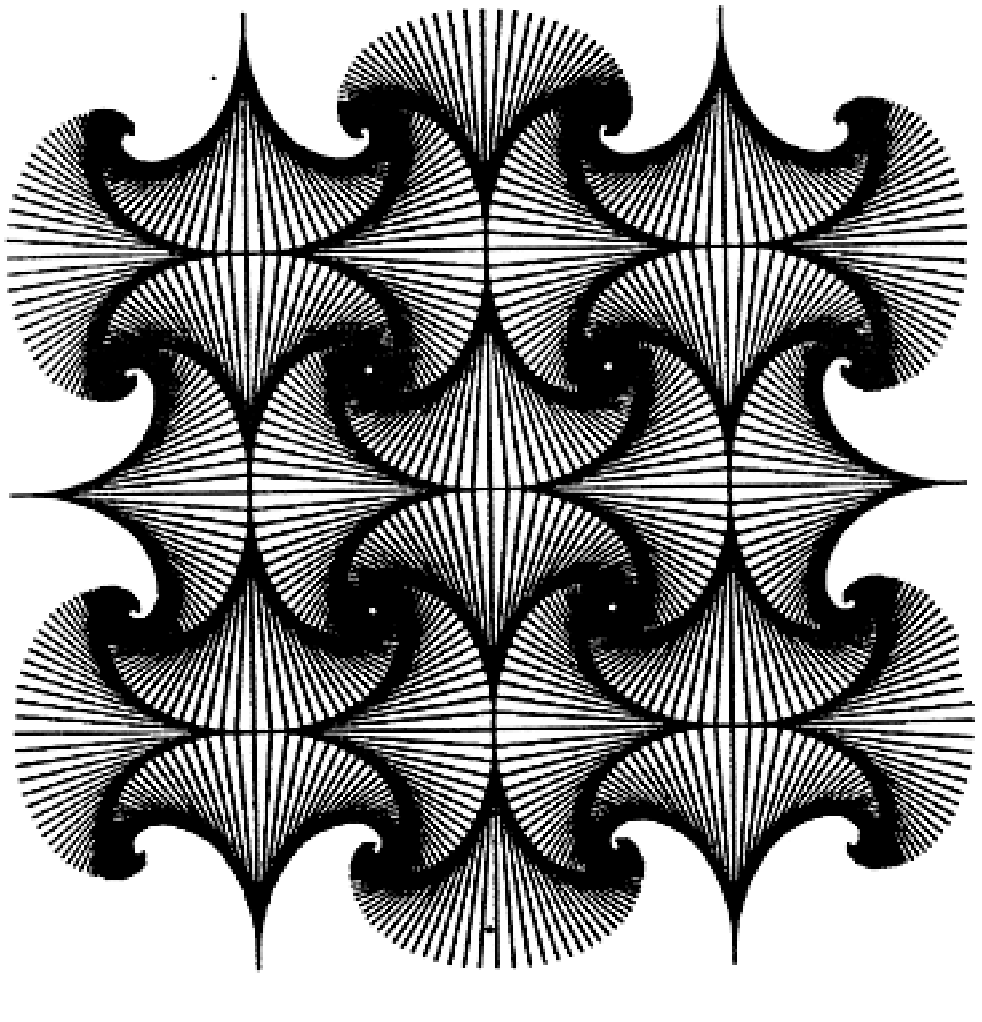
物理

COMPUTER GRAPHICS AND PHYSICS
GREGORY PAT SCANDALIS



CALIFORNIA STATE POLYTECHNIC STATE UNIVERSITY
SAN LUIS OBISPO, CALIFORNIA
1984

Physics 461
Physics 462

Date Submitted: 10/24/84
Project Advisor: Dr. Chris de Latour

CONTENTS

LIST OF FIGURES

1 - INTRODUCTION

1.1 THE COMPUTER

Perhaps no other tool of the Twentieth Century will change as many peoples lives as the 'computer'. The computer is a computational tool capable of doing hundreds of hours of tedious work in seconds; capable of making a bookeepers nightmare manageable; capable of turning a mildly complicated mathematical problem into several pounds of paper! Computer- Godsend or beast?

Two decades ago, computers, in particular computer output, began to take on new and exciting forms. Computers were used to do complicated tasks on spacecrafts; musicians at Columbia University used computers to control another new tool, the electronic synthesizer, creating avant-garde new music; the word 'digital' became a household word; the 'word processor' or text editor made the composition of written material more accessible.

One of the most exciting of the new output media is called 'Computer Graphics', a development that had to wait for today's computer with its rapid graphic capability. This media takes on a many diverse forms, from the simplest video game, to fantastically detailed movies used for entertainment and to verify complex scientific models. Certainly, a 3-D plot of the temperature distribution on a square plate tells the viewer a great deal more than several pounds of numerical output.

One of the fantastic things about studying Physics in the last few decades is that one no longer needs to be a Gauss or a Fourier; spending days doing tedious calculations, struggling to gain an insight into higher mathematics. Not to belittle the heuristic value of doing hand calculations, but no amount of tedious number crunching replaces the 'intuitive' feeling one gets from a picture. With the aid of computing machines, it is now possible to graphically explore mathematical properties of equations and models in Physics.

1.2 COMPUTER GRAPHICS AND PHYSICS

How do computer generated movies relate to Physics. Physics seeks to model what we see in nature with mathematics. Many times I've wanted to graphically verify the validity of a complex mathematical equation. I have felt that a picture would greatly inprove my understanding of such an equation.

I am reminded of a interesting story dealing with a man, blind from birth, who had his sight restored by modern surgery. When he was blind he had learned to operate a metal lathe. After his sight was restored he told friends that he wished to see what a lathe looked like. They took him to a local factory to see a lathe. When showed the lathe, he was very disappointed, and insisted he could not see it. He asked if he could feel it. After exploring, for several minutes, he announced that now he could see it. Apparently he had to see the lathe in familiar terms, by touch, before he could understand it in unfamiliar terms, sight.

Perhaps with much wasted time and frustration the cured blind man could have 'seen' the lathe, but by allowing him to feel the lathe, the task became not one of frustration, but one of enjoyable enlightenment. I share our blind/sighted man's plight. I want to 'feel' the mathematics that I have studied. By analogy to our blind/sighted friend, I think that 'feeling' what I am learning would turn a frustrating tedious task into an enjoyable and enlighting experience. As elegant as an equation may be, it is worthless without a deep understanding of what that equation says. How does one achieve a deeper understanding?

The present method requires that the student wrestle with some problems from some new conceptual material, until connections are made to more familiar concepts. To augment this, the students are often given an opportunity, in the lab, to play with the concepts they are learning in the classroom. But in the lab it is often difficult to distinguish ideal point particles, gasses, frictionless surfaces, etc. etc... , from the real world (Physics does model the real world after all !!!)

Like our blind/sighted man, students must be able to 'feel' what they are doing first. This notion is not new, but what is relatively new is the idea of using computer graphics as a tool to graphically explore the realm of Physics.

As an example consider the Fourier series. I was frustrated by the mechanistic level on which I first understood, eveness, oddness, half range expansions, etc.,etc... What I did not understand I simply memorized. The expected occured, I promptly forgot what I memorized. Then, I started this senior project. One of my first programs allowed me to explore the Fourier series. I then spent an intense and enjoyable session playing with Fourier series. Of course, I could have done the same exploration with graph paper and a calculator, but it would have required more time and resolve than I possess. Computer graphics afforded me a very gratifying learning experience.

It should be noted, that exploring the Fourier series with a computer is not a replacement for learning the Fourier equations and their properties. Computer graphics simply gives one another view of the Fourier series. Both methods of learning are complementary.

The Physicist of days gone by had mathematics as his main 'hand tool'. The Physicist of the Twentieth Century now has within reach a new 'power tool', the computer. This tool has had a major impact on the commercial and scientific communities. so why is it given such a minor role in the educational community? Hopefully my senior project will play at least some small role in changing that.

1) The Brain, Richard M. Restak,M.D. chapter 6.

2 - THE PROJECT

2.1 THE PROJECT

The title of my senior project is 'Computer Generated Movies and Physics'. I created computer generated movies of vibrating systems, that I hope will provide the viewer with insights not experienced before. The programs display four parameters, x,y,z and t. The movie format allows the display of time dependent solutions to various problems! The project was interdisciplinary, it involved concepts from Physics, Mathematics, Computer Science, Electronics and Photography.

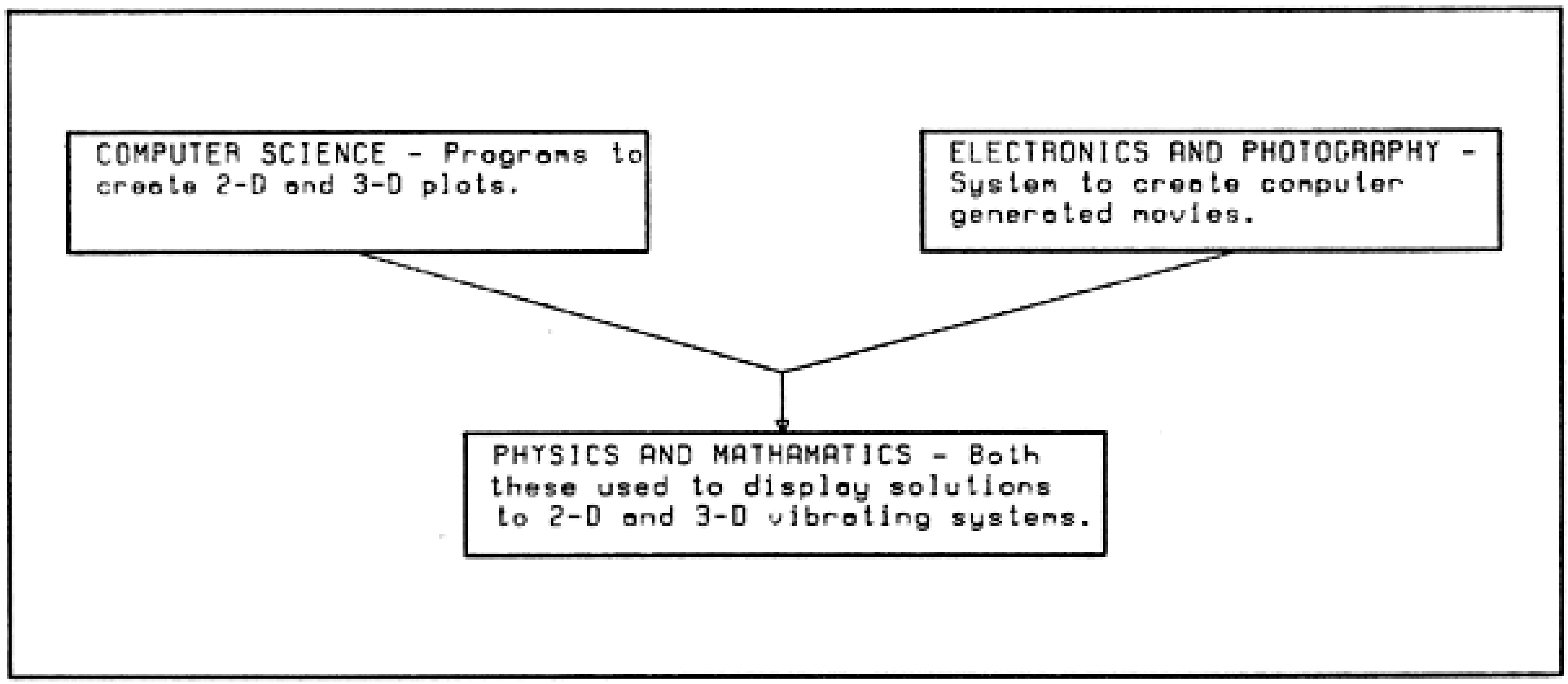

COMPUTER SCIENCE - Programs to create 2-D and 3-D plots.
ELECTRONICS AND PHOTOGRAPHY - System to create computer generated movies.
PHYSICS AND MATHAMATICS - Both these used to display solutions to 2-D and 3-D vibrating systems.

FIGURE 1 - STRUCTURE OF THE SENIOR PROJECT. The project was interdisiplinary, drawing from many different fields.

Considering large amount of computer time that is required to generate a movie, only a few movies were made. The following are the situations that were generated:

1) D'Alembert solution to the 1-D wave equation.
2) Fourier solution to the plucked string.
3) Solution to the vibrating square membrane.

2.2 HOW A MOVIE IS MADE

The method of creating a computer generated movie is simple. Computer pictures are drawn on a special terminal known as a graphics terminal. Once a picture is generated it is known as a 'frame'. This frame is then photographed off the screen with a movie camera set up to photograph single frames. To conserve computing time three pictures are taken of each frame.

The camera is a movie camera that can be checked out from Cal Poly A.V. This camera can be triggered by simply closing a switch. I could let the computer generate a frame and then trigger the camera by hand, however this would be very tedious. An alternative is to trigger the camera with a sound operated switch. After the computer finishes generating a frame, a command in the program rings the bell in the terminal. The bell sets off the sound operated switch which in turn triggers the camera.

Finally when all the frames are taken and the film is processed the resulting movie is a computer generated movie. Incidentaly, 1 second of the movie is composed of 18 frames. This means that a 3.5 minute movie is about 3600 frames. It takes about 1 minute to draw each frame, so 3.5 minute of movie is about 60 hours of computer time!

2.3 A FEW WORDS ABOUT THIS WRITE UP

This paper has been written in a sort of users guide style. I intend for this paper to provide a sort of reference on how to do graphics at Cal Poly, and how to write the programs used in the project. I have included all source code for the programs.

3 - GRAPHICS

3.1 THE TERMINAL

As I said earlier, the graphics are done on a special computer terminal known as a Graphics Terminal. This is a special terminal that allows the user to draw pictures. The terminal that I used was a TEKTRONIX 4006-M. The TEKTRONIX terminal has a direct-view storage tube (DVST) in place of a standard cathode-ray tube (CRT). This tube is much like the tubes used in a storage oscilloscope. Lines written on the DVST will persist for a long time. It is not possible to selectively erase parts of the DVST.

The type of graphics done on the TEKTRONIX terminal are known as 'vector graphics'. On the TEKTRONIX terminal it is possible to draw very straight lines or 'vectors' between two points. This is different from most 'raster scan' displays in which lines are created from discrete elements known as pixels.

The DVST, like a CRT, has a writing beam electron gun and a phosphor coated screen in addition it has a flood electron gun. The writing beam gun does not write directly on the phosphor, but on a fine mesh dielectric screen known as the storage grid. The storage grid is mounted between the writing beam gun and the phosphor screen. Initially the storage grid is uniformly negatively charged. High speed electrons strike this grid and dislodge some of the surface electrons. These electrons are captured by the positively charged collector. This leaves a net positive charge on the storage grid! Because the storage grid is a dielectric, other electrons cannot migrate, and a positive charged pattern is stored.

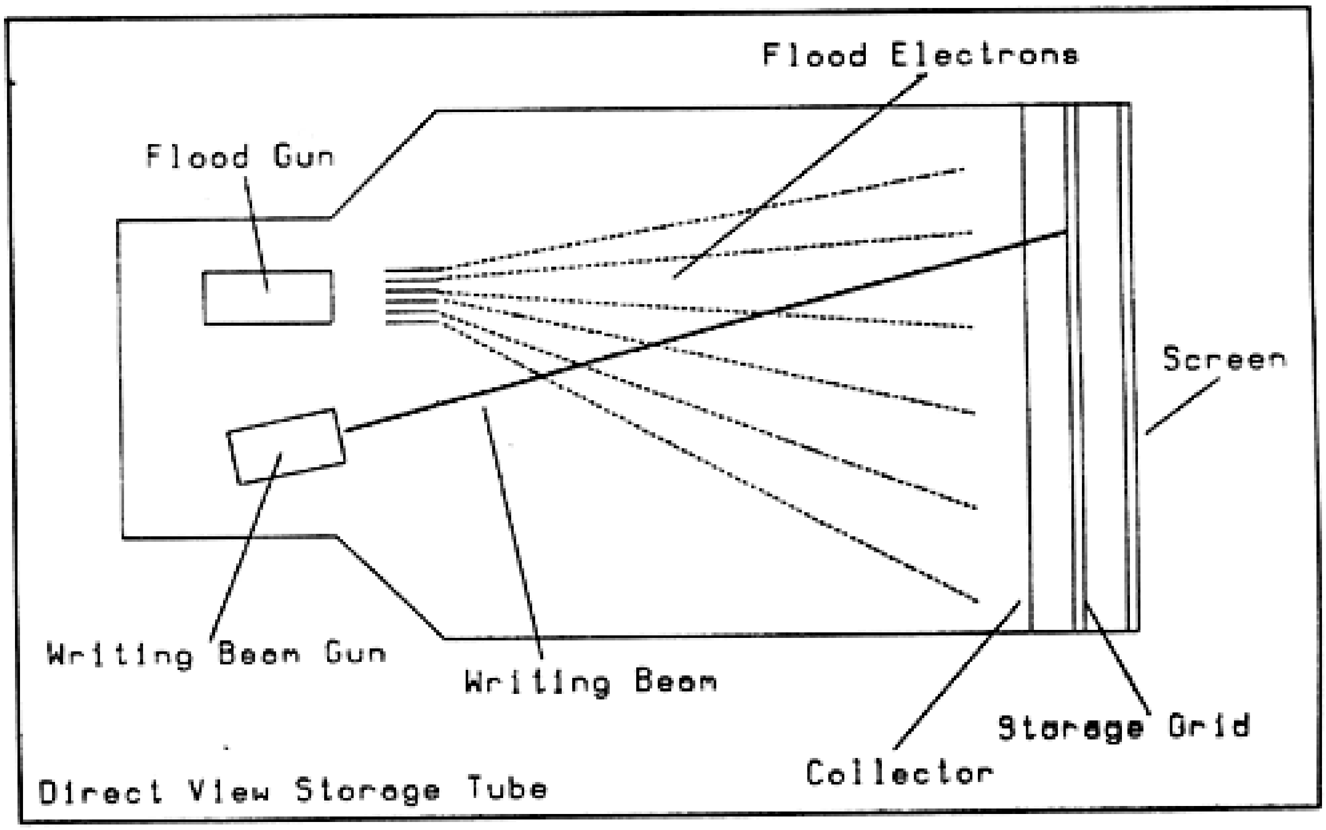


FIGURE 2

Independent of the writing gun is the flood gun. This gun uniformly covers (floods) the storage grid with low velocity electrons. Electrons approaching a positively charged region of the grid will pass through and strike the phosphor screen, causing light emission. Electrons approaching other parts of the of the grid will be repelled and will not pass through. Thus, the charge pattern stored on the grid is reproduced on the phosphor screen!

The terminal has two modes: normal alphanumeric mode and graphic mode. The screen has 1023 (x) by 780 (y) points that can be addressed. These are the screen points. With software, vectors can be drawn between any two points. It is also possible, with software, to define the screen coordinates to be any real coordinates. This is called windowing.

