Sun

Earth Illumination Simulator

Björn Odelfalk*

April 20, 2004

Abstract

In this paper I will describe how the utility Sun for TI-89 is working. You can download the utility at http://www.ticalc.org and search for Sun under TI-89/ASM/PROGRAMS. I will comprehensively explain the theory and the mathematical background to this problem. I will also describe how to convert the mathematics into computer code.

Computers without a FPU often lacks in speed when evaluating functions like $\sin(x)$ or $\arccos(x)$. I will show how you can optimize the code so that the program calculates the value of 4 trigonometric functions instead of 144000.

1 Theory

My way of solving this problem is to start with the geometric algebra. We want to present the results on a Mercator map projection, i.e. longitudes and latitudes are linear to map coordinates. Therefore we define our map coordinates

$$[\theta, \phi] \tag{1}$$

that ranges as

$$[0 \leq \theta < 2\pi, \quad 0 \leq \phi < \pi]$$

*Email: bjorn.odelfalk@chello.se Webpage: http://members.chello.se/odelfalk

to cover all surface on the sphere. Now we have to transform there map coordinates into 3D cartesian coordinates

$$[x_2, y_2, z_2] = [\sin(\phi)\cos(\theta), \sin(\phi)\sin(\theta), \cos(\phi)]. \tag{2}$$

The earth rotates diurnal around its z-axis and we have to define an angle $0 \leq \alpha < 2\pi$ that can be calculated from the specific time of day. If $0 \leq t < 24$ is the time in hours we have the relationship

$$\alpha = 2\pi\frac{t}{24} = \frac{\pi t}{12}. \tag{3}$$

The rotation around the z-axis can be expressed like this:

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}. \tag{4}$$

Everyone knows by now that Earth is in orbit around the sun and since Earths' axis of rotation is tilting we have to consider the declination. Therefore we have to define $-d_c \leq \beta \leq d_c$, the declination angle with a maximum of $d_c = 23.5°$. If we define $1 \leq m < 13$ as the month[1] we have the relationship:

$$\beta = -d_c\frac{\pi}{180}\cos\left(2\pi\frac{m-1}{12}\right). \tag{5}$$

The rotation is now around the y-axis:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos(\beta) & 0 & -\sin(\beta) \\ 0 & 1 & 0 \\ \sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}. \tag{6}$$

Now we have this vector:

$$[x(\alpha, \beta, \theta, \phi), y(\alpha, \beta, \theta, \phi), z(\alpha, \beta, \theta, \phi)]$$

which we want to calculate the angular difference from the sun. Let's say the sun i placed at

$$[x_s, y_s, z_s] = [\lim_{x\to\infty} x, 0, 0].$$

[1]This variable is ofcourse continous, days are the decimals of months and so on.

The calculations becomes much more easier if we assume the sun is placed in infinity, since exactly 50% of the earth is illuminated. Therefore, if the angle, γ, between $[x, y, z]$ and $[x_s, y_s, z_s]$ is

$$-\frac{\pi}{2} < \gamma < \frac{\pi}{2}$$

the Earth is illuminated, otherwise it's not. We can calculate γ with the dot-product as

$$\gamma = \arccos\left(\frac{[x, y, z] \cdot [x_s, y_s, z_s]}{|[x, y, z]||[x_s, y_s, z_s]|}\right)$$

where $|[x, y, z]| = 1$ since we are on a sphere of unity and $|[x_s, y_s, z_s]| = \lim_{x\to\infty} x$. The equation can be simplifed to

$$\gamma = \arccos\left(x\frac{\lim_{x\to\infty} x}{\lim_{x\to\infty} x}\right) = \arccos(x). \tag{7}$$

