Antialiased DNI

by

Pablo Alliende

Computer Graphics Laboratory
University of Waterloo

An essay
presented to the University of Waterloo
in partial fulfillment of the
requirements for the degree of
Master of Mathematics
in
Computer Science.

Waterloo, Ontario, 1982

ABSTRACT

ATOMLLL and DNI are a pair of computer programs which produce full shaded images of molecular models. ATOMLLL performs visible surface calculation given the spatial coordinates of the individual atoms and bonds. It produces an intermediate description used by DNI to compute the final shading, and to do the output.

This essay was focused on improving DNI. Pictures produced using DNI had aliasing problems. Two antialiasing techniques were implemented to improve the final image quality.

Other problems with the program were attacked, device independency being the most important of them.

1. Introduction

The ATOMS program, written by Ken Knowlton and Lorinda Cherry at Bell Labs [Know77], receives as its input the description of atoms and bonds forming molecules. It performs a visible surface calculation, and its output consists of a description of the non-overlapping visible portions of the atoms and bonds. Steve Levine of Lawrence Livermore Laboratory (now Lawrence Livermore National Laboratory) wrote a version of ATOMS called ATOMLLL.

Subsequently Nelson Max (also of Lawrence Livermore Laboratory) added a post-processor called DNI, which displays the output of ATOMLLL on a Dicomed D48 graphic film recorder. The pictures output by ATOMLLL originally were in solid color. DNI added realistic shading and highlights. The versions available at Waterloo were obtained from Max in 1977.

Cathy Johnson, of this department, rewrote DNI in Fortran 77, making the flow control more understandable. Her version added a capability to display output on an Ikonas RDS 3000 frame buffer. She called the new version DNIF77

[John82].

For the current work DNIF77 was ported from a PDP-11/45 to a VAX 11/780, both under UNIX. Simple antialiasing capabilities were added. Several bugs were isolated and corrected. Much of the effort attempted a rationalization of the program. In particular, the device dependencies were centralized, making future addition of new devices easier.

2. ATOMLLL

The input for DNI is produced by the program ATOMLLL described in [Max79a, Max79b]. The version implemented on MFCF's Honeywell 66/60 computer was ported to UNIX as part of this work. Only minor changes were required for Fortran 77 compatibility.

The graphical preview feature of the Honeywell version was modified on UNIX. On the Honeywell, the outlines of the molecule being depicted can be displayed on an HP-2648A terminal. On UNIX this was changed to produce a file containing commands for the virtual device of the CS 488/688 graphics package. The outlines of the molecule can thus be previewed on any device for which the package is implemented, using a driver for the virtual device; this currently involves HP2648-A, Telidon, Versatec, and others. A description of such a driver is in Appendix IV.

The following problems were detected in ATOMLLL:

- The program fails when more than one frame is input, for no apparent reason. This problem disappeared when porting the program to UNIX.
- Some large molecules put the program in an infinite loop.
- Sometimes (unpredictably, but apparently when molecules are large) the output is incorrect.

For these reasons, it is suggested that ATOMLLL be rewritten. In addition to correctness, the following features could be added:

- Allow the user to specify coordinates with more generality. In the original program by Knowlton and Cherry, the user could state the camera's position and orientation with respect to the molecule being depicted. This feature was suppressed in Nelson Max's version. As a temporary solution for this problem, I wrote a preprocessor for ATOMLLL, called **preatom**. Its description is in Appendix III.
- De-Fortranize the user interface. Free format should be used, as provided by Fortran 77. The number of atoms and bonds should be counted by the program, not by the human user. Words should be used as input when necessary. In particular, when colors must be entered, the name of a color is better than a numeric code. Specifying atoms by numeric position is error prone. A name could be given to specific atoms (to be used in the bonds part) or to kinds of atoms (as an abbreviation for a color and a radius). When movies are produced, a more complicated processing is required, to view the scene from different angles. An interface like the one of **preatom** would become tedious to use, and some repetitive commands would be desirable. The best solution might be having a small "picture language" which could be written using Yacc or Lex.

