

An Expert System Approach To Non-Photo Realistic Rendering

DONABY H HENTON

dhenton@users.sourceforge.net
<http://inkulator.sourceforge.net>

—

This paper describes a new approach to non-photorealistic rendering, by imitating the inking style of cartooning normally associated with graphic novels. An algorithm is described that simulates the artist's working methods and goals, rather than focusing on a physical modeling and lighting. It adapts an expert system approach, where artistic knowledge is embedded within the program logic. Normal maps are created for a mesh object and the information is used to determine if a given pixel will be inked or left white, by comparing to user-defined thresholds for the six sides of a cube. The bitmap is then converted to vector format for simplification and anti aliasing.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation - Display algorithms

Additional KeyWords: non-photorealistic rendering, graphic novels

—

1. INTRODUCTION

Non photo realistic rendering has been the subject of extensive research and study (Hertzmann 1998). Numerous approaches have focused on creating models of lighting, silhouetting, the detection of creases, and frame coherence for stroke patterns. All of this research can be generally described as identifying data structures and/or algorithms that produce simplified images from geometric data. This simplification process is primarily computation/mathematics centric, focusing on the how model information can be used to generate the image, involving significant, complex calculations.

This paper approaches this simplification process by examining the steps that artists use to simplify subject matter and communicate through their imagery. Certain aspects of an artist's target goals can be expressed as a statement or aphorism which can be incorporated into the NPR pipeline. The idea is to say something like “Artists always strive to do X”, then describe X, and model X in such a way that it can be part of the computer-generated image. The approach would be that of an expert system where the program embodies knowledge drawn from the study of artists in action.

2. WHAT ARTISTS TRY TO ACHIEVE

There are numerous possible goals art tries to achieve but for simplicity we'll limit the discussion to visual goals,--rather than literary or emotional--and make the following proposal: *One property of a good drawing is that it clearly delineates the sides or planes of an object.* For example, in a portrait, the viewer clearly knows where the left, bottom, and top of the planes of the nose are. The lower and upper side of the lips can be easily located, the planes of face are discernible. We may not know a lot of things about Batman, but in every frame we know where the left side of his chest, arms and legs are.

This article proposes that the clear expression of the planes of an object's surface is one of those criteria. This overrides lighting: the need to communicate the “sidedness” of objects determines the arrangement of light and dark; lighting and the physics of such lighting is secondary. The author bases this proposal on his own experience with natural media, specifically pen and colored pencil.

The author can make no real claims to prove his statements, but rather offers them as an assumption which may be validated after the fact by examining the resultant images .

3. ALGORITHM DETAILS

The goal of the algorithm was to provide the rendering software necessary information concerning the orientation or “sidedness” of a given object and its components. Simple rules for operating on this data would be part of the algorithm as well.

Normal maps were made using the method of Philippe Decaudin [1996]. The color channels of the images represented the degree to which the mesh normals were parallel to the light direction or side. This would be the primary data structure used to symbolize or represent the information concerning planar orientation that an artist might use in determining if a given two dimensional image region would be inked. Decaudin used these maps to find silhouettes and internal lines and discontinuities. In this case, they are used to assign a threshold based on planar orientation. Operator input consists of specifying threshold levels for inking. For example, the user might specify the left and bottom sides to be inked a certain amount, by inputting values that are clamped between 0 and 1. These values are associated with the float values stored in the RGB channels of the normal maps.

A description of the programming pipeline should serve as an example.

Step 1: A mesh is loaded containing vertex and normal information.

Step 2: The user inputs the inking levels for each of the six sides, plus camera orientation

Step 3: Using OpenGL, a normal map is created for light coming from the top (red), left (green), and front (blue) channels. A second map is created for the bottom, right, and back oriented lights. The lights are world axis aligned.

Step 4: The method described in [Lander 2000] is used to create a silhouette. This involves rendering the model twice, once in wireframe mode and once in polygon mode, the latter render obscuring the previous except for silhouette edges (the effect depending on line width settings).

Step 5: Each pixel in the target image is assigned either black or white based on the normal maps of Step 3 and the threshold levels of Step 2. For example, if the Red channel of map 1 (left) is below the specified left threshold, the pixel is set to black. Pixels that are part of the silhouette of Step 4 are always black.

Step 6: The composite image is vectorized using the AutoTrace API (an open source raster to vector library) and final output is saved in AI (Adobe Illustrator 88) format.

4. DEMONSTRATION IMAGE

Figure 1 shows an example of output from the program. All hardware was consumer quality hardware. Poser 5 was used to create the mesh. Mesh objects were obtained from consumer suppliers.

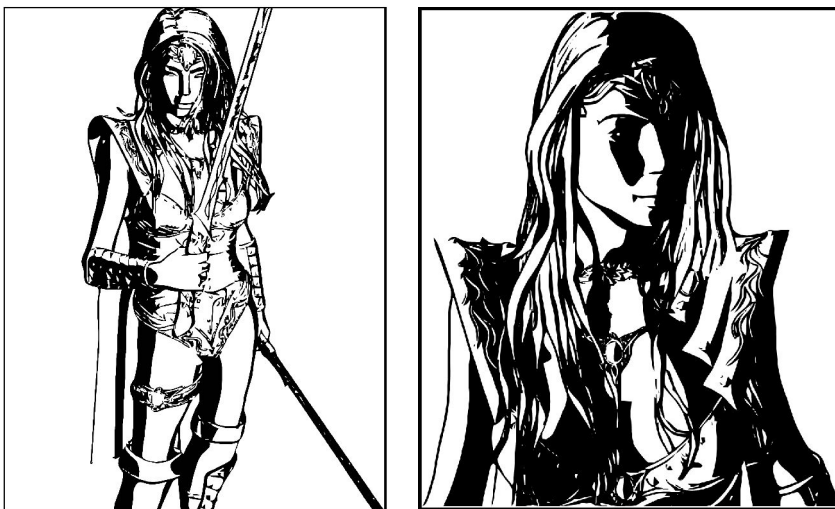


Fig. 1. Output Samples

This image shows that the expert system approach used here can create acceptable results with no special hardware requirements.

ACKNOWLEDGMENTS

Many thanks to the Autotracer project (<http://autotracer.sourceforge.net>) which made vectorization possible.

Poser 5: <http://www.curiouslabs.com>

Mesh Figure:Daz3D <http://www.daz3d.com>

Clothing: BVH of renderosity.com

More information, samples, and source code are available at
<http://inkulator.sourceforge.net>

REFERENCES

DECAUDIN P. 1996 Cartoon Rendering of 3-D Scenes , INRIA,Research Report 2919.

HERTZMANN, A 1998 Introduction to 3D Non-Photorealistic Rendering:
Silhouettes and Outlines, Media Research Laboratory Department of Computer Science, New York University

LANDER, J. 2000 Under the Shade of the Rendering Tree, Game Developer Magazine, vol 7. no 2, pp.17-21.