Oct 1, 2006

A CLAY CUBE DIAL
Horizontal, Vertical
and two Meridian dial plates

This project was a cube dial, on an 8 by 8 by 8 cinder clock, with the faces or dial plates in clay.

This dial will be for Silver City, NM, whose coordinates are:

location lat:	32.75°	N
location long:	108.2°	W
magnetic declination:	10.6°	E

Because a lot of diallists have been using DeltaCAD, this project was going to use DeltaCAD macros and clay for the dial plates of which there were to be four. One would be a horizontal dial, one a vertical, with the two remaining being meridian dials. This would be a non decliner, so there would be no alignment issues, however these dials would be longitude corrected.

First, there was no connection between using DeltaCAD and using clay. The drive to use DeltaCAD was so the author could have a way of drawing dial plates that would be easy to use with clay slabs. They could be used for glass also. The other reason was because the author wanted to master DeltaCAD macros, which are a form of BASIC with some extras thrown in.

The author mostly uses glass and copper on concrete columns or walls for his outside dials, however he was trained in sculpture and pottery in the days of his youth, and he had some kilns lying around the place. In his youth he spent many hours kiln sitting.

Some clay was acquired, also three slip colors, and two glazes. There are many clays and the author was not fussy. The clay came in plastic bags and when opened, some clay was removed and squished between the fingers along with some water, and when creamy, a batch of slip was made. Slip is nothing more than colored clay, and by using the same clay for the home made slip as for the dial plate itself, coefficients of expansion should cause the slip to remain when dried out, and subsequently fired at around 1850f, called bisque firing. The slip pigment suppliers have complex mixing methods, teh author simply does what he was taught back in the 1960's. Whereas a slip is colored clay, a glaze is actually a powdered glass, or equivalent to it. After the bisque firing which causes the clay and slip to be permanent if not waterproof, the glaze is added which is at a higher temperature, potters would call this "cone 6", and that melts the glaze and fuses it with the clay creating a waterproof dial plate. Cone 6 also generates a dial plate not likely to suffer from freeze thaw cycles, and fires at about 2230 degrees F.

So this case study really covers several divergent issues.

1. Using a different CAD system
2. Using the CAD macros
3. Using clay
4. Making slips
5. Making glazes
6. Firing at two different temperatures
7. The final dial in situ.

The author's first career before winding up as an airline pilot was in programming computers. Some commercial, but mostly operating system software. The first system he programmed was an IBM 1401 in Autocoder. Then an IBM 360 in BAL (Basic Assembler Language), with a bit of PL/I, RPG, COBOL, and FORTRAN. All his work on the IBM 370 and later machines was in BAL and on some machines in C and C++. Operating systems used were BOS, DOS, MFT, and MVT on the IBM 360s, and VS1 and MVS on the IBM 370s, and GCS under VM also.

One pet peeve the author has is that documentation for languages is drawn up by programmers as language specifications, and when the human interface is covered, it is always somewhat academic. This tendency became worse when object oriented programming techniques became the standard, and the novice is faced with a bit of a struggle. Thus the examples in this section for the BASIC macro language for DeltaCAD are intended to be "conversational" as opposed to transaction oriented. In other words less object oriented and more of a natural flow.

As background, the author designed FIDO and PATCHES which were early spooling systems on IBM 360 mainframes in the 1970s, and TOTO and later SHADOW which were teleprocessing programs running under DOS, MFT, and MVS on the IBM 370 and later ranges which were sold worldwide from 1972 until 1997. SHADOW in particular made over $55m in sales before the author lost track and interest.

USING A CAD SYSTEM and USING CAD MACROS

A CAD system is a computer program that draws, usually rather better than the average human. The 2d systems such as DeltaCAD are simple to use and provide professional drawings. The author prefers TurboCAD deluxe which is a full 3d modeling system and which was used for most of the pictorials in this series of books.

However, DeltaCAD is clearly better than TurboCAD if one wishes to have the computer do all of the work, including calculating angles and then drawing them. However, computers do what they are told and thus special techniques are needed to do simple things a human automatically can do, such as constraining a line to the boundaries of a box.

