Computer Map Production
using an
Icosahedral Projection Design

an exercise in
The Design Initiative

prepared for

The World Game

by

John Kirk

Los Angeles, January 1971

Acknowledgements

I am deeply indebted to Ann Porch, writer and programmer-analyst, Los Angeles, for her extensive contribution, both to the physical production of the computer program and this paper, and to my morale as I try to communicate what I have learned.

I am grateful to Dr. Norman J. W. Thrower, Professor of Geography, UCLA, for his interest in, and sponsorship of, this work.

I wish to acknowledge the help of numerous members of the CCN staff and the academic community at UCLA upon whose time I presumed during the latter half of 1970 while coding and debugging the computer program.

This work was accomplished with the aid of intramural funding received through the auspices of Campus Computing Network, UCLA. The development and debugging of the program was done on the IBM 360 mod 91 computer at CCN.

Table of Contents

Computer Map Production

using an

Icosahedral Projection Design

Section 1

How this design relates to the problem

The design this paper describes, addresses itself to the discrepancy between some existing needs that are not being met and some existing resources that are not being fully utilized.

Most maps to which people have been exposed have several considerable drawbacks. Historically, these drawbacks have been insignificant but more recently only tolerable. In the last fifty years, they have become crucial bottlenecks to the flow of effective, realistic thinking about worldwide events.

One drawback is the distortion caused by the uneven "stretching" and "contracting" of features in one area of the globe relative to another. This distortion in the Mercator Projection, for instance, has resulted in people having such ingrained experiences as thinking Greenland to be as large as the United States, or as thinking of Antartica to have a coastline nearly as long as the Equator.

Another form of distortion is the cutting of a land mass into separate pieces in the process of spreading the map out flat. Those of us used to maps published in the United States tend to see Asia in our minds as way out on the fringes of the world, part here, part there.

A further drawback in the present day is the East-West orientation of most traditional maps, with North at the "top" of the world and South at the "bottom" of the world. This traditional presentation of the map was certainly optimum for the hundreds of years when the only world-wide traveling was done by ship, using the East-West winds and ocean currents in the temperate zones, and when indeed the only global thinking being done was by admirals of the sea.

In the U.S. today, even though it is becoming ever more obvious that each individual's day-to-day survival depends on the long-run survival and even the prosperity of one hundred percent of mankind, we still find it difficult to picture Asia, the focus of the world's population, as being just over the horizon, North of us, instead of mysteriously distant over land and sea to the East or West.

This paper introduces a particular implementation of R. Buckminster Fuller's Dymaxion Sky-Ocean World map drawn by computer. The projection described has the advantage of no noticeable "stretching" distortion, and at the same time, no breaks in land masses. The particular orientation of the land masses in relation to each other gives, in our age of air transportation, a truer picture of the distances between points by havint the North Pole and thus the shorter polar routes directly measurable and at the center of attention.

At present the high cost and lack of precision involved in producing maps by traditional "hand-craft" techniques results in even our most creative and intelligent thinkers having grown up under the crippling handicap of imprecise, localistic, planar perceiving of their essentially spherical environment. Applying computer

technology, through the icosahedral projection, to cartography can add precision and truer representation of geographical reality to traditional static map production. It can also provide man, for the first time, with dynamic, flexible and interactive image tools with which to perceive his world as a whole. Such tools might catalyse his powers of intuition in conceiving of designs which could lead him to physical success in Universe.

Section 2

How the computer works

Features on a globe or flat map are represented graphically as shaded areas, lines and points. Such information can be represented in the computer even though the computer does not store information directly in these graphic forms.

Computers are designed to store, retrieve and transform numbers. They do so with great speed and effectiveness. Fortunately, shaded areas can be made up of lines; lines can be made up of points; and the location of a point can be indicated by two numbers. These numbers are called its coordinates, and are measurements of its distance up or down from the middle of the page and left or right from the middle of the page.

