Employing Approximate 3D Models to Enrich Traditional Computer Assisted Animation

Fabian Di Fiore Frank Van Reeth

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

Abstract

Although computer assistance for traditional animation is gaining a lot of attention during recent years, it still has to cope with many limitations. Part of the current research focuses on employing full 3D input models, which are rendered and even animated in many different non–photorealistic (NPR) styles. Disadvantages are the need to create complicated 3D models and the many difficulties to achieve lively movements. Purely 2D approaches, on the other hand, need many elaborated single drawings. Getting perspective right and retaining volumes are a major problem in that approach to computer assisted traditional animation, due to the complete lack of 3D information. Unfortunately, the employed software is ignorant about the approximate 3D representation in the animator's mind. In this paper, we present a novel tool for traditional animation, based on an approximate 3D model. This tool helps retaining volumes and proportions, and ensures frame–to–frame coherence.

Keywords: animation, 2.5D animation, computer animation, traditional animation, computer assisted animation, rendering, non–photorealistic rendering.

1. Introduction

A very important factor in the art of bringing hand–drawn characters to life happens in the animator's mind. The animator relies on his ready knowledge of the 3D object he is about to draw. Without that — often partial or approximate — knowledge, 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 1(a–b)). More difficulties arise when the animator chooses to replace the simpler cartoon style of figure 1(a) to the more painterly style of figure 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.

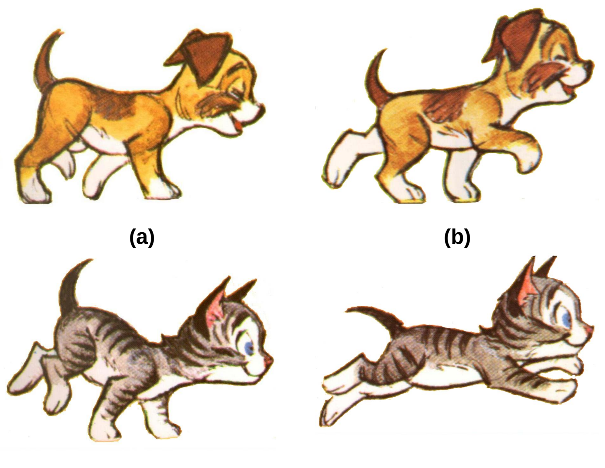


Figure 1. Copyright ©1994 Preston Blair. a–b) Two different key poses of a dog. In both versions the volume is the same. c–d) Two different poses of a cat. The stripes of the cat in figure (d) have to correspond with the stripes in figure (c) in order to avoid a "noisy" animation.

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) providing frame–to–frame coherence.

This paper is organized as follows. Section 2 gives an overview of related work and indicates the differences with our new approach. Section 3 elucidates modelling and animation in 2.5D. The central theme of our paper, creation and use of approximate 3D, is elaborated in Section 4, while Section 5 illustrates the obtained results. We end with our conclusions and topics for our ongoing future research (Section 6).

2. Related Work

Our approach is based on 2.5D modelling and animation techniques: at various stages of the animation process, we enrich the traditional computer assisted animation with approximate 3D representations. This section elaborates on computer assisted traditional animation, techniques to create simple 3D objects and some approaches found in the non-photorealistic domain.

2.1. Computer Assisted Traditional Animation

In the late seventies, Ed Catmull [3] was among the first to discuss important issues underlying computer–assisted animation. He indicated that the lack of explicit 3D information constitutes a major problem in 2D hand–drawn cartoon pictures. In particular, a 2D picture does not contain the 3D information present in the animator's mind, but still everybody expects the 2D representations to behave in similar ways as our 3D mental models do.

Recently, we introduced an automatic in–betweening method based on novel 2.5D modelling and animation techniques [5]. Multi–level 2D strokes, interpolation techniques and on–the–fly resorting are used to create convincing 3D–alike animations starting from pure 2D information. Unlike purely 3D based approaches, our animation still has many lively aspects akin to 2D animation. A rigid 3D look is avoided through varying line thickness and the ability to have subtle outline changes that are either impossible or hard to achieve utilizing 3D models. For example, an animator could draw the ears of a rabbit anatomically incorrect just in order to focus the attention on them.

Our current paper builds further on this 2.5 approach and consequently we will make use of the functionality provided in that system, such as explicit 2.5D modelling and powerful automatic in–betweening.

Over the last few years various commercial software packages have been developed for assisting the animator in the production of traditional cartoon animation. Griffin [8] provides a good overview on the available systems, including practical case studies and step–by–step descriptions of how to create many astonishing animations.