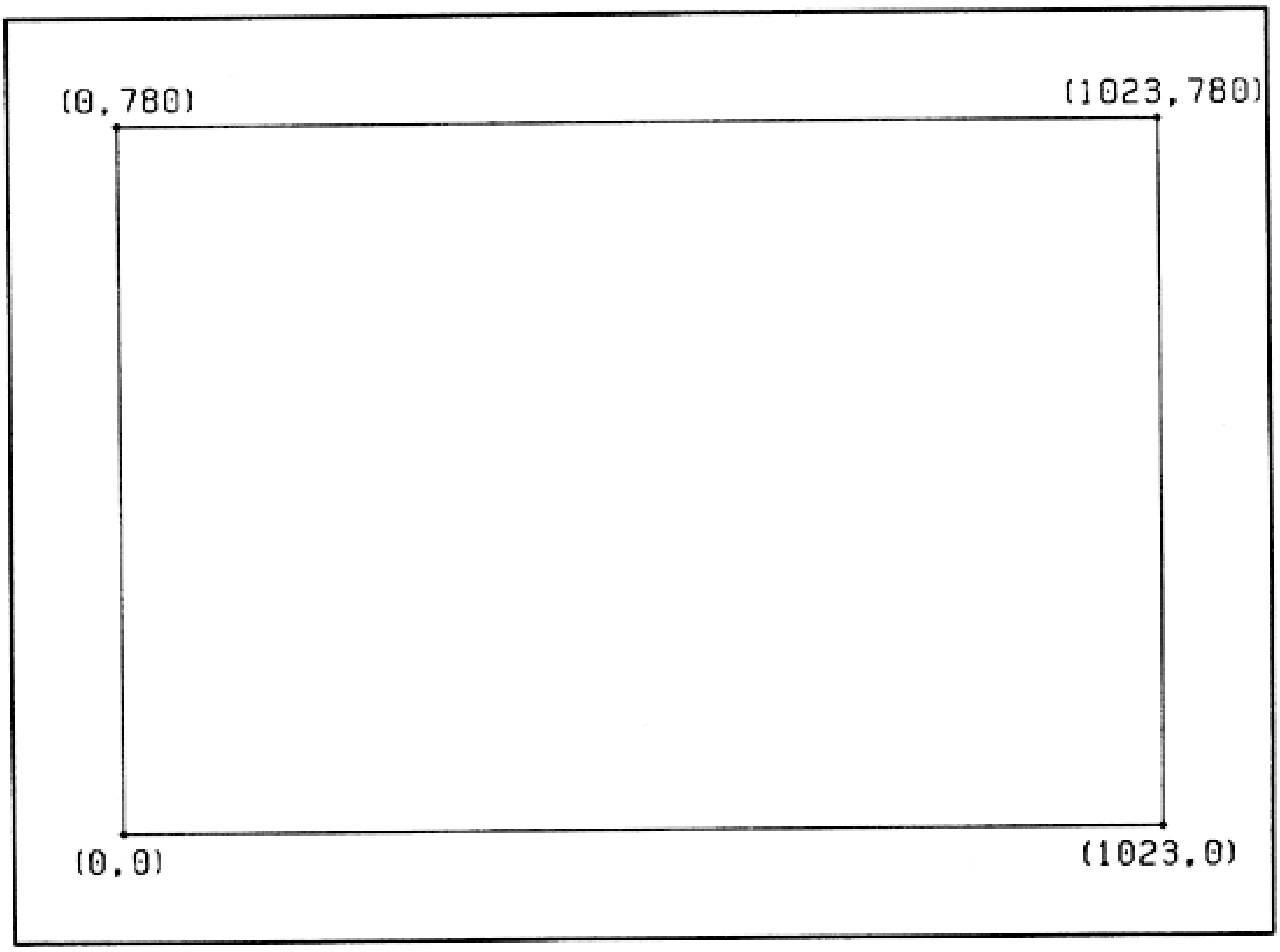


FIGURE 3 - SCREEN DIVISIONS OF THE TEKTRONIX TERMINAL. The screen is 1023 (x) by 780 (y).

3.2 WRITING A GRAPHICS PROGRAM

A program that draws a picture is a conventional program written in FORTRAN IV. In order to draw the pictures, the program must call a subroutine that will do the drawing on the terminal screen. These subroutines are found in a library of subroutines called PLOT10. The main types of subroutines that I use are of the form:

```
                         CALL INITT(IBAUD)
                         CALL TWINDO(MINX,MAXX,MINY,MAXY)
                         CALL DWINDO(XMIN,XMAX,YMIN,YMAX)
                         CALL MOVEA(X1,Y1)
                         CALL DRAWA(X2,Y2)
                         CALL FINITT(IX,IY)
```

INITT puts the terminal into the graphic mode, IBAUD is the (baud rate/10) that the terminal is running at. My terminal ran at 1200 baud rate, thus IBAUD was 120.

TWINDO AND DWINDO are subroutines that preform a windowing transformation. Twindo defines the screen coordinates of the part of the screen that will be used. MINX, MAXX, MINY, MAXY are the screen boundries of the window.

DWINDO defines the real coordinates that will be defined in the window. XMIN, XMAX, YMIN, YMAX are the real boundries of the window.

MOVEA moves the cursor to the point (X1,Y1). MOVEA does not cause the write beam to draw a line. DRAWA causes the write beam to draw a vector from the current cursor position to the point (X2,Y2).

FINITT dumps the line buffer, that is it dumps the last of the graphic commands. FINITT also puts the terminal back into normal alphanumeric mode and places the cursor at screen position (IX,IY).

These subroutines allow a graphic program to be structured (top down structured programing). An example of a FORTRAN fragment that would define the screen to be the cartesian coordinate system , (-2<x<2, -2<y<2), and draw the coordinate axes:

```
                                 .
                                 .
                                 .
                          CALL INITT(120)
                          CALL TWINDOW(122,902,0,780)
                          CALL DWINDO(-2.0,2.0,-2.0,2.0)
                          CALL MOVEA(0.0,-2.0)
                          CALL DRAWA(0.0,2.0)
                          CALL MOVEA(-2.0,0.0)
                          CALL DRAWA(2.0,0.0)
                          CALL FINITT(0,780)
                                 .
                                 .
                                 .
                          STOP
                          END
```

These basic five subroutines are used to do all the graphics.

3.3 RUNNING A GRAPHICS PROGRAM

The computer that I chose for my senior project for was Cal Poly's CDC CYBER 174, known as the Local Cyber. As I said before, the graphics programs are standard FORTRAN IV programs. Because the PLOT10 library was called it was necessary to use the older FORTRAN IV instead of FORTRAN 77. Programs can be written with and without line numbers. To run a graphics program without line numbers on the Cyber the user types:

```
-,MOVIE,PROGRAM NAME,BATCH
```

To run a graphics program with line numbers on the Cyber the user types:

```
-,MOVIE,PROGRAM NAME
```

MOVIE is a procedure file written in Cyber Control Language that: allows the user to use the PLOT10 library; calls the FORTRAN compiler; disables the normal limits on computing time (a copy of MOVIE is in Appendix 1). It is necessary to disable the normal limits on computing time because the movie program typically runs for 4-6 hours

4 - THE HARDWARE

4.1 THE CAMERA

The camera used to photograph the screen of the graphics terminal was a SANYO SUPER-8 movie camera. This camera was checked out from Cal Poly's Audio Visual department. The SANYO movie camera has a single frame feature that allows the user to do animations. The camera shutter is triggered by electrically shorting the two terminals of a microjack input.

4.2 THE TRIGGER BOX

A trigger box was constructed to trigger the camera. This box consisted of:

1) A 12 V. and a 5 V. power supply.
2) A sound operated switch.
3) A divide by two decimal counter and LED display.

The sound activated switch required 12 V. and the counter required 5 V. The sound activated switch was purchased from Radio Shack. It was necessary to divide by two, the number times the bell triggered the switch because, the sound activated switch had two states: open and closed. Thus, the first ring of the bell would cause the switch to close and trigger the camera; the second ring of the bell would set the switch to open, which would not trigger the camera. The LED display of the counter showed the number of Pictures taken, and not the number of times the terminal bell had rung.

I built three versions of the trigger box. The first was soldered on a proto board. This first version performed erratically. The second version was wire wrapped with what is called a 'just wrap' wire wrap tool. The just wrap tool uses unstripped wire that is wrapped around square pegs. The idea is that the square corners of the pegs will cut into the insulation, and form a gas tight bond to the wires. This did not work at all. The final version was built with a conventional stripped wire wrap tool. This final version worked very well. The schematic for the trigger box can be found in Appendix 2.

4.3 PROCESSING THE FILM

The film used to make the movies was Kodak Tri-x Super-8 movie film. It was necessary for me to do all the film processing for three reasons:

1) The film was not reversed to positive the way most movie film is; it was developed directly to negative. This was done to make the movie easier to view; the bright green lines of the graphics terminal would be dark black lines in the movie.

2) The film was developed with D-19 developer for 15 minutes. This is a high contrast developer that does not develop any grey tones; the film will only have black and white tones. This was done to eliminate any fogging from ambient light being emitted by the screen.

3) The film was over developed or 'pushed'. The screen of the graphics terminal was very dim; thus, it was necessary to overdeveloped the film to obtain greater detail.

The film was developed in a Arkay G-3 hand cranked movie processing tank.

5 - 3-D GRAPHICS

5.1 TRANSFORMATIONS

As explained earlier, the Tektronix graphics terminal can draw straight lines or vectors between any two points. In general an object to be drawn is represented as a set of points that vectors are to be drawn between. A 3-D object would be represented by a set of points:

$$P_n(x_n, y_n, z_n)$$

Transformations are used to manipulate the object in cartesian space. There are three kinds of geometrical transformations:

1) Translation
2) Scaling
3) Rotation

In addition to the geometrical transformations a fourth transformation is used to map the 3-D down to 2-D:

4) The perspective transformation

Each of the 3-D geometrical transformations can be represented in a uniform way by a 4 x 4 matrix. A point is transformed by operating on it with a transformation matrix.

$$[P'] = [P][A] \qquad (5\text{-}1)$$

$$[P'] = [x' \; y' \; z' \; 1] \qquad (5\text{-}2)$$

$$[P] = [x \; y \; z \; 1] \qquad (5\text{-}3)$$

$$[A] = \begin{vmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & 1 \end{vmatrix} \qquad (5\text{-}4)$$

5.2 TRANSLATION

Points in xyz space can be translated to new positions by adding a translation amount to the coordinates of the points. The form of the translation transformation:

$$\begin{aligned} x' &= x + Dx \\ y' &= y + Dy \\ z' &= z + Dz \end{aligned} \qquad (5\text{-}6)$$

In matrix form:

$$[T] = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Dx & Dy & Dz & 1 \end{vmatrix} \qquad (5\text{-}7)$$

5.3 SCALING

Scaling stretches or compresses the image along a particular axis with respect to the origin. The scaling transformations:

$$\begin{aligned} x' &= xS_x \\ y' &= yS_y \\ z' &= zS_z \end{aligned} \qquad (5\text{-}8)$$

In matrix form:

$$[S] = \begin{vmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad (5\text{-}9)$$

5.4 ROTATIONS

Most of us are familiar with right-handed coordinates. Unfortunately, the most logical coordinate system for 3-D graphics would be the left-handed system. This gives a natural interpretation of larger z values being into the screen of the terminal. The matrix form of the x, y, and z rotation operators in left handed coordinates:

$$[R_X(e)] = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos e & \sin e & 0 \\ 0 & -\sin e & \cos e & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad (5\text{-}10)$$

$$[R_Y(e)] = \begin{vmatrix} \cos e & 0 & -\sin e & 0 \\ 0 & 1 & 0 & 0 \\ \sin e & 0 & \cos e & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad (5\text{-}11)$$

$$[R_Z(e)] = \begin{vmatrix} \cos e & \sin e & 0 & 0 \\ -\sin e & \cos e & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad (5\text{-}12)$$

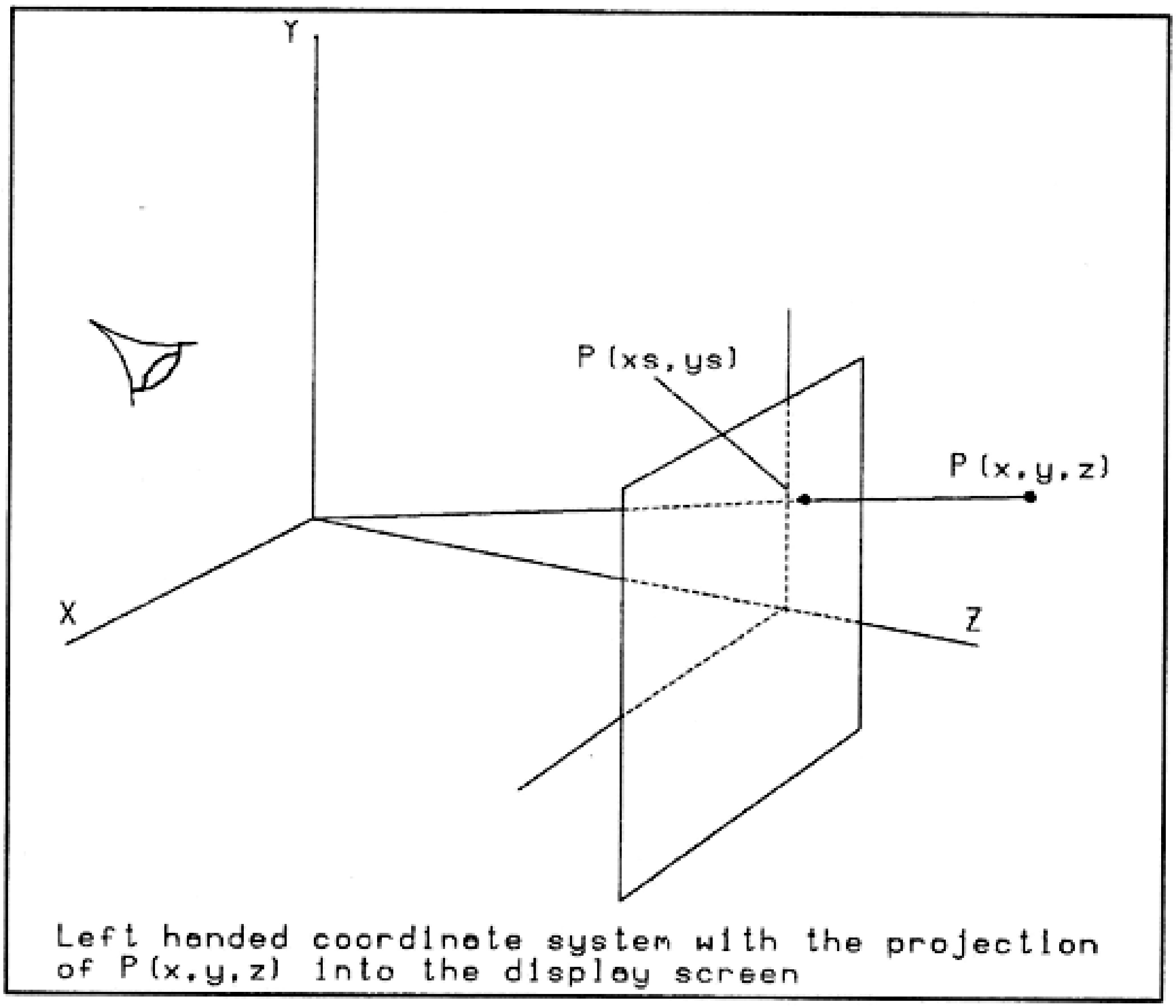
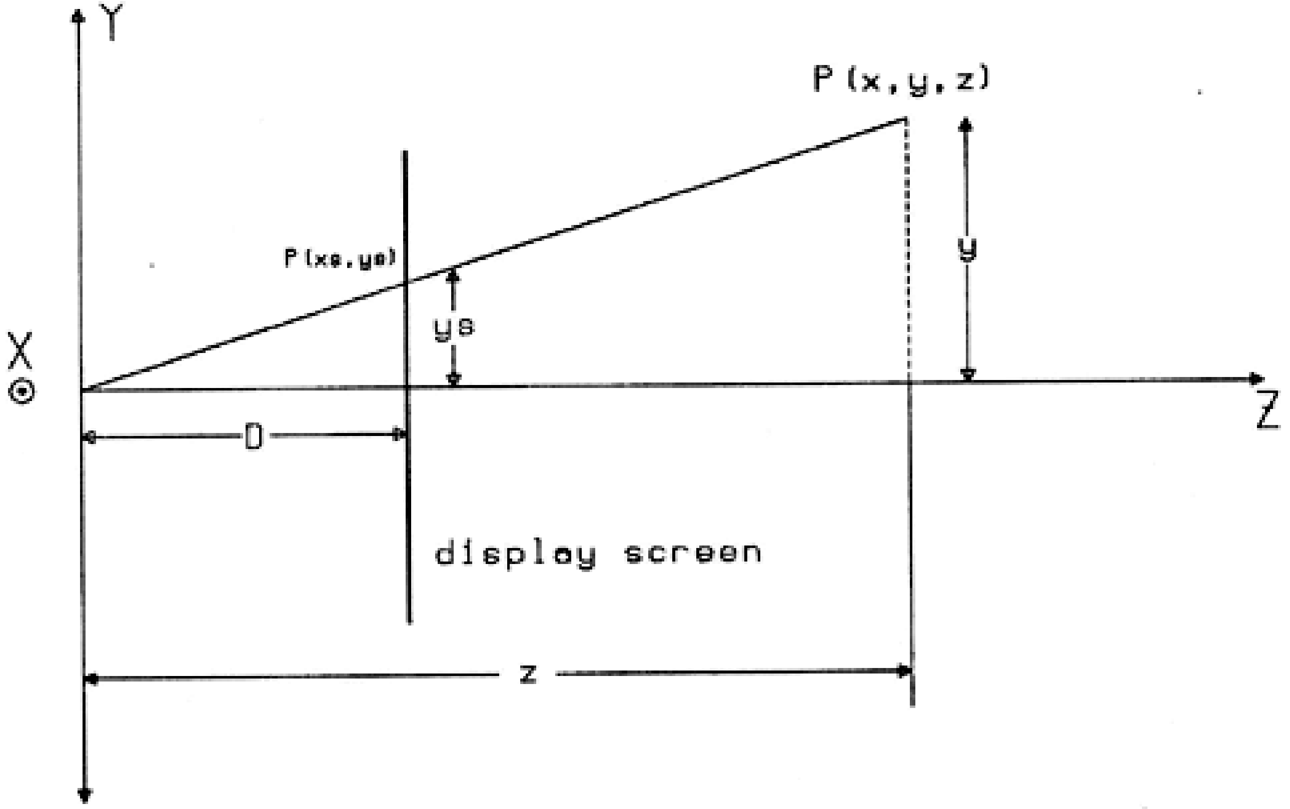

Y
X
Z
P (xs, ys)
P (x, y, z)
Left handed coordinate system with the projection of P (x, y, z) into the display screen

FIGURE 4

5.5 THE PERSPECTIVE TRANSFORMATION

The perspective transformation is used to map the 3-D object coordinates down to 2-D screen coordinates. The perspective transformation projects the 3-D object onto the display screen. The projection of a 3-D object is defined by straight projection rays emanating from a center of projection, intersecting the projection plane (in this case the display screen), and terminating on each point in the 3-D object. The perspective transformation can be derived from similar triangles. The center of projection is set at the origin. x,y,z are the object coordinates. x_s and y_s are screen coordinates.

$$x_s = D(x/z) \qquad (5\text{-}13)$$
$$y_s = D(y/z)$$



The perspective transformation can be derived from similar triangles. (ys/D) = (y/z) thus ys = D(y/z)

FIGURE 5

6 - MATRIX PROPERTIES

6.1 BASIC PROPERTIES

The basic geometrical operations in 3-D are represented as matricies. The task of manipulating a 3-D object is reduced to matrix multiplications. It is therefore, important to understand some of the basic properties of matrix multiplication. Given two matricies:

[A] with elements (a_{ij}) and size m x p

[B] with elements (b_{ij}) and size p x n

Their product is:

[C] with elements (c_{ij}) and size m x n

$$(c_{ij}) = \overset{p}{\underset{k = 1}{\quad}} a_{ik}b_{kj} \qquad (6\text{-}1)$$

In general matrix multiplication has the following properties:

$[A][B] \neq [B][A]$	COMMUTATIVE LAW
$([A][B])[C] = [A]([B][C])$	ASSOCIATIVE LAW

6.2 COMMUTING OPERATIONS

In general Matrix operators do not commute; however, several important cases of commuting operators can be found in 3-D graphics. Recall the basic operator matricies used in 3-D graphics:

[T]	TRANSLATION	
[S]	SCALING	
$[R_e(\theta)]$	ROTATION	e = x,y or z

Commuting operations:

$$[T_1][T_2] = [T_2][T_1] \qquad (6\text{-}2)$$

$$[S_1][S_2] = [S_2][S_1] \qquad (6\text{-}3)$$

$$[R_e(\theta)_1][R_e(\theta)_2] = [R_e(\theta)_2][R_e(\theta)_1] \qquad (6\text{-}4)$$

The commuting properties of these operators can be understood geometrically. A net translation is independent of the order the translations are performed. A net scaling is independent of the order the scalings are performed. Finally, a net rotation about a given axis is independent of the order that the rotations are performed.

