This is to certify that the

thesis entitled

DEVELOPMENT OF TWO AND THREE-DIMENSIONAL CARTOGRAPHIC ANIMATIONS TO VISUALIZE POPULATION CHANGE

presented by

Jill K. Hallden

has been accepted towards fulfillment
of the requirements for

M.A. degree in Geography

Richard Groop
Major professor

Date 5/13/99

DEVELOPMENT OF TWO AND THREE-DIMENSIONAL CARTOGRAPHIC ANIMATIONS TO VISUALIZE POPULATION CHANGE

By

Jill K. Hallden

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF ARTS

Department of Geography

1999

ABSTRACT

DEVELOPMENT OF TWO AND THREE-DIMENSIONAL CARTOGRAPHIC ANIMATIONS OF POPULATION CHANGE

By

Jill K. Hallden

Despite a recent surge in research, cartographic animations remain a frontier area in Geography. As late as 1990, authors such as Campbell and Egbert bemoaned the lack of development of temporal mapping techniques. They point to pioneering work by Thrower (1959), Tobler (1970), and Moellering (1980, 1984) as evidence of the promising future animation holds, but note it has not yet come to pass. Recently, animation in cartography has been researched more thoroughly. Yet, articles resulting from this research tend more towards the theoretical, and less towards an explanation of specific development techniques. This research answers the challenge set down by Campbell and Egbert to develop new methods for visualizing spatio-temporal data, and, in the process, leads to the development of a new technique for creating three-dimensional animations. I will describe the construction of the animations in detail in the hope that others pursuing similar goals can use the techniques described. The animations are employed to show U.S. population density change by county from 1790 to 1990. An expert panel of professional cartographers, through the use of a questionnaire, informally reviewed the animations. The results of the questionnaire support animation as a technique and warrant continued research into the both the two and three-dimensional animations.

In loving memory of
my dear friend

Mark Johnson
1969 – 1999

and his sister

Jennifer Johnson
1975 -1999

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Dick Groop for all of his encouragement, support, and guidance through this technical thesis. As my advisor, employer, co-collaborator, and fellow Spartan hockey fan, he has made many positive influences on me. I would also like to thank Gary Manson and Joe Darden for their thoughtful commentary and valuable input.

Over the course of this project, several members of the Department of Geography and the Center for Remote Sensing and GIS have helped guide me along, much to my appreciation. In particular, I wish to thank Sharon Ruggles for answering all my questions with a smile, and the Atlas of Michigan gang for their support, suggestions, and stress relieving Wednesday lunches.

My friends and family, at Michigan State and across the U.S., have kept me going through this thesis. My brother-in-law, Evan Harsha, deserves special recognition for his help in creating the Convert-o-rama data extraction program. Without his efforts at the beginning of this research, the resulting animations may not have been created. My parents, Al and Sue Hallden, have supported and encouraged me throughout the course of my education. In so many ways, they are my role models.

And lastly, but most importantly, I wish to thank my husband, Peter Harsha, for more things than I can ever list here. He has held so many roles over the course of this thesis: bug tester, computer tech support, laptop hauler, grammer checker, dinner provider, etc. Most importantly, he has kept me sane through this ordeal, and for that I am forever grateful.

PREFACE

This work is accompanied by a CD-ROM which contains the animations that were designed for the project. The animations can be viewed on either Apple Macintosh computers or Windows-based PC-compatible computers. Minimum hardware requirements can be found in Appendix E.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter One

CARTOGRAPHIC ANIMATION

In their 1990 essay, "*Animated Cartography: Thirty Years of Scratching the Surface*," Craig Campbell and Stephen Egbert make the case that animation, as applied to thematic cartography, has been underutilized by geographers (Campbell and Egbert, 1990). They point to pioneering work by Thrower (1959), Tobler (1970), and Moellering (1973, 1980a, 1984), who all predicted a promising future for cartographic animation. Thrower, in his 1959 essay *Animated Cartography*, described a method for animating maps by shooting individual frames of animation and then playing each back in quick succession, just as cinematographers animated cartoons. Thrower recognized the impact the ability to incorporate time and space could have on geography, and predicted its eventual widespread use (1959). Tobler was the first to tap the potential of computer-generated cartographic animation when he developed a three-dimensional surface that changed over time to represent urban growth in Detroit (1970). Three years later, Moellering used a computer to generate an animation of traffic accidents to highlight dangerous intersections (1973). Moellering's research in 1980 to map, in three dimensions, U.S. population density change provided a model for future animation construction, but highlighted the limitations of the hardware and software available (1980a). Though provoking, Moellering's animation is necessarily crude because of the limited number of polygons his computer could display at any one time.

Despite this pioneering work, Campbell and Egbert lament that in an age with powerful computers and sophisticated software readily available, that promising future

"has not come to pass" (Campbell and Egbert, 1990). Throwing down the metaphorical gantlet, they call for

> ...[r]esearch on topics such as the development of different types of animated thematic maps and the uses of color in dynamic sequences. Efficient, human-orientated, computer production techniques need to be researched and perfected. The problem of product media choices and dissemination procedures for animated cartography must be confronted (Campbell and Egbert, 1990).

In the eight years since Campbell and Egbert's essay, it appears that a growing number of geographers have picked up that gantlet. More than 35 articles on cartographic animation or cartographic visualization have appeared in academic journals since 1990. The 1997 and 1998 annual meetings of the Association of American Geographers have each featured a number of sessions devoted to animation and visualization. And in 1993, the International Cartographic Association Commission on Map Use recognized the growing role of visualization in cartography and formed a visualization working group to address its implications (MacEachren, 1998).

Despite the significant amount of research on cartographic animation and cartographic visualization, the focus of much of the current research has been on the principles of animation design and studies of its effectiveness rather than specific illustrations of development techniques. For example, Gersmehl (1990) examined some of the tools available for the creation of cartographic animations, but did not delve into the specific steps required to use each of the tools to produce an animation. In their paper, "*Visualizing spatial relationships among health, environmental, and demographic statistics: Interface design issues*," MacEachren et. al., discussed the development of a

prototype interactive animation of health statistics, and how that development followed a hierarchical approach to visualization design (conceptual, operational, and implementational) (MacEachren et. al., 1997). The discussion included descriptions of the two pieces of software used in the creation of their animation (ESRI ArcView and Macromedia Director), but does not provide explicit instructions for their use.

DiBiase (1994) provided considerable detail describing the challenges his cartographic laboratory (GeoSystems) faced developing animated maps for a multimedia encyclopedia. His discussion of the workflow followed by GeoSystems in creating the animations is instructive in much the same way as MacEachren's hierarchical approach to visualization design – namely, it provides a possible path for those pursuing similar goals. But DiBiase devoted more time to the particular pieces of software employed, including the detailed procedure GeoSystems used in creating animations. This "step-by-step" description is important not only because it provides a method for future cartographers to replicate GeoSystem's procedure – perhaps saving valuable time in the implementational phase of development for their particular project– but also because DiBiase's instructions include a discussion of the reasons for taking many of those steps. This discussion ensures that the errors made by GeoSystems need not be repeated by others working on similar projects.

My research is intended to answer the challenge of Campbell and Egbert by developing animated thematic maps in two and three dimensions. I will describe the construction of the animations in detail in the hope that others pursuing similar goals can use the techniques described. In particular, the technique employed for the creation of the three-dimensional map represents, to my knowledge, a new method for animating maps in

three-dimensions. The animations are employed to show U.S. population density change by county from 1790 to 1990.

Chapter 2

CREATING TWO AND THREE-DIMENSIONAL ANIMATIONS

My objective was to animate U.S. Population density change, by county, from 1790 to 1990, using two different animation techniques: two dimensions plus time, and three dimensions plus time. The two-dimensional animation took the form of a choropleth map of the United States (an Albers equal-area projection of the 48 contiguous states), on which counties change color as their population density values change. The three-dimension map used the same U.S. map, but elevated the counties and changed their color values as the population density values changed. The development proceeded along a path illustrated by the following process diagram:

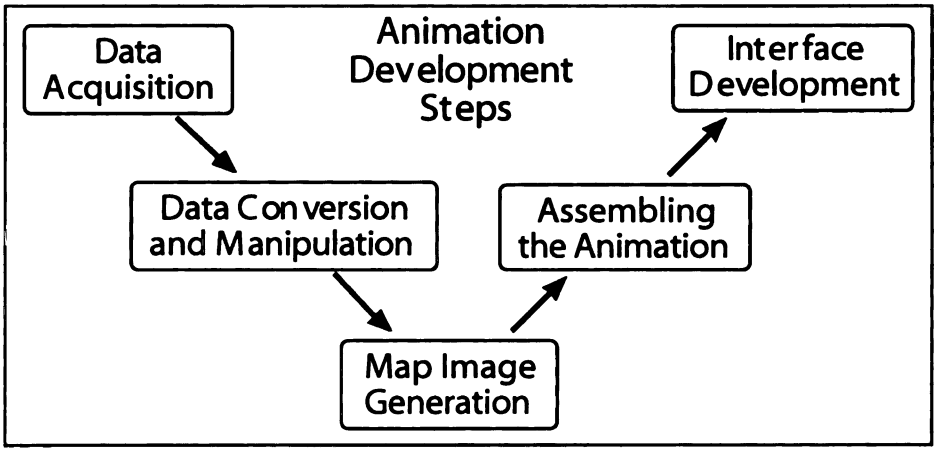


Figure 1. Process Diagram of Animation Development Steps

Acquisition of Data

The acquisition of data proved to be one of the more difficult steps of the project. Though the Internet, and the World Wide Web in particular, was in widespread use at the

beginning of this project in 1996, much of the data a researcher would expect to find on-line today had not yet appeared. While finding county-level historical U.S. Census data on-line is relatively easy in 1998 (it is, in fact, available from the U.S. Census Bureau's web site at http://www.census.gov) in 1996 the only location that offered the data in digital form was the Inter-University Consortium for Political and Social Research (http://www.icpsr.umich.edu/ICPSR/home.html), of which Michigan State University is a member. These data contained all of the information collected about each county surveyed – information such as the total aggregate population, the racial breakdown (often by age groups) of each county, and even the number of churches (by denomination). It was thorough and massive in size.

Because I only needed county names, populations, and county FIPS codes for my research, I helped develop – with Evan Harsha, then a computer science student at the University of California at Berkeley – a program written in C++ that extracted these fields from the ICPSR data. This task was made more difficult because each historical census year (each decade) used a varying number of fields and method of storing the population value. Some census years, for example, used an aggregated population total, whereas other years had separated totals in two categories (slaves and non-slaves, urban and rural, male and female) requiring a double search. The conversion program, called "Convert-o-rama," used a different search routine for each census year, and used a FIPS conversion routine to compare the ICPSR-assigned state and county identification numbers to a table of current FIPS codes. Because counties have changed, sometimes significantly, over the last 200 years, matching the historical data to current county locations was sometimes problematic. Within the dataset, ICPSR attempted to match

historical county data with comparable current counties, but where that was not possible, values are simply recorded as "no data." As a result, what began as a small, rather insubstantial text-conversion tool blossomed into a rather substantial program (see Appendix A). Ultimately, Convert-o-rama was able to output converted data files in tab-delimited text form, suitable for importing into a Microsoft Excel database. As a result of the conversion, the data files also slimmed down from their 170 megabyte size to a much more manageable 1 megabyte size.

By opening each data file within Microsoft Excel, I was able to "cut and paste" the contents of a particular census year into a master file containing all 20 census decades. I was also able to "paste" in the area values for the counties – values needed to calculate the population densities of the counties ($population\ density = \frac{county\ population}{county\ area}$). Because I wanted to create "in-between" years for my two-dimensional animation, I used a linear interpolation between population density values for census decades to calculate population density values for the decade +2, +4, +6 and +8 years (see Appendix B for formula). The resulting file contained population density values for every even-numbered year from 1790 to 1990. This file was then saved in database format (*.DBF) so as to be easily read by ESRI ArcView, the program I used to generate my U.S. county maps.

Map Generation

Both the two-dimensional and the three-dimensional animations utilized the same base map generated on an Apple Macintosh in ESRI's ArcView program: an Albers' equal-area projection of the 48 contiguous United States at the county level. Using the same base map ensured a consistency of appearance between the two animations – an

important consideration should any future research focus on the relative effectiveness of the animations. Also, because I planned to bundle both animations together within a common interface, I hoped the consistency of appearance would minimize any difficulty a viewer might encounter trying to comprehend what data were represented when moving between animations.