This result tells us we can neglect alot of unnecessary calculations. Let's start by calculating x from (6):

$$x = \cos(\beta)x_1 - \sin(\beta)z_1$$

both x_1 and z_1 can be calculated from (4) and we have

$$x_1 = \cos(\alpha)x_2 - \sin(\alpha)y_2, \quad z_1 = z_2.$$

We can get x_2, y_2 and z_2 from (2) which gives

$$x = \cos(\beta)\Big(\cos(\alpha)\sin(\phi)\cos(\theta) - \sin(\alpha)\sin(\phi)\sin(\theta)\Big) - \sin(\beta)\cos(\phi)$$

so that (7) becomes

$$\gamma = \arccos(\cos(\beta)\sin(\phi)\Big(\cos(\alpha)\cos(\theta) - \sin(\alpha)\sin(\theta)\Big) - \sin(\beta)\cos(\phi)) \tag{8}$$

since

$$\cos(\alpha)\cos(\theta) - \sin(\alpha)\sin(\theta) = \cos(\alpha + \theta)$$

it simplifies to

$$\gamma = \arccos(\cos(\beta)\sin(\phi)\cos(\alpha + \theta) - \sin(\beta)\cos(\phi)). \tag{9}$$

Believe it or not, but there in no way to further simplifying this equation.

2 Implementation

In this section I will describe how this mathematical formulae can be converted into computer code. There is a big difference how this code can be written depending on if your system has a FPU (floating point unit) or not.

Functions like $\sin(x)$ gets really slow if your system hasn't got a FPU. This can be solved, however, by precalculating function values that does not change if some parameters are static.

2.1 Computer with FPU

If your system includes a FPU you don't have to be shy about using trigonometric funtions. But this does not mean that you don't have to examine if there could be any simple optimization to do.

If you have a screen with dimensions `[0, 0, XMAX-1, YMAX-1]` let the variable `X` go from `0` to `XMAX-1` so that `THETA = X/XMAX*2*PI`. Let `Y` go from `0` to `YMAX-1` so that `PHI = Y/YMAX*PI`.

`ALPHA` is calculated as suggested in (3) so that `ALPHA = PI*T/12` where `T` is the current time. `BETA` is calculated as suggested in (5) so that `BETA = -DC*PI/180*COS(2*PI*(M-1)/12)` where `M` is the current month and `DC = 23.5`.

`GAMMA` is now calculated as in (9) so that

```
GAMMA=ACOS(COS(BETA)*SIN(PHI)*COS(ALPHA+THETA)-SIN(BETA)*COS(PHI))
```

The only thing left to do is the evaluation of `GAMMA`. If `GAMMA` $< \frac{\pi}{2}$ sun is up and is `ABS(PI/2-GAMMA)` degrees over the horizon.

That's it!

2.2 Computer without FPU (TI-89)

The code in the subsection above can be optimized alot if you are willing to sacrifice two things. The first thing is that size will replace speed. The second thing is that you have to have a static resolution, which the TI-89 has luckily.

The screen on a TI-89 has the dimensions `[0, 0, 159, 99]`. Let `X` go from `0` to `159` and let `Y` go from `0` to `99`.

If we use equation (8) to calculate `GAMMA` we see that `SIN(PHI)`, `COS(THETA)`, `SIN(THETA)` and `COS(PHI)` will be fixed values since `(X, Y)` always goes from `(0, 0)` to `(159, 99)`. Therefore we can precalculate these values and store them corresponding arrays. We'll create `SINPHI[0 to 99]=SIN((0 to 99)/100*PI)`

and `SINTHETA[0 to 159]=SIN((0 to 159)/160*2*PI)`, the same with arrays `COSPHI` and `COSTHETA`. These arrays has to be static, i.e. calculated before you compile so that the data is part of the program.

Left in (8) is the `COS(BETA)`, `COS(ALPHA)`, `SIN(ALPHA)` and `SIN(ALPHA)`. These variables are only calculated once for every run. Therefore you should calculate their values before the `FOR`-loop as `COSBETA=COS(BETA)` and analogous to the other three variables.

The only remainder in (8) is now the `ACOS()` function. But we know from algebra that

$$\arccos(x) = \frac{\pi}{2} - \arcsin(x)$$

and that x and $\arcsin(x)$ switches sign and have the same derivative at $x = 0$. This means that we can get ris of the `ACOS()` function still evaluate `GAMMA` > 0 as the situation when the sun is over the horizon.

By the way, `GAMMA=COSBETA*SINPHI[Y]*(COSALPHA*COSTHETA[X]-SINALPHA*SINTHETA[X])-SINBETA*COSPHI[Y]`. That's it!

3 Conclusions

Nothing to say really. I like the idea of this program, but if you want it to be user-friendly it really has to be fast. I hope some of you programmers have read this paper and get the inspiration to produce your own replica.

Please send me comments about this paper or about the program.