6.3 NON-COMMUTING OPERATIONS

It is important to be aware of some of the non-commuting operations. Non-commuting operators:

$$[T][S] \neq [S][T] \qquad (6\text{-}5)$$

$$[T][R_e(\theta)] \neq [R_e(\theta)][T] \qquad (6\text{-}6)$$

$$[S][R_e(\theta)] \neq [R_e(\theta)][S] \qquad (6\text{-}7)$$

The order of the operators is important in mixed operations because the scaling and rotation operators work with respect to the origin.

6.4 COMMON CONCATENATED OPERATIONS

Because of the origin dependence of the scaling and rotation operators the task of enlarging or rotating an object in space is not just a matter of operating on the object with a single operator. For example to rotate an object about its geometric center would require three operations: Translate it to the origin, rotate it about the origin and translate it back to its original position. Some common concatnated operations:

Given:

$[T_o]$ is a translation operator that translates to the origin.

$[T_o^{-1}]$ is a translation operator that translates back to the original position.

Rotate an object about its geometric center:

$$[T_o][R_e(\theta)][T_o^{-1}] \qquad (6\text{-}8)$$

Enlarge an object about its geometric center:

$$[T_o][S][T_o^{-1}] \qquad (6\text{-}9)$$

Rotate and enlarge an object about its geometric center:

$$[T_o][S][R_e(\theta)][T_o^{-1}] \tag{6-10}$$

Translate a tumbling object:

$$[T_o][R_e(\theta)][T_o^{-1}][T] \tag{6-11}$$

7 - GRAFIT

7.1 THE PROGRAM

GRAFIT is a FORTRAN subroutine that is used to create 3-D plots of single valued functions of two variables. A single valued function of two variables is of the form f(x,y). For every ordered pair (x,y) the function has a unique value. GRAFIT displays f(x,y) as an altitude above the xy plane. The resulting plots created by GRAFIT are surfaces. The user must provide GRAFIT with an array containing all of the z coordinates. GRAFIT removes any lines that are hidden from view or obscured. GRAFIT also allows the user to specify the view of the object by specifying three parameters R, THETA and PHI.

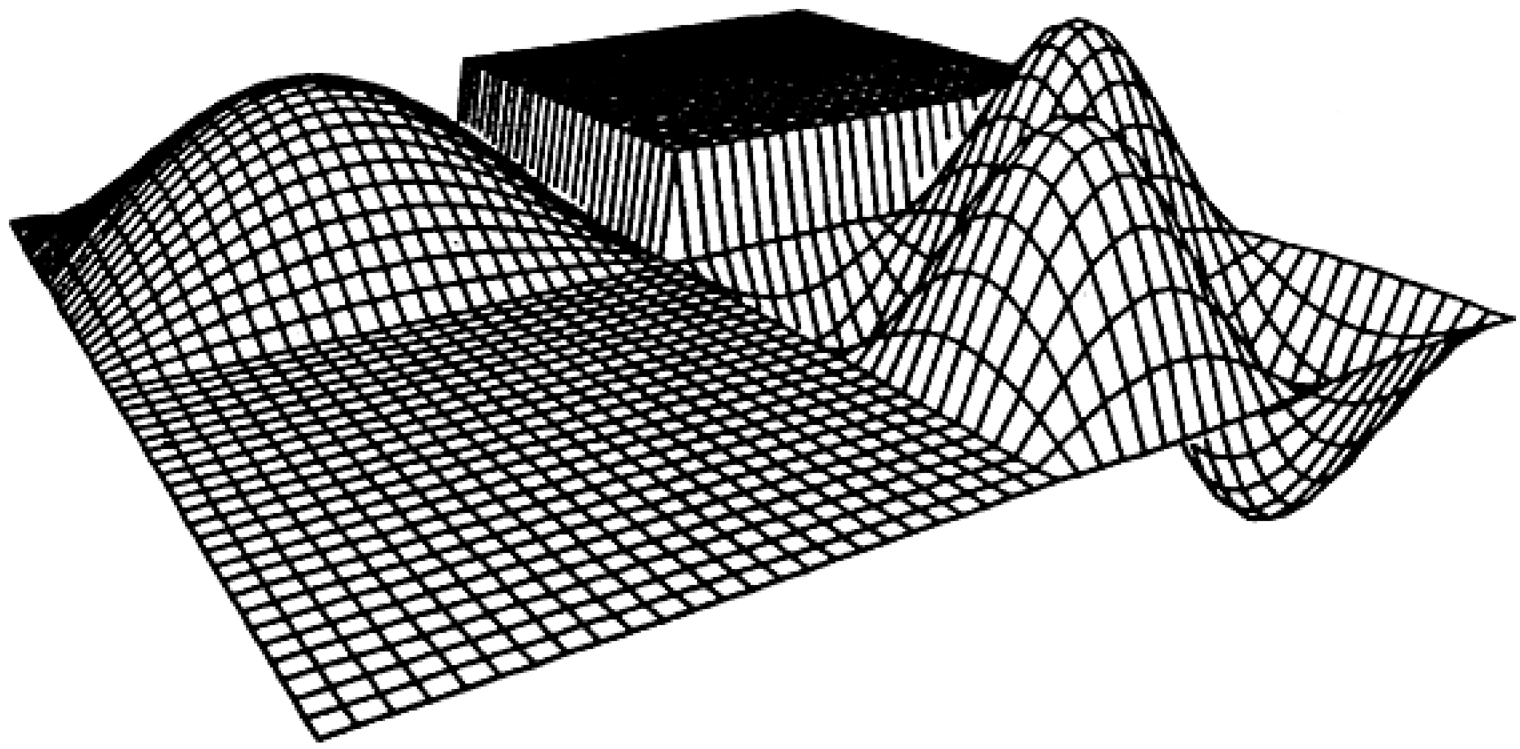


FIGURE 6 - TEST SURFACE. Test surface drawn with GRAFIT. Notice that hidden lines are removed.

7.2 3-D TRANSFORMATIONS USED IN GRAFIT

Grafit represents a special case of 3-D graphics. It was not necessary to have the program do the matrix transformations. The transformations were worked out by hand and defined in the program as functions. Initially the display screen represents the xy plane and the z coordinate is into the display. Next, the z axis rotation operator is applied to the coordinates; this rotates the xy plane of the surface through an angle THETA. Then, the x-axis rotation operator is applied; this tilts the surface through an angle PHI. Finally, the surface is translated forward and scaled so that it fits onto the screen. The operations:

$$[R_Z(\theta)][R_X(\phi)][T_Z][S] \qquad (7-1)$$

The form of these transformations:

$$xs = D[x\cos(\theta) - y\sin(\theta)]/Q \quad (7\text{-}2)$$

$$ys = D[x\sin(\theta) + y\cos(\theta)]/Q \quad (7\text{-}3)$$

$$Q = [(x\sin(\theta) + y\cos(\theta)]\sin(\phi) - z\cos(\phi) \quad (7\text{-}4)$$

Where

x_s and y_s are coordinates on the screen.
D is the distance between the viewer and the screen.
θ is the angle of rotation with respect to the Z axis.
ϕ is the angle of rotation with respect to the X axis.

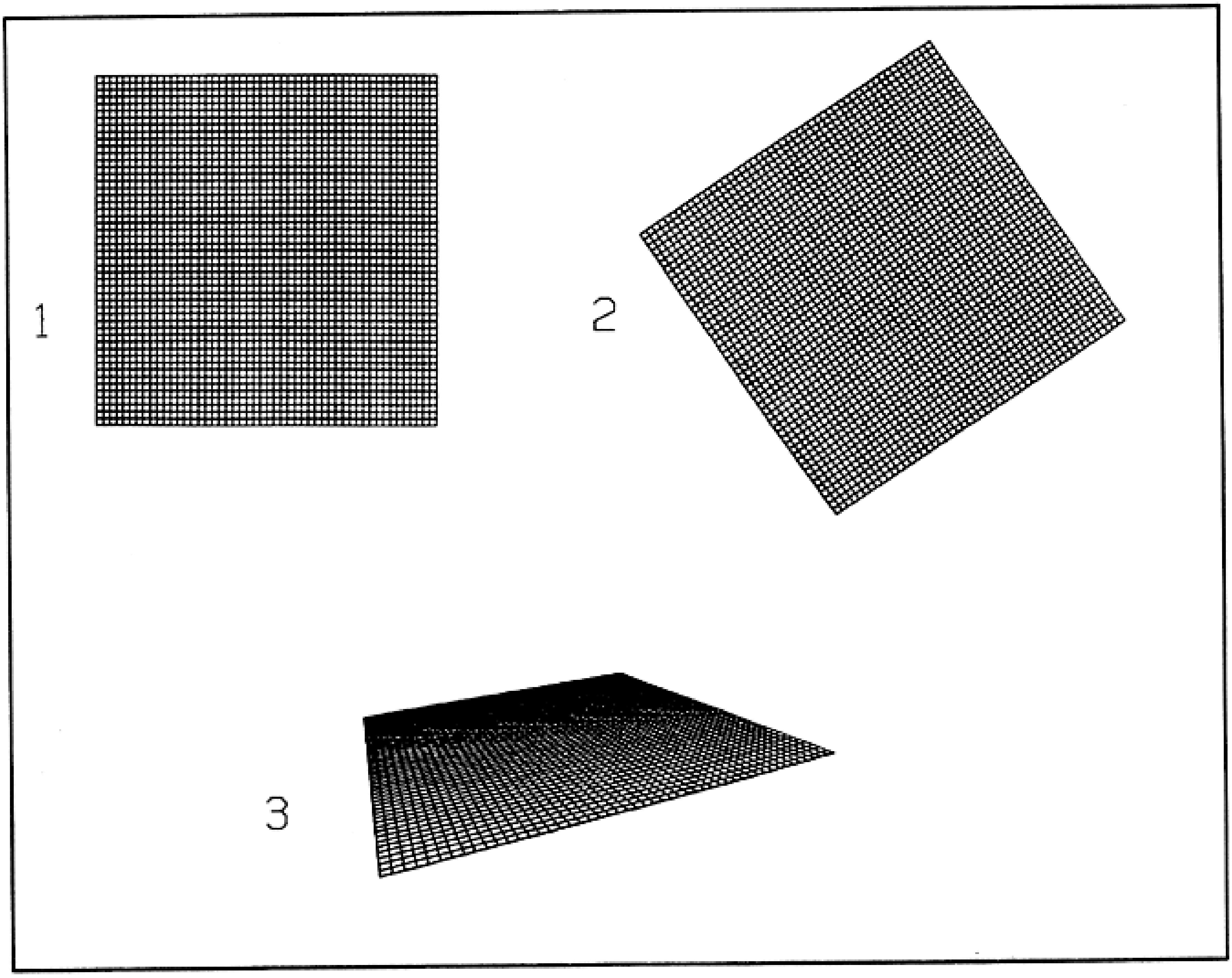


FIGURE 7 - OPERATIONS DONE BY GRAFIT ON A PLANE Initially the plane is scaled and translated to optimize the use of the screen. The plane is then rotated respect to the Z axis, through an angle THETA. Finally the plane is rotated with respect to the X axis through an angle PHI.

7.3 HIDDEN LINE REMOVAL

The algorithm chosen to remove hidden lines from the surfaces is known as Wright's algorithm. This algorithm is described in great detail in the reference 2. I looked at many hidden line algorithms, most involved clipping the hidden parts of vectors before the perspective projection onto the screen. For these algorithms, the criteria for visibility was often elusive. These methods were often brute force solutions to the hidden line problem. Wright's algorithm is very simple to implement and has virtually nothing to do with the three dimensional nature of the picture being drawn. This algorithm clips the hidden parts of vectors after the perspective transformation.

Wright's algorithm can be stated as this: Draw the picture from front to back, and don't draw where you have already drawn.

7.4 IMPLEMENTING WRIGHT'S ALGORITHM

The implementation of Wrights algorithm can be broken into several steps:

1) The surface is represented as a regular array of x and y points. Each P(x,y) has an altitude represented by the z coordinate. Thus the surface is like rubber graph paper.

2) The surface is broken into segments of constant x and constant y.

3) The surface is drawn front to back using segments of constant x and constant y.

4) A visibility test is applied to the end points of each segment. If both ends are visible then the segment is drawn. If both ends are obscured then the segment is not drawn. If one end is visible and one end is obscured then the intersection between the surface and the segment is calculated and the segment is drawn from the intersection to the one visible point.

7.5 VISIBILITY TEST

The second half of the algorithm is: Don't draw where you have already drawn. This is achieved by maintaining 2 arrays, one of the upper bound of the surface and one of the lower bound of the surface, as a function of screen x coordinate. The visibility test is then used to test both end points of a segment to see if they lie between the lower bound and the upper bound.

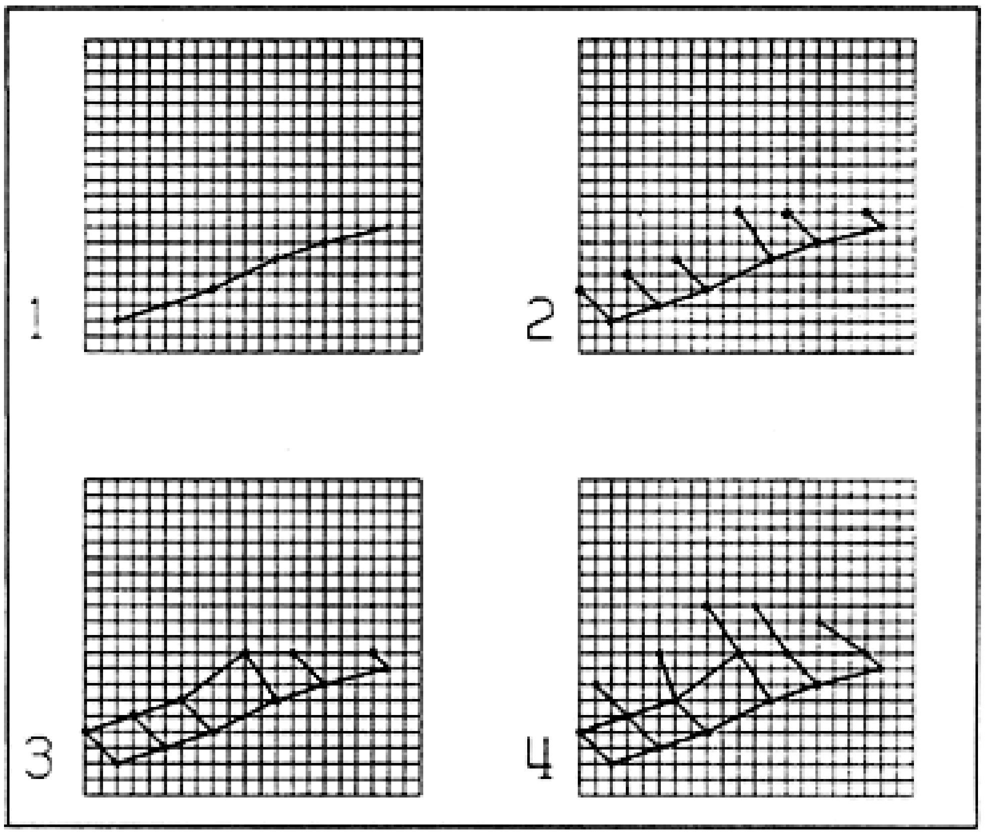


FIGURE 8 - WRIGHT's ALGORITHM. First segments of constant x are drawn; then segments of constant y are draw. This process continues until the whole surface is generated.

