```
; Pallo
; -----
; 2006 Hannu Nuotio

; Pallo is a remake of the crap QBasic game of the same name (and author).

; Start of project: 26.8.2006
; v.1.1 - 20.10.2006 - added joyport constants
; v.1.0 - 16.9.2006 - seems to work

; Compiles with ACME 0.91
; # acme --cpu 6502 -f cbm -o pallo.prg pallo.a

; Type SYS 4096 to start or use crunched version.

; Known bugs:
;  - If warning collides with pallo and food, pallo does not get food.
;    This is done intentionally to avoid nasty coordinate check code :)

; TODO:
;  - music/sfx
;  - 2 player mode

; Memory map:
; $0000-$0fff : unused
; $1000-$xxxx : code
; $xxxx-$yyyy : variables (reused sprite data)
; $yyyy-$zzzz : sintable, strings, logo
; $5300-$5aff : character set
; $5b00-$5bff : sprites
; $5c00-$5ff7 : screen memory (color data)
; $6000-$7ff7 : font memory   (pixel data)
; $7ff8-$7fff : sprite pointers
; $8000-$ffff : unused

; Sprites:
; 0 = $5b00 = Pallo's eyes (white)
; 1 = $5b40 = Pallo's edges (brown)
; 2 = $5bc0 = Pallo's body (green)
; 3 = $5b80 = Food (light red)
; 4 = $5b80 = Warning (dark grey)
; 5 = $5b00 = Pallo2's eyes (white)
; 6 = $5b40 = Pallo2's edges (brown)
; 7 = $5bc0 = Pallo2's body (light blue)

; Interrupt states:
; $x0 - do nothing
; $01: - update "random" numbers
;   - check collisions:
;       - if none,   state = $02
;       - if food,   state = $40
```

```
;       - if killer, state = $80
;       - if both,   state = $C0
; $02 - update sprite positions, set state = $01

; Gameloop states:
;  a = if collision goto c. read joystick, explode bomb, update sprite pos.
;  b = decrease bomb & warning timeout, draw killer
;  c = check which collision(s), move food & warning, erase killer, update
score or end game

; Pallo movement:
;  basic idea:
;    X += sin(dir), Y += cos(dir)
;
;  variables:
;    pallo1x(h)/y = sprite coordinates (= X/Y)
;    pallo1subx/y = subpixel coordinates (= x/y)
;    pallo1dir = direction of movement (= dir)
;
;  direction circle:
;         $80
;       /--|--\
;      / 3 | 2 \ <- segment
; $C0 |---------| $40
;      \ 4 | 1 /
;       \--|--/
;         $00 (dir)
;
; case dir =
;  $00: Y++
;  $40: X++
;  $80: Y--
;  $C0: X--
;
; (if dir & $3f != 0 ) case segment =
;  1: X.x += sin(dir), Y.y += cos(dir)
;  2: X.x += cos(dir), Y.y -= sin(dir)
;  3: X.x -= sin(dir), Y.y -= cos(dir)
;  4: X.x -= cos(dir), Y.y += sin(dir)
;
; sintable:
;  sintable[i] = 256*sin(2*pi*i/256), i = [0,$3f]
;  sin(dir) -> sintable[dir]
;  cos(dir) -> sintable[$40-dir]



; --- Macros

; SpriteLine - for easy definition of sprites
; from "ddrv.a" by Marco Baye
!macro SpriteLine .v {
```

```
!by .v>>16, (.v>>8)&255, .v&255
}


; --- Constants

joyport = $dc00     ; joystick port
joypddr = $dc02     ; joystick port data direction register

bombpoints = 15     ; points required to get a bomb
bombtimeout = 35    ; min. time between explosions
warningtimeout = 50 ; warning->killer interval


; --- Variables

highscoretext = startscrtext10+12

tmp = Sprites

; last joystick state
lastjoy = Sprites+1

; interrupt state (explained above)
intstate = Sprites+2

; sprite coordinates (0=1=2=Pallo, 5=6=7=Pallo2)
pallo1x   = Sprites+3
pallo1xh  = Sprites+4
pallo1y   = Sprites+5
foodx     = Sprites+6
foodxh    = Sprites+7
foody     = Sprites+8
warningx  = Sprites+9
warningxh = Sprites+10
warningy  = Sprites+11
pallo2x   = Sprites+12
pallo2xh  = Sprites+13
pallo2y   = Sprites+14

; pallo dir & subpixel coordinates
pallo1subx = Sprites + 15
pallo1suby = Sprites + 16
pallo1dir  = Sprites + 17
pallo2subx = Sprites + 18
pallo2suby = Sprites + 19
pallo2dir  = Sprites + 20

; "random" variables
rnd1 = Sprites + 21
rnd2 = Sprites + 22 ; rnd[2:1] = 0..308
```

```
rnd3 = Sprites + 23
rnd4 = Sprites + 24 ; rnd[4:3] = 0..308
rnd5 = Sprites + 25 ; rnd5 = 0..182
rnd6 = Sprites + 26 ; rnd6 = 0..182

; score & bombs
bombs  = Sprites + 27   ; bomb amount
bombc  = Sprites + 28   ; bomb counter (0..15)
hscore = Sprites + 29   ; high score
score  = Sprites + 30   ; score

; timeouts
btime = Sprites + 31    ; bomb timeout
wtime = Sprites + 32    ; warning timeout

; temp char for drawbomb
tmpchar = Sprites + 33
;next variable = Sprites + 41


; --- Main

; start of program
*=$1000
mlcodeentry:

; - interrupt setup
; from "An Introduction to Programming C-64 Demos" by Puterman aka Linus
Åkerlund
; http://user.tninet.se/~uxm165t/demo_programming/demo_prog/demo_prog.html
; ... + modifications
;
sei     ; interrupts off
lda #$7f
ldx #$01
sta $dc0d   ; Turn off CIA 1 interrupts
sta $dd0d   ; Turn off CIA 2 interrupts
stx $d01a   ; Turn on raster interrupts
lda #<int    ; low part of address of interrupt handler code
ldx #>int    ; high part of address of interrupt handler code
ldy #250    ; line to trigger interrupt
sta $0314   ; store in interrupt vector
stx $0315
sty $d012
lda #<nmi    ; low part of address of NMI handler code
ldx #>nmi    ; high part of address of NMI handler code
sta $0318   ; store in NMI vector
stx $0319
lda #0
sta intstate    ; set interrupt state to 0
lda $dc0d   ; ACK CIA 1 interrupts
```

```
lda $dd0d   ; ACK CIA 2 interrupts
asl $d019   ; ACK VIC interrupts
cli     ; interrupts on

; disable bcd-mode
cld

; copy character rom to $5300-$5aff
; copy sprites to $5b40-$5bfe
ldx #0      ; 255 loops
sei     ; interrups off
lda $1
and #$fb
sta $1      ; character rom on
-   lda $d000,x    ; load from char-rom
    sta $5300,x    ; store to ram
    lda $d100,x    ; load from char-rom
    sta $5400,x    ; store to ram
    lda $d200,x    ; load from char-rom
    sta $5500,x    ; store to ram
