. [20] interpolate the length of the edges of the polygon and the angles between them, while [1] use Delauney triangulated polygons, and instead of the outline, transform the triangles of the resulting mesh. Some researchers [4] propose an algorithm to automatically establish the correspondence. Others [10, 21] 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 6 shows some key frames of an animation sequence automatically generated using the four extreme frames in figure 5 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 center of the house) from sideways in front, upwards to up-front and then down again to end in front of the house.

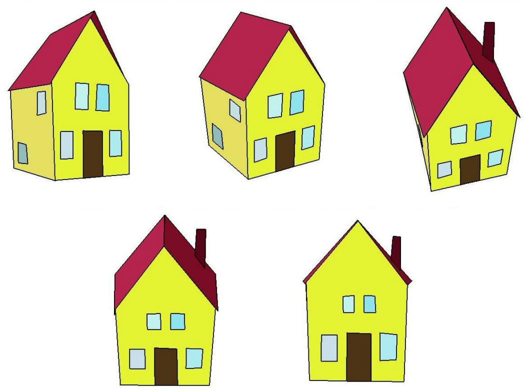


Figure 6. Key frames of an animation sequence using the extreme frames in figure 5.

3.2.4. Level 2: 3D information by means of 3D skeletons.

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 3.2.1 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 two slider bars for setting the 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 7) 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 7 shows a typical skeleton supporting composition and manipulation of a 2.5D animation character.

Figure 7. (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 [7, 25] or other physics based approaches.

3.2.5. 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 8).

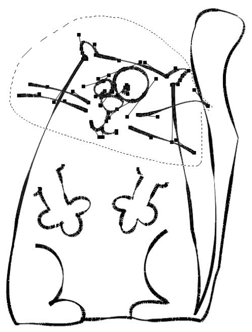


Figure 8. Group of control points.

The control points within the selection will be marked as a group. The control points in figure 8 are shown to visualize 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 9).

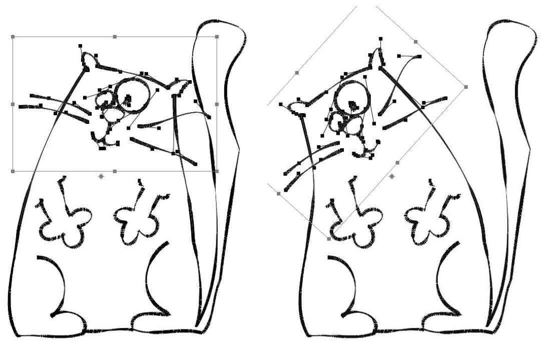


Figure 9. High-level deformation tool.

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).

3.3. Curve/outline representation

For research purposes outside the context of this paper, we decided to utilize subdivision curves to represent sub-objects at level 0 in our solution. We refer the reader to [28] 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 [23]. This allows us to use only a limited number of control points to fully control a subject with an irregular outline (cf. figure 10). Other curve types such as Bézier curves or NURBS [14] could have been used just as well.

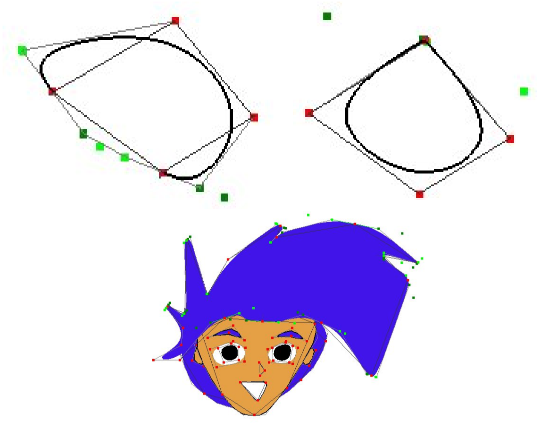


Figure 10. (a) shows a open subdivision curve that has two interpolating control points while (b) shows the ability of tension control. These kinds of curves are used to model the character of figure (c).

Each curve can be closed or opened and can be coloured (outlines as well as internally). Also, the curves can have a varying line thickness.