3. Antialiasing

Aliasing may occur when we have to quantize an image for reproduction. It is manifested by jagged edges and lost detail. Sometimes we may get extraneous patterns not present in the original. These artifacts detract from the visual quality, and it is important to eliminate them. Here we are concerned primarily with adjacent areas of different colors within an image; the shading calculation within DNI does a very good job of handling the interior of atoms and bonds.

When the boundary of an area is not a vertical or horizontal straight line, we get aliasing problems in the form of jagged edges. Solutions to the aliasing problems are called *antialiasing techniques*.

To have less aliasing without increasing the resolution of the output device, we must take into account the fact that a given pixel is representing not an infinitesimal point of the original image, but instead an area of a very small, but measurable size [Crow77]. A good solution for the aliasing problem is to assign a given pixel a color and an intensity which are the averages of those for all the points in the figure which correspond to that pixel.

In the ideal situation we can calculate exactly the average of all colors and intensities involved, by integrating through the area of the pixel. In many practical situations we can consider that the pixels are small enough so that a given pixel either is all of the same color and intensity or there is a line which divides it into two zones, each of a different color and intensity. Then the intensity for that pixel is simply the weighted average of the intensities of each zone, where the weights are the corresponding areas that each intensity covers. Molecular models, having only straight lines and arcs of circles, can be approximated by this simplification.

For DNI two simpler methods were implemented, with the Ikonas frame buffer as output device. One solution has the program do its computation using double resolution, the intensity and color of each device pixel being the average of four computed pixels, instead of the value at just one point. This idea is attractive because no major changes have to be made to the program; there is already a feature to compute up to 4096 by 4096 resolution, and the Ikonas device resolution is only 512 by 512.

Another idea, which attempts to reduce the factor of four in the intensity computations introduced by the method above, is to take as an estimate for the

value of each pixel a weighted average of the central point of the considered pixel and of the pixels surrounding it. This is equivalent to blurring the image, which helps to smooth jagged edges. The term *dithering* is frequently used for techniques like this. This method is useful when calculating the intensity for a given pixel is costly, and a weighted average of nine (or five) intensities at neighboring pixels is less expensive than the calculation and averaging of four intensities at each pixel. The intensities in the pictures we are dealing with are easy to calculate, but nevertheless the availability of the weight method provides a good way to evaluate its results. As expected, the weight method is not as good as when double resolution antialiasing is used.

A simple way of being more time efficient with either of the methods described is to note that aliasing occurs only at the extremes of each scanline within an atom or bond. For many scanlines we can plot just its first and last points using an antialiasing technique, and all other points in the normal way.

The two methods described here belong to a family of filtering methods, described in detail by Crow [Crow77]. They correspond to applying a lowpass filter to the digital image.

4. Resolution

DNI originally could output in three different resolutions to the Dicomed: 1024 by 1024, 2048 by 2048, and 4096 by 4096. When the Ikonas output was added a problem was presented: none of these three resolutions is available in the Ikonas, whose resolution is 512 by 512. To solve this, the lowest resolution of the Dicomed was used (1024 by 1024), and only a fourth of the picture was plotted.

To add more generality the program was changed so that Ikonas output has a wider range of resolutions. A resolution of 512 by 512 allows viewing the com-

plete picture; 256 by 256 may be used for a quick preview of a picture. Finally, the resolutions 2048 by 2048 and 4096 by 4096 are now also possible using the Ikonas, following the same scheme used for 1024 by 1024 resolution, namely, only part of the picture is represented.

These last two resolutions are not very useful, because only a sixteenth or sixty-fourth of the picture can be seen at once, tremendously magnified. However, since there was no reason to forbid these resolutions, they were allowed. There is one exception to the statement above: when double resolution is being used, the resolution 4096 by 4096 cannot be selected. Since DNIF77 would need to be changed to accept an internal resolution of 8192 by 8192, and since the change would not achieve a very useful purpose, this restriction exists in the current version of the program.

