Rendering Artistic and Believable Trees for Cartoon Animation

Fabian Di Fiore   William Van Haevre   Frank Van Reeth

Limburgs Universitair Centrum
Expertise Center for Digital Media
Universitaire Campus
B-3590 Diepenbeek, Belgium
E-mail: {`fabian.difiore, william.vanhaevre, frank.vanreeth`}`@luc.ac.be`

Abstract

We present a novel approach to design artistic and believable trees in a cartoon-like style, which can be rendered by an animated camera to produce a convincing 3D-like experience. While computer assisted traditional animation is able to generate the desired effects, this approach still depends too much on the creation of many hand drawn images. Existing approaches fully depending on 3D geometries, on the other hand, give little artistic freedom and have difficulties to avoid artifacts popping up in successive frames.

In order to provide good solutions to these difficulties, we present a hybrid (2.5D) framework, combining the advantages of both 2D and 3D approaches. From an underlying 3D geometry we get the necessary information to obtain an acceptable level of 3D behavior and a good frame-to-frame coherence. In the same framework, 2D artistic input is employed to obtain any desired 'look', both of the rendering and of the animation.

Keywords: natural phenomena, 2.5D animation, computer animation, non–photorealistic rendering.

1. Introduction

During the last years, the use of CGI elements in the production of cartoon animations has become a very common thing. This paper deals with the category of CGI effects which classifies the CGI standards: things that are too numerous or tedious to draw. In our case, the animation of cartoon-like trees. Figure 1 shows some snapshots of the kind of animations we are striving after.

Consider for example figure 1. From an animator's point of view two technical issues arise. The first issue concerns the animator's ability to picture in mind the objects he is about to draw. For characters this is relatively easy, since he can rely on his ready knowledge. However this is not the case for trees, due to their complex, recursive structure (what does a tree look like viewed from aside or behind?). The second issue deals with the numerous branches and leaves that a tree consists of. For example, when walking around a tree, the branches and leaves may not suddenly appear and disappear. Without such coherence, the temporal aliasing makes the final animation hard to enjoy.

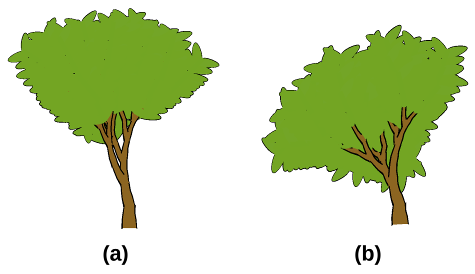

Figure 1. Some snapshots of an animated tree. a) Regular view. b) View from below.

Therefore, within the boundaries of our study, we aim at tackling these issues. Furthermore, it is also our goal to give the animator the same freedom of expressing the artistic style he is bearing in mind as if drawing on paper.

To establish these goals, we present a hybrid (2.5D) approach that benefits from existing 2D and 3D approaches. On the one hand, 3D geometrical models are exploited to extract 3D information from, which is necessary to provide for frame-to-frame coherence, while in order to preserve the animator's freedom of creativity 2.5D modeling and animation techniques are used.

This paper is organized as follows. Section 2 gives an overview of related work in the field and indicates the differences with our philosophy. Section 3 first elucidates modeling and animation in 2.5D. Then, the central theme of our paper, rendering and animating cartoon-like trees, is elaborated, while Section 4 provides clarifying results. We end with our conclusions and topics for future research (Section 5).

2. Related Work

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

2.1. Pure 2D Approaches

In this section we enlarge on research conducted in the field of pure 2D animation where both the modeling and animation stage integrally take place in 2D.

For Walt Disney's feature animation *'Mulan'* [7], CGI was used to animate Bamboo plants. Each plant is composed of several 2D layers and this for certain camera angles. The advantage of this approach lies in the fact that the animator has full control over the final style. However, a major drawback is that the artist still has to paint the plant manually for all camera angles. As a result, it is up to the animator to know how the plants look like from different views, and consequently to provide for frame-to-frame coherence.

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

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

2.2. Starting from 3D: Non-photorealistic Rendering Techniques

Recently popular, 'Toon Rendering' (a subcategory in the non-photorealistic rendering (NPR) domain [6]) is used to (automatically) generate stylized cartoon renderings starting from 3D geometrical models.