4. Results

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

Figure 11 shows some snapshots of animation exploiting the data presented in figures 5 and 6, acting as a representative of 2.5D animation of a typical background/décor element for which level 1 functionality suffices.

Figure 13 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 12 for generating the sequence.

Figure 14 consists of three images depicting some extreme frames of an animated hand, modelled as a skeleton based sub-object in level 2. Snapshots of an animation sequence exploiting these extreme frames are shown in figure 15.

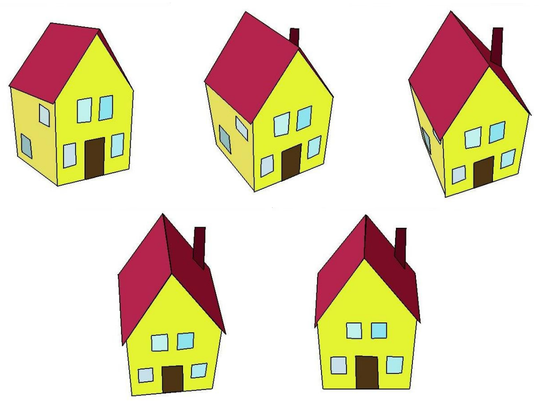


Figure 11. Snapshots of an animation of a typical background/décor element.

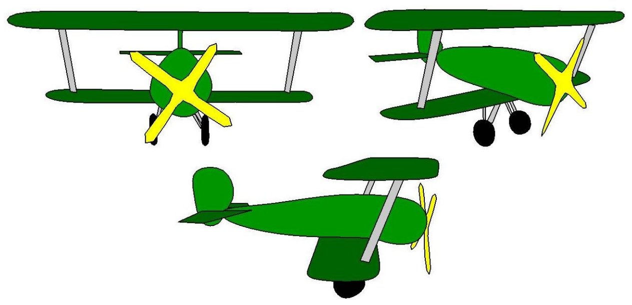


Figure 12. These pictures show the extreme frames of a 2.5D airplane.

The current implementation is written in C++ (MFC) with the use of OpenGL [8]. It offers real-time displaying and editing of the results, maximizing the comfort of the animator who wishes to adapt the animation to his artistic needs.

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.

5. Conclusions and future work

In this paper, 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 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

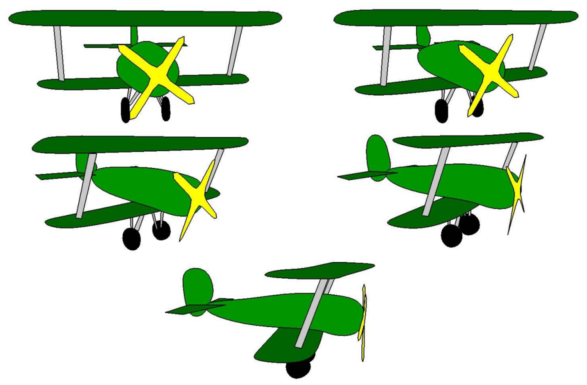


Figure 13. Snapshots of an animation of a typical animation object undergoing only affine transformations.

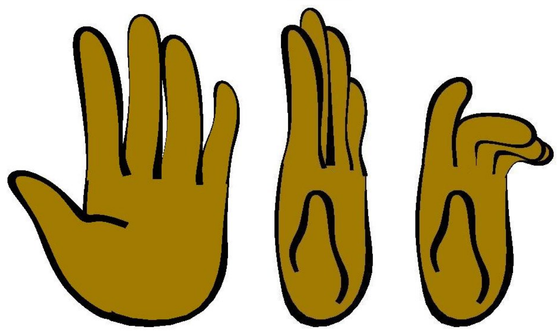


Figure 14. Extreme frames in a skeleton-based hand animation: (a) shows the hand at the start of the rotation while (b) depicts the end of it. (c) shows the same hand as after the rotation but of which all fingers are bend down somewhat.