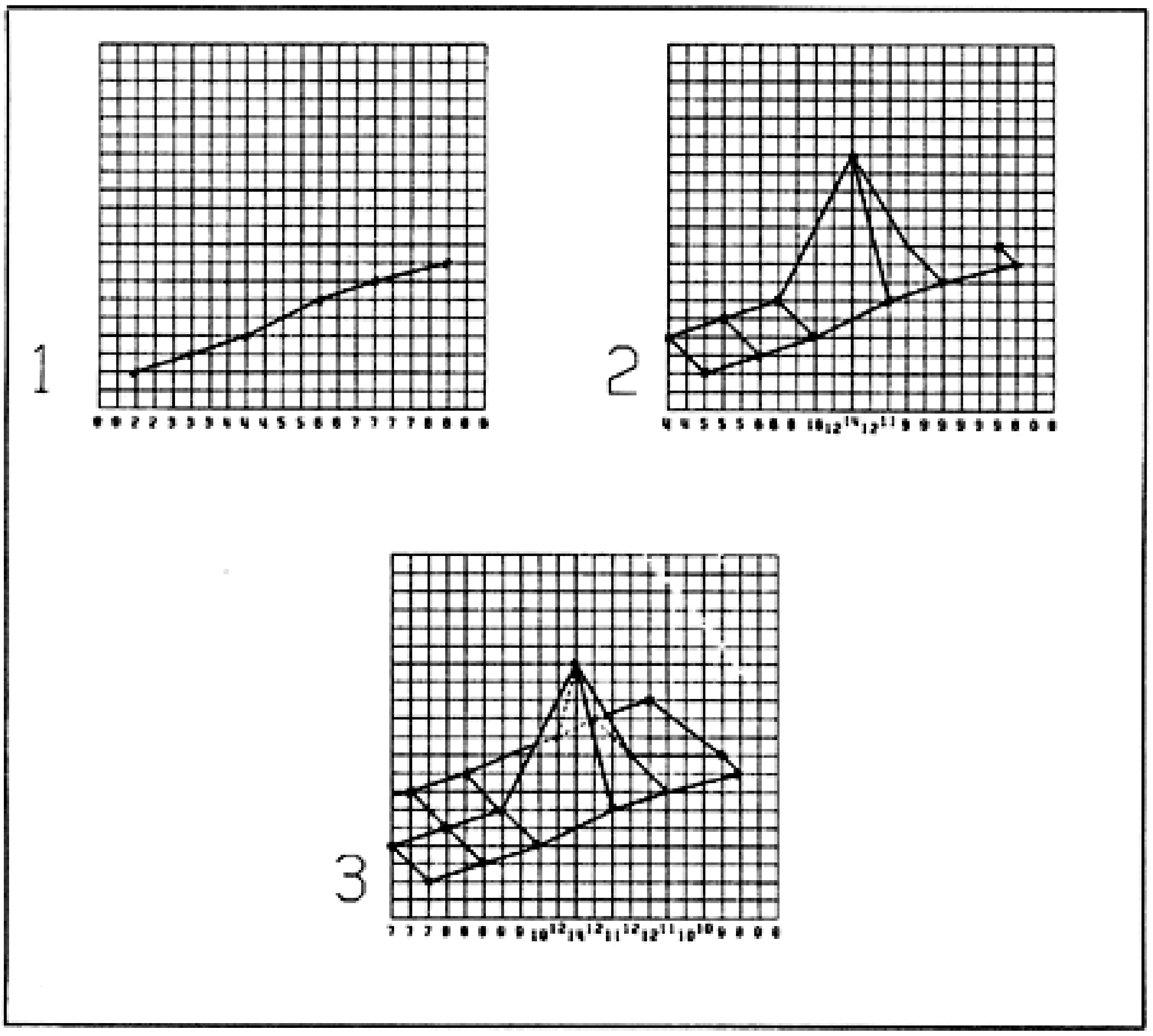


FIGURE 9 - VISIBILITY TEST. Below each picture is the current upperbound for the surface. If one of the end points of the segment is below the upper bound then the intersection of the segment and the surface is calculated. The segment is only drawn to the intersection.

7.6 STEREO VIEWS

Since the user can specify the view of the surface it is very easy to generate stereo views of surfaces. Below is a stereo view generated by GRAFIT. This view is mirrored. It may be viewed by placing a mirror between the two views. One eye should look directly at the first picture, while the second eye should view the reflection of the second picture.

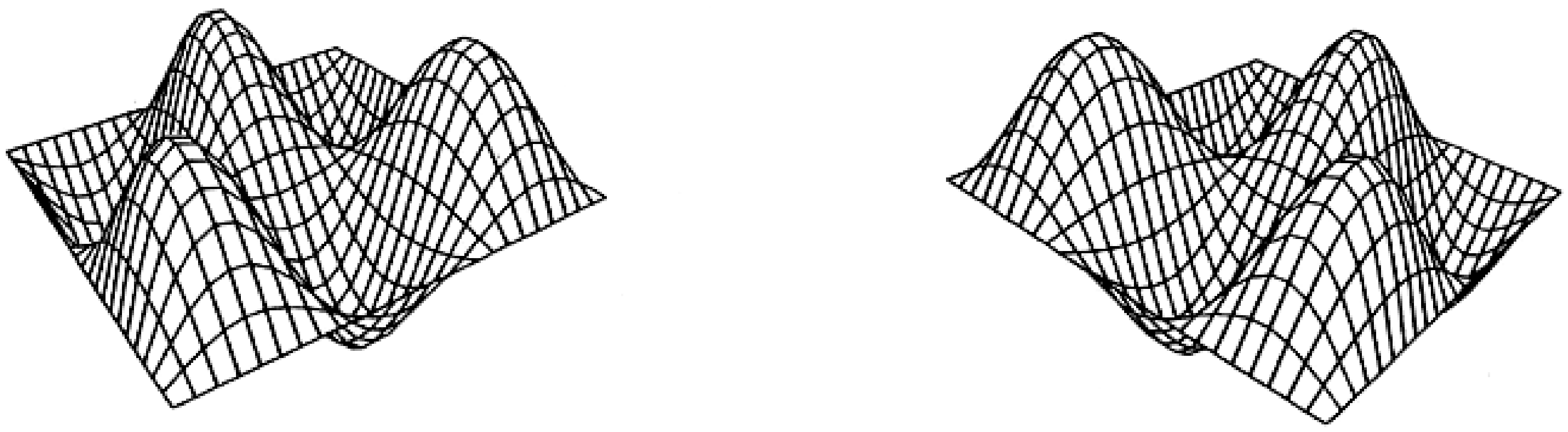


FIGURE 10 - STEREO VIEWS GENERATED BY GRAFIT.

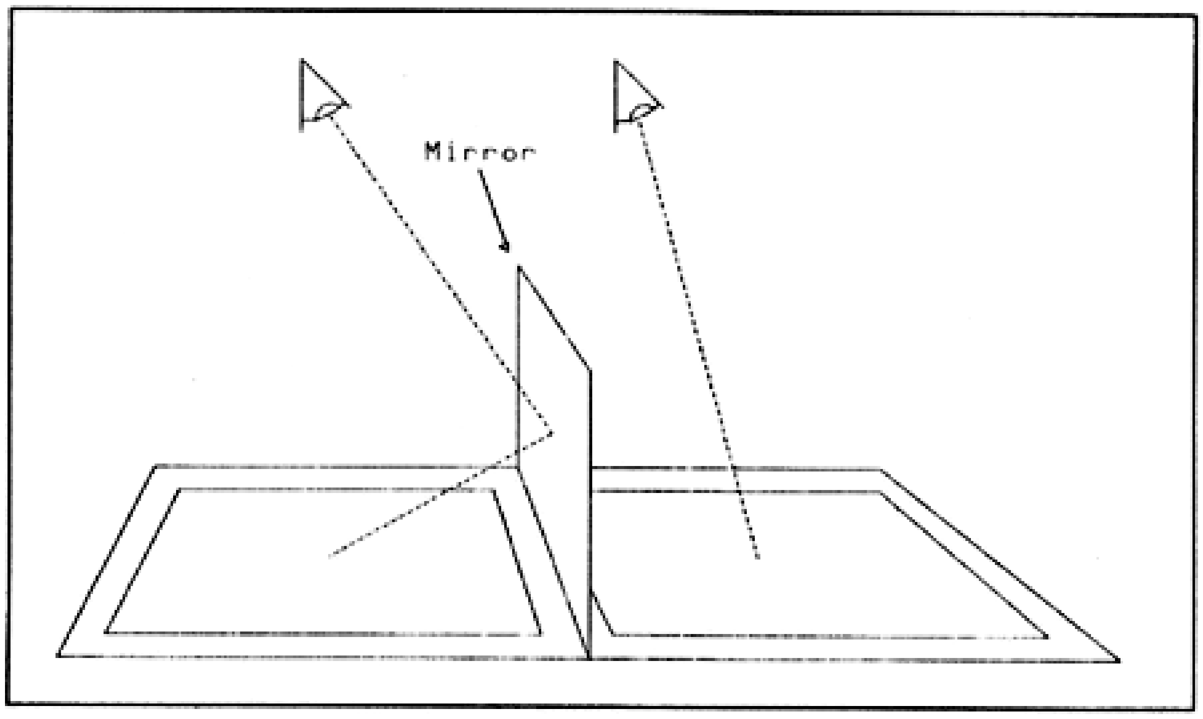


FIGURE 11 - VIEWING THE STEREO VIEWS GENERATED BY GRAFIT.

8 - WAVE EQUATIONS

8.1 WAVE EQUATIONS

My Senior Project considered a special class of partial differential equations (PDE) known as wave equations. A wave equation is a linear second order PDE of the form:

$$\nabla^2 U = \partial^2 U/\partial t^2 \qquad (8\text{-}1)$$

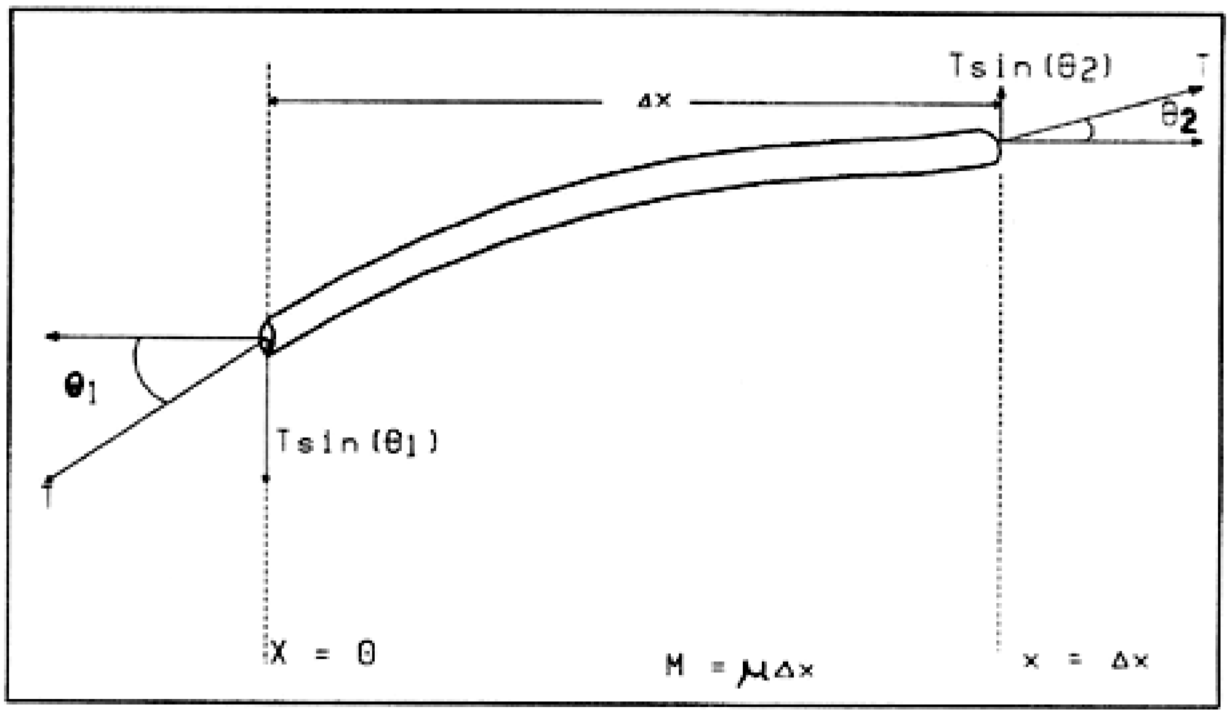
A set of conditions on the continuity of U and its derivatives are known as boundary conditions. The wave equation and boundary conditions constitute a boundary value problem. The solution of a boundary value problem is known as a wave function. A wave function is some function that will satisfy the wave equation and any boundary conditions that are imposed on the wave equation.

8.2 1-D WAVE EQUATION

The 1-D wave equation has the form:

$$\partial^2 y/\partial x^2 = (1/v^2)\partial^2 y/\partial t^2 \qquad (8\text{-}2)$$

The 1-D wave equation can be derived from Newton's second law, $\vec{F} = m\vec{a}$ applied to a small segment of the string. Consider the figure:



FIGIRE 12 - SEGMENT OF STRING USED TO DERIVE THE 1-D WAVE EQUATION. The string has a linear density (mass per unit length) of .

Several assumptions must be made to simplify the derivation of the wave equation. We must assume that the amplitude of the wave is small enough for θ and θ to be small. This gives several simplifications:

1) x is ~ the length of the segment.
2) sinθ ~ tanθ

Applying Newton's second law:

$$T\sin(\theta_1) - T\sin(\theta_2) = ma \qquad (8\text{-}3)$$

But for small angles sin(θ) ~ tan(θ):

$$T[\tan(\theta_1) - \tan(\theta_2)] = ma \qquad (8\text{-}4)$$

substituting $m = \mu\Delta x$ and $a = d^2y/dt^2$:

$$T[\tan(\theta_1) - \tan(\theta_2)] = (\mu\Delta x)d^2y/dt^2 \qquad (8\text{-}5)$$

But tan(θ) = dy/dx:

$$T[(dy/dx)|_{x=o} - (dy/dx)|_{x=\Delta x}] = (\mu\Delta x)d^2y/dt^2 \qquad (8\text{-}6)$$

Dividing through by Δx:

$$T[(dy/dx)|_{x=o} - (dy/dx)|_{x=\Delta x}]/\Delta x = d^2y/dt^2 \qquad (8\text{-}7)$$

But: $\lim_{x \to o}([(dy/dx)|_{x=o} - (dy/dx)|_{x=\Delta x}]/\Delta x) = (d^2y/dx^2)$

Therefore:

$$\partial^2y/\partial x^2 = (\mu/T)\partial^2y/\partial t^2 \qquad (8\text{-}8)$$

Where the velocity of the media is:

$$v = (T/\mu)^{1/2} \qquad (8\text{-}9)$$

8.3 2-D WAVE EQUATION

The 2-D wave equation in cartesian coordinates has the form:

$$\partial^2 y/\partial x^2 + \partial^2 y/\partial z^2 = (1/v^2)\partial^2 y/\partial t^2 \qquad (8\text{-}10)$$

The 2-D wave equation can be similarly derived from Newton's second law, $\vec{F} = m\vec{a}$, applied to a small segment of a square membrane. Consider the figure:

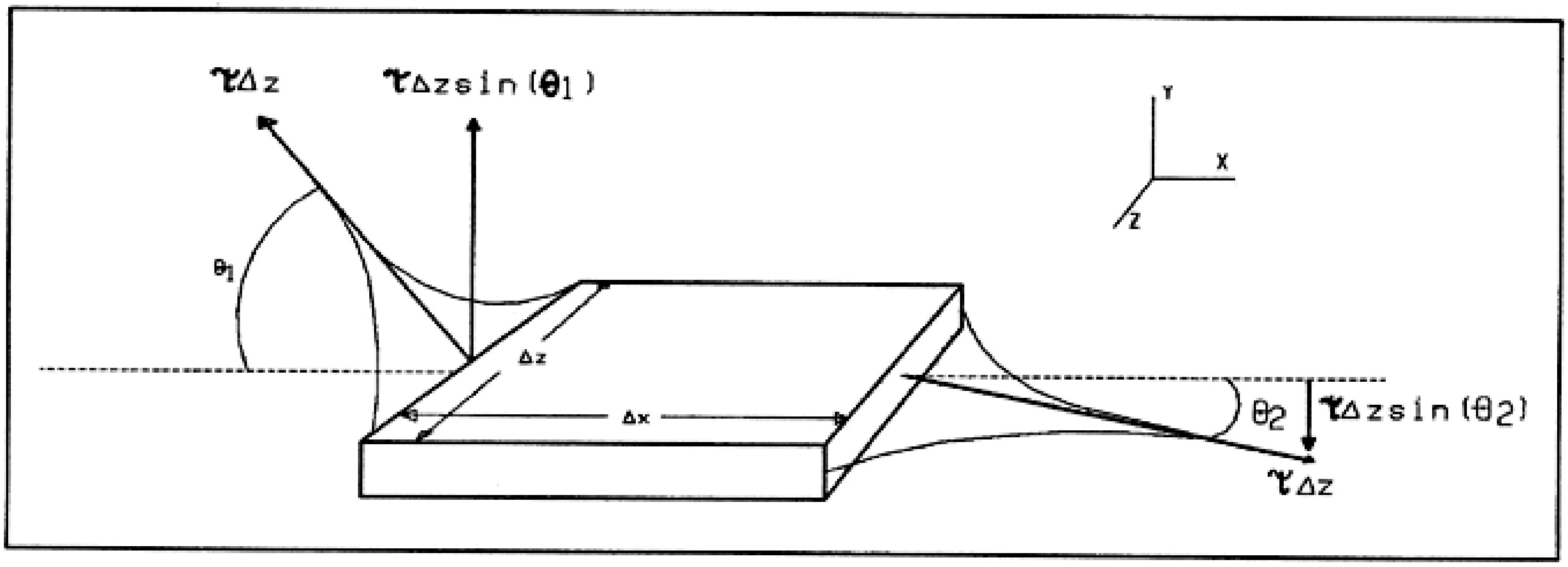


FIGURE 13 - SEGMENT OF MEMBRANE USED TO DERIVE THE 2-D WAVE EQUATION. The membrane has a surface density (mass per unit area) of ρ. τ Is the linear tension (tension per unit length).