2.2. Techniques to Rapidly Create Simple 3D Objects

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

Igarashi et al. [9] present a gesture–based sketching interface (TEDDY) to quickly and easily design freeform models. In their system, the user first interactively draws a 2D silhouette consisting of several 2D freeform strokes. A 3D polygonal object is automatically constructed by inflating the region surrounding the silhouette, making wide areas fat and narrow areas thin.

Bimber [1] extends this sketching interface with support for dynamic gesture recognition. 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 freeform strokes to draw the outlines and working with simplified 3D models (cf. Section 4.1).

2.3. Painterly Rendering and Similar NPR Techniques

Barbara Meier [12] 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. 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. Barbara 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, but 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. Semi–transparency is hard to avoid.

Kowalski et al. [10] presented a method to render a 3D scene in a stylized manner. 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 (frame–to–frame) 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 [11] extended the work of Kowalski and presented a new framework that addressed some of the issues. The framework also introduced the concept of *tufts* which manage the multiresolution behavior of *graftals* according to specifications of the scene designer. This new framework addresses many of the problems encountered with the previous one but it is considerably slower for complex scenes and animators still have to create textual files manually to define looks and behaviors for *graftals*.

A major issue in these systems based so much on 3D, is that they generate a very 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 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 favorite 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 [14], providing an extensive overview of published work, covering these and many other techniques employed in the fascinating world of non-photorealistic rendering.

3. Modelling and Animation in 2.5D

In traditional 2D animation [2, 13] the modelling and drawing processes 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 the remaining *in–between frames* of the animation.

In prior work [5] we defined a 2.5D method for automatic in–betweening, which clearly distinguishes a modelling phase and an animation phase. This is implemented as a multi–layered system starting with basic 2D drawing primitives (curves) at level 0, over explicit 2.5D modelling structures at level 1. Level 2 includes hierarchical information by means of skeletons and level 3 offers the opportunity to include high–level tools (for example a deformation tool or a sketching tool).

In the next section we elaborate on how to aid the animator in the creation of the *extreme frames*, which lay the foundation of the animation, thereby making use of the existing functionality of the system.

4. 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.

4.1. 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 [2, 13], when drawing the objects in different poses and actions (i.e. drawing the extreme frames), animators first make reference drawings, 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. In fact, the sketched objects are only constructed from circular and rounded "3D–ish" base forms (see figure 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.

First, the user sketches (and possibly modifies) 2D circular and rounded forms as if drawing on paper. We use "free form stroke techniques" [6, 15] to create and alter these 2D objects.

Our system then interprets these circular and rounded forms to automatically construct a 3D polygonal object of revolution. This approach enables the easy and rapid con-

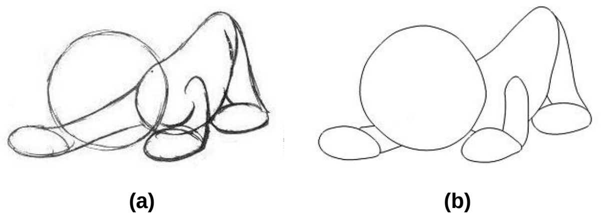


Figure 2. a) Copyright ©1994 Preston Blair. b) The same extreme frame as shown in (a) by using approximate 3D objects.

struction 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 2(b), we only show the silhouettes [4] 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 added 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), 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 he creates a new extreme frame.

To summarize, in this section we explained the creation of approximate 3D models starting from 2D freeform strokes and showed how they effectively can be used to create different extreme poses of the basic shape of an object while retaining the proportions of the object.

4.2. 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 characterized 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 he is assured that the shapes of the characters are preserved.

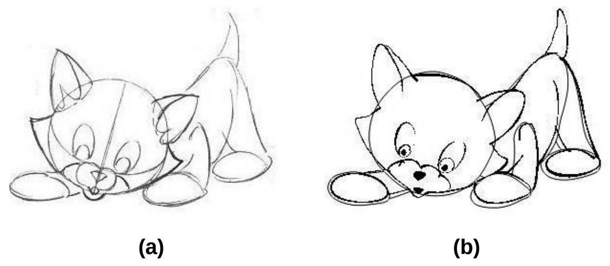


Figure 3. a) Copyright ©1994 Preston Blair. b) The same extreme frame as shown in (a) by using our system.

Figure 3(a) shows the version an animator is likely to draw the outlines compared to our results (figure 3(b)).

