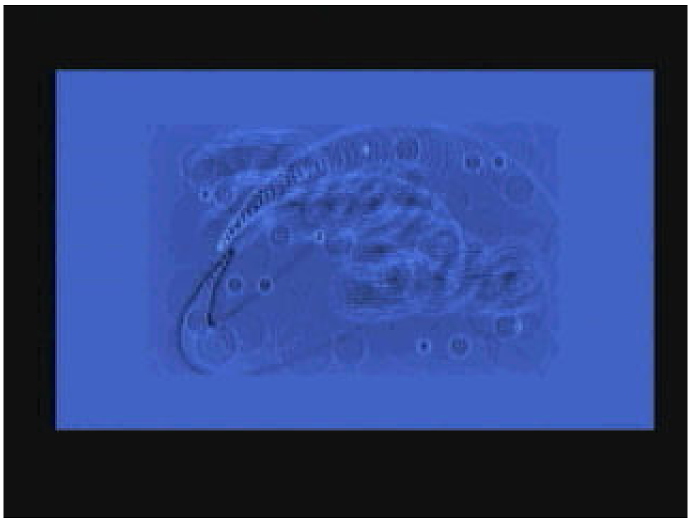
Water-Compo

The mission



Water compo ? Sounds like sun'n'fun. But well, of course you have to code something - a routine simulating water in a 2d-array. These routines are probably known to most people, but it seems to me that still many people dont really know, what makes them tick. So I added some links, where you can read more about this.

- Kimmo Riomelas Homepage

Again we will have a "two-compos-in-one" package: **fastest waterroutine** and **shortest waterroutine**.

The deadline was: Sunday, December 14th, 1997.

Rules:

- Your contribution has to contain up to three routines: **Water_Init**, which will be called once to initialise your tables or whatever, **Water_drop** this has to initiate water movement at a given position of your water-array (like throwing a stone in it, or whatever :)) and **Water_do** which is rendering the next water frame to a given buffer.
- All tables or buffers have to be in a bss-section. Otherwhise they will be counted to the routines length.
- **Water_do** is getting a pointer to a chunky screenbuffer in **A0**. All other registers contain unpredictable values, when the routine is called.
- The screenbuffers size is 256x256. 0 is the darkest color, 255 the brightest.
- For the size-compo only the length of **Water_init** and **Water_do** is measured.
- The speed of the routines will probably be measured on 060/50. I will try to measure it also on 030/50 and 040/40.
- **Water_drop** will get the drops position in x=D0.l and y=D1.l.
- Try to avoid overflows , when there are lots of waves !!
- Your entry will be published here, when the compo is over. In case you dont want that, please add a little note.

Here is a piece of the test-code for the water routines. Try it to make sure, that your routine doesnt produce any overflow with it!

```
                moveq   #31,d7
        .lop1
                move.l last,d4
                ror.l    d4,d4
                add.l    d7,d4
                move.l d4,last
                moveq    #0,d0
```

```
moveq    #0,d1
move.b d4,d0
lsr.w    #8,d4
move.b d4,d1
jsr      water_drop
dbf      d7,.lop1
```

This is called once per frame, before the routine **water_do** is called. It is producing a kind of "rain" consisting of 32 "drops" per frame.

The Results

This time the participation was really low - only three entries in the "fastest" compo and four entries in the "shortest" compo. Not much to say about the results. The "shortest" compo was won by me (**Azure**), the "fastest" compo was won by **Graham**. All routines were tested on 68060/50.

Strangely this time several people had problems with the rules. **R.A.Y.** even did far more, than was needed. His shortest entry is also doing a kind of bumpmapping additionally, but despite of that his routine has some flaws. (Drops where no drops should be :) **Shin** forgot to adjust the water to the palette. Well - due to these problems I simply judged both entries as third placers.

Shortest Water Results:

Place	Contributor	Length
1.	Azure	44/46
2.	Graham	54
3.	Shin	(58)
3.	R.A.Y.	(60)

Fastest Water Results:

Place	Contributor	Speed
1.	Graham	126
2.	PG	134
3.	Romeo	187

How was it done ?

The winning routine of the "Shortest Water" compo by **Azure**, 46 bytes version: (the 44 bytes version has a little flaw)

```
        lea             data-5,a1
        move.l          (a1),d7
        lea             5(a1,d7.l),a2
        eor.w           #1,(a1)
        add.l           (a1)+,a1
.loop
        move.b          (a1)+,d0
        add.b           -256(a1),d0
        add.b           256(a1),d0
        add.w           (a1),d0
        asr.b           #1,d0
        sub.b           (a2),d0
        spl             d1
        and.b           d1,d0       ; damping ??!? :)
        move.b          d0,(a2)+
        move.b          d0,(a0)+
        addq.w          #1,d7
        bne.b           .loop
```

Download

In case you want to see the contributions code. Download the package here ***Please remember that, even if 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