Following the assumptions made for the 1-D equation, this equation may be derived in a similar fashion. The force contribution in the y direction due to the x direction, F_x:

$$(\tau\Delta z)\sin(\theta_1) - (\tau\Delta z)\sin(\theta_2) = F_x \qquad (8\text{-}11)$$

But for small angles $\sin(\theta) \sim \tan(\theta)$:

$$(\tau\Delta z)[\tan(\theta_1) - \tan(\theta_2)] = F_x \qquad (8\text{-}12)$$

But $\tan(\theta) = dy/dx$ and $F_x = ma_x = \rho\Delta x\Delta z(a_x)$:

$$(\tau\Delta z)[(dy/dx)|_{x=o} - (dy/dx)|_{x=\Delta x}] = \rho\Delta x\Delta z(a_x) \tag{8-13}$$

Dividing through by $\rho\Delta x\Delta z$:

$$(\tau/\rho)[(dy/dx)|_{x=o} - (dy/dx)|_{x=\Delta x}]/\Delta x = (a_x) \tag{8-14}$$

But, $\lim_{x\to o}([(dy/dx)|_{x=o} - (dy/dx)|_{x=\Delta x}]/\Delta x) = (d^2y/dx^2)$

Therefore:

$$d^2y/dx^2 = (\rho/\tau)a_x \tag{8-15}$$

similarly:

$$d^2y/dz^2 = (\rho/\tau)a_z \tag{8-16}$$

The total force on the membrane is the sum of the x and z contributions:

$$a_x + a_z = d^2y/dt^2 \tag{8-17}$$

Thus:

$$\partial^2y/\partial x^2 + \partial^2y/\partial z^2 = (\rho/\tau)\partial^2y/\partial t^2 \tag{8-18}$$

Where the velocity of the media is:

$$v = (\tau/\rho)^{1/2} \tag{8-19}$$

8.4 SOLUTIONS TO WAVE EQUATIONS

Many techniques exist to solve PDEs. I chose two different techniques to solve the wave equation:

1) D'Alemberts solution
2) Separation of variables

9 - MOVIE 1, D'ALEMBERT'S SOLUTION TO THE 1-D WAVE EQUATION

9.1 - D'ALEMBERT'S SOLUTION TO THE 1-D WAVE EQUATION

The most general solution to the 1-D wave equation is:

$$Y(x,t) = 1/2[f(x + vt) + f(x - vt)] \tag{9-1}$$

Where $f(x)$ is a function that represents the shape of the string at time $t = 0$, and v is the wave velocity of the string. The string's motion can be thought of as the averaged superposition of two traveling waves moving in opposite directions. The story boards for the first movie visually demonstrate this.

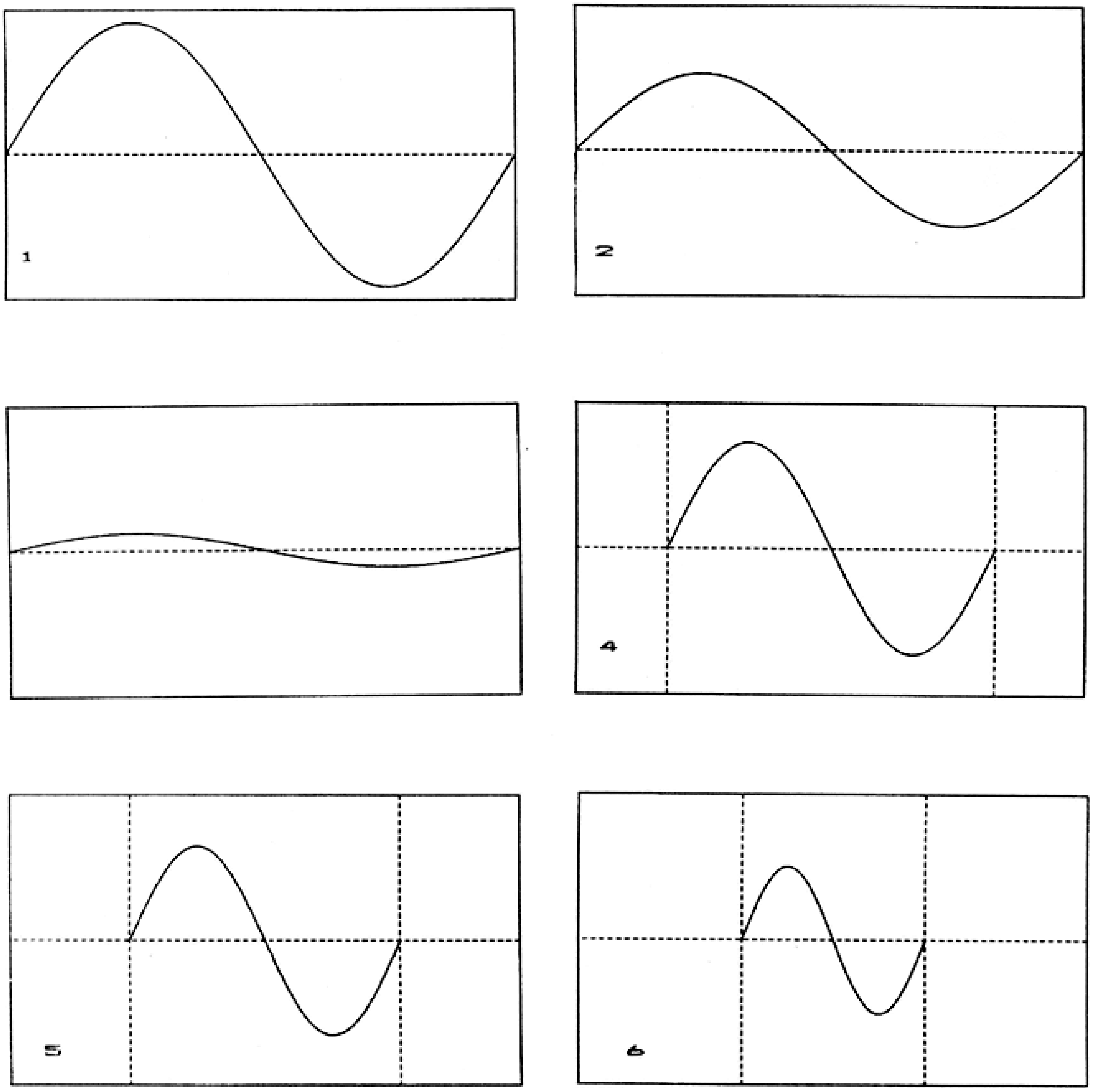


FIGURE 14 - STORY BOARD FOR D'ALEMBERTS SOLUTION TO THE 1-D WAVE EQUATION. Frames 1 through 3: the movie shows a string in the second mode. Frames 4 through 6 the string stops vibrating and the screen is panned back from the string.

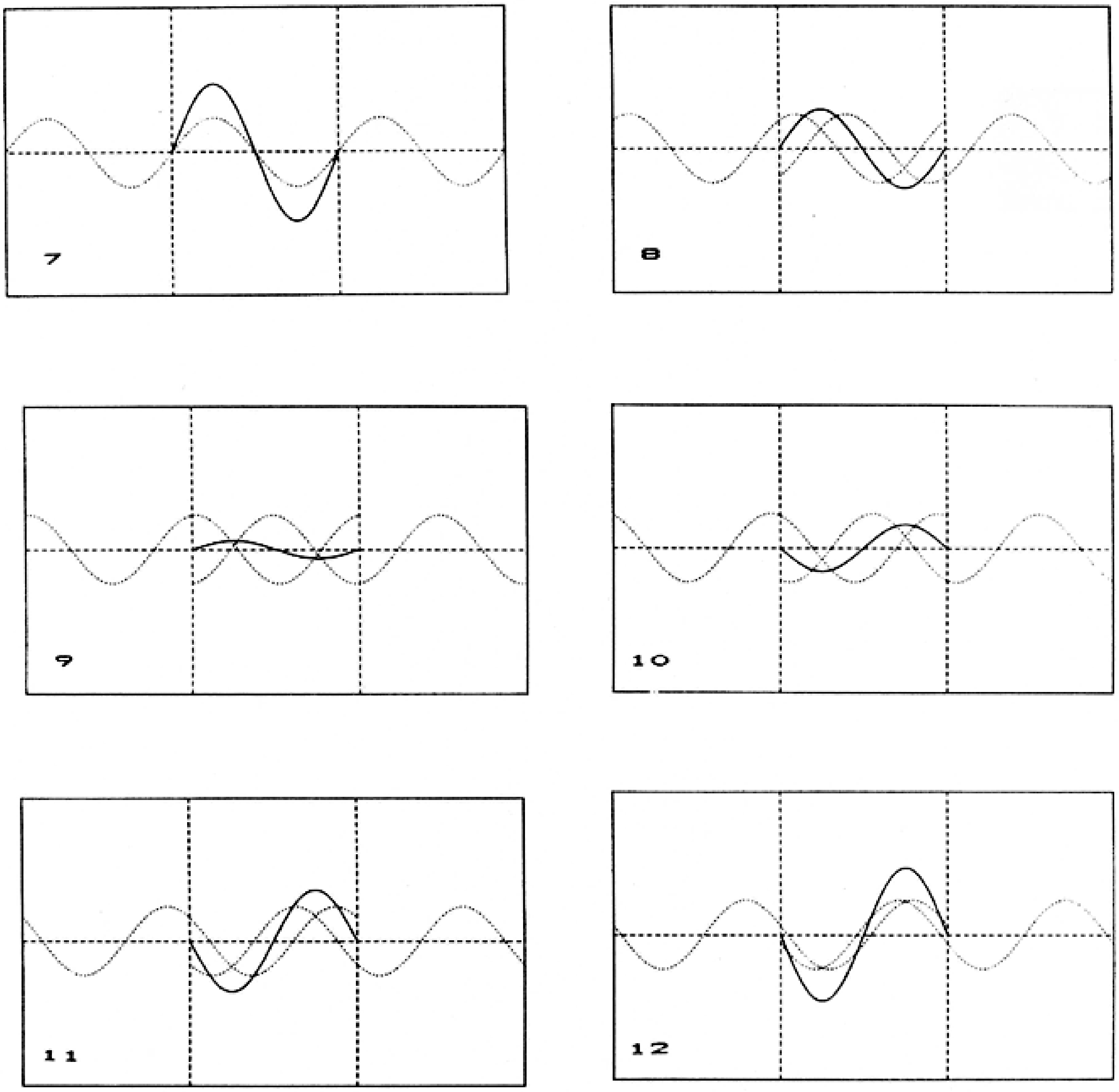


FIGURE 15 - Frames 6 through 12: we see that the string's motion is a superposition of two traveling waves moving in opposite directions.

$$Y(x,t) = 1/2[f(x + vt) + f(x - vt)]$$

10 - MOVIE 2, FOURIER SOLUTION TO TO THE 1-D WAVE EQUATION

10.1 SEPARATION OF VARIABLES

This is a powerful technique for solving PDEs. Separation of variables can be applied to the 1-D wave equation boundry value problem. Adopting a notion that is easier to type:

$$y_{xx} = (1/v^2) y_{tt} \qquad (10\text{-}1)$$
$$y(0,t) = 0 \quad \text{Right end bound} \qquad (10\text{-}2)$$
$$y(c,t) = 0 \quad \text{Left end bound} \qquad (10\text{-}3)$$
$$y(x,0) = f(x) \quad \text{Initial shape of the string} \qquad (10\text{-}4)$$
$$y_t(x,0) = 0 \quad \text{Initial velocity is zero} \qquad (10\text{-}5)$$

Where c is the length of the string and v is the wave velocity. We assume that y(x,t) is the product of a spatially dependent function and a temporally dependent function.

$$y(x,t) = X(x)T(t) \qquad (10\text{-}6)$$

Substituting this expression into the wave equation and separating variables gives:

$$(X''(x)/X(x)) = (T''(t)/v^2T(t)) = -\lambda \qquad (10\text{-}7)$$

Both sides are equal to a constant. The choice of $-\lambda$ is for the convevience of notation. The seperation can be completed to form two ODEs. These ODEs plus boundry conditions form a Sturm-Liouville Problem.

$$X''(x) + \lambda X(x) = 0 \qquad (10\text{-}8)$$
$$T''(t) + (\lambda v^2)T(t) = 0 \qquad (10\text{-}9)$$
$$X(0) = 0 \qquad (10\text{-}10)$$
$$X(c) = 0 \qquad (10\text{-}11)$$
$$T'(0) = 0 \qquad (10\text{-}12)$$

$-\infty < > < \infty$ Thus, three possible problems must be considered: $\lambda < 0$, $\lambda = 0$, and $\lambda > 0$. It can be shown that all cases except $\lambda > 0$ yield trivial solutions. Let $\lambda = -b^2$, The solutions to the ODEs are:

$$X(x) = A\sin(bx) + B\cos(bx) \qquad (10\text{-}13)$$
$$T(t) = C\sin(bv^2t) + D\cos(bv^2t) \qquad (10\text{-}14)$$

Applying the boundry conditions gives:

$$B = 0 \qquad (10\text{-}15)$$
$$C = 0 \qquad (10\text{-}16)$$
$$b = n\pi/c \qquad (10\text{-}17)$$

The numbers $= n^2\pi^2/c^2$ are called eigenvalues.

Substituting the eigenvalues into the solutions gives a set of solutions known as eigenfunctions:

$$X_n(x) = A\sin(n\pi x/c) \qquad (10\text{-}18)$$
$$T_n(t) = D\cos(n\pi t/vc) \qquad (10\text{-}19)$$

Forming the product solution:

$$y_n(x,t) = b_n\sin(n\pi x/c)\cos(n\pi vt/c) \qquad (10\text{-}20)$$

A linear combination of solutions to a PDE is also a solution to the PDE. This is known as the principle of superposition. The next step is to sum up all of the eigen functions into a single solution:

$$y(x,t) = \sum_{n=1}^{\infty} b_n\sin(n\pi x/c)\cos(n\pi vt/c) \qquad (10\text{-}21)$$

If we apply the final condition regarding the initial shape of the string (10-4):

$$f(x) = \sum_{n=1}^{\infty} b_n\sin(n\pi x/c) \qquad (10\text{-}22)$$

If we multiply both sides by sin(mπx/c) and integrate over the length of the string.

$$\int_0^c f(x)\sin(m\pi x/c)dx = \sum_{n=1}^{\infty} b_n\int_0^c \sin(n\pi x/c)\sin(m\pi x/c)dx \qquad (10\text{-}23)$$

But

$$\int_0^c \sin(n\pi x/c)\sin(m\pi x/c)dx = (c/2)\delta_{nm} \qquad (10\text{-}24)$$

Thus the only non-zero members of the series are when n=m. The coefficients b_n can be given the values:

$$b_n = (2/c)\int_0^c f(x)\sin(n\pi x/c)dx \quad (n = 1,2, \ldots) \qquad (10\text{-}25)$$

The complete solution to the 1-D wave equation is thus:

$$\boxed{\begin{aligned} y(x,t) &= \sum_{n=1}^{\infty} b_n\sin(n\pi x/c)\cos(n\pi vt/c) & (10\text{-}26)\\ b_n &= (2/c)\int_0^c f(x)\sin(n\pi x/c)dx \quad (n = 1,2, \ldots) & (10\text{-}27)\end{aligned}}$$

10.2 The Plucked String

The plucked string is a special case of the 1-D wave equation. In general for a string of height h with a length c the initial shape is defined as:

$$f(x) = \left[\begin{array}{ll} 2h(x/c) & 0 < x < c/2 \\ 2h(1-(x/c)) & c/2 < x < c \end{array}\right. \qquad (10\text{-}28)$$

b_n can be calculated from 10-16 as:

$$b_n = (4hc/n^2\pi^2)\sin(n\pi/2) \qquad (10\text{-}29)$$

This gives us a complete solution:

$$y(x,t) = (4hc/\pi^2)\sum_{n=1}^{\infty} (1/n^2)\sin(n\pi x/c)\cos(n\pi vt/c) \tag{10-30}$$

Let $k_n = (n\pi/c)$ and $w_n = (n\pi v/c)$ we have:

$$\boxed{\begin{aligned} y(x,t) &= (4hc/\pi^2)\sum_{n=1}^{\infty}(1/n^2)\sin(k_n x)\cos(w_n t) && (10\text{-}31) \\ w_n &= vk_n && (10\text{-}32) \end{aligned}}$$

10.3 PHYSICAL INTERPRETATION

First this general solution to the wave is intended to model the behavior of a plucked string. The solution is a function of space and time. The equation for the solution has four distinct parts. The constant in front of the summation scales the total solution for the plucked string. The sine terms are the spatial harmonics that describe the initial shape of the string. the sum of the spatial harmonics gives the complete shape of the string. the cosine terms are the temporal harmonics that describe the motion of the string. the sum of the temporal harmonics gives the complete motion of the The temporal harmonics are proportional to the frequencies that the string is vibrating with. The $(1/n^2)$ indicates that the contribution of the spatial and temporal terms each fall off like $(1/n)$, the harmonic series.

The spatial part of the solution is fixed with respect to time. The spatial part of the solution is modulated by the temporal part of the solution. w_n is related to the freqencies that the string is vibrating at:

$$w_n = 2\pi f_n \tag{10-33}$$

k_n is related to the wavelength of a particular frequency:

$$k_n = 2\pi/\lambda \tag{10-34}$$

Note that the ratio of (w_n/k_n) is always the velocity. This says that each temporal harmonic of the solution has the same velocity. If this were not true then the media would be dispersive. We assumed initially that our media was non-dispersive, that is v was a constant in the 1-D wave equation. A real string is probably dispersive, however this solution provides a close model for the behavior of the string.