4.3. Rendering the Animation

In the previous sections we showed how approximate 3D models provide a very effective means to assist the animator throughout the modelling stage (i.e. modelling extreme frames) when creating an animation.

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

It is not our intention to introduce a new render style but to indicate that these 3D models play a significant role in the painting stage of an animation.

When looking at the traditional animation process, animators have to paint by hand every single frame of the animation [8, 13]. 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) coloring 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 the brush strokes randomly change with each frame. This lack of frame–to–frame coherence is due to the difficulty to estimate the position of the same brush stroke in different extreme frames.

For research purposes we implemented a painterly rendering style, although other rendering styles are possible as well.

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 the these are immediately reflected in the other frames.

The painting process itself is as described in listing 1.

While modelling:

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

During animation (at run-time):

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

Listing 1. The painting process of our system.

In our application, brush strokes are polygons textured with RGBA 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.

Regarding the painting process, first of all, the animator has to select one of the underlying curve primitives to which brush strokes are attributed. That way the underlying drawing order of the curve primitives specified in the extreme frames is utilized to determine the drawing order of brush strokes. As a result, this solves the problem of self–occlusions (this is not a topic of this paper and is described in detail in prior work [5]).

Painting itself occurs by gesturing strokes [5] upon the currently selected curve primitive. At this moment, the animator does not see the underlying 3D models but only the silhouettes he has drawn. 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 underlying 3D surface at that moment. This is done at discrete moments in time and delivers us a set of 3D points (which lie on the surface of the underlying 3D object).

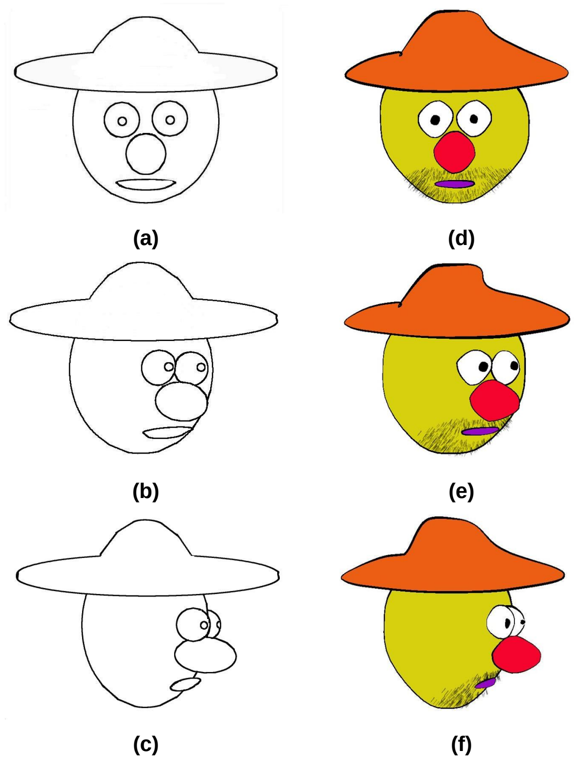


Figure 4. a–c) Three extreme positions of a person's head constructed with circular and rounded forms. d–f) The same three extreme frames, after inking the outlines, filling the primitives and painting some brush strokes.

Then, with each of the points we associate a particle which stores the current 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: that way, our system can handle the brush strokes in the same way as the underlying drawing primitives (curves) and so we can exploit the provided powerful automatic in–betweening method (The generation of in–between frames is a completely different problem, which is outside the scope of this paper. The reader interested in how we implemented the in–betweening stage is referred to [5]). 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 extreme 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, of course 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 utilizing 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 utilize the drawing order of the underlying curve primitives to determine the drawing order of the brush strokes.

Figures 4(d–f) show three painted extreme frames of a person's head. These are based on the 3D representations in figures 4(a–c). We only applied some brush strokes to figure 4(d), depicting the hairs of an untidy beard and moustache. As you can see in figures 4(e–f), the same brush strokes are displayed automatically at where we would expect them to appear: the moustache is still displayed under the nose and above the mouth. Notice that the character's hat in figures 4(d–f) is not geometrically correct with respect to 3D 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. Results

In this section we show some results of our presented method.

Figure 5 shows some snapshots of an animation exploiting the extreme frames presented in figure 4.

In figure 7 we show some animation snapshots of an airplane. We used the extreme frames shown in figure 6.

The extreme frames (without the outlines drawn) of an exotic bird are shown in figure 8. Note that we did not use curves to draw the wing of the bird. Instead it is completely painted upon the body. Figure 9 shows some snapshots of an animation of the bird.