At any rate, the resolutions 2048 by 2048 and 4096 by 4096 can be used to see the detail of a picture at the same resolution that would be plotted on a Dicomed. It may be useful to see such details on the Ikonas when preparing output for the Dicomed.

5. Device Independency

It is desirable that any program whose output is device dependent be designed in such a way that implementing a new device is an easy task. This can be achieved if all output is centralized in a clearly defined set of routines.

The original DNI program was designed for the Dicomed, without aiming at device independency, and hence, whenever a routine which deals with atoms and bonds has to perform output, it will emit the corresponding control words for the Dicomed with the calculated intensities in Dicomed format.

To overcome this problem DNIF77 centralized everything in one routine, called **output**, which receives two parameters: the first one is a Dicomed control

word, and the second one tells what the word is for. The Ikonas implementation, and any other device that might be added, becomes with this method a virtual Dicomed interpreter.

This is not a desirable solution, not only esthetically (because the routines which do the calculations know too much about the Dicomed and too little about other devices), but also practically (because the Dicomed requires its input in an unnatural way, making the Ikonas output more complicated than necessary).

To solve this problem, a set of high level routines was defined. All output must be performed using these routines, and the rest of the program need not know the special characteristics of the output devices.

The functions are the following:

- **Beg frm**: begin frame.

  For the Dicomed this means to output a frame header. For the Ikonas it means clearing the screen or its representation in memory. For other devices this could be nothing at all.

- **B sc lin(n, packed)**: Begin scan line.

  Be prepared to output a scanline of **n** pixels. The logical parameter **packed** is useful only for the Dicomed.

- **Emit i**: Emit intensity

  The given intensity is output to the device, using the selected coordinates. The **x** coordinate is advanced by one.

- **End frm**: End frame

  This means flushing the output to the device.

- **Init**

  Initialize a viewing session.

- **Set pos (x, y)**: Set beginning plotting position.

  x and **y** are stored and used by **Emit 1** as plotting points.

These routines now centralize all output to the graphics devices. A typical such routine looks like the following:

```
if( Device .eq. DICOMD ) then
   call Dc subroutine
else if( Device .eq. IKONAS ) then
   call Ik subroutine
endif
```

As new devices are added, more cases can be added to the **if**.

Alternatively, different versions of the routines can be loaded for each different device. This is useful when we do not have enough memory space to load a very large program. Output to a virtual file could also be generated, to have the calculated intensities in a form ready to be displayed on any device by a special driver.

6. Ikonas Output

Three output modes can be used with the Ikonas: the actual device, the Ikonas memory image, and the high precision memory image.

6.1. The actual device

The values are stored in the Ikonas, and when antialiasing is used, the spare byte in an Ikonas word is used for guard bits. Three of the spare bits are used for each of red and green, and the other two for blue. (One color had to be given only two bits, and blue was selected because the human eye has less discriminatory resolution for blue than for the other two primary colors).

Originally, the output for the Ikonas was along horizontal scanlines, whereas the input for DNIF77 comes in a format suitable only for vertical scanlines. Because of this fact, pictures obtained using DNIF77 have their x and y

coordinates inverted. Since the possibility of drawing full vertical scanlines is not available in the current interface with the Ikonas, I had to choose between having pictures with their coordinates inverted, or make the output to the Ikonas one pixel at a time, which is slower than using full scanlines. (Modifying the Ikonas interface was out of the question). I decided that output should be accurate rather than fast, so now pictures may take a little longer than with the other method, but they are correct.

Another possibility would be to change ATOMLLL so that its output is suitable for horizontal scanlines. This would not be a solution useful only for the Ikonas. On the contrary, many devices, including the Dicomed, are capable of producing only horizontal scanlines. The Dicomed version manages to compensate for this by having the hardware invert x and y a second time.

6.2. The memory image

The results are stored in memory in the same format used by the Ikonas. This is useful for saving images in secondary storage. The advantage of this method is that it uses only four bytes per pixel, being more economical in memory than the high precision format. This method can be used when we employ double resolution antialiasing or no antialiasing, because we need 2 guard bits per pixel in the first case and no guard bits in the second case.