    lda $d300,x    ; load from char-rom
    sta $5600,x    ; store to ram
    lda $d400,x    ; load from char-rom
    sta $5700,x    ; store to ram
    lda $d500,x    ; load from char-rom
    sta $5800,x    ; store to ram
    lda $d500,x    ; load from char-rom
    sta $5900,x    ; store to ram
    lda $d600,x    ; load from char-rom
    sta $5a00,x    ; store to ram
    lda Sprites,x    ; load from sprites (& sine table)
    sta $5b00,x    ; save to ram
    inx
    bne -
lda $1
ora #$04
sta $1      ; character rom off
asl $d019   ; ACK VIC interrupts
cli     ; interrupts on

; generate pallo's body sprite
ldx #64
-   lda $5b80,x    ; a = food sprite
    eor $5b40,x    ; a xor edges
    eor $5b00,x    ; a xor eyes
    sta $5bc0,x    ; save to body sprite
    dex
    bne -

; bank 1 ($4000)
lda #2
```

```
sta $dd00

; hires bitmap
; 25 rows
; file scroll = 2
; raster compare msb = 0
; blank
lda #$2b
sta $d011

; font = "screen" = $6000
; screen = "color" = $5c00
lda #$78
sta $d018

; sprite block pointers
ldx #108    ; -> $5b00
stx $5ff8   ; p1eyes
stx $5ffd   ; p2eyes
inx     ; -> $5b40
stx $5ff9   ; p1edge
stx $5ffe   ; p2edge
inx     ; -> $5b80
stx $5ffb   ; food
stx $5ffc   ; warning
inx     ; -> $5bc0
stx $5ffa   ; p1body
stx $5fff   ; p2body

; sprite colors
lda #1      ; white
sta $d027   ; p1eyes
sta $d02c   ; p2eyes
lda #9      ; brown
sta $d028   ; p1edge
sta $d02d   ; p2edge
lda #5      ; green
sta $d029   ; p1body
lda #10     ; light red
sta $d02a   ; food
lda #11     ; dark gray
sta $d02b   ; warning
lda #14     ; light blue
sta $d02e   ; p2body

; misc. vic settings
lda #0
sta $d020   ; border color
sta $d021   ; background color
sta $d01c   ; sprites = hi-res
lda #$10    ; sprite 4 < backgnd
```

```
sta $d01b   ; sprite-backgnd priority

; init random variables
lda #0
sta rnd2
sta rnd4
sta rnd5
sta rnd6

; set joyport to input
lda #0
sta joypddr

; - Pre-start screen
lda #0
sta hscore  ; high score = 0
lda #$f0    ; fg = gray, bg = black
jsr setscreencolor

; - Start screen
; clear screen
; draw logo & text
; draw highscore
; position sprites
; draw killer
; wait for fire
;
startscreen:
jsr clearscreen

; draw logo
lda #<pallologo
sta logosl
lda #>pallologo
sta logosh  ; point to logo (144x88)
lda #<$6000+8*2+320*1
sta logodl
lda #>$6000+8*2+320*1
sta logodh

ldy #11     ; y = 88/8 = 11
-       ; y loop
ldx #0      ; x = 0
--      ; x loop
    logosl = *+1
    logosh = *+2
    lda $0000,x    ; load bitmap
    logodl = *+1
    logodh = *+2
    sta $0000,x    ; store screen
    inx        ; next x
```

```
    cpx #144
    bne --        ; loop until x=144
clc
lda #64
adc logodl
sta logodl
lda #1
adc logodh
sta logodh      ; destination += 320
clc
lda #144
adc logosl
sta logosl
lda #0
adc logosh
sta logosh      ; source += 144
dey
bne -           ; next y

; draw text
lda #<startscrtext1
sta strptrl
lda #>startscrtext1
sta strptrh
ldx #28
ldy #3
jsr printstr    ; = pallo
lda #<startscrtext2
sta strptrl
lda #>startscrtext2
sta strptrh
ldx #28
ldy #5
jsr printstr    ; = food
lda #<startscrtext3
sta strptrl
lda #>startscrtext3
sta strptrh
ldx #28
ldy #7
jsr printstr    ; = warning
lda #<startscrtext4
sta strptrl
lda #>startscrtext4
sta strptrh
ldx #28
ldy #9
jsr printstr    ; = killer
lda #<startscrtext5
sta strptrl
lda #>startscrtext5
```

```
sta strptrh
ldx #7
ldy #13
jsr printstr    ; use joystick...
lda #<startscrtext6
sta strptrl
lda #>startscrtext6
sta strptrh
ldx #7
ldy #14
jsr printstr    ; < and > ...
lda #<startscrtext7
sta strptrl
lda #>startscrtext7
sta strptrh
ldx #7
ldy #15
jsr printstr    ; fire ...
lda #<startscrtext8
sta strptrl
lda #>startscrtext8
sta strptrh
ldx #4
ldy #17
jsr printstr    ; collect food...
lda #<startscrtext9
sta strptrl
lda #>startscrtext9
sta strptrh
ldx #4
ldy #18
jsr printstr    ; a bomb ...
lda #<startscrtext10
sta strptrl
lda #>startscrtext10
sta strptrh
ldx #12
ldy #21
jsr printstr    ; high score
lda #<startscrtext11
sta strptrl
lda #>startscrtext11
sta strptrh
ldx #4
ldy #24
jsr printstr

; activate sprites
lda #$1f
sta $d015
```

```
; position sprites
ldx #231
lda #73
ldy #0
stx pallo1x
sty pallo1xh
sta pallo1y
lda #89
stx foodx
sty foodxh
sta foody
lda #105
stx warningx
sty warningxh
sta warningy
sty pallo2x
sty pallo2y
sty pallo2xh

; calc & draw killer
clc
ldx #231-24
ldy #121-50
jsr calckiller
jsr drawkiller

; update sprites
lda #2
sta intstate
-   cmp intstate
    beq -    ; wait for state change
lda $d01f   ; clear sprite-data collision
lda $d01e   ; clear sprite-sprite collision

; unblank screen
lda #$3b
sta $d011

; wait for joystick fire
jsr waitfirej

; - Entering game
; clear screen
; draw game screen & text
; position sprites
; init variables
; clear any collisions
;
lda #0
sta intstate    ; interrupt does nothing
```

```
; blank screen
lda #$2b
sta $d011

; draw game screen & text
jsr clearscreen
lda #<statustext
sta strptrl
lda #>statustext
sta strptrh
ldx #0
ldy #0
jsr printstr
jsr drawborders

; init variables & sprite positions
lda #$c0
sta pallo1dir
lda #0
sta warningy    ; warning is offscreen
sta btime
sta wtime
sta score
sta bombs
sta bombc
lda #$30    ; a = char "0"
sta bombstext+0
sta bombstext+1
sta scoretext+0
sta scoretext+1
sta scoretext+2

lda #2
sta intstate    ; update sprites
-   cmp intstate
    beq -    ; wait for state change
lda $d01f   ; clear sprite-data collision
lda $d01e   ; clear sprite-sprite collision

; unblank screen
lda #$3b
sta $d011

; - Game loop
; states:
;  a = if collision goto c. read joystick, update sprite pos.