In 1996, Barbara Meier [11] presented a painterly rendering algorithm in which particles attached to the 3D model are used to eliminate the temporal aliasing effect that disturbs many other NPR approaches. This approach leads to very impressive results for rigid objects, but requires extensive modeling and animation if it were applied to fully animated trees.

Kowalski et al. [8] presented a method exploiting procedural stroke based textures, *graftals*, to render a 3D scene in a stylized manner. These *graftals* place geometric elements procedurally into the scene and so produce effects including fur, grass and trees. Because these *graftals* stick to the 3D surfaces, this approach is suitable for preventing temporal aliasing. 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 [10] extended the work of Kowalski and presented a new framework that addressed some of the issues. The framework also introduced the concept of *tufts* managing the multiresolution behavior of *graftals*. 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 script files manually to define looks and behaviors for *graftals*.

Recently, Deussen [3] presented a method of rendering pen-and-ink illustrations of trees automatically. He starts with detailed 3D tree models consisting of a tree skeleton and leaves. Then, a computer-generated pen-and-ink illustration is achieved by applying existing NPR-techniques to the tree skeleton and by drawing the leaves using abstract drawing primitives. This approach enables us to generate illustrations with different drawing styles and levels of abstraction. Yet, with respect to the creation of cartoon animations, two important issues arise. In the first, the animator's contribution to the rendering of the tree skeleton is limited since standard NPR-techniques have to be used. Secondly, the used 3D models already contain foliage data and that way the artist's creativity is constrained by the overall shape of the 3D model.

In a nutshell, a major issue in these systems based so much on 3D, is that they also *generate* a very 3D look which we especially want to avoid. Furthermore, many details are extremely hard to construct in 3D, while it is much simpler to design very convincing look-alikes in 2D.

3. Our Approach

As will be clear from the subsequent subsections, we opt for a 2.5D methodology which clearly distinguishes a modeling phase and an animation phase. This methodology [4] has been proven to be very useful for the purpose of creating convincing 3D-like animations starting from pure 2D information.

Considering traditional hand-drawn animation [1] from an artist's standpoint; preserving shapes, getting perspective right and ensuring frame-to-frame coherence are a major problem. Existing software to assist traditional animation either lacks the 3D representation needed to tackle this kind of shortcomings, or imposes constraints hampering the animation artists' creativity.

Consequently we identify two parts in the modeling phase. For the first part, which narrows down to rectifying the lack of 3D representation, we discuss incorporating 3D information by means of realistic underlying 3D models. For the second part, preserving the artist's style, we explain the animator's role of providing explicit modeling information.

Some technical issues of the animation phase (e.g. inbetweening process) and the modeling phase (e.g. basic drawing primitives) have already been handled by our previous work [4]. This is briefly discussed in Section 3.1. The two parts of the modeling phase are elaborated in Sections 3.2.1 and 3.2.2.

3.1. Modeling and Animation in 2.5D

The modeling and animation context of this paper is situated in our prior work [4, 5] in which we defined a 2.5D method for automatic in-betweening, which clearly distinguishes a modeling phase and an animation phase.

This is implemented as a multi-layered system starting with basic 2D drawing primitives (in our case sets of attributed 2D curves) at level 0. Level 1 manages and processes explicit 2.5D modeling information and is fundamental in the realization of transformations outside the drawing plane. Objects are modeled as sets of depth ordered primitives with respect to the X-axis and Y-axis rotations. For each set of 'important' x-y-rotations of the object relative to the virtual camera, the animator draws a set of ordered curve primitives, functionally comparable to the extreme frames in traditional animation [1]. Level 2 incorporates 3D information by means of 3D skeletons or approximate 3D objects [5] and level 3 offers the opportunity to include high-level tools (for example a deformation tool or a sketching tool).

Multi-level 2D strokes, interpolation techniques and on-the-fly resorting are used to create convincing 3D-like 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.

3.2. Modeling Cartoon Trees

In this section we elaborate on the two different parts of the modeling phase. First, we discuss how to incorporate 3D information by means of realistic 3D models. For the second part, we'll explain the animator's role of providing explicit modeling information.

3.2.1. Incorporating 3D Information. When creating animated characters, it makes sense to let an animator hand-draw the extreme views of the whole character from scratch. That is because an animator relies on his ready knowledge of the 3D character he is about to draw. The same idea counts when one is animating more complex characters (e.g. a walking dog) where the difficulty lies in retaining the proportions and overall volume of the character. Here the animator can be guided by means of rapidly created *approximate 3D models* (starting from 2D rounded forms) [5] which main task is assisting the animator.