6.3. The high precision memory image

With this method, each of red, green and blue is assigned 16 bits of intensity resolution, instead of 11 or 10, resulting in 6 bytes per pixel. The resulting memory image can be copied to the Ikonas or to a file, rounding the intensities to 8 bits. This scheme is useful for weighted antialiasing, because it overcomes roundoff error.

7. Input/Output Format

The output of ATOMLLL consists of records describing spheres (atoms) and cylinders (bonds), and the "trapezoids" which form each of those. In the original version, up to 200 records were read at a time, because the Dicomed needs two passes over the data, and 200 was the number of records chosen to be processed in a pass; it also was a convenient blocking factor for the magnetic tape that the Dicomed read. These records have eleven fields, and were stored as arrays.

It is much faster to read 200 records if they are stored consecutively in memory. Apparently for efficiency reasons, the records were stored in eleven consecutive arrays, and were read with one read operation. This organization forces the input to be ordered by fields instead of ordered by records. Note that using a matrix instead of 11 arrays would have solved the problem of having an unnatural order.

I decided that until both ATOMLLL and DNIF77 are running in a production environment, the savings provided by fast reading are marginal, and hence, fast reading was discarded as an objective. Therefore, I changed both ATOMLLL and DNIF77 so that the data representation has the records consecutively stored in the file, which is the natural representation for a human being trying to understand the output of ATOMLLL.

8. Gamma Correction

It is common that the intensities for an output device are not linearly distributed. An increase in output intensity by a given percentage may imply different increases on perceived intensity for different starting intensities. This problem is device dependent and its solution is well known. The intensities provided by a program must be *gamma corrected* to provide an apparently linear

response.

The original DNI program had an unorthodox gamma correction for the Dicomed, and the Ikonas implementation used the same method without adaptation. This results in the Ikonas having visible Mach bands when highlights are used. The above fact explains why on the first DNIF77 adding highlights is an option, instead of being the only possibility: the Mach bands were so notorious that the picture was less realistic with highlights than without them.

Now it is possible to add gamma correction to the Ikonas with a special command **(set_map 0)** or a library function **(Ik_gamma_lut)**. This allows us to ignore gamma correction in the program, and to use the command or library function instead. It turns out that the best results are obtained when gamma correction is done both by the program and by the Ikonas command, probably because the unorthodox method used by Nelson Max is complementary with the one provided by **Ik_gamma_lut**. Therefore, the highlighting part of the program was not changed, and Mach bands are suppressed by initializing the Ikonas with **Ik_gamma_lut**.

9. Summary

Antialiasing techniques can be used to successfully improve the quality of an image. Two methods were implemented on an Ikonas frame buffer.

As with any large program, maintenance can be a time consuming task. Several changes and enhancements were accomplished. These are not reflected in the image quality, but they help to increase the quality of the software.

APPENDIX I

ATOMLLL User's Guide

This program reads molecular descriptions from its standard input, and issues to its standard output a list of visible patches to be used by DNI.

The outlines of the picture being generated can be output to a file in the current directory, called **preview**. If **preview** existed, it will be erased prior to the writing of outlines. The file **preview** can be displayed using the command **virtual**, described in Appendix IV.

If abnormal conditions arise, an error message will be written to the file **fort.8**. (This should be changed, and the errors should be sent to the terminal).

Input format for ATOMLLL is described in [Max79a], and also in the file **atoms/atomlll/input-format**.

APPENDIX II

DNIF77 User's Guide

dnif77 asks the user several questions, most of them of a very obvious meaning. The name of the input file, the output device, and the kind of antialiasing desired are among these queries. An invalid response to any query will regenerate the same question until an acceptable response is obtained. The meaning of the less obvious queries is explained below. It is suggested that in future versions of DNI all this information be taken from the command line.

Input Format

The input of **dnif77** consists in records with a format **1117**. The meaning of the eleven fields is explained in [Max77a]. Note that records are not blocked in **dnif77**.