;  b = decrease bomb & warning timeout, draw killer (entry point)
;  c = check which collision(s), move food & warning, erase killer, update
score or end game

; state b
```

```
gstateb:
; decrease bomb timeout
lda btime   ; a = btime (Z if 0)
beq +       ; skip if 0
dec btime   ; btime--
; decrease warning timeout & draw killer if timeout -> 0
+ lda wtime ; a = wtime (Z if 0)
beq +       ; skip if 0
dec wtime   ; wtime--
bne +       ; skip if > 0
jsr drawkiller  ; draw killer
lda #0
sta warningy    ; move warning offscreen
+ lda #1    ; next state
-   cmp intstate
    beq -    ; wait for state change

; state a
gstatea:
; check for collisions
lda #3
bit intstate    ; intstate & 3, N & V=intstate 7 & 6
bne +       ; if !Z, proceed
jmp gstatec ; if Z, jump to state c
; read joystick
; test fire button (falling edge)
+ ldy lastjoy   ; y = lastjoy (for later use)
lda joyport ; a = joy
tax     ; save to x
eor lastjoy ; joy xor lastjoy
stx lastjoy ; update lastjoy
and #$10    ; test fire
beq gstatea_p2  ; jump if no change
txa     ; a = joy
and #$10    ; test fire
bne gstatea_p2  ; jump if rising edge
; fire pressed
; check timeout and available bombs
lda bombs   ; a = bombs (Z if 0)
beq gstatea_p2  ; jump if no bombs
lda btime   ; a = btime (Z if 0)
bne gstatea_p2  ; jump if bomb timeout
; bomb explosion
lda #1
sta $d020   ; set border white
lda #0
sta intstate    ; next state = 0
; decrease bomb amount
dec bombs
ldy #$39    ; y = char "9"
lda #$2f    ; a = char "0"-1
```

```
dec bombstext+1 ; bombstext--
cmp bombstext+1 ; test BCD underflow
bne +
sty bombstext+1 ; save "0" to bombstext
dec bombstext   ; bombstext--
; print bombs
+ jsr printbombs
; explode bomb
lda pallo1y ; a = pallo1y
sec
sbc #50     ; remove sprite offset
tay     ; y = bomb y
lda pallo1x ; a = pallo1x
sec
sbc #24     ; remove sprite offset
tax     ; x = bomb x
lda pallo1xh    ; a = pallo1xh
sbc #0
clc
beq +
sec     ; c = pallo1xh
+ jsr drawbomb  ; drawbomb at x(c=msb),y
lda #1
sta intstate    ; next state = 1
; set timeout
lda #bombtimeout
sta btime
lda #0
sta $d020   ; set border black

; test left & right (use lastjoy for inertia)
gstatea_p2:
txa     ; a = joy
and #$04    ; test left
bne +       ; jump if not active
inc pallo1dir   ; pallo1dir++
inc pallo1dir   ; pallo1dir++
+ tya       ; a = lastjoy
and #$04    ; test left
bne +       ; jump if not active
inc pallo1dir   ; pallo1dir++
+ txa       ; a = joy
and #$08    ; test right
bne +       ; jump if not active
dec pallo1dir   ; pallo1dir--
dec pallo1dir   ; pallo1dir--
+ tya       ; a = lastjoy
and #$08    ; test right
bne gstatea_c   ; jump if not active
dec pallo1dir   ; pallo1dir--
```

```
; calculate movement & update sprite position
; check if movement is straight or from sintable
gstatea_c:
lda #$3f    ; a = $3f
bit pallo1dir   ; Z = p1dir && $3f, NV = bit7&6
php     ; put flags to stack
lda #$3f    ; a = $3f
and pallo1dir   ; a = p1dir & $3f
bne +       ; skip if !Z (sintable movement)
jmp gstatea_s   ; jump if Z (straight movement)
; sintable movement (flags in stack tell the segment)
+ tax       ; x = p1dir & $3f = dir
eor #$ff    ; a = !a
clc     ; c = 0
adc #1      ; a++
and #$3f    ; a = a & $3f
tay     ; y = $40-dir
lda sintable,x  ; a = sin(dir)
tax     ; x = sin(dir)
lda sintable,y  ; a = sin($40-dir)
tay     ; y = sin($40-dir)
plp     ; pull flags from stack
bmi gstatea_34  ; jump if N (segment 3 or 4)
bvs gstatea_2   ; jump if V (segment 2)
clc     ; (segment 1:x+,y+) c = 0
adc pallo1suby  ; pallo1suby += sin($40-dir)
sta pallo1suby  ; save pallo1suby
bcc +       ; skip if no carry
inc pallo1y ; pallo1y++
+ clc       ; c = 0
txa     ; a = sin(dir)
adc pallo1subx  ; pallo1subx += sin(dir)
sta pallo1subx  ; save pallo1subx
bcs gstatea_r   ; inc x if carry
jmp gstatea_f   ; jump to state a finish
gstatea_2:  ; 2:x+,y-
stx tmp     ; tmp = sin(dir)
lda pallo1suby  ; a = pallo1suby
sec     ; c = 1
sbc tmp     ; a = pallo1suby - sin(dir)
sta pallo1suby  ; save pallo1suby
bcs +       ; skip if carry (no borrow)
dec pallo1y ; pallo1y--
+ clc       ; c = 0
tya     ; a = sin($40-dir)
adc pallo1subx  ; pallo1subx += sin($40-dir)
sta pallo1subx  ; save pallo1subx
bcs gstatea_r   ; inc x if carry
jmp gstatea_f   ; jump to state a finish
gstatea_34: ; 3 or 4
bvs gstatea_4   ; jump if V (4)
```

```
sty tmp     ; (3:x-,y-) tmp = sin($40-dir)
lda pallo1suby  ; a = pallo1suby
sec     ; c = 1
sbc tmp     ; a = pallo1suby - sin($40-dir)
sta pallo1suby  ; save pallo1suby
bcs +       ; skip if carry (no borrow)
dec pallo1y ; pallo1y--
+ lda pallo1subx    ; a = pallo1subx
stx tmp     ; tmp = sin(dir)
sec     ; c = 1
sbc tmp     ; a = pallo1subx - sin(dir)
sta pallo1subx  ; save pallo1subx
bcc gstatea_l   ; dec x if no carry (borrow)
jmp gstatea_f   ; jump to state a finish
gstatea_4:  ; 4:x-,y+
clc     ; c = 0
txa     ; a = sin(dir)
adc pallo1suby  ; pallo1suby += sin(dir)
sta pallo1suby  ; save pallo1suby
bcc +       ; skip if no carry