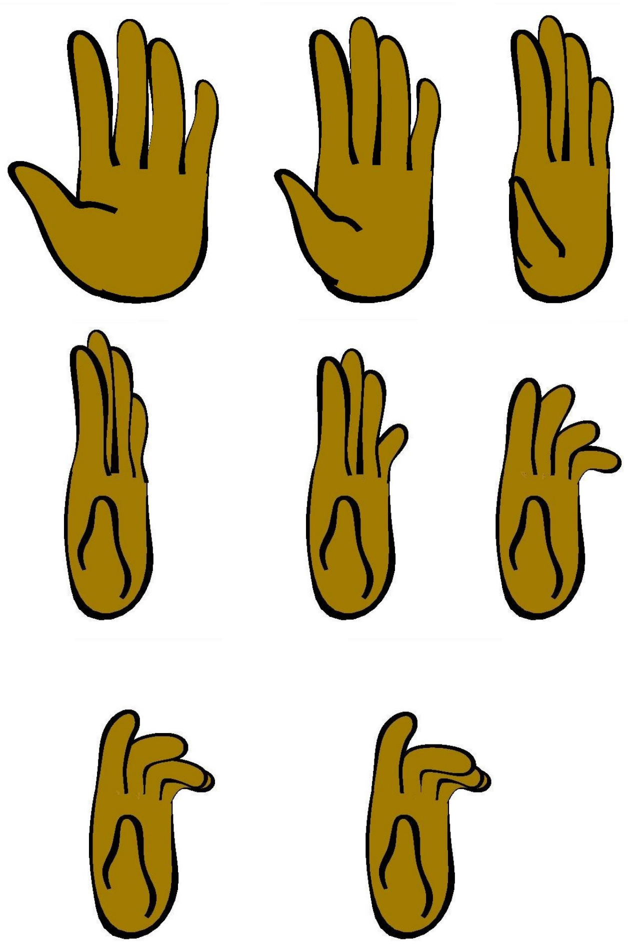


Figure 15. These pictures show some snapshots of an animation sequence generated from the extreme frames in figure 14.

underlying concepts and principles have been detailed and illustrative 2.5D animation examples have been given.

In future research, we want to introduce new techniques and tools for drawing and manipulating stylised curves. It is 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. Hence, we plan to integrate the approach proposed by our colleagues [24], which is based on real-time processing of (possibly pressure sensitive) stylus data for generating the underlying curves on the fly. Their solution implies that graphics artists no longer need to manipulate (i.e. point, click and drag) control points of underlying splines, but rather exploit direct manipulation tools on the curves themselves. This simplifies the interaction drastically. (As a downside on such functionality, more advanced methods for solving the correspondence problem (cf. section 3.2.3) will then have to be implemented as well.)

An interesting level 2 extension would be to incorporate approximate (=heavily simplified) 3D models aside the 3D skeletons to support 2.5D model construction at level 0 and 1. This should aid animators in keeping the 'volume' of their 2D characters when animating them.

We are also interested in doing further research on level 3 of our proposed solution; especially the inclusion of high-level tools for modelling and animation of facial expressions deserves additional attention. One possible way is to integrate into our system the techniques alike the ones reported upon in [16] and [19].

Acknowledgements

We gratefully express our gratitude to the European Fund for Regional Development and the Flemish Government, which are kindly funding part of the research reported in this paper.

Part of the work is funded by the European research project IST-1999-56412 ’3DINCTRAP’. In this context, we would like to express our thanks to the various people in the project providing ideas and for helping with the implementation.

Our appreciations also go out to Joan Cabot and ’Xemi’ Morales for their artistic contribution and to Koen Beets for implementing earlier versions of the lower level curve techniques.

Furthermore we would like to acknowledge ANDROME NV for freely putting available to us their CreaToon plugin SDK.

References

[1] M. Alexa, D. Cohen-Or, and D. Levin. As-rigid-as-possible shape interpolation. *ACM Computer Graphics (Proceedings of SIGGRAPH 2000)*, pages 157–164, 2000.

[2] P. Blair. *Cartoon Animation*. Walter Foster Publishing Inc., ISBN: 1-56010-084-2, 1994.