Dicomed Output

A file called **film_densities** should be on the current directory, containing the measured film densities required for the Dicomed. If such a file is not found, default data will be used. The default data is the same that is stored in **atoms/dni/data/film_densities**.

The input file will be asked for interactively, and the output will be stored in the file **Dicomed.out** in the current directory.

One of the queries that you will be asked is:

Wait between frames, double film advance, debug trace:
(0: no, 1: yes) 311 format:

(1) Select 'wait between frames' if special operator handling is required before signaling restart to the film recorder.

(2) Select 'double film advance' if the exposed film is to be mounted as 35 mm. slides.

(3) Selection of 'debug trace' option generates output on the file 'fort.7' in the current directory.

Ikonas Output

When Ikonas output is selected, it is useful to select output to a disk file if a picture is desired quickly, or the frame buffer is in use. The resulting file can be displayed later, as explained below.

Resolution in the Ikonas is selected as one device pixel per picture pixel. This implies that picture size will increase with its resolution. When the selected resolution is too large for the Ikonas the following message will be issued:

Since this resolution is too high for the Ikonas, only a fraction of the picture can be displayed.
By default this portion is the center of the picture.
Do you wish to change area displayed?

If the user wants to change the portion of the picture that will be displayed, then the coordinates of the center of the area must be entered. Depending on how close the chosen center is to the boundaries of the ATOMLLL data, the dimension of the displayed region may be less than 512 pixels in either direction.

When you want antialiasing with weights you will be asked the following query:

Enter the weights you want to use as fractions of 256.
(center, side, corner, where center + 4*side + 4*corner = 256)

This means that the intensity of a pixel will be added to its own location with a weight of **center**/256, to the locations with which it shares a side with a weight of **side**/256, and to the locations with which its shares a corner with a weight of **corner**/256.

Saving an Image Generated Directly on Ikonas Display

Issue the command

atoms/util/save_raster destination_save_file_name

This file can be displayed with the command **draw_raster**, as indicated below.

Manipulation of Frame Buffer Output on Disk

The commands

atoms/bin/fb_init
/u/daplebon/ikonas/bin/draw_raster frame_buffer

display on the Ikonas the file **frame_buffer**, which must be the representation of an Ikonas screenful in the format used by **dnif77** or **save_raster**.

APPENDIX III

preaton

In the original version of ATOMS the user could specify viewing parameters such as the position of the camera or its distance from the picture plane. In ATOMLLL this feature was deleted, and the camera is assumed to be at the origin, looking trough the z-axis. The only viewing parameter that the user can set is the distance of the camera from the picture plane. The picture is assumed to have a width of 2 units.

Clearly this situation is too restrictive for a serious user, because one is forced to choose a reference coordinate system which may not be appropriate for the application; and because to see the same molecule from a different point of view one cannot "move the camera". Instead one must "move the molecule", i.e., specify all the coordinates from another point of view.

I decided that viewing flexibility should be restored to ATOMLLL, so I made a preprocessor to do that. Eventually the preprocessor could be built in ATOMLLL itself. The reasons for not doing so now is that I wanted to incorporate Level 4 of CS 488/688 graphics package into the preprocessor, and that package is written in Pascal.

The preprocessor, called **preaton**, receives as input the description of a molecule in a format similar to that required by ATOMLLL, but the coordinates are in any reference system the user wants to have. This molecule description is preceded by control commands specifying the characteristics of the coordinate system.

User Documentation

preaton receives as its first argument a file containing a sequence of commands specifying a reference coordinate system, and the description of

molecules, and emits to its second argument the description for the same molecules in the reference system required by ATOMLLL.

The commands that describe the reference coordinate system are as follows:

- **c** x y z

  indicates the location of the camera. Its default is (0, 0, 0).

- **u** x y z

  is a vector which tells what direction is to be considered to point upward. Default is (0, 1, 0).

- **s** x y z

  indicates the sight vector, i.e., in which direction the camera points. Default is (0, 0, 1).