Since trees are far more complex than everyday characters these *approximate 3D models* are unsuitable. This is due to the complex, recursive structure of trees which is extremely hard to model starting from only 2D rounded forms. Moreover, since trees are composed of numerous branches and leaves one also has to deal with temporal coherence between successive frames of an animation. Without such coherence, the temporal aliasing (branches and leaves suddenly pop up and disappear) makes the final animation hard to enjoy.

For those reasons, instead of using *approximate 3D models* the underlying models in our approach are realistic 3D geometries of trees. That way we can incorporate necessary 3D information in order to preserve the overall shape and to ensure frame-to-frame coherence. This is explained in Section 3.3.1.

As regards traditional animation [1], an artist always draws the outlines of the tree skeleton but never draws individual leaves. Instead, the leaves are grouped together per branch or per tree. As a result, we use simple tree models consisting only of a tree skeleton. As will be clear from Section 3.3.3 this information is also sufficient to draw the foliage.

For research purposes, the underlying models are realistic 3D geometries generated by L-systems [12, 13], however other approaches (for example [9]) generating the necessary 3D information can be used as well. Figures 2(a–c) show some input models that we use.

3.2.2. Explicit 2.5D Modeling Information. In this section we show how we preserve the animator's freedom to express the artistic style he is bearing in mind.

One of the purposes of the *approximate 3D models* in [5] is to ink the outlines of the desired object rapidly by tracing silhouette lines and marker lines of the approximate object. This works well when animating characters since these can usually be built from multiple *approximate 3D objects* [1]. That way, tracing all outlines of all approximate

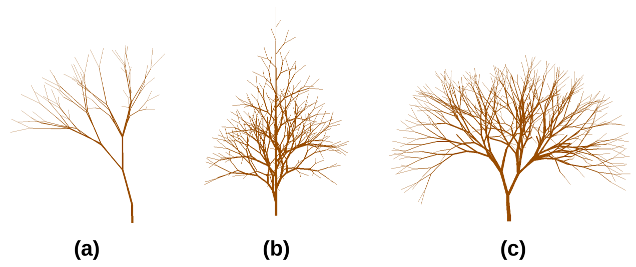

Figure 2. 3D rendered images of our input models.

objects soon results in a 2D layered model which perfectly fits in our 2.5D methodology.

Nevertheless, as explained in the previous section, we do only have one underlying 3D model. Consequently, letting the animator create a layered model is nearly impossible (which of all the branches comes first or last or in-between?).

Therefore, instead of modeling the whole tree at once, we have the basic parts of a tree (branches and foliage) drawn by the animator independent of the input model. In this way, the animator creates a kind of repository which does not depend on the underlying model and as a result is highly reusable.

To resume, in order to achieve convincing 3D-like animations, our system [4] requires the object to be modeled as seen from different viewpoints. These different viewpoints can be seen as the extreme frames of [1] and will be used by our in-betweening method in the animation phase. This is elucidated into detail in Section 3.3.

Figures 3 and 4 show some extreme frames of a branch and foliage as the animator bears them in mind. Notice that both the branch and foliage are geometrically incorrect when compared to real life. However, it is exemplified clearly that it is entirely up to the animator to choose the artistic style for every component of the final animation.

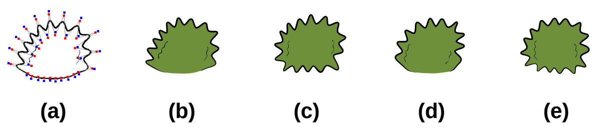

Figure 3. Some extreme frames of foliage.

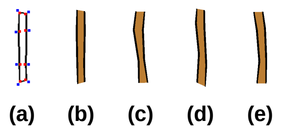

Figure 4. Some extreme frames of a branch.

To recapitulate, the separate modeling of branches and foliage, independent of the underlying 3D model, enables the animator to easily create a repository of tree parts that is reusable with any other tree model. In addition, the animator is entirely free in expressing the artistic style he is bearing in mind.

3.3. Animating Cartoon Trees

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

3.3.1. Overview of the Animation Process. In the previous section we introduced the incorporation of 3D information by means of geometric tree models, and the separate modeling of branches and foliage.