Change to a picture that you can define graphically such as making the picture twice as big or moving it three inches to the right, can also be represented mathematically. For instance, making the picture twice as big can be represented in the computer by multiplying by two the numbers representing the picture. Moving it three inches to the right can be represented by adding three to cretain of the numbers representing the picture.

At this point, a useful concept may be noted. When we look at small plastic airplanes or cars, we are in the habit of calling them "models" of things in the real world. Children learn about cars and planes by playing with these models. Designers of real cars and airplanes use very precise models in their work. We relate to, manipulate, and otherwise study models in order to better understand or deal with the real world because the model is more manageable than is the real world.

In exactly a parallel manner, the numerical representation previously discussed can function as a model. Such a "mathematical model" when placed in a computer can function far more powerfully than other kinds of models, since manipulating mathematical models through the computer is more dynamic, flexible and interactive.

Since people relate rather ineffectively to numbers, and relate spontaneously with great effectiveness (intuition) to graphic images, in recent years machines have been developed which can draw pictures under computer control.

At present there are two basic forms of peripheral machines which produce graphic images. One has a roll of paper which can be moved back and forth lengthwise over a rotating drum. A pen which can b moved crossways to the paper, and picked up or put down, draws the map or picture. The other graphic machine is like a TV screen where an electron beam can be aimed at any point on the screen and turned on whenever a visible mark is desired. A picture of the map is made up of many such marks on the screen. A permanent copy of the picture can be kept by having the computer trigger a camera each time it completes an image.

These graphic machines are controlled very much like any machine with which you are familiar. The computer feeds them numeric measurements for each direction the pen, paper or electronic beam can move, in the same way that you give your TV measurements of loudness or brightness by twisting its knobs.

People control the computer by typing in groups of commands called "programs". The computer is able to respond to these commands in the same way that your TV is able to respond to your knob-twiddling command, "Speak louder!". Knob-twiddling and button-pushing that you are used to can be considered a language in the same way that there are various computer "languages" in which computer programs are written.

Section 3

How this projection method works

There are two steps in the projection method used to produce this map. The first explodes the sphere onto the icosahedron (a shell almost like a sphere, but made of twenty faces, each an equilateral triangle), and the second arranges the twenty faces of the icosahedron into a flat map.

Imagining some things might help to visualize how this projection method works. Imagine two globes, one inside the other. If they're lined up properly, you can pass a thin rod through Los Angeles on the outer globe, through Los Angeles on the inner globe, and it will then go through the center of the globes.

It's clear that the same will be true for any other pair of matching points on the two globes.

Now put two different sized icosohedra around a globe. You can easily see that if the two icosahedra are lined up with each other, a rod from the center, through any feature of the globe, will determine the same point on both icosahedra.

Step one of the projection works in this way, locating each point on the icosahedron corresponding to a point on the globe by passing a line from their mutual center through the feature of the globe to find the corresponding point on the icosahedron. Note that it does not matter what size globe or icosahedron you use, or whether they are touching at any points.

Once you have an icosahedron with the map drawn on its faces, you can proceed with step two.

Imagine taking each of the twenty triangles separately, like tiles, and placing them in any arrangement you like on a flat surface, making sure, whenever you put two edges together, that the map matches properly. For the purpose of seeing ocean currents or whole migrations, you might want an arrangement that keeps the oceans as continuous or unbroken as possible, and puts the land masses out at the fringes. For studying populations of people on land, and their movements and interactions, you might like to have the land masses as unbroken as possible, with the paths of interactions at the center.

The arrangement presented here manages to keep each land mass entirely unbroken, with the major world transportation routes near the center, when the map is fully put together.

Section Four

How this program works

The projection program embodies the task of reading information, point-by-point, in latitude and longitude coordinates, and drawing the information on paper according to the projection method described.

The icosahedron can be placed in any orientation with respect to the globe, by rotating it in any direction, keeping its center the same as the globe's. Therefore, the program reads into the computer, before beginning, the particular position desired.