- **d** x

  indicates the distance of the camera from the projection plane. Default is 5, or if width is specified, default is 2.5*width.

- **w** x

  indicates the width of the resulting scene. Default is 2, or if distance is specified, default is 0.4*distance.

A command letter must be the first character in the line. The letter may be either in upper or lower case. A line beginning with a blank indicates the beginning of data. Data is in free format (except that the first line must begin with a blank) and the order of description of molecules is the same order required by ATOMLLL [Max79a], (see also the file **atoms/atomlll/input-format**) but the first line of a scene contains only the number of atoms, because what Nelson Max calls **pratio**, (half of the viewer's distance from the

screen), can be specified in the command lines. If more than one scene is present, then the default parameters are the ones for the last scene.

APPENDIX IV

Auxiliary Programs

The following programs are related to DNI and ATOMLLL. Their sources are under **atoms/util** and executable files are under **atoms/bin**.

virtual

This program is an interpreter for the virtual device of the CS 488/688 graphics package. It will ask to the terminal for a device to which do the output. It receives the virtual commands generated by ATOMLLL on its standard input, and displays the result in the selected device.

merge file1 file2 file_out

file1 and **file2** are Ikonas images. The result of superimposing **file1** on top of **file2** is stored in **file_out** (all non-zero pixels of **file1** overwrite pixels of **file2**).

move_rect file_in file_out

Six values are read from the standard input: **left**, **right**, **top**, **bottom**, **vec_x**, and **vec_y**. The first four values specify a rectangle of the screen in Ikonas coordinates. The last two values specify a translation vector. **file_out** will have the contents of **file_in** that lie within the rectangle, and displaced in the amount indicated by the vector.

cube

The source for this program is under the directory **atoms/cube** and an executable module is **atoms/bin/cube.b**. This program asks the user for a number **n**, and the coordinates of a "cubic molecule" of side **n** are sent to the standard output. This molecule consists on an array of atoms forming a cube of **n** atoms per side. Each atom is connected to its immediate neighbors by a

bond. This program can be used to test ATOMLLL and DNI with simulated molecules. Real life molecules are not as regular as this cubic molecule, but it served to detect a number of bugs in ATOMLLL. The cube of order 2 produced wrong output and the cube of order 4 put ATOMLLL in an infinite loop. The executable file **atoms/bin/cube** receives one argument. It calls **cube.b** and **preatom** (see Appendix III) and puts into the file argument the coordinates of a cubic molecule in a format appropriate for ATOMLLL.

APPENDIX V

Sample Program Output

The following figures are Polaroid prints taken using a Dunn 631 Color Camera connected to the Ikonas frame buffer system.

Figure 1

A "molecule" used to compare output using three different methods. We have the same picture computed using double resolution (top), no antialiasing (center), and weight antialiasing (bottom). Note how straight lines get smoother with both types of antialiasing, and how circular arcs are also improved with double resolution antialiasing.

Figure 2

This is an enlargement of a section of Figure 1. Again, we have double resolution (top), no resolution (center), and weight antialiasing (bottom).