This section gives an overview of the rendering process for which the pseudocode is depicted in listing 1.

```
Preprocess( model m)
build hierarchy tree of branches
    for each encountered branch, $b^{3D}$, do
        assign random 2D branch, $b^{2D}$
        if (level $l$ >= FOLIAGE) then
            assign random 2D foliage, $f^{2D}$
        end // if
    end // for
end // build
```

```
Draw( frame f )
for each branch, $b^{3D}$, do
    calculate average depth
end // for
sort branches
for each sorted branch, $b^{3D}$, do
    draw $b^{2D}$ on top
    draw $f^{2D}$ on top, if present
end // for
```

Listing 1. The drawing process of a tree.

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

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

The artist can configure a level of detail, depending on the particular effect he wants, to achieve a specific animation. At the lowest level, only the outlines of the global foliage are shown while at the highest level outlines are shown for the most prominent parts of the tree. The different results are discussed in Section 4. The drawing of branches and foliage itself is explained in Sections 3.3.2 and 3.3.3.

We conclude this section by stating that the incorporation of 3D information is an effective means to ensure a correct drawing order and to provide for frame-to-frame coherence.

3.3.2. Drawing the Branches. Tree skeletons are hierarchical structures by nature and as a result we implemented a drawing procedure which is recursively called. Therefore, it suffices to explain the algorithm on the basis of drawing a single branch corresponding to one execution of the drawing procedure (listing 2).

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

Listing 2. The drawing process of a branch.

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

Given the coordinates of its position and orientation relative to its parent, we first calculate its orientation in world space. This information is then streamed to our inbetweening algorithm [4] which creates a 2D branch, b^{2D} (e.g. figure 5(a)).

At this stage, our system demands that the generated branch is aligned with the upstanding axis. The top and the bottom of the branch need to be flattened as well. That way, the remainder of the algorithm can be fulfilled in a more simple way and guarantees better preservation of the animator's style. In order to align a branch with the upstanding axis, the first step is carried out by the animator during the modeling stage. As can be seen in figure 4, branches are modeled vertically. However, animators can make mistakes and consequently the in-betweening process can introduce some deviations. So, as a final step, our system refines the alignment by rotating b^{2D} around the Z-axis so that the virtual line, which connects the upper and the lower midpoints of the branch, is parallel to the upstanding axis. To flatten the branch, a simple operation suffices which involves only translations of the control points along the curve they belong to. Figure 5 shows a generated in-between branch b^{2D} (a) and its aligned version (b). Note that aligning the branch does not hamper the animator's style since it involves only affine transformations.

When looking at real trees, one notices the different widths of the branches: older branches are thicker than younger ones. So in order to create a believable, attractive tree, we scale the width of each b^{2D} according to b^{3D}'s level

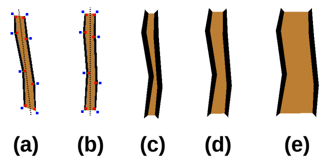

Figure 5. a) A generated in-between branch. b) After alignment. c–e) At different ages.

l in the hierarchy tree. This level corresponds to the age a_l of a real branch. We attribute age a_l to the start of b^{2D} and a_{l-1} to its end. Since each age a_l is associated with a fixed width w_l we are assured of creating a tree becoming narrower when approaching the top. For the scaling itself we can take advantage of the aligned branch since now we just have to rotate the two vertical curves of the branch around the Z-axis in order to fulfill following requirements:

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

Figures 5(c–e) show our branch at different ages. Note that the animator's style is preserved since only rotations around the Z-axis are performed.

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

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

In this section we explained the drawing of a branch. In order to get a visually appealing skeleton, each branch undergoes some *aesthetic* transformations which are affine and so preserve the animator's artistic style.

3.3.3. Drawing the Foliage. The drawing process of foliage (listing 3) is a simplified version of the previous algorithm.

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

Listing 3. The drawing process of foliage.

4. Results

Figure 6 shows some snapshots of an animation of a bare tree. In figures 7, 8 and 10 the same tree is shown, this time with foliage. In figures 7 and 10 we rendered only the outlines of the global foliage which results in a traditional '*cartoon-ish*' tree whereas drawing the outlines of the foliage per branch causes the tree to look completely different (figure 8).

We used the oak model (figure 2(c)) to render the images of figure 9.

