Subject: Re: Speedy Julia Set Fractals
Posted by Chris[6] on Sun, 06 Sep 2009 22:33:49 GMT
View Forum Message <> Reply to Message

On Sep 6, 11:44 am, Caleb <calebwhe...@gmail.com> wrote:

```
> Hello!
>
> I have a quick question about some fractal work I am doing. I know
> that doing matrix multiplications and histograms can exponentiate
> processes that are historically done with for loops. I have been
> trying to think of a way to do this with a fractal program I just
> wrote. Here is a snippet of the code that I want to speed up:
>
> <code>
>
> ; Loop through and do calculations on each point:
>   FOR i = 0, x_size-1 DO BEGIN
>
>     FOR j = 0, y_size-1 DO BEGIN
>
>       ; Initialize number of iterations:
>       num = 0
>
>       ; Complex value of the current coordinate point:
>       z =  COMPLEX(FLOAT(i-X_OFFSET)/(X_OFFSET*SCALE),FLOAT(j-Y_OFFSET) /
> (Y_OFFSET*SCALE))
>
>       ; Calculate value of F(z) at above z:
>       z1 = z^K + c
>
>       ; Take magnitude of the above value (z1):
>       mag = ABS(z1^K + c)
>
>       ; Do loop until mag is greater than threshold or max iterations
> have been calculated:
>       WHILE ((mag LE THRESH) AND (num LT MAX_ITERATION)) DO BEGIN
>
>         ; Re-Calculate value of F(z) at above z1:
>         z1 = z1^K + c
>
>         ; Take magnitude of the above value (z1):
>         mag = ABS(z1^K + c)
>
>         ; Increment iteration variable:
>         num++
>
>       ENDWHILE
>
```

```
>       ; Value of matrix is set to iteration number:
>       grid(i,j) = num
>
>     ENDFOR
>
>   ENDFOR
>
> </code>
>
```

> My problem is that I have a while loop for every iteration of my
> matrix which can run up to 256 iterations if need be. Can I speed of
> these calculations without going to multiple cores?
>
> Oh and if you need more of the code let me know and I'll post it.
>
> Thanks!
>
> Caleb Wherry

This might work (untested)

```
xs = rebin( indgen(x_size), x_size, y_size)
ys = rebin(1#indgen(y_size), x_size, y_size)
 z =  COMPLEX(FLOAT(xs-X_OFFSET)/(X_OFFSET*SCALE),FLOAT(ys-Y_OFFSE T)/
(Y_OFFSET*SCALE))

grid = intarr(x_size, y_size)
todo = grid + 1

for num = 0, num lt maxiter, 1 do begin
     z1 = z^K + c
     mag = ABS(z1^K + c)

     hit = (mag le thresh)
     grid = num * todo * hit + grid * (1 - todo)
     todo = 1 - hit
endfor
```

This avoids the nested loop over x indices and y indices. It pays an
extra penalty of running the iteration on every pixel MAXITER times.
This code assumes that MAG decreases at every step, even after THRESH
is crossed. I'm not sure if this is guaranteed to be true or not,
depending on K and C. Unless most pixels are supposed to be iterated
far fewer than MAXITER times, my guess is that this code will be
faster

Chris