ArcView is a popular desktop mapping and GIS software package. Data, formatted as tab-delimited text, can be imported, joined to geographic shape information and then displayed spatially. The program allows the creation of several map types including dot-density, graduated symbol, and choropleth. Because the data for this project are choroplethic – that is, they represent values for areas rather than for points – the choropleth map suggested itself as the most well-suited for these animations. In order to keep the animation visually uncluttered, I assumed that county boundaries throughout the animation remain constant. Though this does not necessarily reflect reality – counties and county boundaries have changed, sometimes significantly, since 1790 (although little has changed since 1910) – I felt that having the distribution change over a changing area might prove too distracting to a viewer. I was also limited by not having ready access to digitized historical county boundary files. These limitations aside, I was able to generate a choropleth map of population densities for every even-numbered year from 1790 to 1990.

Classing

In order to make sense of the values displayed by ArcView, it was necessary to decide on a classing system for the maps. The extended time period covered by the

animation and the resulting wide range in population densities presented particular data-management and display problems, especially with the range in population densities between the sparsely populated 1790s and the much more urbanized 1990s. In addition, I wanted a high number of classes in order to portray a smooth and gradual population density evolution, as well as to show changes in distribution that might not be apparent using fewer classes with larger intervals. At the top end of the classing system I adopted a large class of 10,000 to 75,000 to accommodate 20th Century metropolitan areas (I did not feel it was necessary to differentiate between metropolises of 10,000 people per square mile and those with 20,000 or 30,000 per square mile). The high end represents the highest observed density at any point in the data set (New York City, 1990).

Creation of the lowest class for the animation was guided by my research of Fredrick Jackson Turner's 1893 paper, "The Significance of the American Frontier." In his paper, Turner observed that as of 1890, the American Frontier, the force he believed "explained American development" (Billington 1973), had ceased to exist. He noted that as of the 1890 census, a clearly demarcated line between settled and unsettled lands was no longer evident. Because this project grew out of my research into Turner's observation (and as a sort of homage to Turner, a historian with a fondness for maps), I set the lowest class of 0 to 2 persons per square mile equal to Turner's definition of "unsettled" land. In this way, I hoped to see the progression of the American frontier across the Western United States. With the high and low classes set, the remaining classes are exponential in nature, with slow increases for the beginning classes and larger jumps as the densities increase (Figure 2).

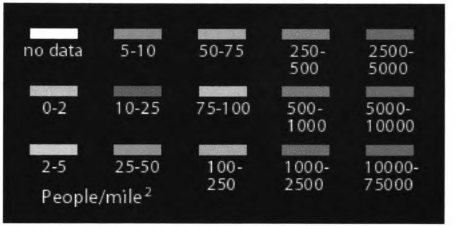


Figure 2. Classes and colors for animations.

Assigning color values to the legend classes proved challenging due to the large number of classes (15). I considered the two approaches open to the cartographer faced with portraying progressive classes: (1) using several hues and (2) using a series of values of one hue (Robinson et. al., 1984). Using the single hue method, I found I could not create as many distinctive values as would be required to allow a user to easily differentiate classes, especially since the maps changed color during the animation. If this classing system were to be used for a series of static maps, the single hue method – or perhaps a double hue method – may be adequate for discerning the classes. But for an animated map series viewed on a computer monitor, I judged the differences in values in the single hue (and double hue) approaches to be inadequate. I settled on a full-spectral color system ranging from pale blues to teals, greens, and yellow, then through oranges, reds, ending in fuchsia as the highest density category. Cool colors symbolized low densities and intense shades indicated high population concentrations. Kumler (1988) found, in a study on continuous tone maps, that "subjects performed better on a spectral scheme involving several colors than on either of two three-color schemes."

Once I determined the number and interval for the classification scheme, I assigned the color system to the classes using ArcView's legend editor. The resulting map legend was saved and could then be applied to each of the years in my data, ultimately creating 100 individual maps classed using the same classification scheme.

Map Creation

Using the legend editor, I was also able to experiment with different looks for the maps. Most significantly, I was able to view the data with and without county boundaries displayed. Because I was not sure whether any particular viewer would wish to see the boundaries, I decided to render the maps without county boundaries and provide a method in the final interface that would allow the user to view the boundaries turned on or off. I also tried a number of different projections for the mapped data before settling on the Albers Equal Area projection – a projection I felt was the most easily recognizable.

Creating the maps in ArcView required the use of ArcView's layout function. The "layout window" uses a printer's paste-up metaphor. One selects a paper size and then assembles upon it the items one would like ArcView to output – maps, legends, explanatory text, or color bitmaps, for example. ArcView can then render the layout for output to a printer, save the layout as an Encapsulated PostScript file, or, on the Macintosh, generate a color bitmap image of the layout called a PICT. For my purposes, I used the layout mode solely to define the extent of my image area and to output the resulting map as a PICT file which I could then manipulate further, if necessary, in an image editing program like Adobe Photoshop. Using the same layout for each map image

is crucial to ensure proper registration and a smooth animation. Any differences between map images like slight size changes or color shifts that might not seem apparent in a series of static maps become quite noticeable when viewed sequentially in an animation.

The actual creation of the 100 separate map images was a matter of loading the legend, selecting the year data I wished to see mapped, and exporting the results from the layout window as a PICT file. The resulting PICT files were then brought into Adobe Photoshop 5.0 on the Macintosh where a background color was added and then each image cropped to create a smaller file size. Photoshop version 5.0, through the use of scripted "actions," makes this process much less arduous than previous versions of the program which required the user manipulate each individual image. With actions, I could make the changes I wanted on the first image in the series, record those changes, and then apply them to every remaining image in the series as a batch process (See Appendix C for a breakdown of this action).

This process created all the necessary maps for the two-dimensional animation. For the three-dimensional animation, an additional set of maps was required.

Assembling the Two-dimensional Animation

To assemble the two-dimension animation, it was necessary to import the 100 individual map images (now cropped and background-colored) into Macromedia's Director 6.0. Director is authoring software designed for creating stand-alone interactive multimedia programs. Director uses the metaphor of movie production to structure its operation. Each "movie" or program created in Director is made up of a series of frames that play according to an author's script. An author creates a "cast" holding imported

images (or images created using Director's drawing tools), places the cast members on Director's "stage" (the area of the screen upon which the action occurs) and then assigns them actions. A "score" helps organize the appearance of cast members upon the screen and provides a map to the structure of the movie by representing it as a series of frames.

In order to create the two-dimensional animation, I imported all 100 map images into Director as individual cast members. I placed the resulting cast members in chronological order in the score, allowing each 10 frames of the animation. At a movie playback speed of 10 frames per second, this would result in each image being onstage for one second, or 100 seconds for the full animation. Like an old-time cartoon flip-book, this quick playback of sequential images resulted in a smooth animation of population density change. Once all 100 cast members were placed in the score, I could view the animation and experiment with the presentation.

One element of the presentation I attempted to add, but ultimately discarded, was the use of a slight "pixel dissolve" transition effect between map years. On the Macintosh (my primary authoring machine), this type of transition determines the pixels that change value from frame to frame and places a quick random pixel by pixel dissolve effect between the frames. The effect was subtle but effective. The counties seemed to quickly "grow" across the country rather than just appear in an instant. However, when porting the animation to the PC platform to check for cross-platform compatibility, I noted that the PC rendered this pixel dissolve effect in a way just different enough to eliminate the smooth effect I was trying to achieve. It appeared that the PC could not render the transition at the level of individual pixels; rather, it rendered small blocks of pixels at a time. This resulted in a much coarser transition – instead of individual pixels almost

indiscernibly changing from one frame to the next, little blocks of pixels changed from one to the next. Because some counties on the map are little more than a few pixels wide, this gave the effect of some small counties changing color all at once and larger counties changing block by block. Reluctantly, I discarded the effect on both the PC and Macintosh versions.

Once I had achieved the basic animation of the map images, I could add other necessary elements to the animation. I created a map legend in Adobe Photoshop, which I then imported into Director. I also created the elements of the time line in Photoshop, then imported and animated them in Director. I added a simple "radio button" that allowed the user to turn county boundaries on or off. As a final step, I set the playback speed of the animation to a rate I believed was slow enough to note particular attributes of the animation, but quick enough to convey a sense of movement across the map. This movie was then saved, ready to be incorporated with its three-dimension partner into the final interface – a process I describe below.

Creation of the Three-dimensional Animation

To create the three-dimensional animation, I turned to a program not usually considered for cartographic animations of choroplethic data. MetaCreations Bryce3D is an inexpensive fractal-based terrain generation program for the Apple Macintosh and PC-compatible platforms, used most often to create surreal landscapes for computer games. I had become familiar with the program after reading a web tutorial on a web site (that Unfortunately no longer exists) describing how to use it to map USGS Digital Elevation Model data in three dimensions. Though Bryce3D cannot directly import the DEM data,

it could accept the grayscale bitmap output from any program that could read and display DEM data in two dimensions (such as DEMReader by Brian Wagner, for the Macintosh). What Bryce3D can do is interpret any grayscale image as a height map, generating a "terrain" the same x, y dimensions as the image with elevations determined by the pixel values in the image (Figure 3).

Figure 3. Example of Bryce3D "terrain" generated from a grayscale image.

Just as importantly, Bryce3D also has the ability to metamorphose, or "morph", between two terrains over time. In this way, the designers of the program hoped that users could create mountains that seemed to rise from the sea or hills that seem to erode

(Bryce Users Manual). I realized that this same "morphing" ability could be put to use animating and interpolating a three-dimensional prism map. By setting up "keyframes," or, in the case of my project, points in the animation where the values are known absolutely, I could direct Bryce3D to interpolate and create the in-between frames automatically. My task then was to create grayscale images of each census year that could then be used as keyframes for animation.

Generating the Grayscale Maps

The grayscale maps were generated in identical fashion to the maps I used for the two-dimensional animation – ArcView was used to map my census data using classes and colors I designed. However, instead of using the full spectral color system for my map legend, I created a grayscale legend in which the higher values (the denser populations) are indicated by the lightest shades of gray. To determine 16 equal classes of gray (one for each population density class plus one for counties with no data), I created a technique in Photoshop using the gradient tool (detailed in Appendix D). With this new legend, I could quickly generate grayscale versions of each census decade (1790, 1800, 1810...etc.) and export them as PICTs as I had with the two-dimensional animation map images. In Photoshop, I created an action that changed the background of the exported images to black and cropped them to reduce file size.

Using Bryce3D

In order to create the appearance of a single prism map changing values over time,

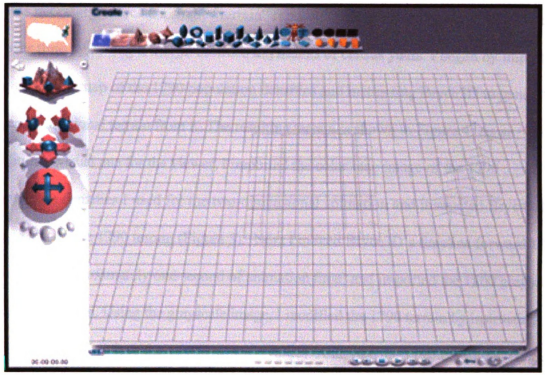


Figure 4. Screen-shot of Bryce3D 1790 "wireframe" terrain.

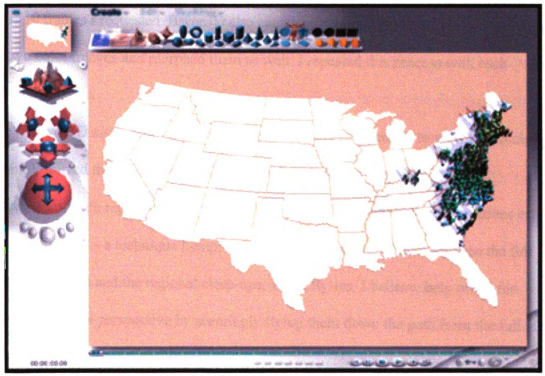


Figure 5. Screen-shot of Bryce3D 1790 terrain with draped 1790 color map image.

it was necessary to create a single "terrain" representing the U.S. county map and modify it over time by having it morph between keyframes of census years. I began by importing the grayscale map of the 1790 population densities and instructing Bryce3D to generate a three-dimensional terrain from it. The resulting terrain is shown in Figure 4. As an additional visual cue for the viewer, I imported the color map image of the 1790 population densities I used for the two-dimensional animation and instructed Bryce3D to drape it over the 1790 terrain (Figure 5). Next I moved Bryce3D's timeline forward 10 frames (or approximately one second of animation time) and imported the 1800 grayscale and color images. Using the grayscale image, I modified the 1790 terrain by instructing Bryce3D to re-map it to the 1800 grayscale values. And, like the 1790 keyframe, I added the 1800 color map image as a drape for the newly modified terrain. Bryce3D then calculated the in-between frames required to morph the terrain between the two decades in the 10 frames I requested. Bryce3D also calculated the changing pixel values between the two color images and morphed them as well. I repeated this process with each remaining decade.