Figure 10 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 11.

The current implementation is written in C++ (MFC) with the use of OpenGL [7]. 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.

6. Conclusions and Future Work

In this paper, 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 first, 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 a very experienced animator easily succeeds in retaining the volume of objects and preventing temporal aliasing.

We showed 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.

In the future we want to explore more ways and techniques to persuade animators to make the shift from traditional hand–drawn to computer assisted animation.

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 IST 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.

Many thanks to Johan Claes for the valuable paper reviews and to Joan Cabot and "Xemi" Morales for their artistic contributions.

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

References

[1] O. Bimber. Rudiments for a 3D freehand sketch based human–computer interface for immersive virtual environ-

ments. In *Proceedings of VRST 1999*, pages 182–183. ACM, December 20–22 1999.

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

[3] E. Catmull. The problems of computer–assisted animation. In *Proceedings of SIGGRAPH 1978*, volume 12, pages 348–353, August 1978.

[4] J. Claes, F. Di Fiore, G. Vansichem, and F. Van Reeth. Fast 3D cartoon rendering with improved quality by exploiting graphics hardware. In *Proceedings of Image and Vision Computing New Zealand (IVCNZ) 2001*, pages 13–18. IVCNZ, November 2001.

[5] F. Di Fiore, P. Schaeken, K. Elens, and F. Van Reeth. Automatic in–betweening in computer assisted animation by exploiting 2.5D modelling techniques. In *Proceedings of Computer Animation 2001*, pages 192–200, November 2001.

[6] F. Di Fiore and F. Van Reeth. A multi–level sketching tool for pencil–and–paper animation. In *Sketch Understanding: Papers from the 2002 American Association for artificial Intelligence (AAAI 2002) Spring Symposium. Technical Report SS–02–08*, pages 32–36, March 2002.

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

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

[9] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A sketching interface for 3D freeform design. In *Proceedings of SIGGRAPH 1999*, pages 409–416. ACM, August 1999.

[10] M. A. Kowalski, L. Markosian, J. D. Northrup, L. Bourdev, R. Barzel, L. S. Holden, and J. F. Hughes. Art–based rendering of fur, grass, and trees. In *Proceedings of SIGGRAPH 1999*, pages 433–438, August 1999.

[11] L. Markosian, B. J. Meier, M. A. Kowalski, L. S. Holden, J. D. Northrup, and J. F. Hughes. Art–based rendering with continuous levels of detail. *Symposium on Non–Photorealistic Animation and Rendering (NPAR2000)*, pages 59–66, June 2000.

[12] B. J. Meier. Painterly rendering for animation. In *Proceedings of SIGGRAPH 1996*, volume 25, pages 477–484, 1996.

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

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

[15] G. Vansichem, E. Wauters, and F. 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.

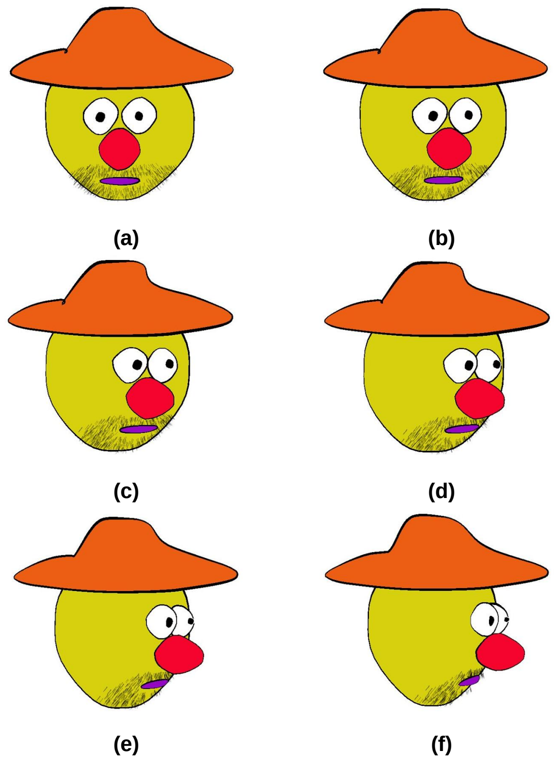


Figure 5. a–f) Some snapshots of an animation sequence generated from the extreme frames in figure 4.