inc pallo1y ; pallo1y++
+ lda pallo1subx    ; a = pallo1subx
sty tmp     ; tmp = sin($40-dir)
sec     ; c = 1
sbc tmp     ; a = pallo1subx - sin($40-dir)
sta pallo1subx  ; save pallo1subx
bcc gstatea_l   ; dec x if no carry (borrow)
jmp gstatea_f   ; jump to state a finish
; straight movement (pallo1)
gstatea_s:
plp     ; pull flags from stack
bmi gstatea_ul  ; jump if N (up or left)
bvs gstatea_r   ; jump if V (right)
gstatea_d:  ; straight movement to down
inc pallo1y ; y++
jmp gstatea_f   ; jump to state a finish
gstatea_r:  ; straight movement to right
inc pallo1x ; x++
bne gstatea_f   ; jump if x!=0 ("carry"=0)
inc pallo1xh    ; x++ (msb)
jmp gstatea_f   ; jump to state a finish
gstatea_ul: ; straight, up or left
bvs gstatea_l   ; jump if V (left)
gstatea_u:  ; straight movement to up
dec pallo1y ; y--
jmp gstatea_f   ; jump to state a finish
gstatea_l:  ; straight movement to left
dec pallo1x ; x--
lda #$ff    ; a = $ff (for "carry" test)
cmp pallo1x ; test "carry"
bne gstatea_f   ; jump if x!=$ff ("carry"=0)
```

```
dec pallo1xh    ; x-- (msb)
jmp gstatea_f   ; jump to state a finish
gstatea_f:
lda #2      ; next state
-   cmp intstate
    beq -    ; wait for state change
jmp gstateb ; back to state b

; state c - a collision happened
gstatec:    ; N->killer, V->food, NV->both
php     ; save flags
bvs +       ; skip if food collision
jmp gstatec_k   ; jump if not food collision
; food collision
; increase score
+ inc score ; score++
lda #$3a    ; a = char "9"+1
ldy #$30    ; y = char "0"
inc scoretext+2 ; scoretext++
cmp scoretext+2 ; test BCD overflow
bne +
sty scoretext+2 ; save "0" to scoretext
inc scoretext+1 ; scoretext++
cmp scoretext+1 ; test BCD overflow
bne +
sty scoretext+1 ; save "0" to scoretext
inc scoretext   ; scoretext++
; print score
+ jsr printscore
; increase bomb couter
inc bombc   ; bomb counter ++
lda #bombpoints ; a = bombpoints
cmp bombc   ; if bomb counter != bombpoints,
bne ++      ; jump
; give a bomb to player
lda #0
sta bombc   ; reset bomb counter
inc bombs   ; bombs++
lda #$3a    ; a = char "9"+1
ldy #$30    ; y = char "0"
inc bombstext+1 ; bombstext++
cmp bombstext+1 ; test BCD overflow
bne +
sty bombstext+1 ; save "0" to bombstext
inc bombstext   ; bombstext++
; print bombs
+ jsr printbombs
; if warning timeout>0 draw killer
++ lda wtime    ; a = wtime (Z if 0)
beq +       ; skip if Z
jsr drawkiller
```

```
; move food & warning
+ clc
lda rnd1    ; a = rnd1
adc #25     ; add sprite x offset + 1
sta foodx   ; save to food x
lda #0      ; a = 0
adc rnd2    ; add msb + c
sta foodxh  ; save to food x msb
clc     ; c = 0
lda rnd5    ; a = rnd5
adc #59     ; add sprite y offset + 9
sta foody   ; save to food y
clc
lda rnd3    ; a = rnd3
adc #25     ; add sprite x offset + 1
sta warningx    ; save to warning x
lda #0      ; a = 0
adc rnd4    ; add msb + c
sta warningxh   ; save to warning x msb
clc     ; c = 0
lda rnd6    ; a = rnd6
adc #59     ; add sprite y offset + 9
sta warningy    ; save to warning y
; calc & erase killer below food
lda rnd5    ; a = rnd5
clc
adc #9      ; a += 9
tay     ; y = rnd5 + 9
ldx rnd1    ; x = rnd1
inx     ; x++ (Z if 255->0)
beq +
lda rnd2    ; a = rnd2
clc
beq ++
+ sec       ; c = rnd[1:2] msb
++ jsr calckiller   ; calc killer at x(c=msb),y
jsr erasekiller ; erase killer under food
; calc killer
lda rnd6    ; a = rnd6
clc
adc #9      ; a += 9
tay     ; y = rnd6 + 9
ldx rnd3    ; x = rnd3
inx     ; x++ (Z if 255->0)
beq +
lda rnd4    ; a = rnd4
clc
beq ++
+ sec       ; c = rnd[3:4] msb
++ jsr calckiller   ; calc killer at x(c=msb),y
; set warning timeout
```

```
+ lda #warningtimeout   ; a = warningtimeout
sta wtime   ; set timeout
gstatec_k:  ; test killer
plp     ; restore flags
bpl gstatec_f   ; jump if not killer collision
; killer collision
; set state to 0
lda #0
sta intstate    ; interrupt does nothing
lda #2
sta $d020   ; set border red
; check score and highscore
lda hscore  ; a = high score
cmp score   ; check if score>highscore
bcs ++      ; jump if not
; print high score
lda #<gothighscoretext
sta strptrl
lda #>gothighscoretext
sta strptrh
ldx #16
ldy #0
jsr printstr
; update highscore
lda score
sta hscore
lda scoretext+0
sta highscoretext+0
lda scoretext+1
sta highscoretext+1
lda scoretext+2
sta highscoretext+2
jmp +++     ; jump
; print game over
++ lda #<gameovertext
sta strptrl
lda #>gameovertext
sta strptrh
ldx #16
ldy #0
jsr printstr
; wait for fire (falling edge)
+++ jsr waitfirej
lda #0
sta $d020   ; set border black
; blank screen
lda #$2b
sta $d011
; goto start screen
jmp startscreen
gstatec_f:
```

```
lda #2
sta intstate    ; next state = 2
-   cmp intstate
    beq -    ; wait for state change
lda $d01f   ; clear sprite-data collision
lda $d01e   ; clear sprite-sprite collision
lda #2
sta intstate    ; next state = 2
jmp gstatea ; back to state a, read joystick


; --- Interrupt routines

; - IRQ
; Interrupt states:
; $x0 - do nothing