Bryce3D also allows the user to select a camera perspective to view the resulting terrain. I utilized this ability to change camera locations and develop "close-up" animations of each region of the country. The process of changing camera positions can also be animated – a technique I employed for the "fly-in" transitions between the full U.S.A. animation and the regional close-ups. These fly-ins, I believe, help orient the viewer to the new perspective by seemingly flying them down the path from the full view to the close-up.

Once I had created the terrain, added the keyframes, and established the camera perspective, I could preview the resulting animation as a wire frame model to get an idea of what the fully rendered animation would look like. Unfortunately, because the wire frame preview is rendered in real-time, it is necessarily coarse (a limitation imposed on Bryce3D by the rendering ability of the average personal computer). Some errors are not obvious in wire frame view, and I was rarely confident I had selected exactly the right perspective before beginning the full rendering process. This was problematic because of Bryce3D's quite lengthy rendering times. Bryce3D has an excellent ray-tracing rendering engine, but because it is designed to run on personal computers and not high-powered graphics workstations, its rendering times can be quite long. Whether rendering on a 250mhz 604e Macintosh PowerPC-compatible machine or 400mhz Pentium II Windows NT machine, rendering times of more than 40 hours were typical for these animations. This large time investment made slight errors in the animations punishing; more often than not, an error would not be discovered until after the rendering had completed. The eleven full animations in this project represent over 440 hours of actual computer rendering time (or 11 full 40-hour weeks), not including the numerous mistakes that required re-rendering. While these times do not represent costs in labor, per se, there is the loss of computer time to be considered. As long as the computer is rendering an animation, it cannot realistically be used for any other work.

Bryce3D outputs these rendered animations as either a collection of individual bitmaps or PICTs, or as a Quicktime (Macintosh) or AVI (PC-compatible) movie. The only difference in rendering output between the two platforms is the choice of video compressors (or codecs) available "out of the box" with the Bryce software. The

Quicktime format for Macintosh supports the use of a codec geared specifically for animation called the "Animation" codec. This created, to my eye, significantly sharper animation than the comparable PC-compatible codec "Cinepak" (also supported by the Macintosh), but at the expense of much larger file sizes. The difference in file size between an animation rendered in high-quality Animation mode versus high-quality Cinepak mode was more than double. The Pacific Southwest close-up, rendered with Quicktime's Animation codec required 43.1MB of file space. The same close-up rendered in Cinepak required 21.0MB of file space. I noticed a significant difference in the look of the animations between the two codecs, but, in a questionnaire I sent with copies of the animations to cartographic professionals nationwide, only two respondents out of 25 also noted the difference.

One concern with rendering these animations in Bryce was the difficulty of including a timeline. Because the animations are rendered as Quicktime movies (or AVIs) and not as a collection of PICTs to import into Director, synchronizing a timeline to the movie is best done by creating another movie with just the timeline and compositing the two together. Using Bryce3D, I was able to create a timeline superimposed on a blue background that progressed at exactly the same rate as the animations. I took this timeline movie and imported it into Adobe Premiere, a video-editing software package, along with each of the animation movies. Premiere was able to composite each of the movies with the timeline by placing the animation movie into the blue screened section of the timeline movie (a process called Chromakey). I saved all of the composited movies in preparation for their inclusion in the final combined interface.

Creating the Combined Interface

It was necessary to develop an interface, or a common program, which would make viewing both animations together a straightforward experience. My goal in designing the interface was to create a single screen from which a viewer could select either the two-dimensional animation or the three-dimensional animation to view. After viewing the animation, perhaps multiple times, the user should find it easy to view the other companion animation. I wanted the animations to run using the full screen of the user's computer, "hiding" any background desktop images or screen menus, so the full focus of the user was directed at the animations. I also wanted the interface to run on both Macintosh and PC-compatible computers.

I chose to create the final interface in Director. Because I had assembled my two-dimensional animation in Director and saved it as a Director movie, it was a simple task to create a link to that movie from my interface. For the three-dimensional animations, I created another Director movie and imported the composited Quicktime movies that made up all of the views of the three-dimensional animation. Using a screen-shot of the blank three-dimensional U.S. surface, I developed an index page that allows the viewer to select a region of the animation (or the "Full U.S.A") he or she wishes to view.

To provide a logical transition between the full size map and the regional perspective – and to aid the viewer's sense of orientation – I rendered a series of "fly-in" animations in which the camera appears to "zoom-in" on the selected region. I felt this step was especially important because the orientation of the regional views seldom put North toward the top of the screen where a user might intuitively expect to find it. The

fly-ins are invoked by an "action" (a pre-scripted behavior for cast members in Director) as soon as the user selects a colored region of the map to view.

Once the fly-in has brought the camera (and the user) to the preset perspective, the composited Quicktime movie of the animation begins playback. I created a horizontal legend bar and simple playback controls for this section (a button that pauses the animation, a play button, and a button that takes the user back to the index screen) to aid user comprehension and facilitate some user interaction. After reaching the end of the animation (1990) the Quicktime movie automatically loops back to the start and begins again. To return to the index screen, the user clicks the right arrow at the end of the legend.

I felt it was important to provide feedback to the user throughout the interface. Every button changes appearance slightly when passed over by the cursor, indicating that selecting that object or text will have some action associated with it. And I attempted to make those actions intuitive; hence, the "?" button links the user to an explanation of the interface, and the "2D" button links the user to the two-dimensional animation. The appearance changes are achieved using two versions of each button: a "normal state" button (or the way the button will appear on the stage without any interaction) and an "on" state. Using a pre-defined script from Director's "Behaviors Library," I instructed Director to switch buttons from the normal state to the "on" state whenever a user passed the cursor over the button (called a "rollover" event). A similar behavior defines what happens after a button is depressed (an "onMouseDown" event) – which, in the case of my animations, typically meant sending the user to another screen within the

animation. This feature could also be used to link to external sources of information, including information contained on the World Wide Web.

Once assembled, the interface movie and all of the supporting files (the Quicktime movies) where placed in a folder and a "Projector" was created. A projector is a stand-alone program designed to "play" Director movies. Using a version of Director on the Macintosh, I created a projector that can play the animations on any Macintosh computer provided it meets the minimum hardware requirements (see Appendix F for the minimum hardware requirements). Moving the cross-platform Director movies to a PC-compatible computer allowed me to create a Windows version of the projector. Switching platforms to create the two different projectors was necessary because Director does not allow the creation of PC projectors on Macintosh systems or vice-versa. With both projectors completed, I moved all the files back onto the Macintosh in order to record them onto a CD-ROM.

Distributing the Animations

The animations, with all supporting files (including a "README.TXT" help file and an informal questionnaire), required nearly 420 MB of disk storage space – a fact that helped rule out several possible distribution methods. Floppy disks, with their 1.4-MB storage capacity, were obviously not an acceptable solution (the project would have required 300 floppy disks). Distribution over the Internet was precluded by the size of the project as well; not because I lacked the server space required to store the file, but by the realization that asking potential users of the software to patiently download 420 MB of information was not practical. Digital Versatile Discs (DVDs), which meet the storage

capacity and data access needs of the animations, do not yet enjoy widespread enough use to justify employing them. Instead, the ubiquitous CD-ROM, with a storage capacity of 650 MB, seemed the obvious choice. Three important factors supported that decision:

1. A majority of computer users now have access to a computer with a CD-ROM drive (http://educationreview.org)
2. CD-ROMs can be used cross-platform (if recorded for that purpose)
3. The Michigan State University Center for Remote Sensing and Geographic Information Science had a CD-ROM recorder available for my use.

Chapter 3

EVALUATING THE ANIMATIONS

Subjective Observations:

In his essay on visualization, MacEachren notes the expanded perspective on map use presented by geographic visualizations (and, by extension, cartographic animations). There are, he argues, three facets of change in map use:

1. In goals of map use (a shift from information retrieval toward information exploration)
2. In target audience for use (a shift from general public towards individuals)
3. In flexibility of use (from inflexible static maps toward highly manipulable dynamic maps). (MacEachren 1998)

I believe these animations demonstrate that these shifts need not be exclusionary – that is, these animations exhibit a blend of each of the attributes MacEachren describes.

In terms of information exploration, one of the original research goals of this project was to see whether the observations Turner noted about the American frontier's westward movement and eventual disappearance in 1890 were observable using these new techniques. In both the two-dimensional and three-dimensional animations, I believe this phenomenon is obvious, even to the casual observer. At nearly every occasion I have had the opportunity to show the animations, the viewers – whether they be fellow geographers, other academics, or non-scholarly observers – almost without exception comment on the appearance of the westward movement of population. I believe the animations succeed at being illustrative of this particular phenomenon.

At the same time, viewers of both the two-dimensional and three-dimensional maps typically bring some of their own specific knowledge to bear on the animations. Amateur genealogists, for example, have used the animations to see whether their ancestors may have been part of the first wave of migration into a particular area. Transportation geographers have viewed the animations to explore the relationships between the building of transportation routes across the country and the growth in population (and have provided suggestions for further research such as the overlaying of railroads and interstates as they are constructed throughout the animation period). The animations seem to be a tool for exploration and discovery as much as they are a demonstration of a particular phenomenon. They also appear to have value for those not necessarily schooled in geography or cartography (the amateur genealogist) as well as those for whom cartography is a specialty (the transportation geographer).

There are a number of phenomena that I did not set out to represent that are nonetheless apparent in these animations. In both animations, the effect of the California Gold Rush of 1849 is particularly apparent as population densities in San Francisco and the counties to the east (covering a portion of the Sierra Nevada mountain range) grow rather dramatically from about 1840 through 1860. The growth of the cities of Salt Lake City, Utah, and Denver, Colorado are also quite apparent – rising, as they do, from the relatively unpopulated counties surrounding them.

In the two-dimensional animation, some slightly subtler phenomena are visible. The effect of the Appalachian Mountains on westward movement is apparent as the population densities seem to stream around the mountain range and down the Ohio River valley, headed west. The increase in population density in the counties adjacent to the

Erie Canal as it is constructed in upstate New York is also quite apparent. And, with the county boundaries turned on, it becomes clear that the population seems to flood across the Midwest and the Great Plains following the Mississippi and the Missouri rivers (because the counties along the rivers use the rivers as one boundary, it is easier to visualize the rivers on the maps when the county boundaries are visible).

The three-dimensional animation emphasizes different aspects of population migration. In particular, counties that exhibit a reduction in population density are much more apparent. Because the movement of the prisms that represent the population density values generally trend upward throughout the animation, those that move downward draw the eye's attention. This proved to be an unexpected benefit of this type of animation. Errors in the data set – which are inevitable given the size of the data set (20 decades of data representing up to 3100 counties each decade) – are immediately apparent. Any county that exhibits a great change in density from one decade to the next due to an error in the data set – usually because of a lack of data for a particular decade – is noticeable because of the quick movement of its prism. While this error would also occur in the two-dimensional animation, the change in color that results is not as apparent as the changing movement of a county prism in the three-dimensional animation. This is not to suggest that the creation of this type of animation is a suitable method of error checking data sets. There are many other methods of error checking not nearly so labor and time intensive as creating a three-dimensional animation. Nevertheless, I was able to subsequently find many of the errors in my data set after viewing the three-dimensional animations (though that error checked data set has not yet been used for another animation).

The three-dimensional animation also seemed effective for noting the growth of cities – though not as effective as it could have been. In the three-dimensional view of Texas and Oklahoma, the growth of the cities of Dallas, Houston, and Oklahoma City are apparent because they seem to sprout from the relatively unpopulated plains surrounding them. In the New England and Middle Atlantic views, however, the growth of cities becomes less apparent because the entire regions experience so much concurrent population growth. Because the steps of my grayscale height maps used linear intervals rather than exponential intervals for higher populations (limited by Bryce3D's ability to differentiate between shades of gray and map them to different intervals), each exponential interval in the classing system is marked by a linear change in the prism height. Using an exponential change for the prism heights would have resulted in cities like New York literally towering over their surrounding areas, as they should given their relative population densities.