The two numbers, latitude and longitude, which represent a point on the globe, are read. They are angles, or fractions of a circle's circumference, and may be measured in degrees and fractions of degrees, or in degrees, minutes, seconds and fractions of seconds, or in radians.

Next the program decides which triangle of the icosahedron the point will fall in, and thus on which surface of the icosahedron to project the point.

Step one of the projection method previously described is the applied. It consists of two parts. The first converts the latitude and longitude of the point to rectangular (x1–y1) coordinates on the icosahedral face where the vertical (y1) axis corresponds to North-South on the globe. The second applies a rotation to get the point in coordinates (x2–y2) relative to a pole of the icosahedron rather than that of the globe.

Step two is applied next. It consists of a rotation (x3–y3) and then a translation (x4–y4) to change the coordinates of the point from its position relative to the center of its triangle, to its final position in the finished map (x4–y4).

The last thing the program does for each point is to give the command to move the pen to the coordinates just computed and supply information about whether or not the connecting line from the previous point is to be drawn.

Section Five

Before and after

... There are a myriad of economic trends and other ultimately vital or lethal evolutionary events transpiring today which are invisible to all humanity only because they are too fast or too slow for man to apprehend and comprehend them.

... Our computerized World Game is designed to accelerate the too slow and decelerate the too fast of all the known vital trendings and thereby to bring them dramatically within popular consideration and our world game's solution.

... As the general system of vital trends becomes visible and its components are seen to integrate synergetically, we also will begin to discern ways of using the world's resources to ever higher and more universal human advantage. We will soon learn popularly how to play the game to explore for ways in which we may use the world's resources so that we may be able to make our whole planet successfully enjoyable by all humanity without any human profiting at the expense of another and without interfering with one another, and how to do so in the shortest possible time.

R. Buckminster Fuller

Denver, June 18, 1969

The computerized mapping design described in this paper accepts information about the location of coastlines, cities, populations, resources, routes, etc. in tabular form as input, and draws maps as output.

Further work is presently underway to make this process progressively more dynamic and interactive. One immediate next step is to produce time-lapse animated films of the map, showing speeded-up or slowed-down trends comprehensive to the whole earth. Another step is achieving the capability of viewing the map on a TV screen, for instance, while shifting back and forth among different "resource" or "need" kinds of information on it for comparison.