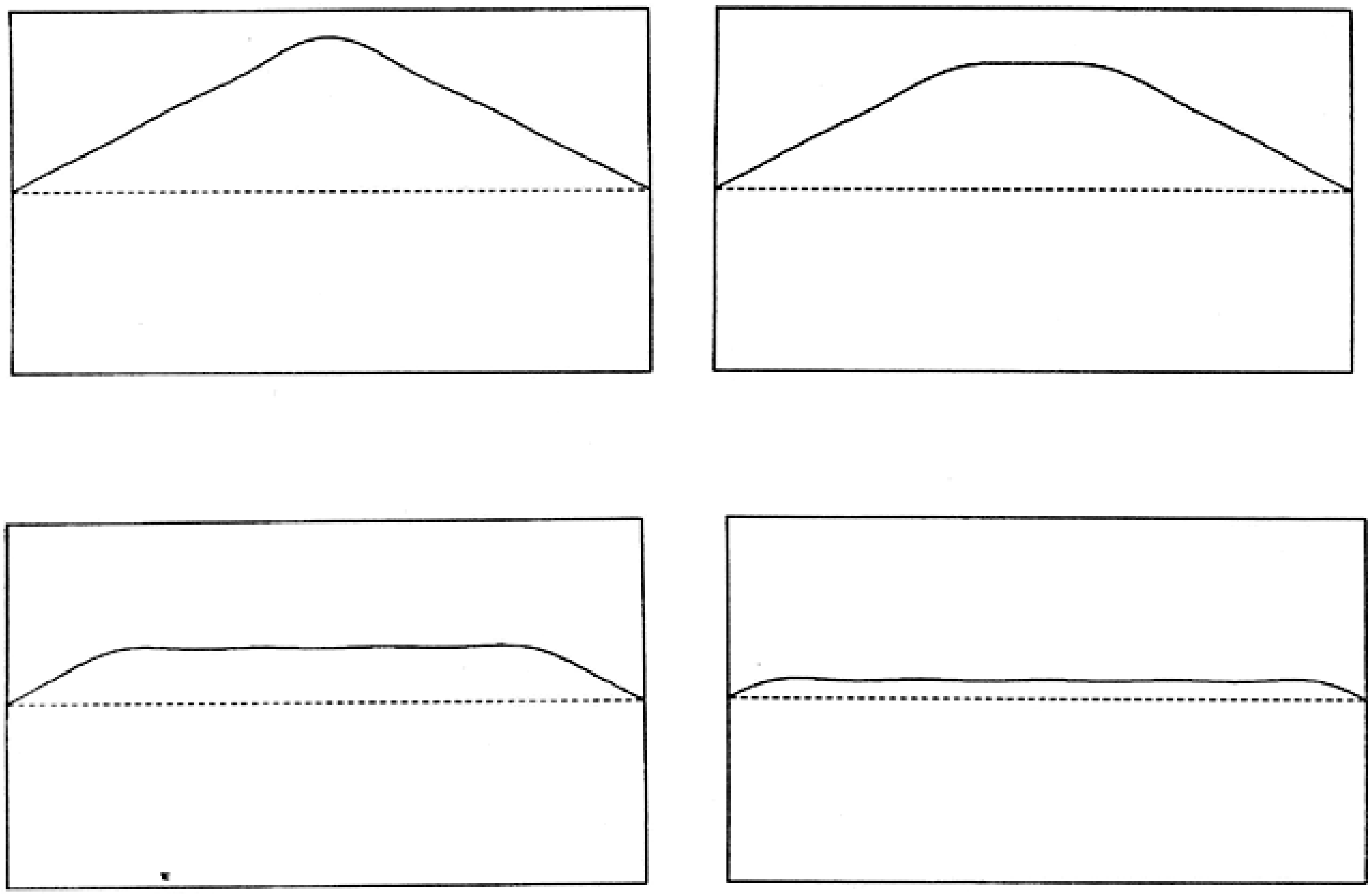


FIGURE 16 - STORY BOARD FOR THE A MOVIE OF THE PLUCKED STRING. This movie demonstrates the time dependent Fourier solution to the plucked string.

$$y(x,t) = (4hc/\pi^2)\sum_{n=1}^{oo} (1/n^2)\sin(k_n x)\cos(w_n t)$$

$w_n = (n\pi v/c)$
$k_n = (n\pi/c)$
h = initial height of the string
v = the velocity of sound in the string
c = the length of the string

11 – MOVIE 3, FOURIER SOLUTION TO THE VIBRATING MEMBRANE

11.1 THE PROBLEM

Consider a flat membrane that is bound on all sides. The membrane is initially displaced with no initial velocity. The problem of the vibrating membrane can be stated as:

$$
\begin{array}{lll}
y_{xx} + y_{zz} = (1/v^2)y_{tt} & & (11\text{–}1)\\
y(0,0,t) = 0 & \left.\right] & (11\text{–}2)\\
y(a,0,t) = 0 & \text{— All sides bound} & (11\text{–}3)\\
y(0,b,t) = 0 & & (11\text{–}4)\\
y(a,b,t) = 0 & & (11\text{–}5)\\
y(x,z,0) = f(x,z) & \text{Initial shape} & (11\text{–}6)\\
y_t(x,z,0) = 0 & \text{Initial velocity is zero} & (11\text{–}7)
\end{array}
$$

11.2 SOLUTION

Using the technique of seperation of variables the solution to this problem can be found.

$$y(x,z,t) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} A_{mn}\sin(k_m x)\sin(k_n z)\cos(w_{mn}t) \qquad (11\text{–}8)$$

$$A_{mn} = (4/ab)\int_0^a\int_0^b f(x,z)\sin(k_m x)\sin(k_n x)\,dx\,dz \qquad (11\text{–}9)$$

$$w_{mn} = (k_n^2+k_m^2)v \qquad (11\ 10)$$

$$k_m = (m\ /a) \qquad (11\text{–}11)$$

$$k_n = (n\ /a) \qquad (11\text{–}12)$$

11.3 THE MOVIE

Because of the enormous number of calculations that are needed to calculate a complete solution to the vibrating membrane problem, I chose to display several of the temporal harmonics in motion.

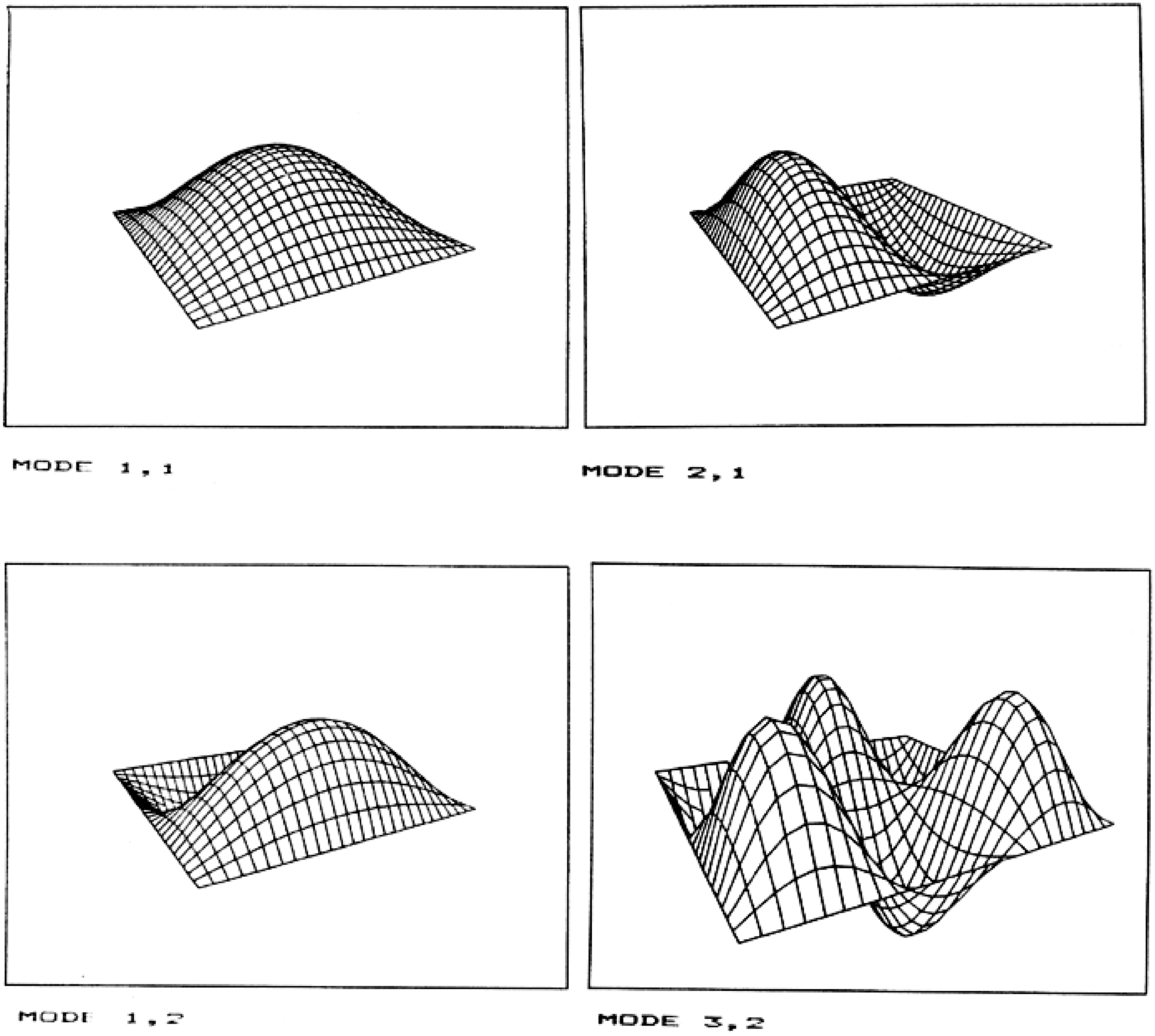


FIGURE 17 – SEVERAL MODES OF THE VIBRATING MEMBRANE

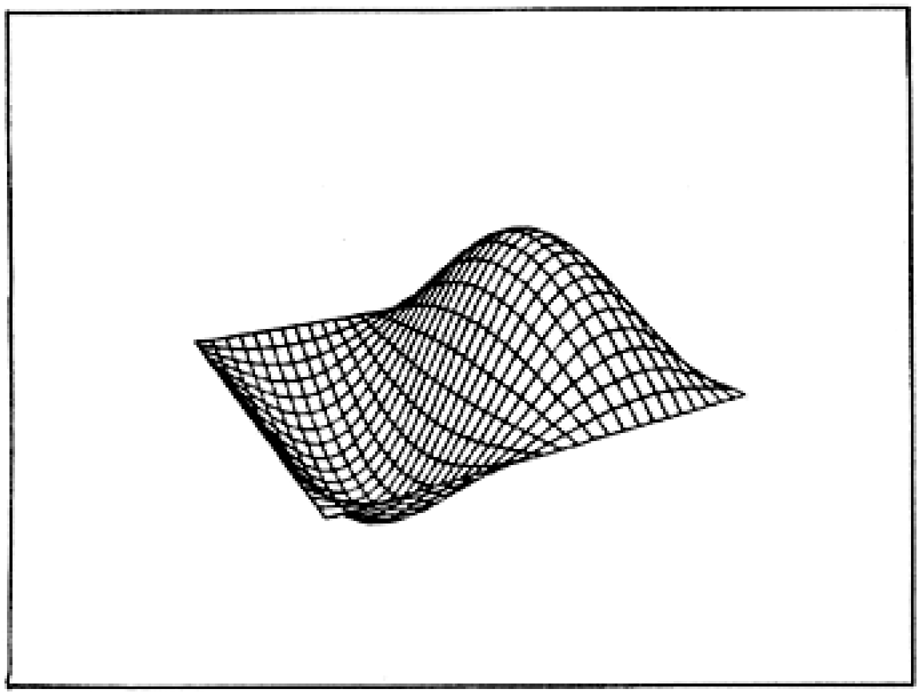
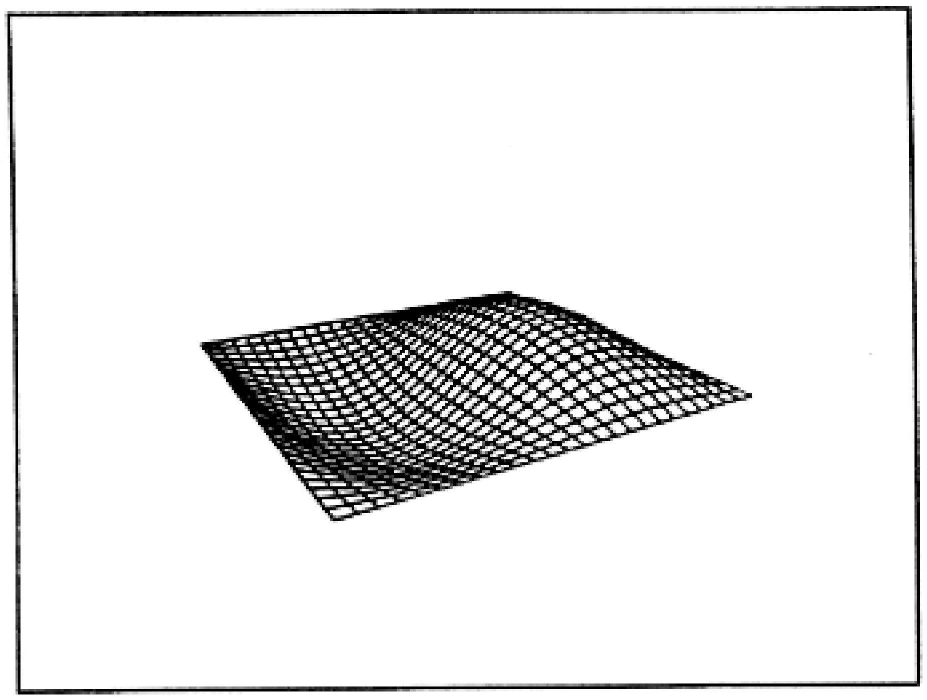
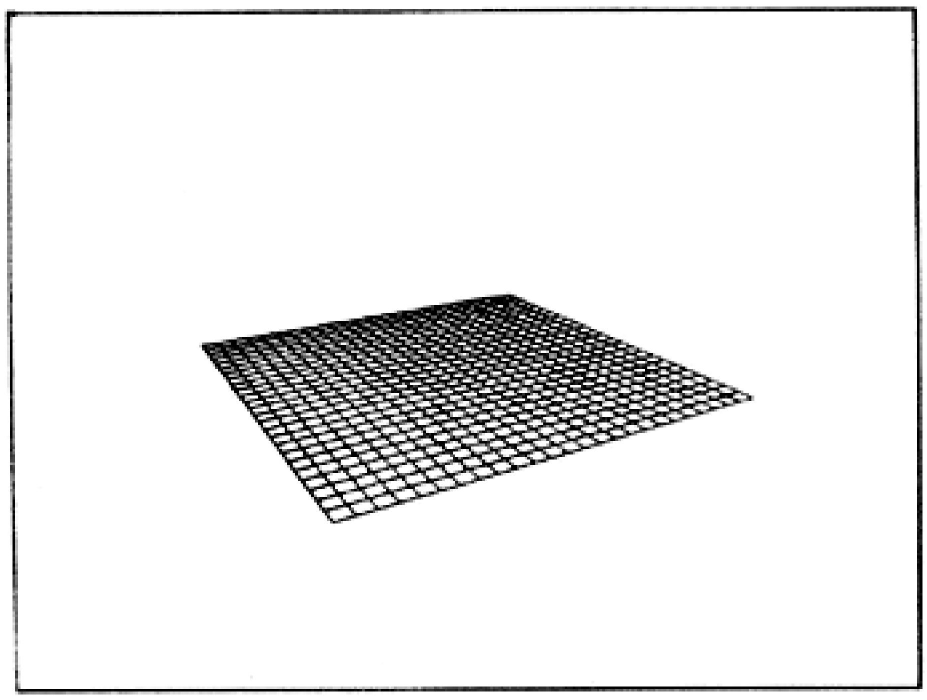
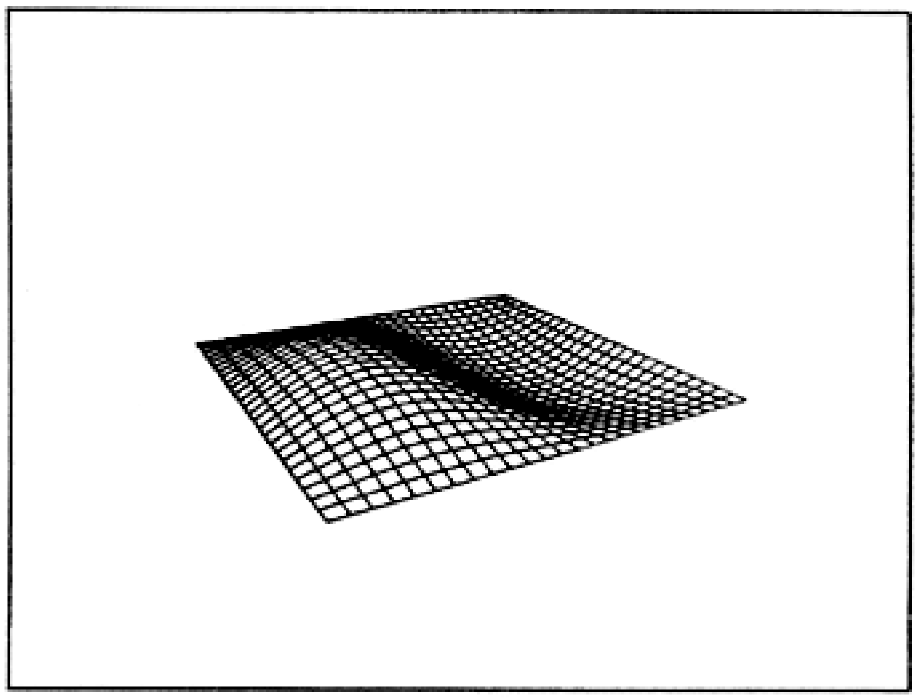
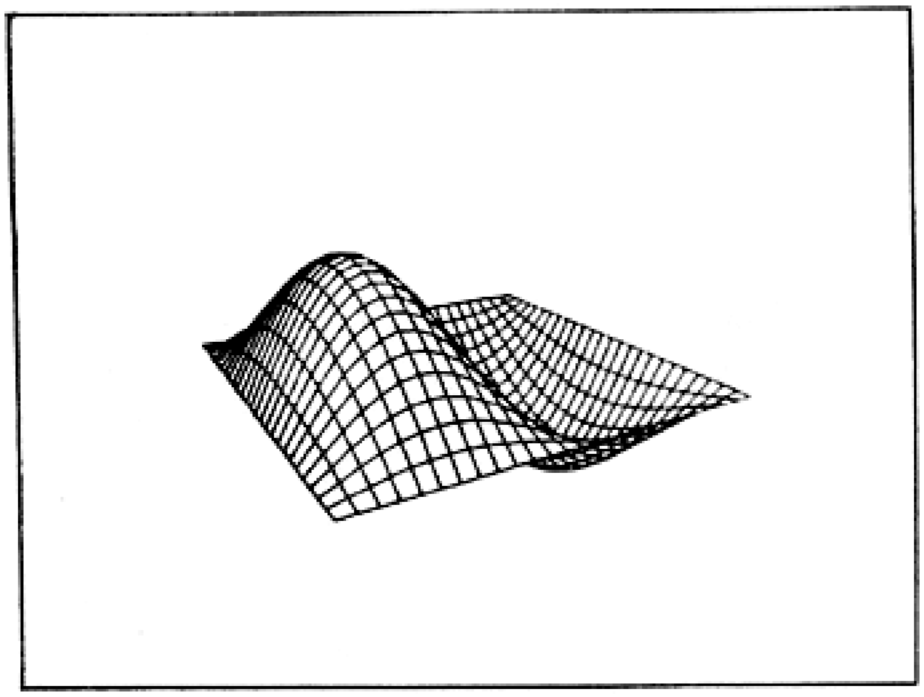






FIGURE 18 – STORY BOARD FOR A MOVIE OF THE VIBRATING SQUARE MEMBRANE. The membrane is in the 2,1 mode.

CONCLUSION

The project was quite a large one. I think that perhaps I bit off more than I could chew. In any case, I learned a great deal from the project. Almost all of the project was home brew, I pretty much started from scratch. The 3-D surface program was used by Ashod Shamelian to display the results of his senior project. Also, I did a colloquium for the Physics department on what I had done.

I would like to take this opportunity to thank the Cal Poly Physics department for all that they gave me in my years as a student.