Objective Observations:

In an attempt to obtain an objective evaluation, I provided copies of the animations on CD-ROM to 34 professional cartographers. Forming an expert panel, the professional cartographers were chosen based on their experience in automated cartographic techniques and the belief that they would be able to offer thoughtful commentary in the allotted time frame. Due to the fact that all of the panel members are acquaintances of my advisor, Dr. Richard Groop, and/or myself, their opinions may not have been purely objective. The decision to use an expert panel, instead of a random or anonymous subject group, stemmed from the desire to obtain an educated evaluation of

the two methods of visualization. An expert panel could review my animations in the framework of previously created animations and could analyze them from a traditional cartographic standpoint. A random subject group would not necessarily be familiar with the current level of animations in Cartography or be trained in cartographic presentation guidelines.

Panel members were first contacted via email to ask their willingness to take part in the research and offer some informal commentary. Out of 35 inquiries, 34 accepted the invitation and were sent the animations and questionnaire. Of the 34 who received the animations, 25 returned the questionnaire I included on the CD-ROM. A copy of the questionnaire is included in Appendix H.

The questions on the questionnaire were open-ended with the aim of facilitating candid comments. The panel members were given only basic information about the project – data set used, base map, interpolation method, etc. – in the hope of getting honest reactions to the animations, not feedback on development techniques.

The questionnaire began with five questions designed to ascertain each panel member's background in cartographic animation and equipment used for playback. Of the 25 responses, 20 members used PC-compatible systems and only 5 used Macintosh computers. A handful tried the software on a series of different machines, both to test differences in displays and speeds and to improve playback of all components.

The range of hardware and software configurations used by the panel members helped highlight an incompatibility with the three-dimensional animations and some systems – an incompatibility of which I was not aware and that had not shown up in any of my pre-release testing. Windows 95, 98 or NT users with Quicktime version 3.0

installed noticed that the three-dimensional animations would not play or even appear on-screen. This problem was traced to a conflict between Quicktime 3 on the Windows platform and Director version 6.0. In a follow-up e-mail to all participants, I explained two possible "workarounds" for the problem, detailed in Appendix G, which were successful in helping all of the respondents who indicated problems with the three-dimensional animation view the animation.

Table 1. Reviewer's ranking of personal experience level with cartographic animations.

Experience Level with Cartographic Animations	1 None	2 Little	3 Fair	4 Good deal	5 Expert
Total in each category	-	6	7	8	2

Table 2. Reviewer's opinion of cartographic animations as illustration devices.

Are cartographic animations good illustration devices?	Experience Level with Cartographic Animation					Total
	1 None	2 Little	3 Fair	4 Good deal	5 Expert	
Yes	-	4	6	6	1	18
No	-	1	-	-	-	1
Maybe	-	1	1	2	1	6

Reviewers were asked to rank their experience level with cartographic animations, including experience creating them, analyzing them, or critiquing them. On a scale from one to, six chose level 2, seven chose level 3, eight chose level 4, and two chose the highest level (Table 1). No reviewer lacked previous experience with animations.

When asked their opinion as to whether or not cartographic animations are good

illustration devices, 18 strongly agreed and one reviewer disagreed (Table 2). Six reviewers hesitated to stamp all animations "good illustration devices" since they believed many factors are needed for effective presentations. "It depends on what you are trying to illustrate and how the illustration is done," one wrote. Another reviewer felt "some portion of mapping will probably lend themselves to [cartographic animation] but not the others." The one negative answer, from a reviewer who indicated little previous experience with cartographic animations, generally found animations to be "in the cute category...fun to look at but difficult to extract specific information." The reviewer did feel, however, that "they have potential."

Two-dimensional Animation:

The second section of the questionnaire focused on the two-dimensional animation. The panel members were asked to express their opinions about the animation, with suggestions for possible discussion (colors, classes, speed, do they find them interesting?) (Table 3). The majority of the respondents addressed these suggested topics.

The choice of colors in the animation received quite a bit of commentary. Thirteen people stated that the colors were effective and pleasing. Nine people, however, had concerns with one or two of the colors in the range. The highest value, the lowest value, and several in between were considered either too similar to their neighbor or too different to be easily distinguished. One reviewer noted that the colors shifted on different computer systems – a fact that could explain the problems with different colors. (This color shifting most likely occurred because the respondent viewed the animation on a computer system capable of displaying only 256 colors. The animations were designed

Table 3. Summary of impressions regarding two-dimensional animation.

Impressions of Two-dimensional Animation		Number of Comments
Classes	Too many classes	5
	Good number of classes	3
	Want user to be able to pick from a selection of classes	2
Colors	Pleasing colors	13
	Dislike one or two colors in legend	9
	Lack of color differentiation	3
	Colors should cover less of the spectrum	3
	Color of "no data" class and first class too similar	1
Speed	Good speed	6
	Want control of speed	4
	Too fast	3
	Too slow	1
Timeline	Want more timeline control	4
	Liked being able to click decade by decade	2
General	Want indication of when data becomes available for a county	4
	Want regional close-ups	2
	Would like to see a continuous instead of stepped surface	2
	Want to see major city names and locations	1
Overall	Effective/liked best	6
	Interesting	3

for monitors capable of displaying 1000s of colors – the resulting re-mapping of colors on the lower resolution monitors may have made some colors appear to shift). Three reviewers felt that the colors should cover less of the spectrum, perhaps sticking to one or two hues. But another reviewer thought the "spectral colors work better with animation than if this were a static choropleth map."

Classes also were discussed extensively but not with agreement. Five people felt there were too many classes, but three specifically stated they liked the large number. Two respondents offered a compromise idea of allowing the user to pick a legend from a small selection of possible legends, giving more control to the user as well as revealing

more about the nature of the data. Overall, the panel members seemed, as one responded, to "appreciate the difficulty of making a legend that must accommodate such skewed data and that spans such a range of values over time."

Table 4. Viewing preference for county outlines on two-dimensional animation.

Do you prefer county outlines "on" or "off"?	On	Off	Both
	4	16	4

In the interface, users have the option of turning black county boundaries "on" or "off". When asked if there was a preference, the majority chose to turn them off (Table 4). One person "wanted to see them, but the scale of the map kept [them] from doing so." Another felt they "distracted from [their] impression of the population distribution pattern." Four reviewers preferred to leave them on, though a few wanted to change the black lines to a lighter gray (or even let the users determine the level of gray). Four other reviewers liked the county boundary lines both on and off, depending on the use. "For specific patterns or values for a particular county or region, they help a great deal. For general trends I prefer them off."

Overall, the two-dimensional animation was considered "very effective" and particularly interesting.

Three-dimensional Animation:

The third section of the questionnaire focused on the three-dimensional animation, which received a much wider range of reaction (Table 5). Several panel members did not

Table 5. Summary of impressions regarding three-dimensional animation.

Impressions of Three-dimensional Animation		Number of Comments
Prisms	Black sides dominated	8
	Want smoother prism edges	2
	Want to see prisms without choropleth drape	1
Interface	Want to control and/or change viewing angle and/or vertical exaggeration	7
	Want to turn shadows off	6
	Want to turn off boundaries	2
General	Could see population "ups" and "downs" better	5
	Liked "fly-bys"	1
	Would like to see a continuous instead of stepped surface	1
Timeline	Want control of timeline	2
Overall	Not as effective/liked less than 2D animation	7
	Difficult to follow	5
	Visually more interesting	3

like the three-dimensional animation finding it difficult to follow and less effective than the two-dimensional animation. There were, however, several members that felt they could see changes in the population density values better in the three-dimensional animation and found it visually more appealing. The main problems cited by respondents involved the dominance of black in the sides of the county prisms. The shadows were also cited as distracting to many viewers. Seven of the respondents wished for some user control of the vertical exaggeration of the animation.

When asked if the close-up, regional animations enhanced the three-dimensional view of the full contiguous U.S.A., the reviewers overwhelmingly agreed (Table 6). However, many noted that the effectiveness of the different close-up movies were

"inconsistent." The dense regions, such as New England and the Middle Atlantic states, provided little usable information because the counties grew so quickly into a forest of multicolored prisms. Other views, like the West Coast, as one respondent wrote, were "very effective because the contrasts were very apparent."

Table 6. Reviewer's opinion of "close-up" animations.

Do the "close-up" animations enhance the full USA map animation?		Number of Comments
Overall	Yes they enhance	16
	Some "close-ups" were very effective, others were not	3
	Provide independent view, but do not enhance full USA	2
	No	2
Comments	"Close-up" became too busy	4
	Want to identify counties in dense regions	3
	Need closer view for denser regions	2
	Want timeline control	2
	Want view angle control	1
	Pixelization distracting	1

Another question asked whether or not any differences were apparent between the close-up animations since some were developed using a slightly different development technique. The aim of the question was to see if the different compression codec effects were noticeable. Although nine questionnaires noted differences between the movies, only one reviewer specifically mentioned the effect of the compression-loss. "It appeared that some of the animations, such as the Great Plains and the Pacific Northwest 'shimmered'. These displays were not as crisp as the others were, such as the Great Lakes and Californian close ups." The others noticed minor differences between some animations.

Comparing Two and Three-dimensional Animations:

With opinions of both animations established, the fourth section asked for comparisons (Table 7). First, the reviewer was asked, "which of the two animation types was most effective at portraying U.S. population density change?" Seventy-two percent felt the two-dimensional animation was the most effective. Although the three-dimensional animation was "more engaging" it was apparent that most respondents felt it was more difficult to read. The "three-dimensional animation reveals variations not so apparent in the 2D, but at the cost of geographic context," one wrote. When asked if the full three-dimensional view of the contiguous U.S.A., minus close-up animations, was "more effective" than the two-dimensional animation, the majority disagreed. Only two people liked the full three-dimensional view of the contiguous U.S.A. over the two-dimensional view. When asked if the close-up animations, minus the full three-dimensional view of the contiguous U.S.A., were "more effective" than the two-dimensional animation, the same number disagreed. Three people felt it would be necessary to view close-ups of the two-dimensional animation in order to make an even comparison with the three-dimensional close-ups.

When asked what changes to the animations they would make, the reviewers came up with an extensive list (Table 8). Many suggested cosmetic changes – adjusting legend and outline colors, trying continuous shading, using a different color for areas not yet covered by census data, creating crisper prisms. Several wanted more user control – the ability to pick out regions, rollover counties and see their name and value, zoom in on the two-dimensional animation, and pick their own viewing angle. Interface changes were also desired – more interaction, more timeline choices including the ability to run animations

Table 7. Comparison of two and three-dimensional animations.

Which of the two animation types is most effective at portraying U.S. population density change?		Number of Comments
Overall	2D animation most effective 3D animation most effective Depends on the use	18 4 2
Comments on 2D animation	Better for specific population change Shows westward expansion Simpler and more effective	2 1 1
Comments on 3D animation	Better for overall trends and overall relative density More engaging, but difficult to read Gets better the more it is viewed Good adjunct, but should not stand alone Reveals variations not so apparent in the 2D, but at the cost of geographic context	2 1 1 1 1
Is the 3D full USA animation, minus "close-up" animations, more effective than the 2D animation at portraying U.S. population density change?		
Overall	No Depends on use Yes	17 3 2
Comments	Problems with vertical exaggeration Growing prisms detract from seeing the patterns Prisms look like point symbols 3D better than 2D, but "close-ups" better than 3D	1 1 1 1
Are the "close-up" animations, minus the full USA animation, more effective than the 2D animation at portraying U.S. population density change?		
Overall	No Yes Want to see 2D "close-ups" to judge 3D "close-ups" Equal	17 6 3 1
Comments	Lost spatial context with dense "close-up" regions Too hard to identify counties in "close-ups" Can see decennial population "drops" better in the "close-ups"	1 1 1

Table 8. Summary of suggested improvements.

