back

Fast Truecolor C2Ps

How to make a fast 12bit/15bit truecolour c2p on amiga

by Rune Stensland (runebs@ifi.uio.no), February 1998. HTML-Version by Tim Böscke.

The amiga is getting older and older, and the AGA chipset doesn't support screen modes that are common on other platforms. such as trueclour chunky modes. I've recently been studying how to make fast truecolour chunky2planar routines on MC68030+, and I've come up with a couple of very interesting solutions. (sourcecode included at bottom of txt.)

The 12bit Algorithm

I use a 1280*xxx shires screen in 6bpl ham6 mode. This means that 4 shires pixels make one truecolour pixel in low (320*xxx) resolution. The upper two bitplanes form a RGBB mask. (If you don't know how ham works read about it in the amiga reference manual). To simplify the chunky2planar routine and get a copyspeed on 030+ solution I've decided to use scrambled truecolour. This means that the bit arrangements of each 12 bit chunky pixel goes like this:

```
r0g0b0b0 r3g3b3b3 r1g1b1b1 r2g2b2b2
```

instead of:

```
00000000 r3r2r1r0 g3g2g1g0 b3b2b1b0
```

or similar.

As you can see I use 4 red, 4 blue, 4 green bits + a copy of the 4 blue bits to fill up the word. 4 and 4 bits are going to be moved to the same bitplane using scrambled colour, this removes 2 merges in the c2p algoritm (If you are unfamilar with the merge logic take a look at the c2p doc by Michael Kalms)

4 longwords containing 8 12bit truecolor pixels.

```
1 r0g0b0b0 r3g3b3b3 r1g1b1b1 r2g2b2b2 R0G0B0B0 R3G3B3B3 R1G1B1B1 R2G2B2B2
2 r0g0b0b0 r3g3b3b3 r1g1b1b1 r2g2b2b2 R0G0B0B0 R3G3B3B3 R1G1B1B1 R2G2B2B2
3 r0g0b0b0 r3g3b3b3 r1g1b1b1 r2g2b2b2 R0G0B0B0 R3G3B3B3 R1G1B1B1 R2G2B2B2
4 r0g0b0b0 r3g3b3b3 r1g1b1b1 r2g2b2b2 R0G0B0B0 R3G3B3B3 R1G1B1B1 R2G2B2B2

Swap 16 (1X3) (2X4)                                    12*2 cycles (030)
```

```
1 r0g0b0b0 r3g3b3b3 r1g1b1b1 r2g2b2b2 r0g0b0b0 r3g3b3b3 r1g1b1b1 r2g2b2b2
3 R0G0B0B0 R3G3B3B3 R1G1B1B1 R2G2B2B2 R0G0B0B0 R3G3B3B3 R1G1B1B1 R2G2B2B2
2 r0g0b0b0 r3g3b3b3 r1g1b1b1 r2g2b2b2 r0g0b0b0 r3g3b3b3 r1g1b1b1 r2g2b2b2
4 R0G0B0B0 R3G3B3B3 R1G1B1B1 R2G2B2B2 R0G0B0B0 R3G3B3B3 R1G1B1B1 R2G2B2B2

Swap 4 (1X3) (2X4)                                          18*2 cycles (030)

1 r0g0b0b0 R0G0B0B0 r1g1b1b1 R1G1B1B1 r0g0b0b0 R0G0B0B0 r1g1b1b1 R1G1B1B1
3 r3g3b3b3 R3G3B3B3 r2g2b2b2 R2G2B2B2 r3g3b3b3 R3G3B3B3 r2g2b2b2 R2G2B2B2
2 r0g0b0b0 R0G0B0B0 r1g1b1b1 R1G1B1B1 r0g0b0b0 R0G0B0B0 r1g1b1b1 R1G1B1B1
4 r3g3b3b3 R3G3B3B3 r2g2b2b2 R2G2B2B2 r3g3b3b3 R3G3B3B3 r2g2b2b2 R2G2B2B2

Swap 8 (1X2) (3X4)                                          18*2 cycles (030)

1 r0g0b0b0 R0G0B0B0 r0g0b0b0 R0G0B0B0 r0g0b0b0 R0G0B0B0 r0g0b0b0 R0G0B0B0
2 r1g1b1b1 R1G1B1B1 r1g1b1b1 R1G1B1B1 r1g1b1b1 R1G1B1B1 r1g1b1b1 R1G1B1B1
3 r3g3b3b3 R3G3B3B3 r3g3b3b3 R3G3B3B3 r3g3b3b3 R3G3B3B3 r3g3b3b3 R3G3B3B3
4 r2g2b2b2 R2G2B2B2 r2g2b2b2 R2G2B2B2 r2g2b2b2 R2G2B2B2 r2g2b2b2 R2G2B2B2
```

12*2+18*4 = 96 cycles

96/4 = 24 cycles + longword chipwrites (loop and register misses not counted)

On my 030 50mhz I can memory pipeline 26 cycles during a longword chipmem write.

The 15bit Algorithm

In the 15 bit c2p I use a 8bpl ham8 screen in 1280*xxx, And writes to 5 planes. My screen mask is now placed in the 2 first bitplanes instead of the 2 last (ham8 design). My mask is now RGBR, look at end of description for details.