APPENDIX 1

Listing of the Cyber Control Language (CCL) procedure file, MOVIE

```
.PROC,MOVIE,P,BATCH=FALSE/TRUE.
.*
.************************************************************************
.* GREGORY PAT SCANDALIS
.* MARCH 1,1983
.* THIS IA A CCL (CYBER CONTROL LANGUAGE PROCEDURE) THAT INTERFACES THE
.* GRAFIT PROGRAM WITH THE NOS OPERATING SYSTEM. THE PURPOSE OF THIS
.* PROCEDURE IS TO ATTACH THE GRAPHICS AND MATH LIBRARIES
.* AND TO COMPILE AND EXECUTE THE PROGRAM.  THIS PROCEDURE WILL ALSO
.* DISABLE THE NORMAL LIMITS ON CPU TIME.  THIS WILL ALLOW THE MOVIE
.* GENERATION PROGRAM TO RUN CONTINUOUSLY.
.*
.* INVOCATION:  -,MOVIE,PROGRAM_NAME,BATCH  (NO LINE NUMBERS)
.*              -,MOVIE,PROGRAM_NAME,BATCH  (WITH LINE NUMBERS)
.************************************************************************
.*
.************************************************************************
.* LOOK TO SEE IF THE FILE EXISTS
.************************************************************************
.*
$IFE, .NOT.FILE(P,AS), GET IT.
     $GET,P/NA
     $IFE, .NOT.FILE(P,AS), BAD NAME.
          $NOTE.+ +CANNOT FIND P IN THIS ACCOUNT!+ +
          $REVERT
     $ENDIF,BAD NAME.
$ENDIF,GET IT.
.*
.************************************************************************
.* REWIND THE SOURCE CODE FILE
.************************************************************************
.*
$REWIND,P.
.*
.************************************************************************
.* GET THE CYBER SYSTEM INTERFACE LIBRARY
.************************************************************************
.*
$IFE, .NOT.FILE(CSUBS,AS), GET IT 2.
     GET,CSUBS/UN=LETPLOO
$ENDIF, GET IT 2.
.*
.************************************************************************
.* ATTACH THE GRAPHICS SUBROUTINES
.************************************************************************
.*
$IFE, .NOT.FILE(TEKLBR,AS), GET IT 3.
     GET,CSUBS/UN=LIBRARY,NA
$ENDIF, GET IT 3.
.*
.************************************************************************
.* ATTACH THE MATH SUBROUTINES
.************************************************************************
.*
$IFE, .NOT.FILE(ZZZMATH,AS), GET IT 4.
     GET,CSUBS/UN=LBJZLOO.
$ENDIF, GET IT 4.
$RETURN,LGO,DOIT.
.*
.************************************************************************
```

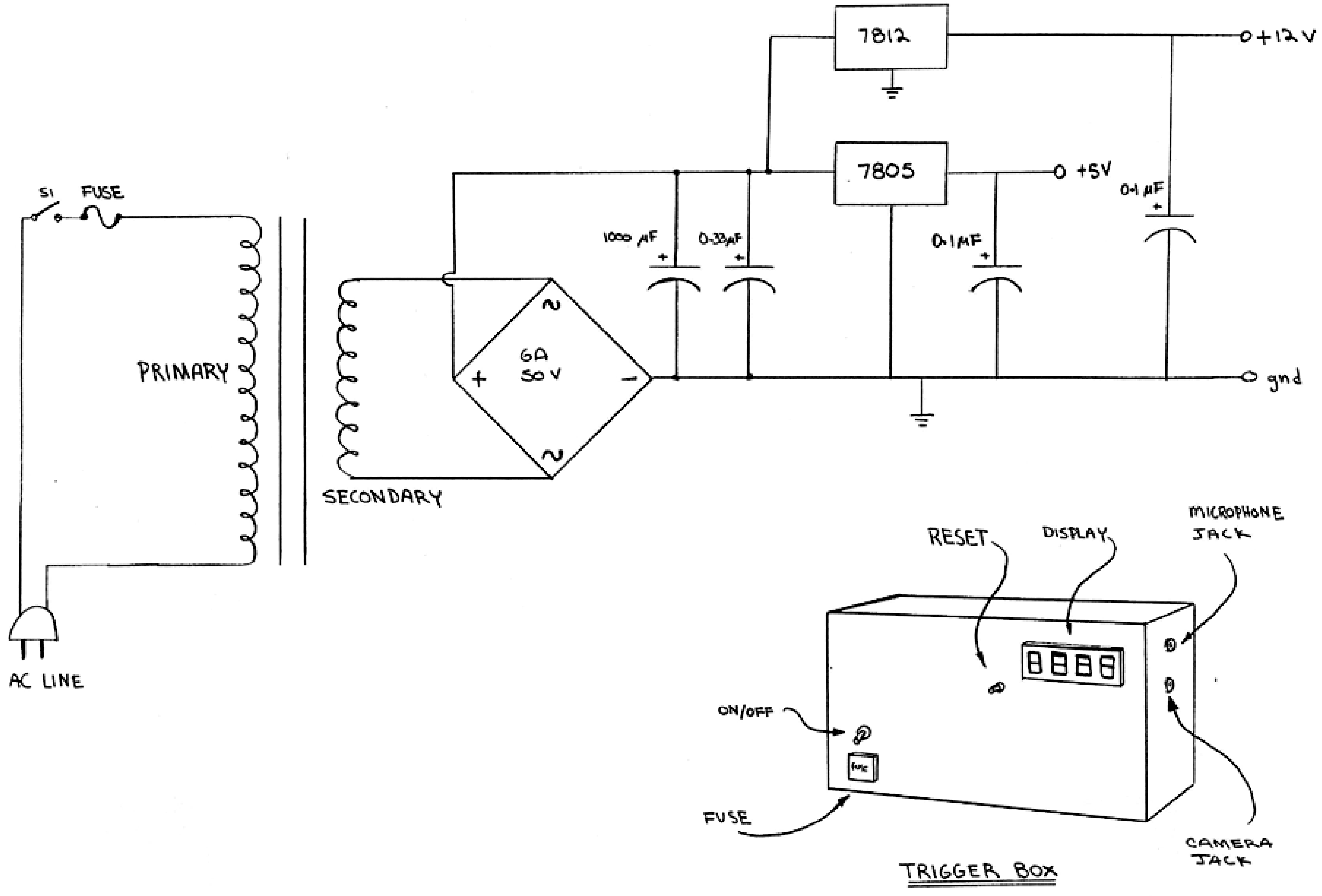
```
.*  CALL THE FTN COMPILER (FORTRAN), USE EITHER THE BATCH OPTION       *
.*  OR THE NO BATCH OPTION.                                            *
.***********************************************************************
.*
$IFE, BATCH, CARD IMAGE.
     $FTN,I=P,L=0,REW.
$ELSE, CARD IMAGE.
     $FTN,I=P,L=0,REW,TS,SEQ.
$ENDIF, CARD IMAGE.
.*
.***********************************************************************
.*  DISABLE THE TIME USE LIMITS                                        *
.***********************************************************************
.*
$SETTL,*.
$SETASL,*.
$SETJSL,*.
.*
.***********************************************************************
.*  LOAD AND EXECUTE, NO MODULE GENERATED.                             *
.***********************************************************************
.*
$LDSET,LIB=ZZZMATH/CSUBS/TEKLBR,PRESET=ZERO.
$LOAD(LGO)
$NOGO(DOIT)
DOIT.
$RETURN,LGO,DOIT.
.*
.***********************************************************************
.*  MAKE THE NAMED FILE THE PRIMARY                                    *
.***********************************************************************
.*
$IFE, .NOT.FILE(P,PT), MAKE PRI.
     $PRIMARY,P.
$ENDIF, MAKE PRI.
$REVERT.
```

APPENDIX 2

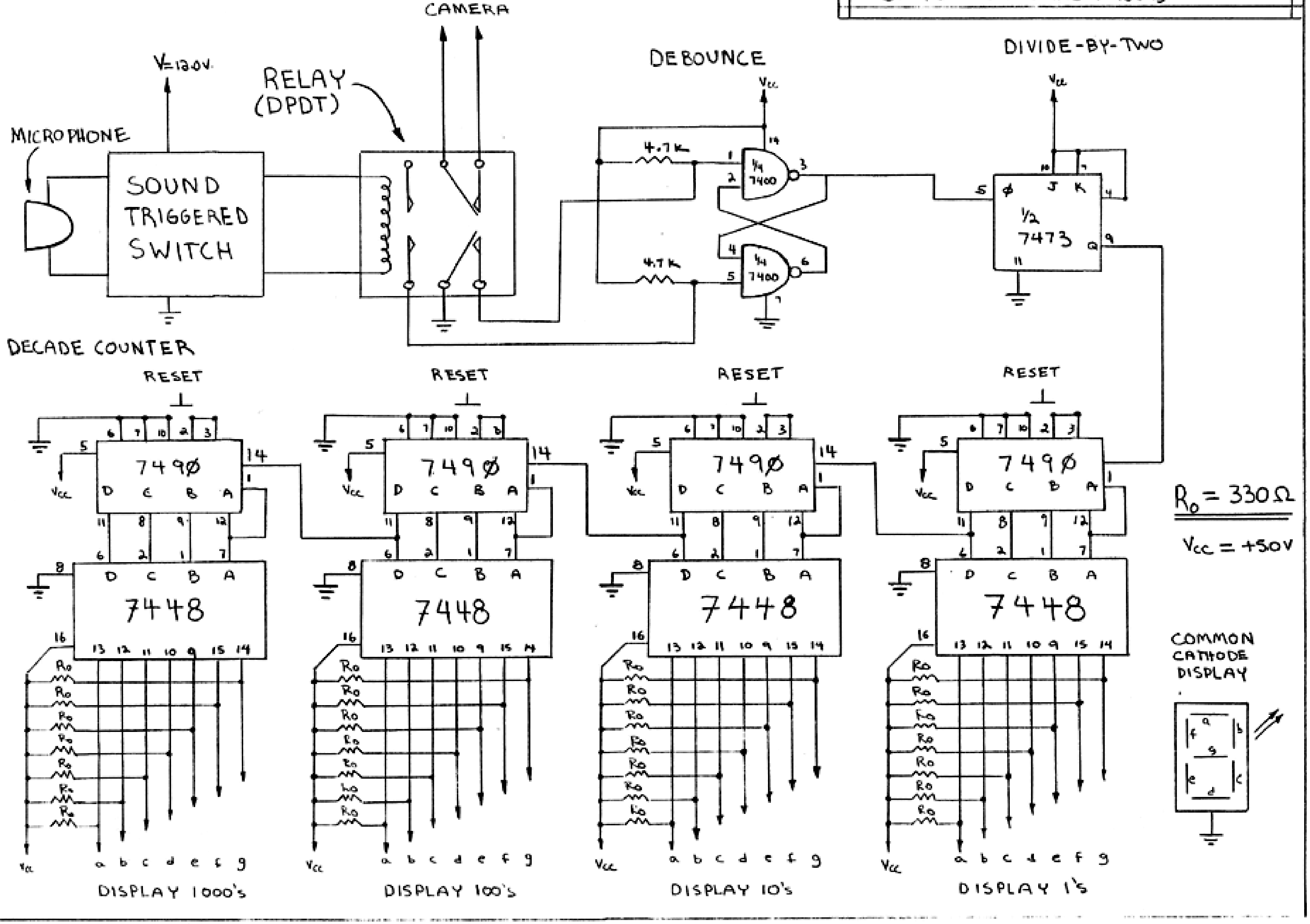
Schematic for the trigger box

Gregory Pat Scandalis 6/10/83

Power Supply for Trigger Box
+5.0 V and +12.0 V.



Gregory Pat Scandalis 6/10/83
Trigger Circuit for Senior Project
"Computer Generated Movies"