; $01 - update random numbers
;     - check collisions:
;   - if none,   state = $02
;   - if food,   state = $40
;   - if killer, state = $80
;   - if both,   state = $c0
; $02 - update sprite positions, set state = $01
;
int:
lda #$0f    ; a = $f
and intstate    ; a = intstate & $f
bne +
jmp int_return  ; if state = $x0, return
+ cmp #2
bne +
jmp int_sprites ; if state = 2, update sprites
; state $01: update "random" numbers
+ clc
lda #13
adc rnd1
sta rnd1
bcc +
inc rnd2    ; rnd[1:2] += 13
+ clc
lda #23
adc rnd3
sta rnd3
bcc +
inc rnd4    ; rnd[3:4] += 23
+ inc rnd5  ; rnd5 ++
clc
lda #29
adc rnd6    ; rnd6 += 29
sta rnd6
; set numbers within limits
```

```
; rnd[1:2] = 0..306
lda rnd2
beq +       ; jump if msb = 0
lda rnd1
sec
sbc #<306    ; a = rnd1-306 (lsb)
bcc +       ; jump if rnd[1:2]<306
sta rnd1    ; rnd1 -= 306 (lsb)
lda #0
sta rnd2    ; rnd[1:2] = rnd[1:2] mod 306
; rnd[3:4] = 0..306
+ lda rnd4
beq +       ; jump if msb = 0
lda rnd3
sec
sbc #<306    ; a = rnd3-306 (lsb)
bcc +       ; jump if rnd[3:4]<306
sta rnd3    ; rnd3 -= 306 (lsb)
lda #0
sta rnd4    ; rnd[3:4] = rnd[3:4] mod 306
; rnd5 = 0..180
+ lda rnd5
sec
sbc #180    ; a = rnd5-180
bcc +       ; jump if rnd5<180
sta rnd5    ; rnd5 = rnd5 mod 180
; rnd6 = 0..180
+ lda rnd6
sec
sbc #180    ; a = rnd6-180
bcc +       ; jump if rnd6<180
sta rnd6    ; rnd6 = rnd6 mod 180
; state $01: check collisions
+ ldy #$00  ; next state = $00
lda $d01e   ; sprite-sprite collision
tax     ; save to x
and #$08    ; test food
beq ++      ; skip if no collision
; test pallo
txa     ; a = sprite-sprite collision
and #$07    ; test pallo collision
beq ++      ; skip if no collision
; test warning (ignore if food+pallo+warning)
txa     ; a = sprite-sprite collision
and #$10    ; test warning collision
bne ++      ; skip if collision
ldy #$40    ; next state = $40
++ lda $d01f    ; sprite-data collision
and #$07    ; test pallo
beq +       ; skip if no collision
tya     ; a = next state
```

```
ora #$80    ; next state |= $80
tay     ; y = next state
+ cpy #$00  ; if next state = 0
bne +       ; jump if collisions
ldy #$02    ; next state = $02
+ sty intstate  ; set next state
jmp int_return  ; return
; state $02: update sprites
int_sprites:
; pallo
ldx pallo1x
lda pallo1y
stx $d000
sta $d001
stx $d002
sta $d003
stx $d004
sta $d005
; food
ldx foodx
lda foody
stx $d006
sta $d007
; warning
ldx warningx
lda warningy
stx $d008
sta $d009
; pallo2
;ldx pallo2x
;lda pallo2y
;stx $d00a
;sta $d00b
;stx $d00c
;sta $d00d
;stx $d00e
;sta $d00f
; sprite pos msbs
lda #0
;ora pallo2xh
;asl
;ora pallo2xh
;asl
;ora pallo2xh
;asl
ora warningxh
asl
ora foodxh
asl
ora pallo1xh
asl
```

```
ora pallo1xh
asl
ora pallo1xh
sta $d010   ; sprite pos msbs
dec intstate    ; next state = $01
int_return:
asl $d019   ; ACK interrupt (to re-enable it)
pla
tay
pla
tax
pla     ; pop y,x and a from stack
rti     ; return

; - NMI
;
nmi:
rti     ; return


; --- Subroutines

; - clearscreen
;
clearscreen:
ldx #0      ; 256 loops
lda #0      ; clear
-   sta $6000,x
    sta $6100,x
    sta $6200,x
    sta $6300,x
    sta $6400,x
    sta $6500,x
    sta $6600,x
    sta $6700,x
    sta $6800,x
    sta $6900,x
    sta $6a00,x
    sta $6b00,x
    sta $6c00,x
    sta $6d00,x
    sta $6e00,x
    sta $6f00,x
    sta $7000,x
    sta $7100,x
    sta $7200,x
    sta $7300,x
    sta $7400,x
    sta $7500,x
    sta $7600,x
    sta $7700,x
```

```
    sta $7800,x
    sta $7900,x
    sta $7a00,x
    sta $7b00,x
    sta $7c00,x
    sta $7d00,x
    sta $7e00,x
    sta $7ee8,x
    inx
    bne -
rts     ; return

; - setscreencolor
; parameters:
; a = screen color (0xfb, f/b=fore/background)
;
setscreencolor:
ldx #0      ; 256 loops
-   sta $5c00,x
    sta $5d00,x
    sta $5e00,x
    sta $5ee8,x
    inx
    bne -
rts     ; return

; - drawbowders
;
drawborders:
; left & right edge
ldx #0      ; 8 loops
--  lda #$61
    sta drawbordersl+1
    lda #$62
    sta drawbordersr+1
    lda #$40
    sta drawbordersl
    lda #$78        ; left dest. = (0,1)
    sta drawbordersr    ; right dest.= (39,1)
    ldy #24    ; 24 loops
-       lda #$80    ; left edge
drawbordersl=*+1
        sta $0000,x    ; save to dest.
         lda #$01    ; right edge
drawbordersr=*+1
        sta $0000,x    ; save to dest.
        clc
        lda #$40
        adc drawbordersl
        sta drawbordersl
        lda #1
```

```
        adc drawbordersl+1
        sta drawbordersl+1    ; left dest. += 320
        clc
        lda #$40
        adc drawbordersr
        sta drawbordersr
        lda #1
        adc drawbordersr+1
        sta drawbordersr+1    ; right dest. += 320
        dey
        bne -
    inx
    cpx #8
    bne --
; top & bottom edge
lda #$61
sta drawborderst+1
lda #$7e
sta drawbordersb+1
lda #$40
sta drawborderst
lda #$07        ; top dest. = (0,1)
sta drawbordersb    ; bottom dest.= (0,23)
ldx #40     ; 40 loops
-   lda #$ff    ; line
drawborderst=*+1
    sta $0000    ; save to dest.