Suggestions for improvement.		Number of Comments
Interface	Want a more interactive interface	6
	Audio track for decades	3
	Be able to run animations backwards	2
	Want to query counties	2
	Want choice of view angle	2
	Want to zoom in on 2D	2
	User defined viewing regions	1
	Adjust animation size to monitor resolution	1
	Smaller full USA animations for greater pattern recognition and then more detailed zooms	1
Design	Separate timeline from legend	3
	Crisper prisms	3
	Less shadows in 3D	3
	Add transitions to 2D	1
	Closer views in dense regions	1
	Multiple legend designs for different uses	1
	Linear timeline	1
	Make population drops more visible	1
Colors	Try a different color scheme	2
	Change colors only when a certain value is reached	2
	Continuous shading instead of stepped	2
	Light colors mean less and darker colors mean more	1
	Prisms with gray/lighter sides	1
	Try 3D in monotone with just prism height movement	1
	Use height in 3D to show density and use color to show rate of change	1
Data	Want information when counties/states form	1
	Raw population values instead of density	1
	Try a different density calculation	1
	Accompanying histograms that update during animation	1
General	Test non-geographers and non-map users/paper maps	2
	Must distribute through web	1

backwards, and adding audio cues for the timeline. A few even recommended testing with non-geographers and/or with paper maps before delving too far into animation modification.

Some of the ideas were very creative. To one respondent, "the 3D maps seem to employ color and height redundantly to represent population density. As it stands, only motion is used to represent change. I recommend representing rate of change with both motion and color. That way rapidly changing counties would be literally highlighted. Let height alone represent population density." Although this might yield an even more complicated visualization, the effect might be dramatic. Another interesting idea was to make the US map "half as big with an eye toward seeing patterns rather than being lured into looking at individual counties. The detailed zoom-in could then be even bigger."

Two reviewers suggested converting the three-dimensional surface from a prism into a continuous surface. They felt the prisms were not the best way to represent the county data. I disagree, on the basis that the choroplethic data is inherently tied to the county unit. Losing the county areas by using a continuous surface would misrepresent the areas around cities like Denver and New Orleans that are considerably denser than their surrounding counties. Instead, a continuous surface would artificially draw up the populations in the areas surrounding the city, like a pole holding up a tent, despite census data to the contrary.

Summary:

As the developer of both the two and three-dimensional animations, my opinions about their relative effectiveness are rather complicated. I feel, as do the overwhelming

majority of the professional reviewers, that the two-dimensional animation is very effective at showing population change and westward expansion, and that the three-dimensional animations are definitely intriguing and show much potential. It would be a relatively simple matter to address some of the difficulties some respondents noted with the three-dimensional animations in future versions. However, though the creation of the three-dimensional animations is not difficult using the method I describe, the time required for each view of the animation to render could limit its usefulness as a tool for researchers trying to explore their data. With the continuing increase in processor speeds – or the porting of this particular program onto a dedicated graphics workstation – it may be possible in the future to use this technique to create near real-time views of a data set. As it stands now, the three-dimensional animation seems to serve its best purpose as a sort of showcase animation – an engaging look at a data set that encourages further exploration. Evidence of this is found in the answers that the respondents gave to the last questions on the questionnaire (Table 9). When asked if they would recommend using the animations for a geography or history class, 24 of 25 professionals said yes. When asked if viewers would find them interesting, regardless of content effectiveness, all 25 reviewers said yes. When asked if they would encourage further development, all 25 again said yes. Clearly the animations were effective, engaging and should be further refined.

Table 9. Summary of overall impressions of two and three-dimensional animations.

Would you recommend using these animations for a geography or history class?		Number of Comments
Overall	Yes No	24 1
Comments	2D much better for learning 3D good for general overall trends	5 1
Regardless of content effectiveness, would viewers find these animations interesting?		
Overall	Yes No	25 -
Comments	Especially the 3D animation with shadows 3D demonstrates a very innovative technique	1 1
Would you encourage further development of these types of animation?		
Overall	Yes No	25 -
Comments	2D has much more potential, 3D maybe 3D has lots of potential, 2D already a proven method	2 2

Chapter 4

CONCLUSIONS

These animations represent a response to the challenge posed by Campbell and Egbert in their 1990 essay, and an attempt to meet the need for a detailed description of the development of a two and three-dimensional animation. As computer hardware and software becomes more powerful and more sophisticated, the creation of these types of animation will inevitably become more common. Therefore, the utility of an explicit description of the process used in creating these animations can help future cartographers (and, increasingly, non-cartographers) develop their own animations by illuminating one possible development path.

Although the development steps outlined in this thesis constitute the best approach I found for creating these types of visualizations, they too will be subject to changes in technology. Therefore, future steps should include investigating new equipment and software in an effort to refine the techniques. New technology also offers the potential for visualizations not feasible or imagined at this time. Only through experimentation will the "state of the art" in visualization be pushed.

The comments received from the reviewers, besides providing instructive criticisms for improving future animations, pointed to an interesting trend. With one exception, all the reviewers agreed that animations in general were, or had the potential to be, good illustration devices. All 25 reviewers also believed that further research into the animation techniques described in this paper merited further research. This is evidence, I believe, of a general acceptance of animation as a technique.

In order to determine the true effectiveness of the two types of animation at portraying population information, formal user testing will be needed in the future. Structured, objective tests will need to be developed and administered to a wide variety of subject groups. It will be interesting to see if non-geographers and/or non-cartographers interpret the visualizations differently than the expert panel. However, the information obtained through the expert panel questionnaire should serve as a good guideline for testing approach and questions.

The development of these animations is also a further step in fulfilling Tobler's, Moellering's, and Thrower's predictions of the future of animations. Indeed, the development of the three-dimensional animation represents a new approach for creating animations. While the actual rendering of the three-dimensional animations was acutely time-intensive, the construction phase was not. With this new technique, any spatio-temporal, choroplethic data that can be mapped in a GIS package (or other mapping software, such as Golden Software's MapViewer), can be rendered as a smooth, interpolated three-dimensional animation, without the need for an expensive graphics workstation or expensive software. As computers become more powerful – and rendering times consequently shorten – this new technique may become more attractive.

However, one hazard of developmental research is the inevitable obsolescence of technique. Even as I was completing this research, new methods for creating these types of animations – new software packages, new hardware capabilities, new presentation technologies – were already being introduced. One assumes that the ability to animate GIS data will become a standard function of ArcView, or some other GIS software, in the very near future. The need to employ multiple pieces of software in the development of this

TR

research may very well be eliminated (a situation I plan to applaud). But the value in studying the development path I chose will not, I hope, similarly fade into obsolescence. As Moellering's description of his workflow (1980a) in creating the first three-dimensional map of U.S. population densities proved invaluable to me, I, in turn, hope my research and process will prove valuable to others.

TR

APPENDICES

APPENDIX A

SOURCE CODE FOR "CONVERT-O-RAMA"

```
// Code written by Evan Harsha and Jill K. Hallden
// copyright 1996

#include <Types.h>
#include <Memory.h>
#include <Quickdraw.h>
#include <Fonts.h>
#include <Events.h>
#include <Menus.h>
#include <Windows.h>
#include <TextEdit.h>
#include <Dialogs.h>
#include <OSUtils.h>
#include <ToolUtils.h>
#include <SegLoad.h>
#include <stdio.h>
#include <string.h>
#include <Files.h>

/* Constants */
#define EOL 13                  // line delimiter is CR (ASCII 13)
#define MAXLINE 256             // caller's buffer size; max line len

/* Globals */
Rect    windRect;
Str63   buf;
enum
{
        dacen790 = 1, dacen800, dacen810, dacen820, dacen830, dacen840, dacen850,
dacen860, dacen870, dacen880, dacen890,dacen900, dacen910, dacen920, dacen930,
dacen940, dacen950, dacen960, dacen970
};
#define cAlertBox 128

/* Prototypes */
void Initialize(void);
void ParseTheFile(StandardFileReply *theFile);
short GetFileTypeFromName(Str63 name);
short StringToNumber(char * string);
short GetFipsFromState(char * state);
short SearchForString(char * findThis,char *inThis);
short ReadLine(ParamBlockRec pbp, char *lineBuf);
long Convert9ToDecimal(char * buffer, short offset);

short StringToNumber(char * string)
{
        short number;
```

```
        number = 100 * (string[0] - 48) + 10 * (string[1] -48) + (string[2]-48);
        return number;

}
short GetFileTypeFromName(Str63 name)
{
        char blah[4];
        long i;
        short number;

        for(i=0;i<4;i++) blah[i] = 0;

        if(name[1] != 'd')
        {
                ParamText("\pYo, file ain't a dacen file",nil,nil,nil);
                Alert(cAlertBox,nil);
                ExitToShell();
        }
        for(i=0;i<3;i++)
                blah[i] = name[i+6];

        number = StringToNumber(blah);
        switch(number)
        {
                case 790:
                        return dacen790;
                break;
                case 800:
                        return dacen800;
                break;
                case 810:
                        return dacen810;
                break;
                case 820:
                        return dacen820;
                break;
                case 830:
                        return dacen830;
                break;
                case 840:
                        return dacen840;
                break;
                case 850:
                        return dacen850;
                break;
                case 860:
                        return dacen860;
                break;
                case 870:
                        return dacen870;
                break;
                case 880:
                        return dacen880;
```

```
                break;
                case 890:
                        return dacen890;
                break;
                case 900:
                        return dacen900;
                break;
                case 910:
                        return dacen910;
                break;
                case 920:
                        return dacen920;
                break;
                case 930:
                        return dacen930;
                break;
                case 940:
                        return dacen940;
                break;
                case 950:
                        return dacen950;
                break;
                case 960:
                        return dacen960;
                break;
                case 970:
                        return dacen970;
                break;
        }
}
short GetFipsFromState(char * state)
{
        if(strcmp(state,"ALABAMA")== 0)
                return 1;
        if(strcmp(state,"ALASKA")== 0)
                return 2;
        if(strcmp(state,"ARIZONA")== 0)
                return 4;
        if(strcmp(state,"ARKANSAS")== 0)
                return 5;
        if(strcmp(state,"CALIFORNIA")== 0)
                return 6;
        if(strcmp(state,"COLORADO")== 0)
                return 8;
        if(strcmp(state,"CONNECTICUT")== 0)
                return 9;
        if(strcmp(state,"DELAWARE")== 0)
                return 10;
        if(strcmp(state,"DISTRICT OF COLUMBIA")== 0)
                return 11;
        if(strcmp(state,"FLORIDA")== 0)
                return 12;
        if(strcmp(state,"GEORGIA")== 0)
                return 13;
```

```
if(strcmp(state,"HAWAII")== 0)
        return 14;
if(strcmp(state,"IDAHO")== 0)
        return 16;
if(strcmp(state,"ILLINOIS")== 0)
        return 17;
if(strcmp(state,"INDIANA")== 0)
        return 18;
if(strcmp(state,"IOWA")== 0)
        return 19;
if(strcmp(state,"KANSAS")== 0)
        return 20;
if(strcmp(state,"KENTUCKY")== 0)
        return 21;
if(strcmp(state,"LOUISIANA")== 0)
        return 22;
if(strcmp(state,"MAINE")== 0)
        return 23;
if(strcmp(state,"MARYLAND")== 0)
        return 24;
if(strcmp(state,"MASSACHUSETTS")== 0)
        return 25;
if(strcmp(state,"MICHIGAN")== 0)
        return 26;
if(strcmp(state,"MINNESOTA")== 0)
        return 27;
if(strcmp(state,"MISSISSIPPI")== 0)
        return 28;
if(strcmp(state,"MISSOURI")== 0)
        return 29;
if(strcmp(state,"MONTANA")== 0)
        return 30;
if(strcmp(state,"NEBRASKA")== 0)
        return 31;
if(strcmp(state,"NEVADA")== 0)
        return 32;
if(strcmp(state,"NEW HAMPSHIRE")== 0)
        return 33;
if(strcmp(state,"NEW JERSEY")== 0)
        return 34;
if(strcmp(state,"NEW MEXICO")== 0)
        return 35;
if(strcmp(state,"NEW YORK")== 0)
        return 36;
if(strcmp(state,"NORTH CAROLINA")== 0)
        return 37;
if(strcmp(state,"NORTH DAKOTA")== 0)
        return 38;
if(strcmp(state,"OHIO")== 0)
        return 39;
if(strcmp(state,"OKLAHOMA")== 0)
        return 40;
if(strcmp(state,"OREGON")== 0)
        return 41;
```

```
        if(strcmp(state,"PENNSYLVANIA")== 0)
                return 42;
        if(strcmp(state,"RHODE ISLAND")== 0)
                return 44;
        if(strcmp(state,"SOUTH CAROLINA")== 0)
                return 45;
        if(strcmp(state,"SOUTH DAKOTA")== 0)
                return 46;
        if(strcmp(state,"TENNESSEE")== 0)
                return 47;
        if(strcmp(state,"TEXAS")== 0)
                return 48;
        if(strcmp(state,"UTAH")== 0)
                return 49;
        if(strcmp(state,"VERMONT")== 0)
                return 50;
        if(strcmp(state,"VIRGINIA")== 0)
                return 51;
        if(strcmp(state,"WASHINGTON")== 0)
                return 53;
        if(strcmp(state,"WEST VIRGINIA")== 0)
                return 54;
        if(strcmp(state,"WISCONSIN")== 0)
                return 55;
        if(strcmp(state,"WYOMING")== 0)
                return 56;
}
short SearchForString(char * findThis,char *inThis)
{
        short   i,j;
        short   length = strlen(inThis);
        short   length2 = strlen(findThis);
        short   count;
        for(i=0;i<length;i+=length2)
        {
                count = 0;
                for(j=0;j<length2;j++)
                {
                        if(findThis[j] == inThis[i+j])
                                count++;
                }
                if(count == length2)
                        return i;
        }
        return 0;
}

short ReadLine(ParamBlockRec pbp, char *lineBuf)
{
        ParamBlockRec        *pb;
        short   len, rc;

        pb = &pbp;                              // make copy of caller's param block
```