Figure 11 consists of images depicting different stylized renderings of the tree in figure 2(b). In (a) a less detailed foliage object is used compared to (b). As a consequence, the first image resembles much more a pine whereas the second one portrays a spruce. The effect of the animator's artistic input is even more apparent when looking at (c). Here the animator drew a detailed foliage object while letting the system render the outlines of each foliage object.

The current performance (Pentium III 600 MHz, GeForce 256 DDR) of the rendering process is shown in figure 12. The frame rate is significantly lower when rendering the foliage. This is due to the use of subdivision curves to represent every foliage object. This subdivision process occurs per frame for each in-between object and results in concave polygons, consisting of many vertices. These polygons, in turn, then have to be tesselated to ensure correct coloring.

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

5. Conclusions and Future Work

In this paper, we presented a novel approach to create artistic and believable cartoon-tree animations. Existing computer animation software either employs full 3D models or uses purely 2D approaches. In the first, highly detailed 3D models are rendered in traditional non-photorealistic styles and leave no room for the animator to express his artistic feelings. On the other hand, purely 2D approaches require many repeated drawings which is very time-consuming. Furthermore, the numerous branches and leaves often cause the final animation to suffer from temporal aliasing.

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

In the future we want to explore the suitability of this technique to other natural sceneries. Furthermore, we want to extend the framework to also incorporate non-rigid animation, like movements influenced by the wind.

Acknowledgements

We gratefully express our gratitude to the European Fund for Regional Development, the Flemish Government and the European IST research project IST-2001-37116 'CUSTODIEV' which are kindly funding part of the research reported in this paper.

References

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

[2] J. M. Cohen, J. F. Hughes, and R. C. Zeleznik. Harold: A world made of drawings. *NPAR 2000: Symposium on Non-Photorealistic Animation and Rendering*, pages 83–90, June 2000.

[3] O. Deussen and T. Strothotte. Computer-generated pen-and-ink illustration of trees. In *Proceedings of SIGGRAPH 2000*, pages 13–18. ACM, 2000.

[4] 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.

[5] F. Di Fiore and F. Van Reeth. Employing approximate 3D models to enrich traditional computer assisted animation. In *Proceedings of Computer Animation 2002*, pages 183–190, June 2002.

[6] B. Gooch and A. A. Gooch. *Non-Photorealistic Rendering*. A. K. Peters Ltd., ISBN: 1568811330, 2001.

[7] E. Guaglione. The art of Disney's Mulan. *SIGGRAPH 1998 Course notes 39*, July 1998.

[8] 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. ACM, August 1999.

[9] B. Lintermann and O. Deussen. Interactive modeling of plants. *IEEE Computer Graphics and Applications*, pages 2–11, 1999.

[10] 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.

[11] B. J. Meier. Painterly rendering for animation. In *Proceedings of SIGGRAPH 1996*, volume 25(4), pages 477–484. ACM, 1996.

[12] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag New York Inc., ISBN: 0–387–97297–8, 1990.

[13] W. Van Haevre and P. Bekaert. A simple but effective algorithm to model the competition of virtual plants for light and space. In *Proceedings of the 11-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2003)*, pages 464–471, February 2003.

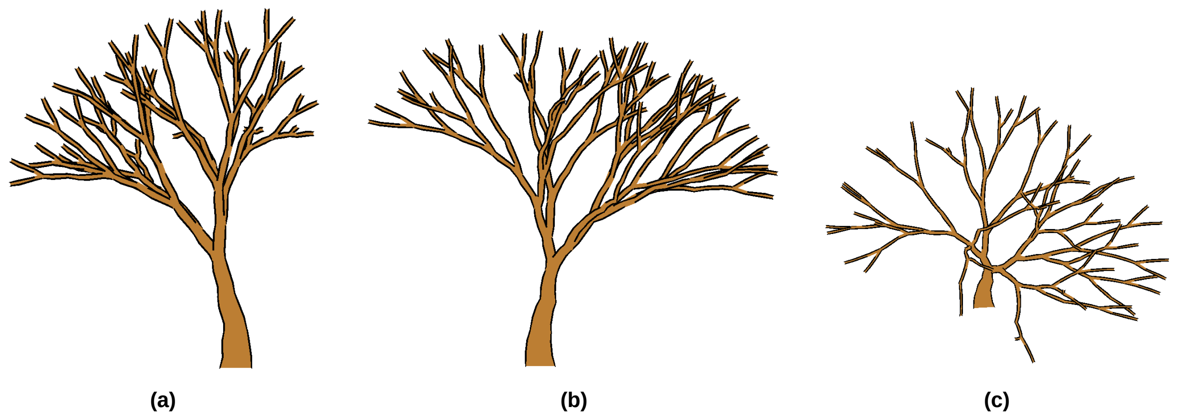

Figure 6. Snapshots of an animation of the bare tree depicted in figure 2(a). a) Regular view. b) Side view. c) View from above.