The chunky format is scrambled similar to my 12 bit solution, but instead of having an extra copy of b3b2b1b0 I place the r4g4b4b4 (the 5th bits.)

```
      **       **       **       **
r0g0b0r4 r3g3b3g4 r1g1b1b4 r2g2b2d4
```

After 3 merges(same as in the ham6 c2p) i get:

```
1 r0g0b0r4 R0G0B0R4 r0g0b0r4 R0G0B0R4 r0g0b0r4 R0G0B0R4 r0g0b0r4 R0G0B0R4
2 r1g1b1b4 R1G1B1B4 r1g1b1b4 R1G1B1B4 r1g1b1b4 R1G1B1B4 r1g1b1b4 R1G1B1B4
3 r3g3b3g4 R3G3B3G4 r3g3b3g4 R3G3B3G4 r3g3b3g4 R3G3B3G4 r3g3b3g4 R3G3B3G4
4 r2g2b2b4 R2G2B2B4 r2g2b2b4 R2G2B2B4 r2g2b2b4 R2G2B2B4 r2g2b2b4 R2G2B2B4
```

I do a seperate merge to convert the last plane data

```
3* and                                          ;6 cycles(030)

1 xxxxxxr4 xxxxxxR4 xxxxxxr4 xxxxxxR4 xxxxxxr4 xxxxxxR4 xxxxxxr4 xxxxxxR4
2 xxxxxxb4 xxxxxxB4 xxxxxxb4 xxxxxxB4 xxxxxxb4 xxxxxxB4 xxxxxxb4 xxxxxxB4
3 xxxxxxg4 xxxxxxG4 xxxxxxg4 xxxxxxG4 xxxxxxg4 xxxxxxG4 xxxxxxg4 xxxxxxG4

        add.l   d0,d0    ;xxxxr4xx...           ;10 cycles(030)
        add.l   d4,d0    ;xxxxr4g4...
```

```
        add.l   d0,d0    ;xxr4g4xx...
        add.l   d2,d0    ;xxr4g4b4...
        add.l   d0,d0    ;r4g4b4xx...

5 r4g4b400 R4G4B400 r4g4b400 R4G4B400 r4g4b400 R4G4B400 r4g4b400 R4G4B400
```

NOTE:

It's important that this longword represents the most significant bits. of the colour. This to reduce the error. The marked bits are not set to r0 R0 r0 etc. this will cause a small error in the colour.

```
      **               **               **               **       **
r0g0b0r4 R0G0B0R4 r0g0b0r4 R0G0B0R4 r0g0b0r4 R0G0B0R4 r0g0b0r4 R0G0B0R4
```

I use RGBR RGBR so the small error in the red component is only valid in 1/4 lowres pixel. This error can easyly be fixed by adding some commands, but harder to get into copyspeed limit on 030.

DMA Problem

Most experienced amiga programmers know that setting up a superhires screen in 1280*192 6 or 8 bpl. will cause the chipram to get really slow. According to my tests a chipmemwrite is around 3 times slower when the screen is displayed than when writes occur outside view area (shires 8bpl).

The 3 * faster than copyspeed trick:

In pal, the screen is divided into around 312 vertical rasterlines. only 0-256 of them are visible. If you use a 1280*192 screen you get 312-192 = 120 rasterlines where the screen is not activated -> 0 screen DMA acces.

The idea behind the 3 * copyspeed trick, is to do the c2p convertion in these 120 rasterlines, where chipmem is 3 times faster. The effect rendering (to fastram chunky buffer) is done while the screen is displayed,(192 rasterlines in my example). If the c2p is done before effect 100% of the time left is used to render the effekt. If the effect is done before the c2p 100% of the time left is used to do the c2p convertion.

So how does such a routine work?

IMPORTANT: *THE 3* COPYSPEED TRICK WILL NOT WORK IN A MULTITASKING ENVIRONMENT REMEMBER TO SHUT DOWN THE SYSTEM.*

I use 2 interrupts triggered by the copper, one when the frame starts, and one when the frame ends. Each time the interrupt is triggered I change state from c2p->effectrender or effektrender->c2p. Inside the interrupt I exgange all registres used by the current routine (a0-a7/d0-d7,PC,SR), the Program counter and status register changes are performed by writing to the supervisor stack, so when I RTE the system will automaticly change to the right PC,Sr)

How much do I save?

On my MBX1230 50mhz board a 1280*192 ham6 screen is converted in around 379 rasterlines, around 1.21 frame (screen DMA turned off). Ham7 is converted in 448 rasterlines (1.4 Frames) If

the c2p should run at 0 screen-dma speed the effekt rendered must take more than 604 rasterlines

379/120 = 3.15 ;number of frames needed to complete c2p.

3.15*192 = 604 rasterlines = 1.93 frame.

The more time your effect use to render, the less time the c2p convertion will use.

Last word from the author

I would like to thank you all for reading this document, now I hope to see some fast 1*1 truecolour rutines on amiga (Except my own :))!
I would also like to thank Brekke/Contraz Espen Brekke (espenbr@ifi.uio.no) who has helped me with the 3 * Copyspeed trick.

Additional code greetings in random order to:

Azure,Kalms,Ludde,Jamie,Eft,Stelios,Shin,Accede,Chauple,Cyberstarr,Romeo Psalt,slummy,Dig-it,rubberduck,Piru,dave,blackwine,chip,raylight,thain,grey duken.

.. and the rest on #amycoders IRC channel!

Any questions? Contact me at:

Rune Stensland (SP^CTZ) runebs@ifi.uio.no

The Sources

I've decided to include my 12 bit and 15 bit c2p's. Please give me credit if you decide use them in your productions :)

Download the sources here (shiftclick)

Last change: 16.01.2001
