back

Gravity Particle Compo

The Mission

Finally, after a long break, here is another competition. Your task is to size optimize a gravity particle routine.

The deadline will be : **Sunday, July 5th 1998 at 18.00 CET**. Everything has to be mailed to amycoders@ibm.net till then.

Your Task:

You have to simulate a lot of particles with a very small mass moving around an object with a big mass. Interaction between the particles is not taken into account.

A pseudocode program:

ob.x, ob.y = Position of the heavy object.
gravity= gravity (a constant)
part[] = array of particles
part[].px, part[],py = position of particle
part[].vx, part[].vy = velocity of particle

1. Loop i=1...n
2. xdif = ob.x - part[i].px
3. ydif = ob.y - part[i].py
4. length = xdif*xdif+ydif*ydif
5. length = length * gravity
6. part[i].vx = part[i].vx + xdif / length ;<-- Please note, that dividing by zero is not very healthy. Do something against it.
7. part[i].vy = part[i].vy + ydif / length
8. part[i].px = part[i].px + part[i].vx
9. part[i].py = part[i].py + part[i].vy
10. *{ ... plotparticle ... }*
11. loopend

This is only an approximation of course.

- Good values for *"gravity"* are 0 .. 1.0. So you *might* leave it away.
- The amount of particles you have to plot is 4096. No less no more. (They may not be visible at once due to clipping)
- The *"heavy object"* must be positioned at the middle of the screen (128;128)
- Your routine will be called with the following parameters:
  - **A0** pointer to a 256x256 pixel chunky screen.
  - **A1** pointer to the particle array. The particle struct is like this:

    0 word px
    2 word py
    4 word vx
    6 word vy

    so the structure has a size of 8 bytes.
    Each word represents a 12.4 bits fixed point number. (You will need fixed point numbers for a working implementation)

- Your plotpartice must set a single pixel of an aribitrary color except 0.
- You should check whether the particle is in the visible screen area before plotting it. That means: All visible particles have to be drawn, no "invisible" particle may be drawn.
- All your data has to be in a BSS section or it will be added to your routines length.

The Results

This time judgeing and ranking the entry has been quite difficult. Four out of eight entries violated the rules in some way. One entry didnt work at all and was taken out because of that. (Daves entry).The other entries had flaws like uninitalized variables (registers) or they were plotting pixels into memory other than the chunky buffer.

I solved this problem by added the number of bytes needed to fix the flaw plus two extra bytes as "penality"

Thanks must go to **nao** for helping to test and rank the routines

Place	**Contributor**	**Length**
1.	Merko	60+2+2p *) ***)
2.	Nao	64+0+2p **)
3.	Villelas&Dr. Slump	66+0+2p **)
	Blueberry	68 ***)

6.	Piru	74 ***)
7.	Psycho	76 ***)

*) Merkos routine didnt initalise the register **D4**. Thus the first particle could be drawn at an erranous position. A **moveq #0,d4** had to be added to fix this, adding two bytes.

) These routines drew pixel into memory areas adjacent to the chunky buffer. This could have been easily fixed by inserting an **eor #$8080,dx to calculate the pixel position offset instead of using an indexed addressing mode. No extra byte required.

***) These routines used an assymetric "particle space" by setting the "heavy object" to (128;128) in the particle space (thats not the screen space). Since the placement of the heavy object in the "particle space" wasnt mentioned in the competition rules I didnt count this as flaw.

Download

In case you want to see the contributions code. Download the package here ***Please remember that even though you can download these routines, they are still not public domain. Ask before using any of these routines, or give credits at least.***

Last change: 16.01.2001