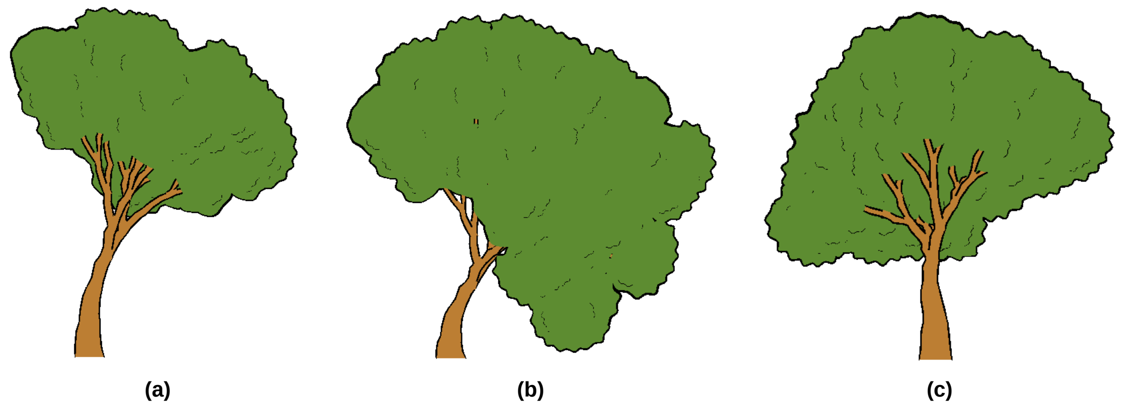

Figure 7. These images show the tree in figure 6 full of leaves. Only the silhouette of the global foliage is drawn. a) Regular view. b) View from above. c) View from below.

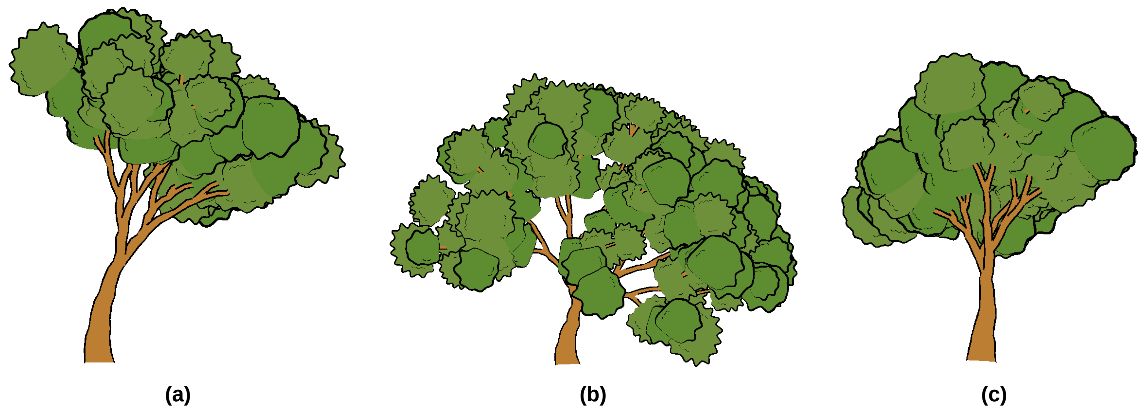

Figure 8. The same tree as in figure 7. The silhouette of each local foliage is drawn which results in a savannah tree. a) Regular view. b) View from above. c) View from below.