APPENDIX 3

Listing of GRAFIT

```
C
C
C
C*******************************************************************
C*******************************************************************
C
C
C
C             THE SUBROUTINE GRAFIT DOES THE 3-D GRAPHICS
C             THE DRIVER MUST PROVIDE THE COMMON ARRAYS:
C             X(26,26),Y(26,26),Z(26,26).  IN ADDITION
C             THE FOLLOWING PARAMETERS ARE PASSED THROUGH
C             GRAFIT:     R- HOW FAR AWAY THE SURFACE IS
C                         THETA- THE AZIMUTHAL ANGLE
C                         PHI- THE COLATITUDE ANGLE
C                         D- THE DISTANCE OF THE OBSERVER
C                         TO THE PROJECTION PLANE
C
C
C
C********************************************************************
C********************************************************************
C
C
C
      SUBROUTINE GRAFIT(R,THETA,PHI,D,IFRAME)
      DIMENSION XS(26,26),YS(26,26),DEPTH(4),HMAX(1024),HMIN(1024)
      COMMON/POINT/X(26,26),Y(26,26),Z(26,26)
+           /DATA/XSC,YSC,COSTH,COSPHI,SINTH,SINPHI,RR,DD
+           /HEIGHT/HMAX(1024),HMIN(1024)
      DO 10 I=1,1024
      HMAX(I)=-38.95
      HMIN(I)=38.95
   10 CONTINUE
      THETAA=THETA*0.017453
      PHII=PHI*0.017453
      COSTH=COS(THETAA)
      COSPHI=COS(PHII)
      SINTH=SIN(THETAA)
      SINPHI=SIN(PHII)
      RR=R
      DD=D
C
C
C
```

```
C**************************************************************
C
C      TRANSFORM AND MAP THE 3-D DOWN TO 2-D
C
C**************************************************************
C
C
C
      DO 400 I=1,26
      DO 300 J=1,26
      CALL TRNSFM(X(I,J),Y(I,J),Z(I,J))
      XS(I,J)=XSC
      YS(I,J)=YSC
  300  CONTINUE
  400  CONTINUE
C
C
C
C**************************************************************
C
C      SET UP THE VIEWPORT
C
C**************************************************************
C
C
C
C
C
C
C**************************************************************
C
C
C      IN ORDER TO HIDE THE OBSCURED LINES GRAFIT MUST
C      DRAW FROM FRONT TO BACK.  THIS NEXT SECTION WILL
C      LOCATE THE CLOSEST AXIS AND SET IFLAG.
C      THE VALUE OF IFLAG IS USED TO IMPLEMENT A CASE CONSTRUCT
C      IF IFLAG= 1     DO MODULE 1
C                2     DO MODULE 2
C                3     DO MODULE 3
C                4     DO MODULE 4
C
C      EACH MODULE IS DESIGNED TO DRAW THE CLOSEST PART OF
C      OF THE SURFACE FIRST.
C
C**************************************************************
C
C
C
      DEPTH(1)=ZZ(X(16,1),Y(16,1),Z(16,1))
      DEPTH(2)=ZZ(X(26,16),Y(26,16),Z(26,16))
      DEPTH(3)=ZZ(X(16,26),Y(16,26),Z(16,26))
      DEPTH(4)=ZZ(X(1,16),Y(1,16),Z(1,16))
      IFLAG=1
      DO 500 I=2,4
      IF(DEPTH(I).LT.DEPTH(IFLAG)) IFLAG=I
  500 CONTINUE
      GO TO(600,700,800,900),IFLAG
```

```
C
C
C
C***************************************************************
C
C
C      MODULE 1 STARTS AT 600
C
C
C****************************************************************
C
C
C
 600  DO 630 J=1,26
      CALL MOVEA(XS(1,J),YS(1,J))
      DO 610 I=2,26
      CALL PLOTEST(XS(I-1,J),YS(I-1,J),XS(I,J),YS(I,J))
  610  CONTINUE
      IF(J.EQ.26) GO TO 1000
      DO 620 I=1,26
      CALL MOVEA(XS(I,J),YS(I,J))
      CALL PLOTEST(XS(I,J),YS(I,J),XS(I,J+1),YS(I,J+1))
  620  CONTINUE
  630  CONTINUE
C
C
C
C***************************************************************
C
C
C      MODULE 2 STARTS AT 700
C
C
C****************************************************************
C
C
C
  700 I=26
  705 CONTINUE
      CALL MOVEA(XS(I,1),YS(I,1))
      DO 710 J=2,26
      CALL PLOTEST(XS(I,J-1),YS(I,J-1),XS(I,J),YS(I,J))
  710 CONTINUE
      IF(I.EQ.1) GO TO 1000
      DO 720 J=1,26
      CALL MOVEA(XS(I,J),YS(I,J))
      CALL PLOTEST(XS(I,J),YS(I,J),XS(I-1,J),YS(I-1,J))
  720 CONTINUE
      I=I-1
      GO TO 705
C
```

```
C
C
C*********************************************************************
C
C
C      MODULE 3 STARTS AT 800
C
C
C*********************************************************************
C
C
C
  800 J=26
  805 CONTINUE
      CALL MOVEA(XS(26,J),YS(26,J))
      I=50
  806 IF(I.EQ.0) GO TO 810
      CALL PLOTEST(XS(I+1,J),YS(I+1,J),XS(I,J),YS(I,J))
      I=I-1
      GO TO 806
  810 CONTINUE
      IF(J.EQ.1) GO TO 1000
      I=26
  815 IF(I.EQ.0) GO TO 820
      CALL MOVEA(XS(I,J),YS(I,J))
      CALL PLOTEST(XS(I,J),YS(I,J),XS(I,J-1),YS(I,J-1))
      I=I-1
      GO TO 815
  820 CONTINUE
      J=J-1
      GO TO 805
C
C
```

```
C
C**********************************************************************
C
C
C       MODULE 4 STARTS AT 900
C
C
C**********************************************************************
C
C
C
  900 DO 930 I=1,26
      CALL MOVEA(XS(I,26),YS(I,26))
      J=50
  905 IF(J.EQ.0) GO TO 910
      CALL PLOTEST(XS(I,J+1),YS(I,J+1),XS(I,J),YS(I,J))
      J=J-1
      GO TO 905
  910 CONTINUE
      IF(I.EQ.26) GO TO 1000
      J=26
  915 IF(J.EQ.0) GO TO 920
      CALL MOVEA(XS(I,J),YS(I,J))
      CALL PLOTEST(XS(I,J),YS(I,J),XS(I+1,J),YS(I+1,J))
      J=J-1
      GO TO 915
  920 CONTINUE
  930 CONTINUE
 1000  CONTINUE
      IFRAMEE=IFRAME*2
      DO 1010 I=1,IFRAMEE
      CALL BELL
      CALL MOVEA(-51.15,-51.15)
      CALL DASHA(-51.15,0.0,3)
 1010 CONTINUE
      RETURN
      END
C
C
```

```
C
C**********************************************************************
C**********************************************************************
C
C
C         SUBROUTINE THAT MAPS 3-D INTO 2-D
C
C
C**********************************************************************
C**********************************************************************
C
C
C
      SUBROUTINE TRNSFM(X,Y,Z)
      COMMON/DATA/XSC,YSC,COSTH,COSPHI,SINTH,SINPHI,RR,DD
      XP=X*COSTH-Y*SINTH
      YINTER=X*SINTH+Y*COSTH
      YP=YINTER*COSPHI+Z*SINPHI
      ZP=-Z*COSPHI+YINTER*SINPHI+RR
      XSC=XP*DD/ZP
      YSC=YP*DD/ZP
      RETURN
      END
C
C
C**********************************************************************
C**********************************************************************
C**********************************************************************
C
C
C         FUNCTION USED TO FIND THE DEPTH OF A POINT
C
C**********************************************************************
C**********************************************************************
C**********************************************************************
C
C
C
      FUNCTION ZZ(X,Y,Z)
      COMMON/DATA/XSC,YSC,COSTH,COSPHI,SINTH,SINPHI,RR,DD
      ZZ=(X*SINTH+Y*COSTH)*SINPHI+RR-Z*COSPHI
      RETURN
      END
C
C
C
```

```
C*******************************************************************
C*******************************************************************
C
C
C
C      THIS IS THE SUBROUTINE THAT HIDES THE OBSCURED LINES.
C      THE METHOD IS KNOWN AS WRIGHT'S METHOD.  A REFRENCE
C      CAN BE FOUND IN IEEE PAPERS ON COMPUTERS----
C
C
C
C*******************************************************************
C*******************************************************************
C
C
C
      SUBROUTINE PLOTEST(XX1,YY1,XX2,YY2)
      COMMON/HEIGHT/HMAX(1024),HMIN(1024)
+     /XYINTER/XINTER,YINTER
      X1=XX1
      X2=XX2
      Y1=YY1
      Y2=YY2
      I1=INT(10.0*X1+512.5)
      I2=INT(10.0*X2+512.5)
      IVIS1=1
      IVIS2=1
      IF(Y1.LT.HMAX(I1).AND.Y1.GT.HMIN(I1)) IVIS1=0
      IF(Y2.LT.HMAX(I2).AND.Y2.GT.HMIN(I2)) IVIS2=0
      IF(Y1.EQ.HMAX(I1).OR.Y1.EQ.HMIN(I1)) IVIS1=2
      IF(Y2.EQ.HMAX(I2).OR.Y2.EQ.HMIN(I2)) IVIS2=2
      IF(IVIS1.EQ.1.AND.IVIS2.EQ.1) GO TO 100
      IF(IVIS1.EQ.0.AND.IVIS2.EQ.0) GO TO 200
      IF(IVIS1.EQ.1.AND.IVIS2.EQ.0) GO TO 300
      IF(IVIS1.EQ.0.AND.IVIS2.EQ.1) GO TO 400
      IF(IVIS1.EQ.0.AND.IVIS2.EQ.2) GO TO 400
      IF(IVIS1.EQ.2.AND.IVIS2.EQ.0) GO TO 200
      IF(IVIS1.EQ.1.AND.IVIS2.EQ.2) GO TO 100
      IF(IVIS1.EQ.2.AND.IVIS2.EQ.1) GO TO 100
      IF(IVIS1.EQ.2.AND.IVIS2.EQ.2) GO TO 100
C
```

```
C**********************************************************************
C
C      CASE 1, BOTH ENDPOINTS VISABLE, STARTS AT 100
C
C**********************************************************************
C
  100 CALL DRAWA(X2,Y2)
      CALL UPDATE(X1,Y1,X2,Y2)
      GO TO 500
C
C**********************************************************************
C
C      CASE 2, BOTH ENDPOINTS OBSCURED, STARTS AT 200
C
C**********************************************************************
C
  200 CALL MOVEA(X2,Y2)
      GO TO 500
C
C**********************************************************************
C
C      CASE 3, POINT 1 VISABLE AND POINT 2 OBSCURED, STARTS AT 300
C
C**********************************************************************
C
  300 CALL SEARCH(X1,X2,IVIS1,X2,Y2,IVIS2)
      CALL DRAWA(XINTER,YINTER)
      CALL MOVEA(X2,Y2)
      CALL UPDATE(X1,Y1,XINTER,YINTER)
      GO TO 500
C
C**********************************************************************
C
C      CASE 4, POINT 1 OBSCURED AND POINT 2 VISABLE, STARTS AT 400
C
C**********************************************************************
C
  400 CALL SEARCH(X1,Y1,IVIS1,X2,Y2,IVIS2)
      CALL MOVEA(XINTER,YINTER)
      CALL DRAWA(X2,Y2)
      CALL UPDATE(XINTER,YINTER,X2,Y2)
C
C
C
  500 RETURN
      END
C
C
C
```

```
C***********************************************************************
C***********************************************************************
C
C
C
C      THIS IS THE SUBROUTINE USED TO UPDATE THE HMIN AND HMAX
C
C
C
C***********************************************************************
C***********************************************************************
C
C
C
      SUBROUTINE UPDATE(X1,Y1,X2,Y2)
      COMMON/HEIGHT/HMAX(1024),HMIN(1024)
      I1=INT(10.0*X1+512.5)
      I2=INT(10.0*X2+512.5)
      IF(Y1.GT.HMAX(I1)) HMAX(I1)=Y1
      IF(Y1.LT.HMIN(I1)) HMIN(I1)=Y1
      IF(Y2.GT.HMAX(I2)) HMAX(I2)=Y2
      IF(Y2.LT.HMIN(I2)) HMIN(I2)=Y2
      IF(X1.EQ.X2) GO TO 40
      IF(I1.GT.I2) GO TO 10
      GO TO 20
   10 ITEMP=I1
      I1=I2
      I2=ITEMP
   20 CONTINUE
      IDELTA=I2-I1
      IF(IDELTA.LE.1) GO TO 40
      I1=I1+1
      I2=I2-1
      DO 30 I=I1,I2
      XI=FLOAT(I)
      Y=((Y2-Y1)/(X2-X1))*((XI-512.5)*.1-X1)+Y1
      IF(Y.GT.HMAX(I)) HMAX(I)=Y
      IF(Y.LT.HMIN(I)) HMIN(I)=Y
   30 CONTINUE
  40  RETURN
      END
C
C
C
```

```
C*******************************************************************************
C
C
C
C      THIS SUBROUTINE FINDS THE INTERSECTION POINT
C      WITH A BINARY SEARCH
C
C
C
C*******************************************************************************
C*******************************************************************************
C
C
C
      SUBROUTINE SEARCH(XX1,YY1,IVIS1,XX2,YY2,IVIS2)
      COMMON/HEIGHT/HMAX(1024),HMIN(1024)
     +/XYINTER/XINTER,YINTER
      X1=XX1
      Y1=YY1
      X2=XX2
      Y2=YY2
      X0=XX1
      Y0=YY1
      IF(X1.EQ.X2) GO TO 6
      GO TO 7
    6 XINTER=X1
       YINTER=Y2
      GO TO 50
    7 CONTINUE
      SLOPE=(Y2-Y1)/(X2-X1)
      I1=INT(10.0*X1+512.5)
      I2=INT(10.0*X2+512.5)
   10 I3=(I2-I1)/2+I1
      XI3=FLOAT(I3)
      X3=(XI3-512.5)/10.0
      Y3=(X3-X0)*SLOPE+Y0
      IVIS3=1
      IF(Y3.LT.HMAX(I3).AND.Y3.GT.HMIN(I3))  IVIS3=0
      IF(IVIS1.EQ.IVIS3) GO TO 20
      IVIS2=IVIS3
      I2=I3
      Y2=Y3
      GO TO 30
   20  IVIS1=IVIS3
      I1=I3
      Y1=Y3
   30 CONTINUE
      IDELTA=IABS(I1-I2)
      IF(IDELTA.NE.1) GO TO 10
      IF(IVIS1.EQ.0) GO TO 40
      XI1=FLOAT(I1)
      XINTER=(XI1-512.5)/10.0
      YINTER=Y1
      GO TO 50
   40 XI2=FLOAT(I2)
      XINTER=(XI2-512.5)/10.0
      YINTER=Y2
   50 RETURN
      END
C
```

```
05320C
05330C
05340C*********************************************************************
05350C
05360C
05370C
05380C      SUBROUTINE USED TO DRAW THE BORDER AND TO INSERT A DELAY
05390C
05400C
05410C
05420C*********************************************************************
05430C
05440C
05450C
05460       SUBROUTINE BORDER
05470       CALL MOVEA(-51.15,-38.95)
05480       CALL DRAWA(-51.15,38.95)
05490       CALL DRAWA(51.15,38.95)
05500       CALL DRAWA(51.15,-38.95)
05510       CALL DRAWA(-51.15,-38.95)
05520       RETURN
05530       END
```

APPENDIX 4

Listing of the program used to generate Movie 1, D'alemberts Solution

```
C**********************************************************************
C
C
C      GREGORY PAT SCANDALIS- SENIOR PROJECT/COMPUTER MODELING
C      DESCRIPTION:  THIS PROGRAM IS THE FIRST IN A SERIES OF
C      COMPUTER MODELS BASED ON PHYSICS. THIS PROGRAM MODELS A
C      VIBRATING STRING. THIS PROGRAM DOES NOT SOLVE THE "WAVE
C      EQUATION". THIS PROGRAM GENERATES A " MOVIE " FRAME BY FRAME
C      ON A TEKTRONIX GRAPHICS TERMINAL. THESE INDIVIDUAL FRAMES
C      OF THE STRING'S MOTION WILL BE PHOTOGRAPHED BY A FRAME
C      CAMERA FRAME BY FRAME. THE PROGRAM USES D'ALEMBERTS SOLUTION
C      TO THE WAVE EQUATION:
C
C
C
C
C      THIS SOLUTION REPRESENTS THE SUM OF TWO WAVES TRAVELING IN
C      OPPOSITE DIRECTIONS.
C
C**********************************************************************
C
       PROGRAM STRING(INPUT,OUTPUT,TAPE61,TAPE62,TAPE5=INPUT,
+       TAPE6=OUTPUT)
        COMMON PI,V,XL,N
C
C        INPUT THE VELOCITY, HARMONIC NUMBER AND THE RUN TIME
C
       PRINT*," INPUT N, THE HARMONIC NUMBER"
       READ*,N
       PRINT*,"ENTER THE INITIAL NUMBER OF STILL SECONDS"
       READ*,J0
       J1=J0*18
       J0=J0*36
       PRINT*,"INPUT THE PERIOD IN REAL TIME"
       READ*,PER
       V=275./(N*PER)
       PRINT*," THE VELOCITY IS : ",V
       PRINT*," INPUT ITMAX, THE NUMBER OF SECONDS GENERATED"
       READ*,ITMAX
       IQUIT=ITMAX*18+J1
       PRINT*,"BETTER QUIT BEFORE THE FRAME COUNTER READS ",IQUIT
       ITMAX=ITMAX*6
       PRINT*,"ENTER THE INITIAL TIME"
       READ*,T0
C
C        SET INITIAL VALUES AND CONSTANTS
C
       PI=3.1415926
       T=T0
       XL=900.0
C
```

```
C        THIS  DO  RESETS EACH FRAME
C
      DO 10 I=1,ITMAX
       X=0.0
C
C        INITIALIZE THE GRAPHICS AREA, DRAW THE BOX
C
      CALL INITT(120)
      CALL DWINDO(0.0,900.0,-1.1,1.1)
      CALL TWINDO(62,962,200,717)
      CALL MOVEA(0.0,-1.1)
      CALL DRAWA(0.0,1.1)
      CALL DRAWA(900.0,1.1)
      CALL DRAWA(900.0,-1.1)
      CALL DRAWA(0.0,-1.1)
      CALL MOVEA(0.0,0.0)
      CALL DASHA(900.0,0.0,3)
C
C        INITIALIZE THE CURSOR
C
      Y=F(X,T)
      CALL MOVEA(X,Y)
C
C         CALCULATE AND PLOT A FRAME OF THE STRING
C
      DO 20 J=1,450
         X=X+2
         Y=F(X,T)
         CALL DRAWA(X,Y)
   20 CONTINUE
      T=T+1
      JJ=6
      IF(T.EQ.1.0) JJ=JO
      DO 11 II=1,JJ
       CALL BELL
      CALL MOVEA(0.0,-0.75)
      CALL DASHA(0.0,0.0,3)
   11 CONTINUE
   10 CONTINUE
      CALL FINITT(0,0)
      STOP
      END
C
      FUNCTION F(X,T)
       COMMON PI,V,XL,N
      F=(.5)*(SIN((N*PI*(X+V*T))/XL)+SIN((N*PI*(X-V*T))/XL))
      RETURN
      END
```

APPENDIX 5

Listing of the program used to generate Movie 2, Fourier Solution

```
C*****************************************************************
C
C
C      GREGORY PAT SCANDALIS- SENIOR PROJECT/COMPUTER MODELING
C      DESCRIPTION:  THIS PROGRAM IS THE FIRST IN A SERIES OF
C      COMPUTER MODELS BASED ON PHYSICS. THIS PROGRAM MODELS A
C      VIBRATING STRING. THIS PROGRAM DOES NOT SOLVE THE "WAVE
C      EQUATION". THIS PROGRAM GENERATES A " MOVIE " FRAME BY FRAME
C      ON A TEKTRONIX GRAPHICS TERMINAL. THESE INDIVIDUAL FRAMES
C      OF THE STRING'S MOTION WILL BE PHOTOGRAPHED BY A FRAME
C      CAMERA FRAME BY FRAME. THE PROGRAM USES D'ALEMBERTS SOLUTION
C      TO THE WAVE EQUATION:
C
C
C
C
C      THIS SOLUTION REPRESENTS THE SUM OF TWO WAVES TRAVELING IN
C      OPPOSITE DIRECTIONS.
C
C*******************************************************************
C
       PROGRAM STRING(INPUT,OUTPUT,TAPE61,TAPE62,TAPE5=INPUT,
+       TAPE6=OUTPUT)
        COMMON PI,V,XL,N,H
C
C         INPUT THE VELOCITY, HARMONIC NUMBER AND THE RUN TIME
C
       PRINT*," INPUT THE STRING VELOCITY"
       READ*, V
       PRINT*," INPUT N, THE HARMONIC NUMBER"
       READ*,N
       PRINT*," INPUT ITMAX, THE NUMBER OF FRAMES GENERATED"
       READ*,ITMAX
       PRINT*,"ENTER THE INITIAL HEIGHT OF THE STRING"
       READ*,H
       PRINT*,"ENTER THE INITIAL NUMBER OF STILL FRAMES"
       READ*,JO
       JO=JO*2
       PRINT*,"ENTER THE INITIAL TIME"
       READ*,TO
C
C         SET INITIAL VALUES AND CONSTANTS
C
       PI=3.1415926
       T=TO
       XL=900.0
C
C        THIS "DO" RESETS EACH FRAME
C
       DO 10 I=1,ITMAX
        X=0.0
C
```

```
C          INITIALIZE THE GRAPHICS AREA, DRAW THE BOX
C
      CALL INITT(120)
      CALL DWINDO(0.0,900.0,-1.1,1.1)
      CALL TWINDO(62,962,200,717)
      CALL MOVEA(0.0,-1.1)
      CALL DRAWA(0.0,1.1)
      CALL DRAWA(900.0,1.1)
      CALL DRAWA(900.0,-1.1)
      CALL DRAWA(0.0,-1.1)
      CALL MOVEA(0.0,0.0)
      CALL DASHA(900.0,0.0,3)
C
C        INITIALIZE THE CURSOR
C
      Y=F(X,T)
      CALL MOVEA(X,Y)
C
C          CALCULATE AND PLOT A FRAME OF THE STRING
C
      DO 20 J=1,450
         X=X+2
         Y=F(X,T)
         CALL DRAWA(X,Y)
   20 CONTINUE
      T=T+1
      JJ=6
      IF(T.EQ.1.0) JJ=J0
      DO 11 II=1,JJ
       CALL BELL
      CALL MOVEA(0.0,-0.75)
      CALL DASHA(0.0,0.0,3)
   11 CONTINUE
   10 CONTINUE
      CALL FINITT(0,0)
      STOP
      END
C
      FUNCTION F(X,T)
       COMMON PI,V,XL,N,H
      P=0.0
      PARTSM=0.0
      DO 10 I=1,N
      XI=FLOAT(I)
      AN=8.0*H*SIN(XI*PI/2.0)/(XI**2.0)*(PI**2.0)
      TAU=COS(XI*PI*V*T/XL)
      XLAM=SIN(XI*PI*X/XL)
      P=AN*TAU*XLAM
      PARTSM=PARTSM+P
   10 CONTINUE
      F=PARTSM
      RETURN
      END
```

APPENDIX 6

Listing of the program used to generate Movie 3, Modes of the Vibrating Membrane

```
C*******************************************************************
C
C
C
C              GREGORY PAT SCANDALIS
C              SENIOR PROJECT
C              PHYSICS
C              COMPUTER GENERATED MOVIES W/3-D GRAPHICS
C
C
C*******************************************************************
C
C
C
C                THE FIRST PART OF THE PROGRAM IS A DRIVER TO
C                GENERATE THE DATA FOR THE TEST SURFACE.
C
C
C
C******************************************************************
C*******************************************************************
C
C
      PROGRAM PATPLOT(INPUT,OUTPUT,TAPE61,TAPE62,TAPE5=INPUT,
+                 TAPE6=OUTPUT)
      DIMENSION X(26,26),Y(26,26),Z(26,26)
      COMMON/POINT/X(26,26),Y(26,26),Z(26,26)
C
C      INITIALIZE THE ARRAYS
C
      DO 20 I=1,26
      DO 10 J=1,26
      X(I,J)=I-16.
      Y(I,J)=J-16.
   10 CONTINUE
   20 CONTINUE
C        INITIAL DATA IS INPUT
C
      PRINT*,"ENTER R,THETA,PHI,D,IFRAME"
      READ*,R,THETA,PHI,D,IFRAME
      XNX=2.0
      YNY=1.0
   15 T=0.0
      DO 3000 K=1,140
      DO 50 I=1,26
      DO 60 J=1,26
      A=SIN((I-1.0)*XNX*.12566)
      B=SIN((J-1.0)*YNY*.12566)
      C=COS(((XNX**2+YNY**2)**.5)*T*.17444)
      Z(I,J)=8.0*A*B*C
   60 CONTINUE
   50 CONTINUE
      T=T+1.0
      CALL INITT(120)
      CALL DWINDO(-51.15,51.15,-38.95,38.95)
      CALL TWINDO(0,1023,0,780)
      CALL BORDER
      CALL GRAFIT(R,THETA,PHI,D,IFRAME)
 3000  CONTINUE
      XNX=XNX+1.0
      YNY=YNY+1.0
      IF(XNX.EQ.3.0) GO TO 15
      CALL FINITT(0,0)
      STOP
      END
```

BIBLIOGRAPHY

Brown, J. W., Churchill R. V., Fourier series and Boundry Value Problems, McGraw-Hill (1941).

Frey, A.R., Kinsler, L.E., Fundementals of Acoustics, John Wiley & Sons (1962).

Myint-U, Tyn, Partial Differential Equations of Mathematical Physics, North Holland (1973).

Newman, W. M., Sproull, R. F., Principles of Interactive Computer Graphics, McGraw-Hill (1979).

Tipler, P. A., Physics vol. 2, Worth Publishers (1978).

Foley, J. D., Van Dam, A., Fundamentals of Interactive Computer Graphics, Addison-Wesley Publishing Company (1982).

Wright, T. J., A Two Space Solution to the Hidden Line Problem for Plotting Functions of Two Variables, IEEE Transactions on Computers c-22, no. 1, January 1973.