Computer-Generated Pen-and-Ink Illustration of Trees

Oliver Deussen and Thomas Strothotte

Presentation by
Dmytro Prykhodko

Overview

- Introduction and Goal
- Related work
- Traditional approach
- Illustration of trees
- Results
- Conclusions

Introduction

- Method for automatically rendering pen-and-ink illustrations of trees
- Input is detailed tree models consisting of tree skeleton and leaves

Goal

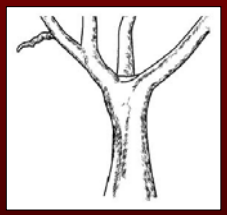
- To provide the user with a transition from a tree illustration with a realistic plant-specific look to an abstract representation consisting of only a few strokes

Related Work

- C. I. Yessios – uses abstract plant symbols in 2D drafting system, 1979
- A.R. Smith – "cartoon tree" with small disks representing bunches of leaves, 1984
- Kowalski et al. – Art-based rendering, 1999

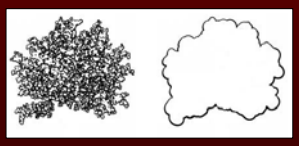
Traditional Approach

- The tree skeleton is usually drawn up to the second branching level, by silhouette lines and crosshatching on the stem surface



Traditional Approach

- Shape of the foliage is presented by
  - an abstract outline, or
  - a collection of many small objects



Traditional Approach

- Foliage divided into
  - Lit area with some details and its outline
  - Half shadow area with more details to compensate for the gray level
  - Shaded area

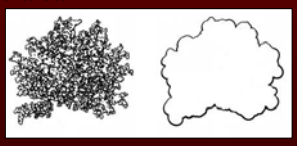


Illustration of Trees

- Create detailed tree model using conventional modeling program (xfrog)

Photo-realistically rendered images of the synthetic sample trees: Tree I: complex tree; Tree II: young lime tree; Tree III: conifer

Illustration of Trees

- Preprocess model into two files:
  1. Geometry of the tree skeleton
  2. Leaves as particles, each with position and normal vector

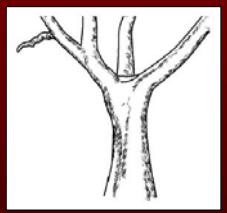
Generating Illustrations

- Trunk and branches are drawn using NPR
- Foliage is rendered by representing each leaf by a disk facing the viewer
- Optionally, cast shadows

http://www.synergizedsolutions.com/simpsons/pictures.shtml

Drawing the Tree Skeleton

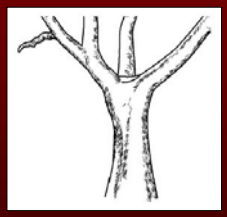
- Tree skeleton consists of cylinders each representing a branch
- Outline generated by analytical silhouette algorithms



Trunk and main branches of Tree I

Drawing the Tree Skeleton

- Crosshatches placed with "Difference Image Algorithm" or variant of Floyd Steinberg method



Trunk and main branches of Tree I

Drawing the Foliage

- Each leaf is represented by the outline of abstract drawing primitive
- Position is determined by the 3D leaf position and the size is controlled by the user

Drawing the Foliage

- Algorithm to draw the foliage
  - Draw the leaves as solids and get depth buffer
  - For each pixel, determine how far it is in front of its neighbors, and store it in a separate buffer
  - Use that data to draw the leaves

Drawing the Foliage

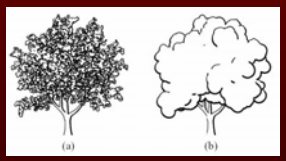
- Depth in camera coordinate system

$$z = \frac{\frac{z_1 z_0 (d_1 - d_0)}{z_1 - z_0}}{d - \frac{(z_1 + z_0)(d_1 - d_0)}{2(z_1 - z_0)} - \frac{(d_1 + d_0)}{2}}$$

- d_0 & d_1 = min and max values represented in the depth buffer
- z_0 & z_1 = corresponding depth values of near and far clipping planes in camera projection
- d = depth value [0..1]

Drawing the Foliage

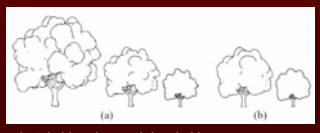
- Depth differences can be computed for camera coordinates or for depth buffer
- Depth difference threshold is determined by a percentage of the depth range of the tree



a) disk size=0.15, threshold=1000; b) disk size=0.7, threshold=2000

Level of Detail

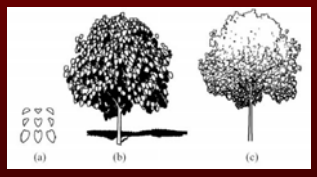
- More detail up front and less in the back because depth buffer has non-linear characteristic
- Changing ratio of z_1 to z_0 alters non-linearity



a) Primitive sizes and threshold are constant
b) Primitive sizes are enlarged with distance

Drawing Primitives

- A number of drawing primitives can be used to represent leaves



a) Leaves are rendered using interpolated polygons from nine given samples
b) Shadow drawn in black, threshold=100
c) Shadow represented by detail, threshold=6000

Software Framework

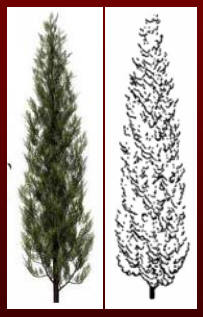
1. Determine depth differences
   - Interactive: stem & foliage rendered together
   - "Non-interactive": stem & foliage rendered separately
2. Software shadows created and stored
3. Draw pixels above the threshold

Software Framework

- For generating high quality images:
  - Bitmaps of the stem and foliage are vectorized
    1) least squares fitting
       (determine global vectors)
    2) index buffer
       (stores primitive IDs at each pixel as a color value)
  - Polygons drawn by spline interpolation

Results

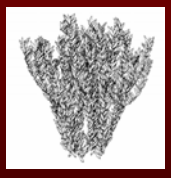
- This tree is drawn using view-facing elliptic primitives of random orientation
- Only the lower part of each ellipse is drawn
- Threshold = 100
- 13,200 particles



Results


16,200 particles
(originally 200,000 leaves)


2,300 particles


Real leaves

Results


90,000 particles + 23,000 for the ground

Conclusion

- Trunk and branches are represented by silhouette lines augmented by crosshatching in dark areas
- Foliage is drawn by using abstract primitives that represent leaves
- Depth differences are used to determine what part of the primitives are drawn

Future Work

- Instead of shadow buffer, maybe cross-hatch the leaves to suggest shadow
- Reduce amount of geometric data. A continuous level-of-detail algorithm for trees will improve performance
- Add new styles and colored versions for cartoons application