Projection Program Flowchart

Start
initialize plot
read parameters
draw outline
draw boundaries inside outline
(1)
read card with 7 points (EOF?)
(yes $->$ 5) (no $->$ 2)
(2)
test for (0,180) end of island
(yes $->$ 4) (no $->$ next)
subtract LON from 180
take radian measure
housekeep new island marker
find triangle the point belongs in
check whether the point belongs in 21 or 22 (other halves of 5 and 17)
apply step 1, part 1 formula
apply step 1, part 2 formula
apply step 2, part 1 formula
apply step 2, part 2 formula
apply scaling factor
plot point
(3)
end of 7 points?
(yes $->$ 1) (no $->$ 2)
(4)
housekeep new island marker
($->$ 3)
(5)
close plot
write summaries
stop

```
      Program prjctn
      real*4 vcoord(12,2),fcoord(22,2),hold,x1,y1,x2,y2,x3,y3,x4,y4
      real*4 trm1(22),trm2(22),lat(7),lon(7),transx(22),transy(22)
      real*4 trm3(22),trm4(22),coord,p1,p2,p3,p4,base,height,factor
      real*4 llat,llon,p5,p6
      integer*4 i,ii,face,cardno,nexfac,island,ffff,buffer(4000)
      integer*4 count(1000),prefac
101   format(i7,3x,14f5.2)
102   format(i5,2f25.16)
103   format(3i5)
104   format(' ',f20.15,20x,f20.15)
105   format('1','vcoord1',33x,'vcoord2')
106   format('1')
107   format(i2)
108   format(' ',i5,6f15.10)
c
c initialize plot
c
      call plots(buffer,4000)
      mult=3.0
      call plot(10.0*mult,-11.0*mult,3)
      call plot(10.0*mult,-5.0*mult,-3)
c
c read parameters
c
      read(1,102) (i,vcoord(i,1),vcoord(i,2),ii=1,12)
      read(1,102) (i,fcoord(i,1),fcoord(i,2),ii=1,22)
      read(1,102) (i,trm3(i),trm4(i),ii=1,22)
      read(2,102) (i,trm1(i),trm2(i),ii=1,22)
      base=1.323164552735024
      height=base*sqrt(3.0e0)/2.0e0
      factor=(10.0e0/(3.0e0*height))*mult
      base=factor*base
      height=factor*height
      p1=5.0e0*height/3
      p2=2.0e0*height/3
      p4=-height/3.0e0
      p5=base/4.0e0
      p6=height/6.0e0
c
c draw outline
c
      call plot( 0           -p5, p2                -p6,3)
      call plot( base        -p5, p2                -p6,2)
```

```
      call plot( 1.5e0*base   -p5, p1                  -p6,2)
      call plot( 2.0e0*base   -p5, p2                  -p6,2)
      call plot( 2.5e0*base   -p5, p1                  -p6,2)
      call plot( 3.0e0*base   -p5, p2                  -p6,2)
      call plot( 3.0e0*base   -p5, p3                  -p6,2)
      call plot( 1.5e0*base   -p5, p3                  -p6,2)
      call plot( 2.0e0*base   -p5, p4                  -p6,2)
      call plot( base         -p5, p4                  -p6,2)
      call plot( 0.5e0*base   -p5, p3                  -p6,2)
      call plot( 0            -p5, p4                  -p6,2)
      call plot( -0.5e0*base  -p5, -height             -p6,2)
      call plot( -0.5e0*base  -p5, p3                  -p6,2)
      call plot( -base        -p5, -p2                 -p6,2)
      call plot( -base        -p5, p4                  -p6,2)
      call plot( -2.0e0*base  -p5, p4                  -p6,2)
      call plot( -1.75e0*base -p5, -5.0e0*height/6.0e0 -p6,2)
      call plot( -2.5e0*base  -p5, p3                  -p6,2)
      call plot( -1.5e0*base  -p5, p3                  -p6,2)
      call plot( -2.0e0*base  -p5, p2                  -p6,2)
      call plot( -base        -p5, p2                  -p6,2)
      call plot( -1.5e0*base  -p5, p1                  -p6,2)
      call plot( 0.5e0*base   -p5, p1                  -p6,2)
      call plot( 0            -p5, p2                  -p6,2)
c
c draw boundaries inside outline
c
      call plot( -1.75e0*base -p5, -5.0e0*height/6.0e0 -p6,3)
      call plot( -0,5e0*base  -p5, p1                  -p6,2)
      call plot( 0            -p5, p2                  -p6,2)
      call plot( -base        -p5, p2                  -p6,2)
      call plot( -0.5e0*base  -p5, p3                  -p6,2)