drawbordersb=*+1
    sta $0000    ; save to dest.
    clc
    lda #8
    adc drawborderst
    sta drawborderst
    lda #0
    adc drawborderst+1
    sta drawborderst+1    ; left dest. += 8
    clc
    lda #$8
    adc drawbordersb
    sta drawbordersb
    lda #0
    adc drawbordersb+1
    sta drawbordersb+1    ; bottom dest. += 8
    dex
    bne -
rts     ; return

; - waitfirej
; returns:
;  lastjoy
;
```

```
waitfirej:
lda joyport ; a = joy
sta lastjoy ; update lastjoy
-
lda joyport ; a = joy
tax     ; save to x
eor lastjoy ; joy xor lastjoy
stx lastjoy ; update lastjoy
and #$10    ; test fire
beq -       ; jump if no change
txa     ; a = joy
and #$10    ; test fire
bne -       ; jump if rising edge
rts

; - drawbomb
; parameters:
;  x = x-coordinate, c = msb
;  y = y-coordinate
;
drawbomb:
; get tx&ty text coords from x,y
; tx-=3, ty-=3
; ex=tx+6+1, ey=ty+6+1 (end coord+1)
; check limits:
;  x: <0 -> 0, >39 -> 39
;  y: <1 -> 1, >24 -> 24
txa     ; a = x-coord.
ror     ; c->bit7
lsr
lsr     ; a = x/8, c = "0.5"
adc #-3     ; a = a-3 + c (round)
bpl +       ; jump if a>0
clc     ; c = 0
adc #7      ; a+=7
sta drawbombex  ; store x end coord
lda #0      ; a = 0
sta drawbombtx  ; store x start coord
jmp ++      ; jump to y coord conversion
+ sta drawbombtx ; store x start coord
clc     ; c = 0
adc #7      ; a+=7
cmp #40     ; check if a>40
bcc +       ; jump if a<=40
lda #40     ; a = 40
+ sta drawbombex ; store to x end address
++ tya      ; a = y-coord.
lsr
lsr
lsr     ; a = y/8, c = "0.5"
adc #-3     ; a = a-3 + c (round)
```

```
bpl +       ; jump if a>0
clc     ; c = 0
adc #7      ; a+=7
sta drawbombey  ; store y end coord
lda #1      ; a = 1
sta drawbombty  ; store y start coord
jmp ++      ; jump
+ bne +     ; jump if a!=0
lda #1      ; a = 1
+ sta drawbombty ; store y start coord
clc     ; c = 0
adc #7      ; a+=7
cmp #25     ; check if a>25
bcc +       ; jump if a<=25
lda #25     ; a = 25
+ sta drawbombey    ; store to y end address
; bombdest = ty*320+tx*8
++ lda drawbombty   ; a = ty
sta bombdesth   ; bombdesth = "256*ty"
lda #0      ; a=0
lsr bombdesth   ; bombdesth = "128*ty", c=lsb
ror         ; c->a,msb
lsr bombdesth   ; bombdesth = "64*ty", c=lsb
ror         ; bombdesth:a = 64*ty
sta bombdestl   ; bombdest = 64*ty
lda drawbombty  ; a = ty
adc bombdesth   ; a = ty+bombdesth
sta bombdesth   ; bombdest = 320*y
; tmp=8*tx
lda #0      ; a=0
ldx drawbombtx  ; x = tx
stx tmp     ; tmp = tx
asl tmp     ; tmp = 2*tx, c=msb
rol     ; c->a,lsb
asl tmp     ; tmp = 4*tx, c=msb
rol
asl tmp     ; tmp = 8*tx, c=msb
rol     ; a:tmp = 8*tx
; bomdest = $6000+320*ty+8*tx
adc bombdesth   ; a = bombdesth + "tmph"
sta bombdesth   ; charyh = a
lda tmp     ; a = tmp
clc     ; c = 0
adc bombdestl   ; a = bombdestl + tmp
sta bombdestl   ; bombdestl = a
sta bombdest2l  ; bombdest2l = a
lda #$60    ; a = $60 (screenmem msb)
adc bombdesth   ; a = $60+bombdesth+c
sta bombdesth   ; bombdest  = $6000 + 320*ty + 8*tx
sta bombdest2h  ; bombdest2 = $6000 + 320*ty + 8*tx
```

```
; for y=ty..ey
drawbombyloop:
ldx #0      ; reset dest x offset
lda drawbombtx  ; a = bomb x offset
sta drawbombx   ; reset bomb x offset

; for x=tx..ex
drawbombxloop:
; make char to print:
;  char = $00
;  if x=0 , set $80 all
;  if x=39, set $01 all
;  if y=1 , or $ff first
;  if y=24, or $ff last
drawbombx=*+1
ldy #$00    ; y = bomb x coord
lda #$00    ; bomb char line = 0
cpy #0      ; check if bomb is at left edge
bne +       ; jump if not
ora #$80    ; don't erase left edge
+ cpy #39   ; check if bomb is at right edge
bne +       ; jump if not
ora #$01    ; don't erase right edge
+ sta tmpchar+0 ; create char
sta tmpchar+1
sta tmpchar+2
sta tmpchar+3
sta tmpchar+4
sta tmpchar+5
sta tmpchar+6
sta tmpchar+7
drawbombty=*+1
ldy #$00    ; y = bomb y coord
cpy #1      ; check if bomb is at top edge
bne +       ; jump if not
lda #$ff    ; dont erase top edge
sta tmpchar+0
+ cpy #24   ; check if bomb is at bottom edge
bne +       ; jump if not
lda #$ff    ; dont erase bottom edge
sta tmpchar+7
+ ldy #0    ; reset char line counter

; printchar - and version
-   lda tmpchar,y    ; a = char line y
bombdestl=*+1
bombdesth=*+2
    and $0000,x    ; a &= screen mem
bombdest2l=*+1
bombdest2h=*+2
    sta $0000,x    ; screen mem = a
```

```
    inx        ; x++
    iny        ; y++
    cpy #8        ; check if y = 8
    bne -        ; jump if not
; next x
inc drawbombx       ; bomb x ++
lda drawbombx       ; a = bomb x
drawbombex=*+1