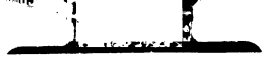
```
        pb->ioParam.ioPosMode = fsAtMark | 0x80 | (256*EOL);
        pb->ioParam.ioReqCount = MAXLINE;       // max line size
        pb->ioParam.ioBuffer = lineBuf;         // transfer to this address

        rc=PBRead( pb, FALSE );     // read one line

        if (rc==eofErr && pb->ioParam.ioActCount==0)
                return ( eofErr );              // end of file reached

        if ((rc==noErr) || (rc==eofErr)) {
                len = pb->ioParam.ioActCount;
                if (len==MAXLINE) return(-1);   // not a delimited file
                if (lineBuf[len-1] != EOL) len++;   // last line has no EOL
                lineBuf[len-1]=0;               // convert to ASCIIZ
        }
        return (rc);
}
long Convert9ToDecimal(char * buffer, short offset)
{
        long value;

        value =     (buffer[offset] - 48) * 100000000;
        value +=  (buffer[offset+1] - 48) *  10000000;
        value +=  (buffer[offset+2] - 48) *   1000000;
        value +=  (buffer[offset+3] - 48) *    100000;
        value +=  (buffer[offset+4] - 48) *     10000;
        value +=  (buffer[offset+5] - 48) *      1000;
        value +=  (buffer[offset+6] - 48) *       100;
        value +=  (buffer[offset+7] - 48) *        10;
        value +=  (buffer[offset+8] - 48) *         1;

        return value;
}
void ParseTheFile(StandardFileReply *theFile)
{
        short   refNum, newFileRefNum;
        short   fileType = GetFileTypeFromName(theFile->sfFile.name);
        long    count;
        char    buffer[100];
        char    buf2[100];
        char    buf3[100];
        Str63   fileName;
        FSSpec theNewFile = theFile->sfFile;
        short   currentFips;
        OSErr   err;
        short   location;
        short lines;
        ParamBlockRec pbp;

        for(count = 0;count < 63;count++)
                fileName[count] =0;

        BlockMove(theFile->sfFile.name,fileName,strlen((char *)theFile->sfFile.name)-3);
        strcat((char *)fileName,".NEW");
```

```
BlockMove(fileName+1,theNewFile.name+1,63);
theNewFile.name[0] = strlen((char *)fileName)-1;

err = FSpCreate(&theNewFile,'TEXT','TEXT',smSystemScript);
if(err)
{
        ParamText("\pError Creating .NEW file",nil,nil,nil);
        Alert(cAlertBox,nil);
        return;
}
err = FSpOpenDF(&theNewFile,fsCurPerm,&newFileRefNum);
if(err)
{
        ParamText("\pError Opening dacenFile.NEW file",nil,nil,nil);
        Alert(cAlertBox,nil);
        return;
}
err = FSpOpenDF(&theFile->sfFile,fsCurPerm,&refNum);
if(err)
{
        ParamText("\pError Opening dacenFile ",nil,nil,nil);
        Alert(cAlertBox,nil);
        return;
}

pbp.ioParam.ioRefNum = refNum;

count = 80;
location = 0;
err = 0;
currentFips = 0;
lines = 0;
while(1)
{
        err = ReadLine(pbp, buffer);
        lines++;
        // are we done?
        if(err == -1 || err == eofErr)
                break;
        // we got a line....gotta see what it is
        if(strlen(buffer) ==0)
                continue;
        // check to see if it is a state line...
        if(buffer[0] == '0' && buffer[1] == '1' && buffer[2] == 'S')
        {
                short i,j=0;
                char state[20];
                for(i=0;i<20;i++)
                        state[i] = 0;

                i = 14;
                while(1)
                {
                        if(buffer[i] == ' ' && buffer[i+1] == ' ')
```

```
                                        break;
                                state[j] = buffer[i];
                                i++;
                                j++;
                        }
                        currentFips = GetFipsFromState(state);
                }
                else
                {
                        switch(fileType)
                        {
// 790
// 870
// 950
                        case dacen790:
                        case dacen870:
                        case dacen950:
                        if(buffer[0] == '0' && buffer[1] == '1' && buffer[2] == 'C')
                        {
                                short i,j=0;
                                char county[20];
                                long temp =0;

                                for(i=0;i<20;i++)
                                        county[i] = 0;
                                i=14;
                                while(buffer[i] != '0')
                                {
                                        county[j] = buffer[i];
                                        i++;
                                        j++;
                                }
                                if(currentFips > 9)
                                        sprintf((char *)buf2,"%d",currentFips);
                                else
                                        sprintf((char *)buf2,"0%d",currentFips);

                                sprintf((char *) buf3,
"%c%c%c\t%s\t%c%c%c%c%c%c%c%c%c\n",
                                                                buffer[10],
                                                                buffer[11],
                                                                buffer[12],
                                                                county,
                                                                buffer[32],
                                                                buffer[33],
                                                                buffer[34],
                                                                buffer[35],
                                                                buffer[36],
                                                                buffer[37],
                                                                buffer[38],
                                                                buffer[39],
                                                                buffer[40]);
                                        strcat(buf2,buf3);
                                        temp = strlen(buf2);
```

```
                                FSWrite(newFileRefNum,&temp,buf2);
                        }
                        break;
// 800
// 810
                case dacen800:
                case dacen810:
                if(buffer[0] == '0' && buffer[1] == '3' && buffer[2] == 'C')
                {
                        short i,j=0;
                        char county[20];
                        long temp =0;

                        for(i=0;i<20;i++)
                                county[i] = 0;
                        i=14;
                        while(buffer[i] != '0')
                        {
                                county[j] = buffer[i];
                                i++;
                                j++;
                        }
                        if(currentFips > 9)
                                sprintf((char *)buf2,"%d",currentFips);
                        else
                                sprintf((char *)buf2,"0%d",currentFips);

                        sprintf((char
*)buf3,"%c%c%c\t%s\t%c%c%c%c%c%c%c%c%c\n",
                                                        buffer[10],
                                                        buffer[11],
                                                        buffer[12],
                                                        county,
                                                        buffer[50],
                                                        buffer[51],
                                                        buffer[52],
                                                        buffer[53],
                                                        buffer[54],
                                                        buffer[55],
                                                        buffer[56],
                                                        buffer[57],
                                                        buffer[58]);
                                strcat(buf2,buf3);
                                temp = strlen(buf2);
                                FSWrite(newFileRefNum,&temp,buf2);
                        }
                        break;
// 820
                case dacen820:
                if(buffer[0] == '0' && buffer[1] == '7' && buffer[2] == 'C')
                        {
                                short i,j=0;
                                char county[20];
                                long temp =0;
```

```
                                        for(i=0;i<20;i++)
                                                county[i] = 0;
                                        i=14;
                                        while(buffer[i] != '0')
                                        {
                                                county[j] = buffer[i];
                                                i++;
                                                j++;
                                        }
                                        if(currentFips > 9)
                                                sprintf((char *)buf2,"%d",currentFips);
                                        else
                                        sprintf((char *)buf2,"0%d",currentFips);

                                        sprintf((char
*)buf3,"%c%c%c\t%s\t%c%c%c%c%c%c%c%c%c\n",
                                                                buffer[10],
                                                                buffer[11],
                                                                buffer[12],
                                                                county,
                                                                buffer[50],
                                                                buffer[51],
                                                                buffer[52],
                                                                buffer[53],
                                                                buffer[54],
                                                                buffer[55],
                                                                buffer[56],
                                                                buffer[57],
                                                                buffer[58]);
                                        strcat(buf2,buf3);
                                        temp = strlen(buf2);
                                        FSWrite(newFileRefNum,&temp,buf2);
                                }
                                break;
// 830
// 840
                        case dacen830:
                        case dacen840:
                        if(buffer[0] == '1' && buffer[1] == '1' && buffer[2] == 'C')
                                {
                                        short i,j=0;
                                        char county[20];
                                        long temp =0;

                                        for(i=0;i<20;i++)
                                                county[i] = 0;
                                        i=14;
                                        while(buffer[i] != '0')
                                        {
                                                county[j] = buffer[i];
                                                i++;
                                                j++;
                                        }
```



```
                                        if(currentFips > 9)
                                                sprintf((char *)buf2,"%d",currentFips);
                                        else
                                        sprintf((char *)buf2,"0%d",currentFips);

                                        sprintf((char
*)buf3,"%c%c%c\t%s\t%c%c%c%c%c%c%c%c%c\n",
                                                                buffer[10],
                                                                buffer[11],
                                                                buffer[12],
                                                                county,
                                                                buffer[31],
                                                                buffer[32],
                                                                buffer[33],
                                                                buffer[34],
                                                                buffer[35],
                                                                buffer[36],
                                                                buffer[37],
                                                                buffer[38],
                                                                buffer[39]);
                                        strcat(buf2,buf3);
                                        temp = strlen(buf2);
                                        FSWrite(newFileRefNum,&temp,buf2);
                                }
                                break;
// 850
                        case dacen850:
                        if(buffer[0] == '2' && buffer[1] == '0' && buffer[2] == 'C')
                                {
                                        short i,j=0;
                                        char county[20];
                                        long temp =0;

                                        for(i=0;i<20;i++)
                                                county[i] = 0;
                                        i=14;
                                        while(buffer[i] != '0')
                                        {
                                                county[j] = buffer[i];
                                                i++;
                                                j++;
                                        }
                                        if(currentFips > 9)
                                                sprintf((char *)buf2,"%d",currentFips);
                                        else
                                        sprintf((char *)buf2,"0%d",currentFips);

                                        sprintf((char
*)buf3,"%c%c%c\t%s\t%c%c%c%c%c%c%c%c%c\n",
                                                                buffer[10],
                                                                buffer[11],
                                                                buffer[12],
                                                                county,
                                                                buffer[68],
```

```
                                                buffer[69],
                                                buffer[70],
                                                buffer[71],
                                                buffer[72],
                                                buffer[73],
                                                buffer[74],
                                                buffer[75],
                                                buffer[76]);
                                strcat(buf2,buf3);
                                temp = strlen(buf2);
                                FSWrite(newFileRefNum,&temp,buf2);
                        }
                        break;

                case dacen860:
                if(buffer[0] == '2' && buffer[1] == '0' && buffer[2] == 'C')
                        {
                                short i,j=0;
                                char county[20];
                                long temp =0;
                                long number1,number2,total;
                                for(i=0;i<20;i++)
                                        county[i] = 0;
                                i=14;
                                while(buffer[i] != '0')
                                {
                                        county[j] = buffer[i];
                                        i++;
                                        j++;
                                }
                                if(currentFips > 9)
                                        sprintf((char *)buf2,"%d",currentFips);
                                else
                                sprintf((char *)buf2,"0%d",currentFips);

                                sprintf((char *)buf3,"%c%c%c\t%s",
//%c%c%c%c%c%c%c%c%c\n",
                                                buffer[10],
                                                buffer[11],
                                                buffer[12],
                                                county);
                                strcat(buf2,buf3);

                                number1 = Convert9ToDecimal(buffer,59);
                                err = ReadLine(pbp, buffer);
                                while(1)
                                {
                        if(buffer[0] == '5' && buffer[1] == '4' && buffer[2] == 'C')
                                                break;
                                        else
                                                err = ReadLine(pbp, buffer);
                                }
                                number2 = Convert9ToDecimal(buffer,59);
                                total = number1 + number2;
```

```
                                sprintf((char *)buf3,"\t%d\n",total);

