Double screen vertical scroller

By Bitbreaker/Oxyron^Arsenic^Nuance

Glues together two hires/koala bitmaps and displays them stacked vertically. Format of the Hires pics (as exported from for e.g. HermIRES):

```
$0000-$1f40 bitmap
$1f40-$2328 screen
background/bordercolor: i fucking don't care :-) Set it on your own in the
source :-)
$2328-$2710 colram (koala only)
```

Use ACME to compile.

```
          * = $4000

y         = $10
y_        = $11
trig      = $12
dir       = $13
bank      = $14

colram    = $d800
screen1   = $0400
screen2   = $c400
bitmap1   = $2000
bitmap2   = $e000

;MULTICOLOR     ;uncomment this if you want a koala version that also copies
colram

          sei
          ;setup raster irq for line $2d
          lda #$01
          sta $d019
          sta $d01a
          lda #$7f
          sta $dc0d
          lda $dc0d
          lda #$18
          sta $d018
!ifdef MULTICOLOR {
          sta $d016
          ;set background color
          lda #$01
          sta $d021
}
          lda #$fa
```

```
          sta $d012
          lda #$03
          sta $dd00
          lda #<irq
          sta $0314
          lda #>irq
          sta $0315

          ;init some values, blank display to avoid glitches on start
          lda #$00
          sta $d011
          sta y
          sta trig
          sta dir
          sta bank
          sta $d020
          cli

!ifdef MULTICOLOR {
          jsr update_colram
}

l_down
          ;scroll down, fill both buffers with fitting (y-offset!) bitmaps
          sta y_
          jsr update_bmp
          lda #$02
          sta y_
          jsr update_bmp

          ;flag new direction to irq
          lda dir
          eor #$01
          sta dir

          ;enable screen on next irq
          lda #$30
          sta en1+1
down
          lda y
          cmp #$cf
          beq l_up
          ;wait until irq triggers us
          lda trig
          beq down

          ;reset flag
          lda #$00
          sta trig
```

```
          ;update offscreen buffer
          jsr update_bmp
          jmp down

l_up
          ;and up we go again...
          lda #50
          sta y_
          jsr update_bmp
          lda #48
          sta y_
          jsr update_bmp

          lda dir
          eor #$01
          sta dir
up
          lda y
          beq l_down
          lda trig
          beq up

          lda #$00
          sta trig

          jsr update_bmp
          jmp up

irq
          dec $d019

          ;update y-offset
          jsr update_y

          jmp $ea7e

update_y
          ;up or down?
          lda dir
          beq up_

          ;down, finished or inc y?
          lda y
          cmp #$cf
          bne +
          rts
+
          ;clc
          ;increment y
          adc #$01
          sta y
```

```
         ;do we need to shift whole bitmap by one charblock?
         and #$07
         tax
         bne out2
         ;y position of next buffered bitmap
         lda y
         ;clc
         adc #$08
         bne out1 ;branch always

up_
         lda y
         bne +
         rts
+
         ;decrement y
         sec
         sbc #$01
         sta y

         ;do we need to update buffer?
         and #$07
         tax
         cmp #$07
         bne out2
         ;calc new y_
         lda y
         ;sec
         sbc #$08
out1
         ;calculate y_ -> y / 4 & $fe to have an index into pointer table
         lsr
         lsr
         and #$fe
         sta y_
         ;toggle vic bank
         lda $dd00
         eor #$03
         sta $dd00
         ;now trigger code outside of irq
         inc trig
!ifdef MULTICOLOR {
         txa
         ora en1+1
         ;set new y-shift
         sta $d011
         ;and update colram
         jmp update_colram
}
```

```
out2
         txa
en1      ora #$00
         ;set new y-shift
         sta $d011
         rts

!ifdef MULTICOLOR {
update_colram
         ;directly copy colram (forward, so that upper lines are copied
first and we can't be overtaken by raster)
         lda y
         lsr
         lsr
         and #$fe
         tay
         lda #>colram
         sta tgt_c+2
         lda pcolram+0,y
         sta src_c+1
         lda pcolram+1,y
         sta src_c+2

         ldx #$04
         ldy #$00
-
src_c    lda $1000,y
tgt_c    sta $1000,y
         iny
         bne -
         inc src_c+2
         inc tgt_c+2
         dex
         bne -
         rts
}

update_bmp
         ;toggle buffer
         lda bank
         eor #$01
         sta bank
         ;fill corresponding buffer
         beq +
         lda #>bitmap1
         jsr copy_bmp
         lda #>screen1
         bne copy_screen
+
         lda #>bitmap2
         jsr copy_bmp
```

```
         lda #>screen2

copy_screen
         ;copy $0400 bytes of screen to target buffer
         sta tgt1+2
         ldx y_
         lda pscreen+1,x
         ldy pscreen+0,x
         ldx #$01   ;1 * $0400 bytes
         bne copy

copy_bmp
         ;copy $2000 bytes of bitmap to target buffer
         sta tgt1+2
         ldx y_
         lda pbitmap+1,x
         ldy pbitmap+0,x
         ldx #$08   ;8 * $0400 bytes = $2000 bytes
copy
         ;slightly optimized copy routine, doing $400 bytes at once
         sty src1+1
         sty src2+1
         sty src3+1
         sty src4+1

         ldy tgt1+2
copy_l
         ;write highbytes of source and target addresses
         sty tgt1+2
         iny
         sty tgt2+2
         iny
         sty tgt3+2
         iny
         sty tgt4+2

         tay
         sta src1+2
         iny
         sty src2+2
         iny
         sty src3+2
         iny
         sty src4+2

         ldy #$00
-
src1     lda $1000,y
tgt1     sta $1000,y
src2     lda $1000,y
```

```
tgt2     sta $1000,y
src3     lda $1000,y
tgt3     sta $1000,y
src4     lda $1000,y
tgt4     sta $1000,y
         iny
         bne -

         lda tgt1+2
         clc
         adc #$04
         tay
         lda src1+2
         clc
         adc #$04
         dex
         bne copy_l
         rts

pbitmap
         ;pointers into picture bitmaps for y=0, 8, 16, 24, ...
         !for .y, 26 {
              !word (26 - .y) * 320 + bitmaps
         }

pscreen
         ;pointers into picture screens for y=0, 8, 16, 24, ...
         !for .y, 26 {
              !word (26 - .y) * 40 + screens
         }

!ifdef MULTICOLOR {
pcolram
         ;pointers into picture colram for y=0, 8, 16, 24, ...
         !for .y, 26 {
              !word (26 - .y) * 40 + colrams
         }
}

screens
;glue together screen data
!bin "pic1.prg",$3e8,$1f42
!bin "pic2.prg",$3e8,$1f42

bitmaps
;glue together bitmap data
!bin "pic1.prg",$1f40,2
!bin "pic2.prg",$1f40,2

!ifdef MULTICOLOR {
colrams
```

```
;glue together colram data
!bin "pic1.prg",$3e8,$2328
!bin "pic2.prg",$3e8,$2328
}
```