This not intended to be a definitive work on DeltaCAD and its associated macro language, but rather a few pointes to explain some of its quirks. The author spent a total of less than a day reading the DeltaCAD manual and the associated macro manual, and then proceeded forth.

This section extracts some of the code and explains what it is doing, and then final dial plates are shown. There are many macros available for DeltaCAD, they are well worth exploring. For completeness the author has his own versions on the web site for those who are interested, and thus complement the spreadsheets.

Programs begin with initial setup, then they define variables to hold information being worked on, and also ask humans what they want, and then they produce the results.

```
' *******************************************************************
' A horizontal dial macro for DeltaCAD but in conversational mode as
' opposed to the more modern object oriented mode, but with notes
' page numbers refer to Manual.pdf or Basic.pdf provided with deltacad
' *******************************************************************
Sub Main()                            ' main procedure is required
' *******************************************************************
' Initial house keeping - clear the screen - set the drafting area unit
' *******************************************************************
' select all objects that may exist on the screen - p223 of Manual
' then erase them all - page 189 of Manual
If (dcSelectAll) Then
   dcEraseSelObjs
End If
' set the entire future drafting area to inches or centimeters, etc
' page 43 of the Manual: 1.0 generates inches, and 2.54 is cm
dcSetDrawingScale 0.80
```

That establishes the general ground rules. Next some programming structures are needed for human interaction. These are not difficult, nor are they as simple as simple BASIC.

```
' *********************************************************************
' A generic definition is required for a screen input area
' ********************************************************************
'
' Here a box on the screen for user dialog is structurally defined,
' it is only a definition of the generic area, it does not create it
' .....  Dialog aaaaa
'
' To create the area, there must be a Dim statement making a label
' relate to this definition
' .....  Dim bbbbb as aaaaa
'
' To use bbbbb there must be a ..... yyy = Dialog(xxxxx) which causes
' human interaction. So...
'
' create an area on the screen starting at x=20, y=20
' whose size is 200 left to right and 100 top to bottom
' whose title is "Location" - where 0,0 is top left

Begin Dialog aaaaa 20, 20, 200,100, "Location"
' the first text string starts at x=5,y=15 on the screen
' and the text string itself starts at x=60 for a height of 10
  Text          5, 15,   60,10,  "Enter latitude"

' the input area starts at x=65 (further right) y=15 (same height) for a
' size of x=50, y=10
  TextBox      65, 15,   50, 10, .mylat

' the second text string starts at x=5 but now y=30, i.e. lower down
' and the text string itself starts at x=60 for a height of 10
  Text          5, 30,   60, 10,  "Enter longitude"

' and the input area starts at x=65 (further right) y=30
' (same height) for a size of x=50, y=10
  TextBox      65, 30,   50, 10, .mylng

' the third text string starts at x=5 y=45
' and the text string itself starts at x=60 for a height of 10
  Text          5, 45,   60, 10,  "Enter ref longitude"

' and the input area starts at x=65 y=30 for a size of x=50, y=10
  TextBox      65, 45,   50, 10, .myref

' and two buttons for what the user means, location first, button size
' next - and all such boxes must have at least one button by the way
  OKButton     65, 65,   40, 10
  CancelButton 65, 85,   40, 10
End Dialog


' ********************************************************************
' The generic definition must then be generated with a name
' ********************************************************************
'
' this defines "bbbbb" as an instance of aaaaa dialog
Dim bbbbb As aaaaa
```

At this point, a few comments might be helpful. The Begin Dialog has nothing to do with a dialog. It is an encyclopedia definition of what you might wish to actually create.

It is created with the Dim statement.