cmp #$00        ; check if bomb x = end bomb x
bne drawbombxloop   ; if not, next x
; next y
inc drawbombty      ; bomb y ++
lda drawbombty      ; a = bomb y
drawbombey=*+1
cmp #$00        ; check if bomb y = end bomb y
beq drawbomb_f      ; if it is, jump to end
; dest+=320
clc
lda #64
adc bombdestl
sta bombdestl
sta bombdest2l
lda #1
adc bombdesth
sta bombdesth
sta bombdest2h  ; ...destination += 320
jmp drawbombyloop
drawbomb_f:
rts     ; return
drawbombtx !by 0    ; start bomb x variable

; - calckiller
; parameters:
;  x = x-coordinate, c = msb
;  y = y-coordinate
; returns:
;  ktemp = killer data
;  kloc(l:h) = killer location
;  kloc2(l:h) = killer location
;
calckiller:
; x&7 -> shifts
txa     ; a = x (c = msb)
and #7      ; a = x&7
sta kxoff   ; kxoff = x&7
; x&$f8 -> x-coordinate
txa     ; a = x (c = msb)
ror     ; c->a,msb; a = x/2
and #$fc    ; a = (x/2)&$fc
tax     ; x = (x/2)&$fc
; y&7 -> line offset
```

```
tya     ; a = y
and #7      ; a = y&7
sta kyoff   ; kyoff = y&7
; (y/8)*320 -> y-coordinate
tya     ; a = y
lsr
lsr
lsr     ; a = y/8
tay     ; y = y/8
sty kloch   ; kloch = "256*y"
lda #0      ; a=0
lsr kloch   ; kloch = "128*y", c=lsb
ror         ; c->a,msb
lsr kloch   ; kloch = "64*y", c=lsb
ror         ; kloch:a = 64*y/8
sta klocl   ; kloc = 64*y/8
tya     ; a = y/8
adc kloch   ; a = y/8+kloch
sta kloch   ; kloc = 320*y/8
; kloc = $6000+320*(y/8)+8*(x/8)
txa     ; a = (x/2)&$fc
asl     ; a = (x/8)*8, c = msb
tax     ; x = (x/8)*8
lda #0      ; a = 0
adc kloch   ; a = kloch + c
sta kloch   ; kloc = 320*y/8 + msb((x/8)*8), c = 0
txa     ; a = lsb((x/8)*8)
adc klocl   ; a += klocl
sta klocl   ; klocl = 320*y/8 + lsb((x/8)*8), c = msb
sta kloc2l  ; kloc2l = 320*y/8 + lsb((x/8)*8), c = msb
lda #$60    ; a = $60 (screenmem msb)
adc kloch   ; a = $60+kloch+c
sta kloch   ; kloc = $6000+320*(y/8)+8*(x/8)
sta kloc2h  ; kloc2 = $6000+320*(y/8)+8*(x/8)
; clear ktemp
ldx #48     ; 48 loops
lda #0      ; clear
-   dex
    sta ktemp,x
    bne -
; copy from kdata to ktemp with y offset
kyoff=*+1   ; y offset
ldy #0      ; y = destination
ldx #0      ; x = source
-   lda kdata,x    ; a = row 1, line x
    sta ktemp,y    ; ktemp line y = a
    lda kdata+16,x    ; a = row 2, line x
    sta ktemp+8,y    ; ktemp line y = a
    inx        ; x++
    iny        ; y++
    cpy #8    ; if y = 8 then y = 24
```

```
    bne +
    ldy #24    ; (see ktemp comments)
+   cpx #9     ; loop until source line = 9
    bne -
; shift ktemp by x offset
lda kxoff   ; load x offset
beq ++      ; if x offset = 0, skip shifting
ldx kyoff   ; line counter = y offset
ldy #9      ; 9 loops
--  tya    ; a = loop counter
kxoff = *+1
    ldy #0    ; y = x offset
-           clc        ; c = 0
            ror ktemp,x    ; 0->ktemp(left) >>1->c
            ror ktemp+8,x    ; c->ktemp(mid)  >>1->c
            ror ktemp+16,x  ; c->ktemp(right)>>1->c
            dey        ; loop for offset times
            bne -
    inx        ; x++
    cpx #8     ; if x = 8 then x = 24
    bne +
    ldx #24    ; (see ktemp comments)
+   tay    ; y = loop counter
    dey    ; y--
    bne --
++
rts     ; return

; - drawkiller
; parameters:
;  kloc = kloc2 = location to print killer
;  kdata = killer data to print
; notes:
;  if x > 308, this "leaks" into the next line
;  (or sprite pointers), but they are just OR'd with 0
;
drawkiller:
ldx #0          ; 48 loops
-   lda ktemp,x    ; a = killer temp data
    klocl=*+1
    kloch=*+2
    ora $0000,x    ; a |= screen data
    kloc2l=*+1
    kloc2h=*+2
    sta $0000,x    ; screen data = a
    inx        ; x++
    cpx #24        ; if x=24 (4. char)...
    bne +
    clc
    lda #40
    adc klocl
```

```
    sta klocl
    sta kloc2l
    lda #1
    adc kloch
    sta kloch
    sta kloc2h    ; ...destination += 320-24
+   cpx #48        ; loop while x < 48
    bne -
rts         ; return

; - erasekiller
; parameters:
;  kloc = kloc2 = location to print killer
;  kdata = killer data to print
; notes:
;  if x > 308, this "leaks" into the next line
;  (or sprite pointers), but they are just AND'd with 1
;
erasekiller:
; copy locations from drawkiller
lda klocl
sta flocl
lda kloch
sta floch
lda kloc2l
sta floc2l
lda kloc2h
sta floc2h
ldx #0          ; 48 loops
-   lda ktemp,x    ; a = killer temp data
    eor #$ff    ; invert a
    flocl=*+1
    floch=*+2
    and $0000,x    ; a &= screen data
    floc2l=*+1
    floc2h=*+2
    sta $0000,x    ; screen data = a
    inx        ; x++
    cpx #24        ; if x=24 (4. char)...