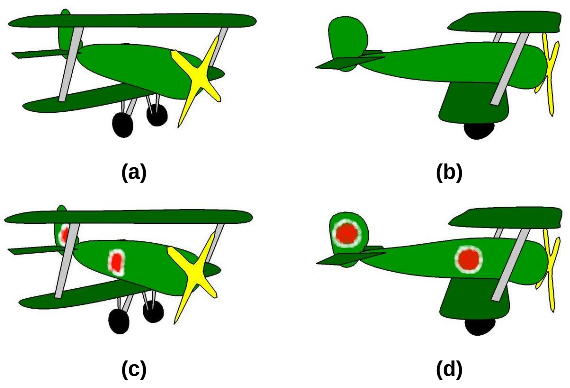


Figure 6. a–b) Some extreme frames, with inked outlines and filled primitives, of an airplane. c–d) The same extreme frames after having painted the object with brush strokes.

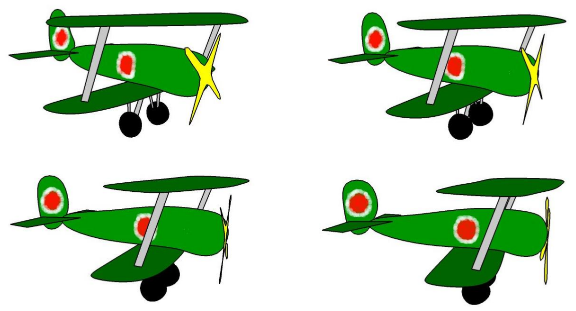


Figure 7. a–d) Snapshots of an animation of the plane shown in figure 6.

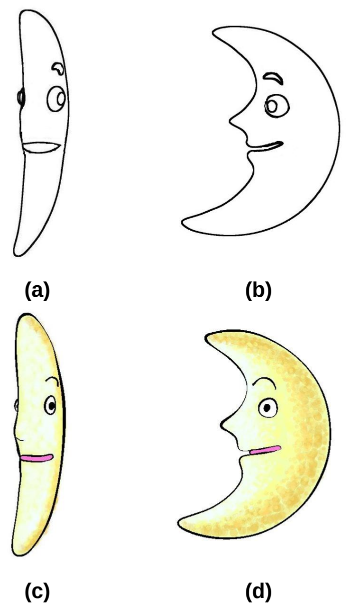


Figure 10. a–b) Some extreme frames, with inked outlines, of a moon. c–d) The same extreme frames after having painted the moon with colored brush strokes to achieve a pointillism–like painting style.

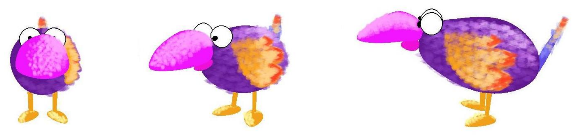
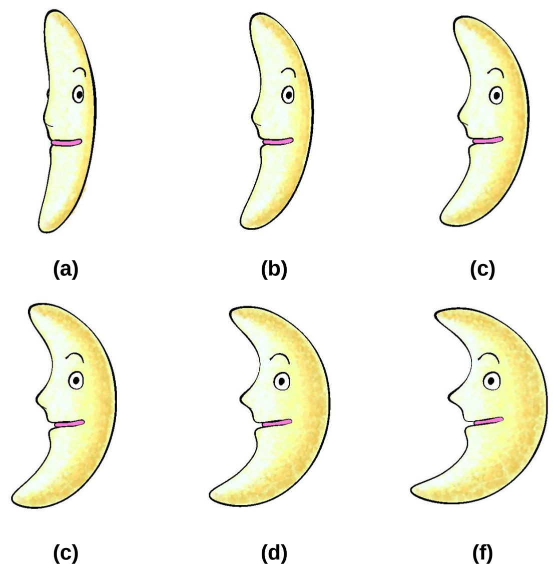


Figure 8. Painted extreme frames, without inked outlines, of an exotic bird.



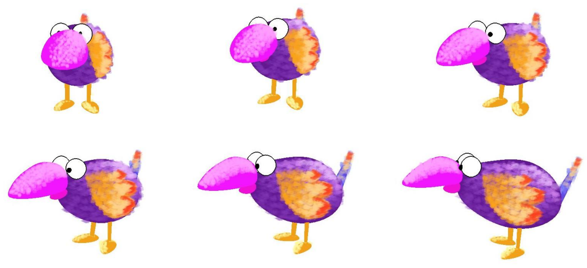


Figure 9. Some snapshots of an animation of the exotic bird depicted in figure 8.

Figure 11. a–f) These pictures show some snapshots of an animation sequence generated from the extreme frames in figure 10.