And used later........., in fact here it is being used!

```
' *********************************************************************
' Now define the initial general working variables
' *********************************************************************
'
' define a lat and a long, and a reference longitude
Dim lat  As single
Dim lng  As single
Dim ref  As single


' *********************************************************************
' Now get the lat, long, and reference longitude
' *********************************************************************

' first set the defaults - here bbbbb.mylat uses the structure
' from aaaaa
bbbbb.mylat = "32.75"
bbbbb.mylng = "108.2"
bbbbb.myref = "105.0"


' here the dialog is invoked and the button results returned to ccccc
' page 20 and 24 etc of Basic discusses the Dialog function
ccccc = Dialog(bbbbb)
' which causes the answer to be returned
lat = bbbbb.mylat
lng = bbbbb.mylng
ref = bbbbb.myref
' CANCEL button returns 0
' OK     button returns -1
' you can determine the button with - Print ccccc, lat, lng, ref
```

A couple of points worth noting. It is considered very bad practice to use hard coded numbers for things, they should always be symbolic, and defined elsewhere. This makes programs easier to change. However, this is intended to explain what is happening, so that rule is being violated.

The rest of the program is straight forward.

```
' *********************************************************************
' ok, what was returned? if "ok button" then do the program itself
' *********************************************************************

' ccccc = -1 means the ok button was used and not the cancel button
If ccccc = -1 Then

   ' ******************************************************************
   ' this is the main program to draw the horizontal dial itself
   ' ******************************************************************
```

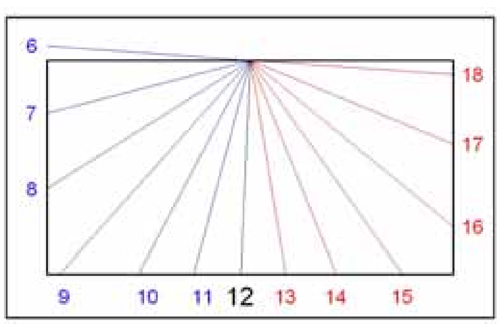
```
' calculate hour line angles next, but first define them
Dim h, hx, hy As Single   ' DeltaCAD is fussy about data attributes

' the formula is... hourlineangle  = atan ( sin(lat) * tan (lha) )
' almost all systems us radians
' the formula also needs adjustment for longitude displacement

' line color  is 0 is black
' line type   is dcsolid
' line weight is dcnormal

' set the text color, font, size, etc also
dcSetTextParms dcBLACK,"Ariel","Bold",0,8, 20,0,0  ' p231 of Manual
dccreatetext -1.25,   -0.3, 0, "Hour and hour line angle H-DIAL"
dccreatetext -1.25,   -0.9, 0, "Lat: "
dccreatetext -0.8,    -0.9, 0, Int(lat)
dccreatetext  0.0,    -0.9, 0, "Long: "
dccreatetext  0.3,    -0.9, 0, Int(lng)

For hr = 6 To 18 Step 1
' ==================================================================
    ' for the hour (hr) calculate the hour line angle (h)
    h = deg(Atn(Sin(rad(lat))*Tan(rad((hr*15) +(ref-lng)))))

    ' show the time in hours
    dcSetTextParms dcBLACK,"Ariel","Bold",0,8, 21,0,0  ' p231 Manual
    dccreatetext (-1.2+((hr-6)/5)), -0.5, 0, Abs(hr)
    ' show the angle
    dcSetTextParms dcBLACK,"Ariel","Bold",0,6, 21,0,0  ' p231 Manual
    dccreatetext (-1.2+((hr-6)/5)), -0.7, 0, Int(h)

    If hr < 12 Then
    '  ------------------------------------------------------------
    '  morning hours ~ NOTE code for keeping lines in a boxed area
    '  ------------------------------------------------------------
       dcsetlineparms dcblue,dcsolid,dcthin  ' page 228 Manual
       If Abs(h) < 45 Then             ' lines touch top of box
          dcSetTextParms dcBLACK,"Ariel","Bold",0,8,21,0,0 ' p231
          hx = Tan(rad((h)))
          dccreateline   0,     0, hx,  1
          dccreatetext  (hx), 1.1,  0,  Abs(hr) ' page 187 of Manual
       Else                             ' lines touch side of box
          dcSetTextParms dcBLACK,"Ariel","Bold",0,8, 20,0,0  ' p231
          hy = Tan(rad((90-h)))
          dccreateline   0,     0, -1,  -hy
          dccreatetext  -1.1, -hy,  0,  Abs(hr) ' page 187 of Manual
       End If

    ElseIf hr = 12 Then
    '  -------------
    '  noon hours
    '  -------------
    ... similar code which is straight forward

    Else
    '  ---------------
    '  afternoon hours
    '  ---------------
    ... similar code except the sign of the angle goes –ve if 90

    End If
' =================================================================
Next h
```