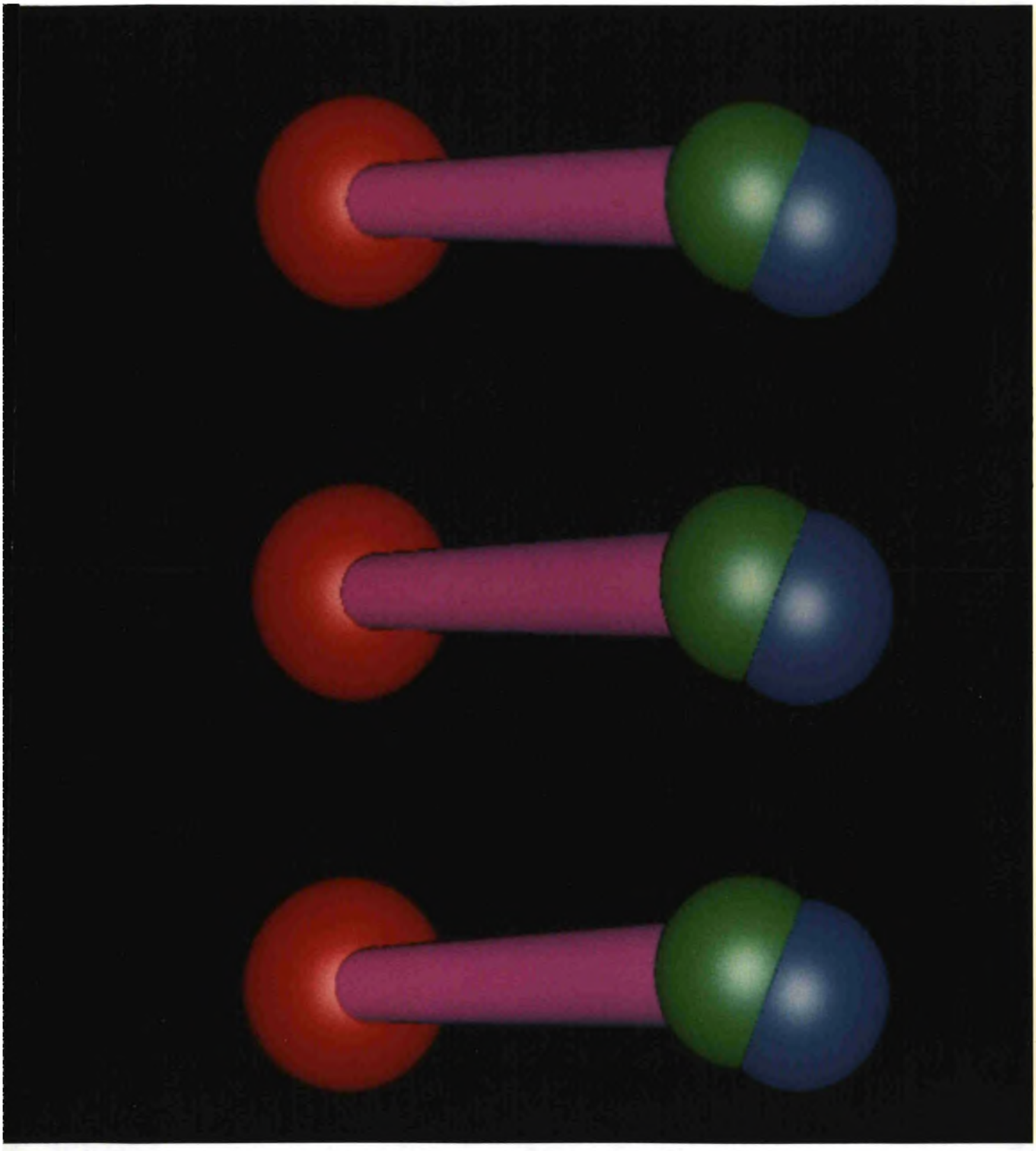

Figure 1

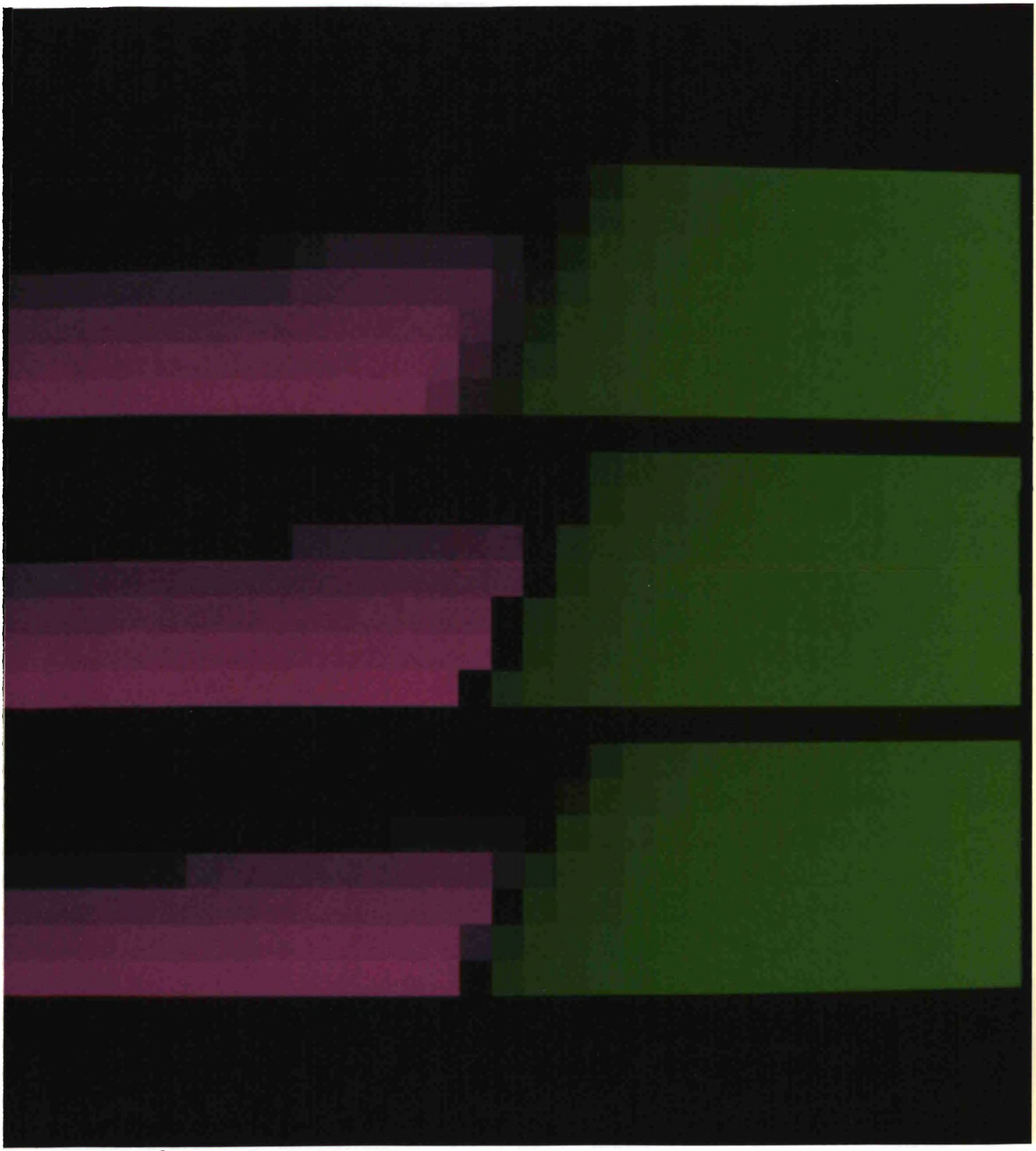

Figure 2

Acknowledgments

I would like to thank my supervisor, Professor Kellogg S. Booth, for suggesting the topic of this essay, and for his guidance during its preparation.

REFERENCES

Crow77 Crow, F. C. "The aliasing problem in computer-generated shaded images", *Communications of the ACM* **20**, 11 (Nov. 1977), 799-805.

John82 Johnson, C. H. "DNIF77: A Program for Shading Molecular Models", Master's Essay, Department of Computer Science, Faculty of Mathematics, University of Waterloo, 1982.

Know77 Knowlton, K., and Cherry, L. "ATOMS — a three-d opaque molecule system — for color pictures of space-filling or ball-and-stick models", *Computers and Chemistry* **1**, 3 (1977), 161-166.

Max79a Max, N. "ATOMLLL — a three-d opaque molecule system, Lawrence Livermore Laboratory version", UCRL-52645, Lawrence Livermore Laboratory, University of California/Livermore, Jan. 1979.

Max79b Max, N. "ATOMLLL: - ATOMS with shading and highlights", *Computer Graphics* **13**, 2 (Aug. 1979), 165-173.