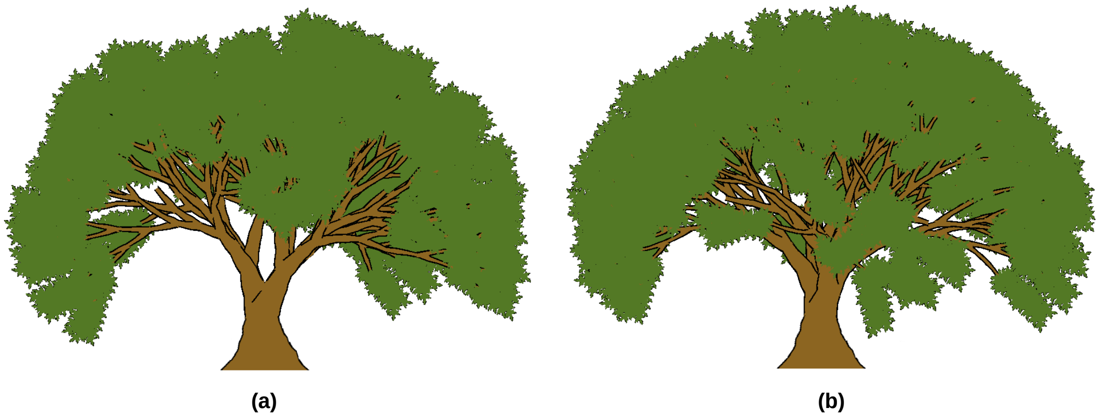

Figure 9. An old oak tree. a) Regular view. b) Side view.

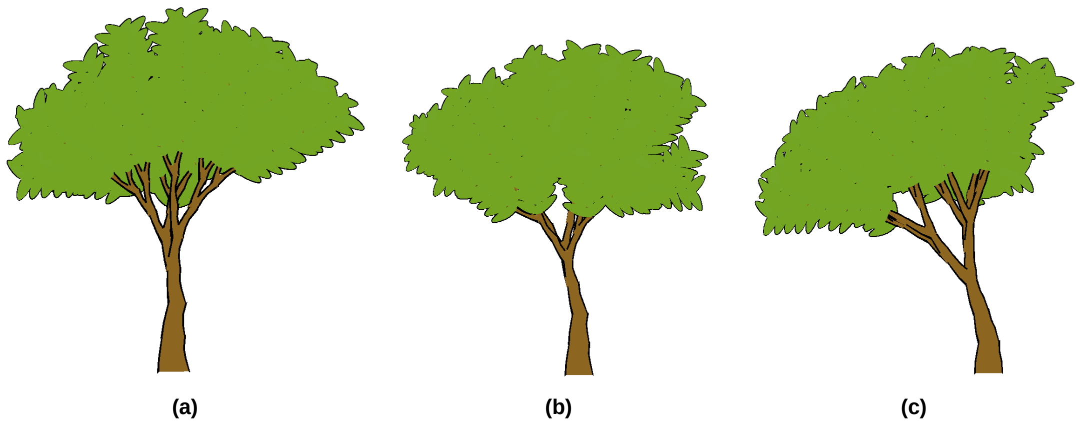

Figure 10. A deciduous tree covered with many little leaves. a) Regular view. b) Side view. c) View from behind.

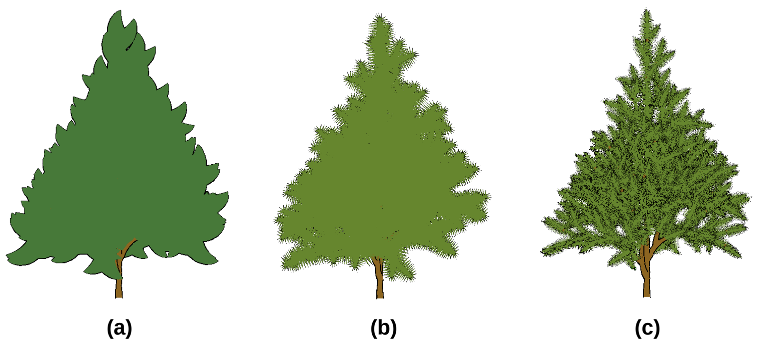

Figure 11. These pictures depict different stylized renderings of the tree in figure 2(b). a) A pine (less detailed foliage). b) A spruce (detailed foliage). c) Highly detailed spruce (foliage with pine-needles, all outlines drawn).

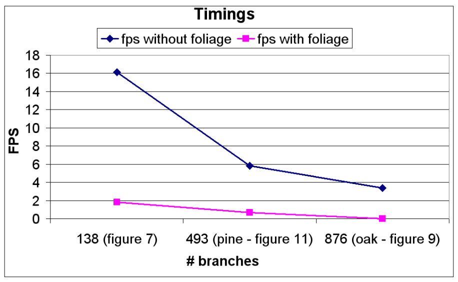

Figure 12. Performance overview.