The program concludes with drawing a couple of boxes. And after the end of the program, there are the DEGREES and RADIANS functions.

```
   ' draw a box around everything
   dccreatebox    -1,      0,     1,     1    ' page 184 Manual
   dccreatebox    -1.2, -.2,  1.2,  1.2    ' page 184 Manual
   dcviewbox      -1.1, -1.1, 1.1,  1.3    ' page 225 Manual
End If

' ***********************************************************************
' *** this ends the entire program
' ***********************************************************************
End Sub

'
***********************************************************************
' Useful routines or functions  - Functions must be defined at the end
' after the main profram which is  sub(xx) ... end sub
' ***********************************************************************

' Convert degrees to radians
Function Rad ( n As single )  As single
' page 83 basic.pdf for functions
    Rad = (n * 2 * 3.14162) / 360
End Function

' Convert radians to degrees
Function Deg ( n As single )  As single
' page 83 basic.pdf for functions
    Deg = (360 * n) / (2 * 3.14162)
End Function
```

The actual code on the web site may differ, however, the objective has been to step through most of the procedural coding. Yes, there are some coding violations, however they have been somewhat intentional in order to make the process of a complete macro, when accompanied by DeltaCAD's two books, MANUAL.PDF and BASIC.PDF, should make more sense of how a DeltaCAD macro or program works.

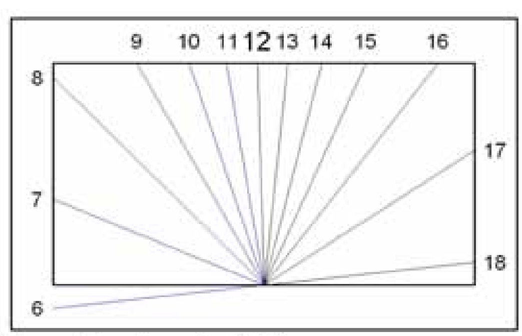
Four dial plates were to be constructed. DeltaCAD produced, using the above and similar code, the four dial plates.