                                strcat(buf2,buf3);
                                temp = strlen(buf2);
                                FSWrite(newFileRefNum,&temp,buf2);
                        }
                        break;
// 880
                case dacen880:
                if(buffer[0] == '0' && buffer[1] == '1' && buffer[2] == 'C')
                        {
                                short i,j=0;
                                char county[20];
                                long temp =0;
                                long number1,number2,total;
                                for(i=0;i<20;i++)
                                        county[i] = 0;
                                i=14;
                                while(buffer[i] != '0')
                                {
                                        county[j] = buffer[i];
                                        i++;
                                        j++;
                                }
                                if(currentFips > 9)
                                        sprintf((char *)buf2,"%d",currentFips);
                                else
                                sprintf((char *)buf2,"0%d",currentFips);

                                sprintf((char *)buf3,"%c%c%c\t%s",
//%c%c%c%c%c%c%c%c%c\n",
                                                        buffer[10],
                                                        buffer[11],
                                                        buffer[12],
                                                        county);
                                strcat(buf2,buf3);

                                number1 = Convert9ToDecimal(buffer,68);
                                err = ReadLine(pbp, buffer);
                                if(strlen(buffer) == 0)
                                        err = ReadLine(pbp, buffer);
                                number2 = Convert9ToDecimal(buffer,32);
                                total = number1 + number2;
                                sprintf((char *)buf3,"\t%d\n",total);

                                strcat(buf2,buf3);
                                temp = strlen(buf2);
                                FSWrite(newFileRefNum,&temp,buf2);
                        }
                        break;
// 890
                case dacen890:
                if(buffer[0] == '0' && buffer[1] == '5' && buffer[2] == 'C')
                        {
```

```
                                short i,j=0;
                                char county[20];
                                long temp =0;
                                long number1,number2,total;
                                for(i=0;i<20;i++)
                                        county[i] = 0;
                                i=14;
                                while(buffer[i] != '0')
                                {
                                        county[j] = buffer[i];
                                        i++;
                                        j++;
                                }
                                if(currentFips > 9)
                                        sprintf((char *)buf2,"%d",currentFips);
                                else
                                        sprintf((char *)buf2,"0%d",currentFips);
                                sprintf((char *)buf3,"%c%c%c\t%s",
//%c%c%c%c%c%c%c%c%c\n",
                                                        buffer[10],
                                                        buffer[11],
                                                        buffer[12],
                                                        county);
                                strcat(buf2,buf3);

                                number1 = Convert9ToDecimal(buffer,68);
                                err = ReadLine(pbp, buffer);
                                if(strlen(buffer) == 0)
                                        err = ReadLine(pbp, buffer);
                                number2 = Convert9ToDecimal(buffer,32);
                                total = number1 + number2;
                                sprintf((char *)buf3,"\t%d\n",total);

                                strcat(buf2,buf3);
                                temp = strlen(buf2);
                                FSWrite(newFileRefNum,&temp,buf2);
                        }
                        break;
// 900
                case dacen900:
                if(buffer[0] == '1' && buffer[1] == '0' && buffer[2] == 'C')
                        {
                                short i,j=0;
                                char county[20];
                                long temp =0;

                                for(i=0;i<20;i++)
                                        county[i] = 0;
                                i=14;
                                while(buffer[i] != '0')
                                {
                                        county[j] = buffer[i];
                                        i++;
                                        j++;
```

```
                                        }
                                        if(currentFips > 9)
                                                sprintf((char *)buf2,"%d",currentFips);
                                        else
                                        sprintf((char *)buf2,"0%d",currentFips);

                                        sprintf((char
*)buf3,"%c%c%c\t%s\t%c%c%c%c%c%c%c%c%c\n",
                                                                buffer[10],
                                                                buffer[11],
                                                                buffer[12],
                                                                county,
                                                                buffer[41],
                                                                buffer[42],
                                                                buffer[43],
                                                                buffer[44],
                                                                buffer[45],
                                                                buffer[46],
                                                                buffer[47],
                                                                buffer[48],
                                                                buffer[49]);
                                        strcat(buf2,buf3);
                                        temp = strlen(buf2);
                                        FSWrite(newFileRefNum,&temp,buf2);
                                }
                                break;
// 910
                        case dacen910:
                        if(buffer[0] == '0' && buffer[1] == '1' && buffer[2] == 'C')
                                {
                                        short i,j=0;
                                        char county[20];
                                        long temp =0;
                                        long number1,number2,total;
                                        for(i=0;i<20;i++)
                                                county[i] = 0;
                                        i=14;
                                        while(buffer[i] != '0')
                                        {
                                                county[j] = buffer[i];
                                                i++;
                                                j++;
                                        }
                                        if(currentFips > 9)
                                                sprintf((char *)buf2,"%d",currentFips);
                                        else
                                        sprintf((char *)buf2,"0%d",currentFips);

                                        sprintf((char *)buf3,"%c%c%c\t%s",
//%c%c%c%c%c%c%c%c%c\n",
                                                                buffer[10],
                                                                buffer[11],
                                                                buffer[12],
                                                                county);
```

```
                                        strcat(buf2,buf3);

                                        number1 = Convert9ToDecimal(buffer,41);
                                        number2 = Convert9ToDecimal(buffer,59);
                                        total = number1 + number2;
                                        sprintf((char *)buf3,"\t%d\n",total);

                                        strcat(buf2,buf3);
                                        temp = strlen(buf2);
                                        FSWrite(newFileRefNum,&temp,buf2);
                                }
                                break;
// 920
// 930
// 940
                        case dacen920:
                        case dacen930:
                        case dacen940:
                        if(buffer[0] == '0' && buffer[1] == '1' && buffer[2] == 'C')
                                {
                                        short i,j=0;
                                        char county[20];
                                        long temp =0;
                                        long number1,number2,total;
                                        for(i=0;i<20;i++)
                                                county[i] = 0;
                                        i=14;
                                        while(buffer[i] != '0')
                                        {
                                                county[j] = buffer[i];
                                                i++;
                                                j++;
                                        }
                                        if(currentFips > 9)
                                                sprintf((char *)buf2,"%d",currentFips);
                                        else
                                        sprintf((char *)buf2,"0%d",currentFips);

                                        sprintf((char *)buf3,"%c%c%c\t%s",
//%c%c%c%c%c%c%c%c%c\n",
                                                                buffer[10],
                                                                buffer[11],
                                                                buffer[12],
                                                                county);
                                        strcat(buf2,buf3);

                                        number1 = Convert9ToDecimal(buffer,32);
                                        number2 = Convert9ToDecimal(buffer,41);
                                        total = number1 + number2;
                                        sprintf((char *)buf3,"\t%d\n",total);

                                        strcat(buf2,buf3);
                                        temp = strlen(buf2);
                                        FSWrite(newFileRefNum,&temp,buf2);
```

```
                }
                break;

        case dacen960:
        if(buffer[0] == '0' && buffer[1] == '8' && buffer[2] == 'C')
                {
                        short i,j=0;
                        char county[20];
                        long temp =0;
                        long number1,number2,total;
                        for(i=0;i<20;i++)
                                county[i] = 0;
                        i=14;
                        while(buffer[i] != '0')
                        {
                                county[j] = buffer[i];
                                i++;
                                j++;
                        }
                        if(currentFips > 9)
                                sprintf((char *)buf2,"%d",currentFips);
                        else
                        sprintf((char *)buf2,"0%d",currentFips);

                        sprintf((char *)buf3,"%c%c%c\t%s",
//%c%c%c%c%c%c%c%c%c\n",
                                                buffer[10],
                                                buffer[11],
                                                buffer[12],
                                                county);
                        strcat(buf2,buf3);

                number1 = Convert9ToDecimal(buffer,50);
                err = ReadLine(pbp, buffer);
                while(1)
                {
                if(buffer[0] == '1' && buffer[1] == '0' && buffer[2] == 'C')
                                        break;
                                else
                                        err = ReadLine(pbp, buffer);
                        }
                        number2 = Convert9ToDecimal(buffer,32);
                        total = number1 + number2;
                        sprintf((char *)buf3,"\t%d\n",total);

                        strcat(buf2,buf3);
                        temp = strlen(buf2);
                        FSWrite(newFileRefNum,&temp,buf2);
                }
                break;
// 970
        case dacen970:
        if(buffer[0] == '1' && buffer[1] == '0' && buffer[2] == 'C')
                {
```

```
                                        short i,j=0;
                                        char county[20];
                                        long temp =0;

                                        for(i=0;i<20;i++)
                                                county[i] = 0;
                                        i=14;
                                        while(buffer[i] != '0')
                                        {
                                                county[j] = buffer[i];
                                                i++;
                                                j++;
                                        }
                                        if(currentFips > 9)
                                                sprintf((char *)buf2,"%d",currentFips);
                                        else
                                        sprintf((char *)buf2,"0%d",currentFips);

                                        sprintf((char
*)buf3,"%c%c%c\t%s\t%c%c%c%c%c%c%c%c\n",
                                                                buffer[10],
                                                                buffer[11],
                                                                buffer[12],
                                                                county,
                                                                buffer[27],
                                                                buffer[28],
                                                                buffer[29],
                                                                buffer[30],
                                                                buffer[31],
                                                                buffer[32],
                                                                buffer[33],
                                                                buffer[34]);
                                        strcat(buf2,buf3);
                                        temp = strlen(buf2);
                                        FSWrite(newFileRefNum,&temp,buf2);
                                }
                                break;
                        }
                }
        }
        FSClose(refNum);
        FSClose(newFileRefNum);
}
//MW specified argument and return type.
void main(void)
{
        StandardFileReply theRecord;

        Initialize();

        // get the file
        StandardGetFile(nil,-1,nil,&theRecord);

        if(theRecord.sfGood)
```

```
        ParseTheFile(&theRecord);
}

//
//      Initialize everything for the program, make sure we can run
//

//MW specified argument and return type.
void Initialize(void)
{
        WindowPtr       mainPtr;
        OSErr           error;
        SysEnvRec       theWorld;

        //
        //      Test the computer to be sure we can do color.
        //      If not we would crash, which would be bad.
        //      If we can't run, just beep and exit.
        //

        error = SysEnvirons(1, &theWorld);
        if (theWorld.hasColorQD == false) {
                SysBeep(50);
                ExitToShell();                  /* If no color QD, we must leave. */
        }

        /* Initialize all the needed managers. */
        InitGraf(&qd.thePort);
        InitFonts();
        InitWindows();
        InitMenus();
        TEInit();
        InitDialogs(nil);
        InitCursor();

        //
        //To make the Random sequences truly random, we need to make the seed start
        //at a different number.  An easy way to do this is to put the current time
        //and date into the seed.  Since it is always incrementing the starting seed
        //will always be different.  Don't for each call of Random, or the sequence
        //will no longer be random.  Only needed once, here in the init.
        //      GetDateTime((unsigned long*) &qd.randSeed);

        //
        //Make a new window for drawing in, and it must be a color window.
        //The window is full screen size, made smaller to make it more visible.
        //
}
```