      call plot( -1.5e0*base  -p5, p3                  -p6,2)
      call plot( -base        -p5, p4                  -p6,2)
      call plot( 0            -p5, p4                  -p6,3)
      call plot( -0.5e0*base  -p5, p3                  -p6,2)
      call plot( 0            -p5, p2                  -p6,2)
      call plot( 0.5e0*base   -p5, p3                  -p6,2)
      call plot( -0.5e0*base  -p5, p3                  -p6,2)
      call plot( 0.5e0*base   -p5, p3                  -p6,3)
      call plot( base         -p5, p2                  -p6,2)
      call plot( 3.0e0*base   -p5, p2                  -p6,2)
      call plot( 2.5e0*base   -p5, p3                  -p6,2)
      call plot( 2.0e0*base   -p5, p2                  -p6,2)
      call plot( base         -p5, p4                  -p6,2)
```

```
      call plot( 0.5e0*base   -p5, p3                 -p6,3)
      call plot( 1.5e0*base   -p5, p3                 -p6,2)
      call plot( base         -p5, p2                 -p6,2)
6     continue
      write(6,122)
      prefac=0
c
c read card with 7 points, EOF?
c
      do 2 i=1,7
c
c test for (0,180), end of island?
c
      if(abs(lat(i)).gt.1.0e-4.or.abs(lon(i)-180.0e0).gt.1.0e-4)
     *  goto 7
c
c housekeep new island marker
c
      nexfac=0
      go to 15
c
c subtract lon from 180
c
7     continue
      lon(i)=180.0e0-lon(i)
c
c take radian measure
c
      lon(i)=rad(lon(i))
      lat(i)=rad(lat(i))
c
c housekeep new island marker
c
      if(nexfac.eq.1) nexfac=2
      if(nexfac.eq.0) nexfac=1
c
c find triangle the point belongs in
c
      hold=100.0
      prefac=face
      do 4 ffff=1,20
      coord=dist(lat(i),lon(i),fcoord(ffff,1),fcoord(ffff,2))
      if(hold.le.coord) goto 3
      face=ffff
```

```
      hold=coord
3     continue
4     continue
c
c check whether belongs in 21, 22 (other halves of 5, 17)
c
      if(face.eq.17.and.dist(lat(i),lon(i),vcoord(4,1),vcoord(4,2)).lt.
     *  dist(lat(i),lon(i),vcoord(9,1),vcoord(9,2))) face=22
      if(face.eq.5.and.dist(lat(i),lon(i),vcoord(4,1),vcoord(4,2)).gt.
     *  dist(lat(i),lon(i),vcoord(12,1),vcoord(12,2)).and.
     *  dist(lat(i),lon(i),vcoord(4,1),vcoord(4,2)).gt.
     *  dist(lat(i),lon(i),vcoord(2,1),vcoord(2,2))) face=21
      if(face.ne.prefac) nexfac=1
c
c apply step 1, part 1 formula
c
      x1=-cos(lat(i))*sin(lon(i)-fcoord(face,2))/
     *    sin(lat(i))*sin(fcoord(face,1))+
     *    cos(lat(i))*cos(fcoord(face,1))*
     *    cos(lon(i)-fcoord(face,2)))
      y1=( sin(lat(i))*cos(fcoord(face,1))-
     *     cos(lat(i))*sin(fcoord(face,1))*cos(lon(i)-fcoord(face,2)))/
     *   ( sin(lat(i))*sin(fcoord(face,1))+
     *     cos(lat(i))*cos(fcoord(face,1))*cos(lon(i)-fcoord(face,2)))
c
c apply step 1, part 2 formula
c
      x2= trm3(face)*x1+trm4(face)*y1
      y2=-trm4(face)*x1+trm3(face)*y1
c
c apply step 2, part 1 formula
c
      x3= trm1(face)*x2+trm2(face)*y2
      y3=-trm2(face)*x2+trm1(face)*y2
c
c apply step 2, part 2 formula
c
      x4=x3+transx(face)
      y4=y3+transy(face)
c
c apply scaling factor
c
      llat=y4*factor
      llon=x4*factor
```

```
c
c plot point
c
      ii=4-nexfac
      call plot(llon,llat,ii)
15    continue
2     continue
c
c end of 7 points? if yes:
c
      goto 5
c
c close plot, write summaries (none here now)
c
1     continue
      call plot(0.0,0.0,999)
12    continue
      stop
      end
c
c function to compute the angular great circle distance between two points
c
      real function dist*4 (lat1,lon1,lat2,lon2)
      real*4 lat1,lon1,lat2,lon2
      dist=arcos(cos(lat1)*cos(lat2)*cos(lon2-lon1)+
     *           sin(lat1)*sin(lat2))
      return
      end
c
c function to take the radian measure of any angle expressed in degrees
c
      real function rad*4 (coord)
      real*4 coord
      rad=(coord/180.0e0)*3.141592653589793e0
      return
      end
```