Hour and hour line angle VERTICAL NON DECLINER

6	7	8	9	10	11	12	13	14	15	16	17	18
86	-77	-60	-44	-29	-16	-3	9	23	36	52	68	86

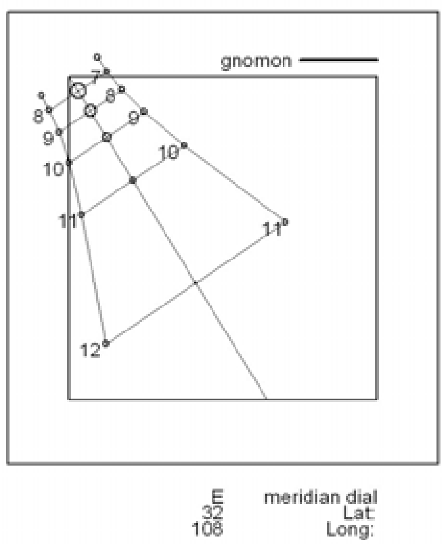
Lat: 32 Long: 108

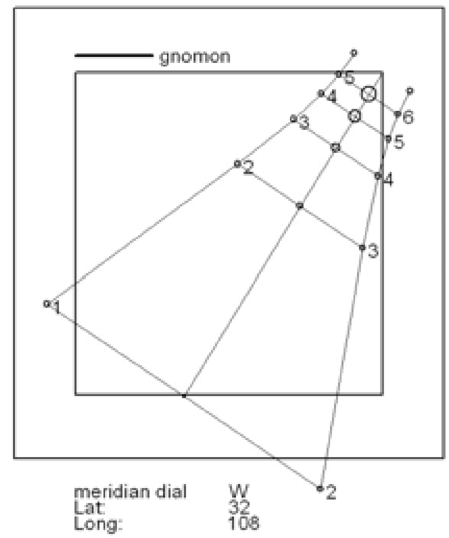


Hour and hour line angle H-DIAL

6	7	8	9	10	11	12	13	14	15	16	17	18
84	-69	-47	-32	-20	-11	-2	6	15	25	39	58	84

Lat: 32 Long: 108





The dial plates when printed on 8 1/2 by 11 inch paper were perfect for a dial plate for an 8x8x8 concrete block. Holes were punched on the hour lines and other key plates, and for the east and west meridian dials, the paper was cut along the calendar lines.

While the horizontal and the vertical dial use an angled gnomon, and in this case no calendar lines were present, this was not true for the meridian dials which had calendar lines, and in addition the gnomon style paralleled the dial plate and thus the critical dimension for the gnomon was its linear height above the sub-style. For this reason the DeltaCAD macro also shows the gnomon linear height, style to sub-style. This line would be inscribed on the meridian dial plate because clay shrinks, so this line should shrink similarly in the correct ration. In fact the line was inscribed twice to show the linear contraction, about 11% in the case of this terra-red clay.

USING CLAY AND MAKING SLIPS

Clay can be purchased at ceramics and pottery supply stores. In this case a clay designed for cone 6 (a measurement of heat) which is about 2230 degrees F (a measurement of temperature). Heat and temperature are not the same thing.

Cone 6 clay was chosen because of the need for stoneware, in other words a need for something that could withstand the freeze thaw cycles of the winter and summer extremes.

A section of clay was cut from the slab and pressed to the approximate size.

The clay surface was prepared by passing a wooden flat sheet over the clay surface, with a small amount of water. This removed scratch and other marks.

The paper printout from DeltaCAD was placed on the clay surface and the holes punched on key points was transcribed to the clay. The dial center, the tips of the hour lines, and the line that for a non longitude corrected dial would be 6 am to 6pm was marked. This is critical for subsequent dial alignment.

The noon line if offset because of longitude correction, and a cutout was made after the slip had been applied for the colored texture, but before firing. The bisque firing is a bit above 1800 degrees F and fixes the slip and clay. They are not waterproof, but they are fixed and can be worked on.

The bisque firing is not done for several days because the moisture must leave the clay body and slip. Do not move the clay pieces, while drying as if you do the clay will remember what you did, and curve up. When cooled down after the bisque firing the glaze is added.

The three pieces can easily be seen, along with the gnomon style cutout, and the alignment piece at the bottom of the two dial plate halves.

The preceding pictures show the horizontal dial plate, below is the vertical dial plate, again before the gnomon sub-style cutout was incised.

When all plates were cured, then bisque fired, then cooled, then glazed, then fired at cone 6 or 2230 degrees F, then the dial plates were affixed to the 8x8x8 concrete block. The medium used was designed for exterior use.

The gnomons were set in place with epoxy, and as it dried the dial plates were set in place temporarily to ensure the gnomon was placed correctly on the block, otherwise you could have a nicely oriented gnomon but a dial plate that didn't fit on the block.

When the gnomons were rigid, then the dial plate fixing medium was used to affix the dial plates to the block. Protractors were used to ensure the gnomons were aligned at latitude, and that the meridian dial equinox line was correct. A pair of dividers was used to pick up the meridian dial liner height from the incised lines for just that purpose, and thus the gnomons were of the correct length.

When all was secure than a sand grout was applied of an adobe color, that is somewhat standard in New Mexico.

The grout was wiped clean off the dial plates with a damp sponge, and when the grout was almost firm, then the same sponge was used to texture the grout.

To the right is the finished dial taken from the southeast, and below is the same dial pictured from the southwest.

Amazingly, it was fairly accurate.

And certainly a welcome addition to the garden.