APPENDIX B

POPULATION DENSITY INTERPOLATION EQUATION

The equation used to determine the interpolated population density values of the "in-between" years in the animation:

$$population\,density\,value = ((Y - X) \times (increase\,of\,years \times 0.10)) + X$$

Where X = the beginning decade value and Y = the ending decade value

For example, here is the equation used to find the interpolated population density of a sample county in 1792:

$$1792\,population\,density = ((8.0 - 2.0) \times (2 \times 0.10)) + 2.0 = 3.2$$

Where 8.0 and 2.0 represent the population densities of 1800 and 1790, respectively.

APPENDIX C

BREAKDOWN OF ACTION TO ADJUST MAP IMAGES USING ADOBE PHOTOSHOP 5.0

1) Open file.

2) Change canvas size (decrease image size by decreasing the size of the image canvas).

3) Duplicate map layer (make a *background copy* layer).

4) Select *background* layer.

5) Fill *background* layer. (foreground color, 100% opacity, normal mode).

6) Select *background copy* layer.

7) Set selection (tolerance = 0) (when setting action, use magic wand to select white area surrounding U.S. outline).

8) Set Add to selection (while holding down the shift key, use magic wand to select all isolated white pixels within the U.S. boundary.

9) Delete (once all white pixels are selected).

10) Set selection to none (deselect).

11) Exchange colors (bring black from background color to foreground color).

12) Color range (fuzziness 145, default settings) (select black state boundaries).

13) Set foreground color (RGB color, R = 38. B = 38, G = 38) (a lighter black color).

14) Fill (foreground color, opacity 100%, normal mode) (change map boundaries to lighter black color).

15) Set selection to none (deselect).

16) Set foreground color (HSV color, H = 0, S = 0, B=0).

17) Convert mode (index color, adaptive palette, best color matching).

18) Save file in pict format.

19) Close file.

APPENDIX D

CREATING EQUAL INTERVALS OF GREY VALUES USING ADOBE PHOTOSHOP 5.0

1 In Photoshop, open a new document by choosing *New*... from the *File* menu

2 Enter "200 pixels" for Width, "480 pixels" for Height. Resolution "72 pixels per inch". Mode "RGB". Contents "White". Click *OK*.

3 Using the Marquee tool, draw a narrow but as-tall-as-possible rectangular box in the open window.

4 Make black the foreground color and white the background color using the color picker.

5 Select the gradient tool. Set options to "Normal", opacity to "100%", and select "foreground to background" as the gradient setting.

6 Beginning at the bottom or the top of the marqueed rectangle, hold down the Shift key and draw a straight line to the opposite side of the rectangle. A gradient from black, through greys, to white should appear in the rectangle.

7 Select *Image→ Adjust→ Posterize*... from the *Image* menu.

8 Choose the number of intervals required.

9 With the eyedropper tool and the *Info* panel visible (if it is not visible, choose *Window→ Show Info*), passing the eyedropper tool over the gray squares reveals their RGB and CMYK values. These values can then be used to create a legend in ArcView.

APPENDIX E

MINIMUM HARDWARE REQUIREMENTS

PC: Windows 95, 98, or NT
Pentium Processor
16MB RAM
Monitor capable of 800x600 resolution
450MB available space on HD (for best performance)
Quicktime 2.5 or later (available at no charge from http://quicktime.apple.com)

Mac: System 7.1 or later
68040 or PowerPC
16MB RAM
Monitor capable of 800x600 resolution
450MB available space on HD (for best performance)
Quicktime 2.5 or later (available at no charge from http://quicktime.apple.com)

Preferred Systems

These animations are quite large and tend to be very processor intensive. In general, the faster the processor, the smoother the playback. To optimize performance, I suggest copying the "COPY_ME" folder on the CD-ROM to your hard drive. Typically, the data transfer rate of a CD-ROM player is much slower than the average hard drive - a fact that will significantly impact the animation performance. It is possible to run the animation entirely from the CD-ROM, but playback will most likely be more sluggish.

APPENDIX F

CARTOGRAPHIC ANIMATION QUESTIONNAIRE

Jill K. Hallden, Department of Geography, Michigan State University
315 Natural Science Building, East Lansing, MI 48823
halldenj@pilot.msu.edu, http://www.msu.edu/~halldenj
(517) 432-0603, (517) 353-1821 fax
Advisor: Dr. Richard Groop

Preamble:

I have created two different styles of cartographic animation illustrating U.S. population density by county. Both animations use the same historical data set: decennial U.S. Census data, by county, from 1790 to 1990. County boundaries are assumed to have remained constant over the course of the animation. Data for the two-dimensional animation has been linearly interpolated at 2 year intervals. The three-dimensional animation utilizes a mathematical "morphing" technique between decade keyframes to interpolate in-between frames.

The purpose of this questionnaire is to solicit your professional opinion of these two methods of visualization. The questions are somewhat open-ended in the hope of facilitating your candid comments. Feel free to use brief keyword-type phrases rather than extended commentary.

You are welcome to keep the CD and use it for academic purposes. If you would like to show it to your classes, you may, though my goal in sending you this questionnaire is to get your opinion on the work rather than the opinion of your students. I think student evaluation is important, and should be done, but it is beyond the scope of this current work.

On the CD-ROM you will find a digital copy of this questionnaire in Microsoft Word 5.0 format. My hope was to ease the burden of returning the questionnaire by providing a document you could attach to an e-mail message and send to halldenj@pilot.msu.edu. But, if you choose to use this paper copy, please return it to me via fax at (517) 353-1821, if possible. If you ask other professionals to evaluate the animations, please duplicate this questionnaire.

Thank you for your time and your input. (Please use additional sheets, if necessary)

Background Information:

1. Name:_____

2. Viewing equipment (circle):

Platform:	PC-compatible	Macintosh		
Operating system:	Windows95	Windows98	WindowsNT	
	MacOS7.__	MacOS8.__	MacOS8.5	other

Processor (model/speed) ex: Pentium/133_____

Monitor size: 15inch 17inch 19+inch

3. With this hardware configuration, did you notice any performance problems?

_____

_____

4. How would you rank your experience level with cartographic animations (creating, analyzing, critiquing, etc.)? (circle appropriate rank):

1-none 2-little 3-fair 4-good deal 5-expert

5. In your opinion, are cartographic animations good illustration devices?

_____

_____

_____

Two-dimensional animation:

1. What are you impressions of the 2-D animation? (Colors? Classes? Speed? Interesting?) _____

_____

2. Does viewing the animation with county outlines "on" enhance or detract?

_____

_____

Three-dimensional animation:

1. What are your impressions of the 3-D animations? _____

_____

_____

2. Do the "close-up" animations enhance the full map animations? _____

_____

_____

3. Several of the "close-up" animations were created using a slightly different development technique from the rest. Were any differences apparent to you?

_____

_____

Comparison:

1. Which of the two animation types is most effective at portraying U.S. population density change? _____

2. Is the 3-D full USA animation, minus "close-up" animations, more effective than the 2-D animation at portraying U.S. population density change? _____

3. Are the "close-up" animations, minus the full USA animation, more effective than the 2-D animation at portraying U.S. population density change? _____

Overall:

1. Would you recommend using these animations for a geography or history class?

2. Regardless of content effectiveness, would viewers find these animations interesting?

3. Would you encourage further development of these types of animation? _____

4. Do you have any suggestions for improving these animations? _____

APPENDIX G

QUICKTIME 3.0 INCOMPATIBILITY WORKAROUNDS

1) View the three-dimensional animations outside the interface by playing them inside Apple's Movie Player application that shipped with Quicktime 3.0. This was a less than satisfactory solution because the participants would then view the animations in an environment different than the one in which they had viewed the two-dimensional animations, and without certain interface elements such as the animation legends.
2) Download Apple Quicktime version 2.12 from an Apple ftp site (ftp://ftp.apple.com/apps/quicktime/QT32EASY.EXE) and install it. This method fixed the incompatibility without overwriting the participants copy of Quicktime 3.0, and was, in my mind, the optimal solution.

These workarounds were successful in helping all of the respondents who indicated problems with the three-dimensional animation view the animation.

LIST OF REFERENCES

LIST OF REFERENCES

Billington, Ray Allen. 1973. *Fredrick Jackson Turner: Historian, Scholar, Teacher*. New York, NY: Oxford Unversity Press.

Campbell, Craig S. and Stephen L. Egbert. 1990. Animated Cartography: Thirty Years of Scratching the Surface. *Cartographica*, vol. 27, no. 2, pp. 24-46.

Cornwell, Bruce and Arthur H. Robinson. 1966. Possibilities for Computer Animated Films in Cartography. *Cartographic Journal,* vol. 3, no. 2, pp. 79-82

DiBiase, David. 1994. Designing Animated Maps for a Multimedia Encyclopedia. *Cartographic Perspectives*, no. 19, pp. 3-7.

Dorling, Daniel. 1992. Stretching Space and Splicing Time: From Cartographic Animation to Interactive Visualization. *Cartography and Geographic Information Systems*, vol. 19, no. 4, pp. 215-227, 267-70.

Gersmehl, Philip J. 1990. Choosing Tools: Nine Metaphors of Four-Dimensional Cartography. *Cartographic Perspectives*, no. 5, pp. 3-17.

Kane, Joseph N. 1972. *The American Counties.* 3rd edition. Metuchen, NJ: Scarecrow Press.

Koussoulakou, A. and M.J. Kraak. 1992. Spatio-temporal Maps and Cartographic Communication. *Cartographic Journal*, vol. 29, pp. 101-108.

Kraak, M. J. 1993. Cartographic Terrain Modeling in a Three-Dimensional GIS Environment. *Cartography and Geographic Information Systems*, vol. 20, no. 1, pp. 13-18.

Kumler, Mark P. 1988. Mapping Smooth Surfaces with Continuous Tone Maps. Unpublished Masters Thesis, Michigan State University, East Lansing, MI.

Langran, Gail and Nicholas Chrisman. 1988. A Framework for Temporal Geographic Information *Cartographica*, vol. 25, no.3, pp. 1-14.

MacEachren, Alan M. 1998. Visualization: Cartography for the 21st Century. Working paper. International Cartographic Association, Commission on Visualization. http://www.geog.psu.edu/ica/ICAvis.html

MacEachren, Alan M. (in collaboration with Polsky, C., D. Haug, D. Brown., F. Boscoe, J. Beedasy, L. Pickle, and M. Marrara). 1997. Visualizing spatial relationships among health, environmental, and demographic statistics: Interface design issues. Proceedings of the 18th International Cartographic Conference. Stockholm, Sweden. June 23-27.

MacEachren, Alan M. 1994. Visualization in Modern Cartography: Setting the Agenda. In MacEachren, Alan M. and D. R. Fraser Taylor (eds.). *Visualization in Modern Cartography*. Modern Cartography, Volume 2. Oxford, U.K. Pergamon/Elsevier Science Ltd.

MacEachren, Alan M. (in collaboration with Buttenfield, B., J. Campbell, D. DiBiase, and M. Monmonier) 1992. Visualization. In Abler, Ronald F., Melvin G. Marcus and Judy M. Olson (eds.). *Geography's Inner Worlds.* New Brunswick, NJ. Rutgers University Press.

MacEachren, Alan M. and David DiBiase. 1991. Animated Maps of Aggregate Data: Conceptual and Practical Problems. *Cartography and Geographic Information Systems*, vol. 18, no. 4, pp. 221-229.

Moellering, Harold. 1973. The Automated Mapping of Traffic Accidents. *Surveying and Mapping*, vol. 33, no. 4, pp. 467-477.

Moellering, Harold. 1980a. The Real-Time Animation of Three-Dimensional Maps. *American Cartographer*, vol. 7, no.1, pp. 67-75.

Moellering, Harold. 1980b. Strategies of Real-Time Cartography. *The Cartographic Journal*, vol. 17. No. 1, pp. 12-15.

Monmonier, Mark. 1990. Strategies for the Visualization of Geographic Time-Series Data. *Cartographica*, vol. 27, no. 1, pp. 30-45.

Olson, Judy M. 1997. Multimedia in Geography: Good, Bad, Ugly, or Cool? *Annals of the Association of American Geographers*, vol. 87, no. 4, pp. 571-578.

Peterson, Michael P. 1994. Cognitive Issues in Cartographic Visualization. In MacEachren, Alan M. and D. R. Fraser Taylor (eds.). *Visualization in Modern Cartography*. Modern Cartography, Volume 2. Oxford, U.K. Pergamon/Elsevier Science Ltd.

Proctor, James D. and Anthony E. Richardson. 1997. Evaluating the Effectiveness of Multimedia Computer Modules as Enrichment Exercises for Introductory Human Geography. *Journal of Geography in Higher Education*, vol. 21, no. 1, pp. 41-55.

Slocum, Terry A. 1994. Visualization Software Tools (Introduction). In MacEachren, Alan M. and D. R. Fraser Taylor (eds.). *Visualization in Modern Cartography*. Modern Cartography, Volume 2. Oxford, U.K. Pergamon/Elsevier Science Ltd.

Thrower, Norman J. W. 1959. Animated Cartography. *Professional Geographer*, vol. 11, no. 6, pp. 9-19.

Thrower, Norman J. W. 1961. Animated Cartography in the United States. *International Yearbook of Cartography*, vol. 1, pp. 20-28.

Tobler, W.R. 1970. A computer movie simulating urban growth in the Detroit Region. *Economic Geographer*. vol. 46, no. 2, pp. 234-240.