[3] N. Burtnyk and M. Wein. Computer-generated key-frame animation. *Journal of the Society Motion Picture and Television Engineers*, 8(3):149–153, 1971.

[4] E. Carmel and D. Cohen-Or. Warp-guided object space morphing. *The Visual Computer*, 13:465–478, 1997.

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

[6] K. Coninx, F. Van Reeth, and E. Flerackers. Interactive specification of $2\frac{1}{2}$D animation by exploiting a real-time 3D rendering architecture. *Proceedings of Eurographics UK, London*, pages 11–18, March 1996.

[7] F. Di Fiore. Physics based modelling of 3D movement in immersive virtual reality environments. Master’s thesis, Catholic University Leuven, 1997.

[8] R. Fosner. *OpenGL Programming for Windows 95 and Windows NT*. Addison-Wesley Developers Press, ISBN: 0-20140-709-4, 1996.

[9] H. Griffin. *The Animator’s Guide to 2D Computer Animation*. Focal Press, ISBN: 0-240-51579-X, 2001.

[10] H. Johan, Y. Koiso, and T. Nishita. Morphing using curves and shape interpolation techniques. *Proceedings of Pacific Graphics 2000, 8th Pacific Conference on Computer Graphics and Applications*, pages 348–358, 2000.

[11] D. Kochanek and R. Bartels. Interpolating splines with local tension, continuity and bias control. *Computer Graphics*, 18(3):33–41, July 1984.

[12] P. C. Litwinowicz. INKWELL: A 2.5D animation system. *Computer Graphics*, 25(4):113–122, 1991.

[13] T. Nettleship and P. Willis. The animachine renderer. *Computer Graphics*, pages 90–97, 1995.

[14] R. J. Oddy and P. J. Willis. Rendering NURB regions for 2D animation. *Computer Graphics Forum, special issue EG*, 11(3):C35–C44, 1992.

[15] J. W. Patterson and P. J. Willis. Computer assisted animation: 2D or not 2D? *The Computer Journal*, 37(10):829–839, 1994.

[16] J. W. Paul ten Hagen. A facial repertoire for animation. *Eurographics*, pages 79–84, 2000.

[17] W. Reeves. Inbetweening for computer animation utilizing moving point constraints. *Computer Graphics*, 15(3):263–269, 1981.

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

[19] Z. Ruttkay and H. Noot. Animated chartoon faces. *NPAR 2000: Symposium on Non-Photorealistic Animation and Rendering*, pages 91–100, June 2000.

[20] T. W. Sederberg and E. Greenwood. A physically based approach to 2D shape blending. *Computer Graphics*, 26:25–34, 1993.

[21] M. Shapira and A. Rappoport. Shape blending using a star-skeleton representation. *IEEE Computer Graphics and Applications*, 15:44–51, 1995.

[22] F. Van Reeth. Integrating $2\frac{1}{2}$D computer animation techniques for supporting traditional animation. *Proceedings Computer Animation*, pages 118–125, 1996.

[23] F. Van Reeth and J. Claes. Interpolatory uniform subdivision curves with normal interpolation and tension control, generating B-splines of any degree. Submitted to “The Visual Computer”, 2001.

[24] G. Vansichem, E. Wauters, and F. Van Reeth. Real-time modelled drawing and manipulation of stylised characters in a cartoon animation context. *Proceedings of the IASTED International Conference, Computer Graphics and Imaging (CGIM 2001)*, pages 44–49, August 2001.

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

[26] G. Wolberg. Image morphing survey. *The Visual Computer*, 14(8/9):360–372, 1998.

[27] J. Yu. Inbetweening for computer animation using polar coordinates linear interpolation. *Research Report 90/R23, Department of Computer Science, University of Glasgow*, September 1990.

[28] D. Zorin, P. Schröder, A. Levin, L. Kobbelt, W. Sweldens, and T. DeRose. Subdivison for modelling and animation. *SIGGRAPH 2000 Course notes 23*, July 2000.