Triangledrawing-Compo

The mission

Another competition of the sort "draw this, short and fast". This time your aim should be either to create a very short or a very fast routine to draw a simple flat triangle.

The choice of your algorithm is free. The rules are below.

The deadline was: **sunday, november 9th, 1997 at 21.00 CET**

Rules:

- The size of the buffer is 256x256, with a color depth of 256.
- Your triangle-routine has to accept following input:
  - D0.l color
  - A0 Pointer to chunky buffer.
  - A1 Pointer to six words containing the screen coordinates of the triangles edges. (X1/Y1/X2/Y2/X3/Y3) Note, that that the coordinates arent sorted in any way
  - The other registers are in undefined state. (Your routine will be disqualified, if it is using input from any other register, than the mentioned ones)
- All registers may be trashed, except a7.
- The Triangle has to be filled. No missing pixels !
- No clipping required
- The triangles size is Xdelta=1-127, Ydelta=1-127
- Table may be precalculated within an init-routine, which is called once at the beginning. Maximum table size is 512kb.
- The routine must be working on 68020-68060! When you are sick enough to use selfmodifying code, make sure that it does also work on 040/060.
- The speed-tests will be done on a 68040/40. The time needed to draw an amount of triangles at different positions and with different sizes will be measured. (small hint: I added some more info in the compo-machine in the linecompo package.

The Results (shortest)

Triangles seem to be a really scary thing for most of the coders. Only 5 contributions in this compo, but it seems that the "best case" in length is still reached. The routines had to draw 4 triangles of different sizes and different positions. Three routines drew the triangles with only 40(!) bytes. But **Bluberry**s routine was the cleanest and with only 300 rasterlines the fastest routine among all contributions to this compo. So it is the clear winner.

Piru pointed out later, that it is even possible to get this routine to 38 bytes with a dirty trick - by using (a0).b as loop counter.

Place	Contributor	Length	Speed	Used Algorithm	Accuracy
1.	Blueberry	40	300	limited dual Interpolation	good
2.	Zuikki *	40	~15000	Subdivision	bad
	Psycho *	40	~187000 (!)	Subdivision	bad
4.	Piru	42	1956	Recursive subdivision	bad
5.	Nao	48	540	Dual interpolation	average

* Zuikkis and Psychos routines were moved down due to their bad accuracy and their extremly low speed.

The winning routine by **Blueberry**: (40 bytes)

```
        movem.l (a1),d1/d3/d4
        sub.l   d1,d3   ; x2-x1 | y2-y1
        sub.l   d1,d4   ; x3-x1 | y3-y1
        lsl.l   #8,d1   ; 8.8 precision
        st.b    d7      ; Loop 256 times.
.loop1: move.l  d1,d2
        move.b  d7,d6   ; Loop n times, this makes a triangle.
.loop2: move.l  d2,d5
        lsr.w   #8,d5
        swap.w  d5
        lsr.l   #8,d5   ; x,y
        move.b  d0,(a0,d5.l)
        add.l   d4,d2   ; Inner position
        subq.b  #1,d6
        bcc.b   .loop2
        add.l   d3,d1   ; Outer position
        subq.b  #1,d7
        bcc.b   .loop1
```

The Results (fastest)

Only 4 contributions here. But still quite a variety, which lead to some problems with judgeing them. All routines are showing totally different habits, depending on the cpu used. In the table are the results (rasterlines for drawing 512+4 polys) for Blizzard 1230/50, Apollo (?) 1240/40 and Apollo 1260/50. As you can see the ranking is different for every cpu.

To get an overall ranking every routine got points depending on their ranking on each seperate cpu. (4= first place, 1=last place) These points were added. As you can see the places are very close, but all over all. **Bolts** routine got the most points. **Blueberrys** routine used longword-aligned writes, which made it very fast on 030. But the table reads, that were neccessary due to this, lowered the performance on 040/060.

Place	Contributor	Length	Speed (030)	Speed (040)	Speed (060)	Points
1.	Bolt	478	1532	940	738	3+4+4=11
2.	Blueberry	374	1304	952	774	4+3+3=10
3.	Piru	570	1633	1050	843	2+2+1=5
4.	Kalms	296	1822	1133	790	1+1+2=4

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