    bne +
    clc
    lda #40
    adc flocl
    sta flocl
    sta floc2l
    lda #1
    adc floch
    sta floch
    sta floc2h    ; ...destination += 320-24
+   cpx #48        ; loop while x < 48
    bne -
```

```
rts         ; return

; - printscore
; uses: printstrnocalc
;
printscore:
lda #>$6000+8*7
sta charyh
lda #<$6000+8*7
sta charyl
lda #>scoretext
sta strptrh
lda #<scoretext
sta strptrl
jmp printstrnocalc

; - printbombs
; uses: printstrnocalc
;
printbombs:
lda #>$6000+8*38
sta charyh
lda #<$6000+8*38
sta charyl
lda #>bombstext
sta strptrh
lda #<bombstext
sta strptrl
jmp printstrnocalc

; - printstr
; parameters:
;  x = x-coordinate
;  y = y-coordinate
;  strptr(l:h) -> string to print
; assumptions:
;  x = 0..39, y = 0..24
; uses:
;  printchar (chary(l:h),charxl)
;
; - printstrnocalc
; parameters:
;  chary -> where to print (320*y+8*x)
;  strptr(l:h) -> string to print
; assumptions:
;  x = 0..39, y = 0..24
; uses:
;  printchar (chary(l:h),charxl)
;
printstr:
;chary=320*y=256*y+64*y
```

```
sty charyh  ; charyh = "256*y"
lda #0      ; a=0
lsr charyh  ; charyh = "128*y", c=lsb
ror         ; c->a,msb
lsr charyh  ; charyh = "64*y", c=lsb
ror         ; charyh:a = 64*y
sta charyl  ; chary = 64*y
tya     ; a = y
adc charyh  ; a = y+charyh
sta charyh  ; chary = 320*y
;charx=8*x
lda #0      ; a=0
stx charxl  ; charxl = x
asl charxl  ; charxl = 2*x, c=msb
rol     ; c->a,lsb
asl charxl  ; charxl = 4*x, c=msb
rol
asl charxl  ; charxl = 8*x, c=msb
rol     ; a:charxl = 8*x
; chary=$6000+320*y+8*x
adc charyh  ; a = charyh + "charxh"
sta charyh  ; charyh = a
lda charxl  ; a = charxl
clc     ; c = 0
adc charyl  ; a = charyl + charxl
sta charyl  ; charyl = a
lda #$60    ; a = $60 (screenmem msb)
adc charyh  ; a = $60+charyh+c
sta charyh  ; chary = $6000 + 320*y + 8*x
printstrnocalc: ; chary is set
ldy #0      ; y = 0
-   ; char loop
    strptrl=*+1
    strptrh=*+2
    lda $0000,y    ; a = char at strptr+y
    beq +        ; if char = 0, leave
    jsr printchar    ; print a to chary
    iny        ; y++, -> next char
    lda #8
clc
adc charyl  ; a = charyl+8, c=msb
sta charyl
lda #0
adc charyh  ; a = 0 +charyh+c
sta charyh  ; chary->next x-pos.
jmp -
+
rts     ; return

; - printchar
; parameters:
```

```
;  chary = where to print (320*y+8*x)
;  a = char (-> charxl=$5300+char*8)
;
printchar:
sta charxl  ; charxl = char
lda #0
asl charxl  ; charxl = 2*x, c=msb
rol     ; c->a,lsb
asl charxl  ; charxl = 4*x, c=msb
rol
asl charxl  ; charxl = 8*x, c=msb
rol     ; a:charxl = 8*x, tmp = 0
adc #$53    ; a:charxl = $5300+char*8
sta charxh
ldx #7      ; 8 lines
-       ; line loop
    charxl=*+1
    charxh=*+2
    lda $0000,x    ; load from char-rom
    charyl=*+1
    charyh=*+2
    sta $0000,x    ; store to screen
    dex        ; next line
bpl -
rts     ; return


; --- Sprites

; ...reused as variables

Sprites
    ; eyes (white)
    ;          765432107654321076543210
    +SpriteLine %..........................
    +SpriteLine %..........................
    +SpriteLine %..........................
    +SpriteLine %..........................
    +SpriteLine %..#.#.#.#.................
    +SpriteLine %..........................
    +SpriteLine %..........................
    +SpriteLine %..........................
    +SpriteLine %..........................
    +SpriteLine %..........................
    +SpriteLine %..........................
    +SpriteLine %..........................
    +SpriteLine %..........................
    +SpriteLine %..........................
    +SpriteLine %..........................
    +SpriteLine %..........................
```

```
    +SpriteLine %........................ ;18
    +SpriteLine %........................ ;19
    +SpriteLine %........................ ;20
    +SpriteLine %........................ ;21
    !byte 0
    ; edges, mouth, eyes (brown)
    ;          765432107654321076543210
    +SpriteLine %...#####............... ;1
    +SpriteLine %..#.....#.............. ;2
    +SpriteLine %##.#...#.##............ ;3
    +SpriteLine %#...#.#...#............ ;4
    +SpriteLine %#..#...#..#............ ;5
    +SpriteLine %#.........#............ ;6
    +SpriteLine %##.#####.##............ ;7
    +SpriteLine %..#.....#.............. ;8
    +SpriteLine %...#####............... ;9
    +SpriteLine %........................ ;10
    +SpriteLine %........................ ;11
    +SpriteLine %........................ ;12
    +SpriteLine %........................ ;13
    +SpriteLine %........................ ;14
    +SpriteLine %........................ ;15
    +SpriteLine %........................ ;16
    +SpriteLine %........................ ;17
    +SpriteLine %........................ ;18
    +SpriteLine %........................ ;19
    +SpriteLine %........................ ;20
    +SpriteLine %........................ ;21
    !byte 0
    ; ball (food, warning)
    ;            765432107654321076543210
    +SpriteLine %...#####............... ;1
    +SpriteLine %..#######.............. ;2
    +SpriteLine %###########............ ;3
    +SpriteLine %###########............ ;4
    +SpriteLine %###########............ ;5
    +SpriteLine %###########............ ;6
    +SpriteLine %###########............ ;7
    +SpriteLine %..#######.............. ;8
    +SpriteLine %...#####............... ;9
    +SpriteLine %........................ ;10
    +SpriteLine %........................ ;11
    +SpriteLine %........................ ;12
    +SpriteLine %........................ ;13
    +SpriteLine %........................ ;14
    +SpriteLine %........................ ;15
    +SpriteLine %........................ ;16
    +SpriteLine %........................ ;17
    +SpriteLine %........................ ;18
    +SpriteLine %........................ ;19
    +SpriteLine %........................ ;20
```

```
    +SpriteLine %..................... ;21
    !byte 0
; pallo's body = ball xor eyes xor edges


; --- Sintable

; 256*sin(2*pi*i/256), i = [0,63]

sintable
!byte $00,$06,$0c,$12,$19,$1f,$25,$2b,$31,$38,$3e,$44,$4a,$50,$56,$5c
!byte $61,$67,$6d,$73,$78,$7e,$83,$88,$8e,$93,$98,$9d,$a2,$a7,$ab,$b0
!byte $b5,$b9,$bd,$c1,$c5,$c9,$cd,$d1,$d4,$d8,$db,$de,$e1,$e4,$e7,$ea
!byte $ec,$ee,$f1,$f3,$f4,$f6,$f8,$f9,$fb,$fc,$fd,$fe,$fe,$ff,$ff,$ff


; --- Strings

!ct scr     ; C64 screencode
;    |---------0---------0---------0--------|
statustext
!tx "score: 000        pallo        bombs: 00",0
scoretext
!tx "000",0   ; x offset 7
bombstext
!tx "00",0    ; x offset 38
gameovertext
!tx                 "game over.",0
gothighscoretext
!tx                 "highscore!",0
startscrtext1
!tx "= pallo",0
startscrtext2
!tx "= food",0
startscrtext3
!tx "= warning",0
startscrtext4
!tx "= killer",0
startscrtext5
!tx "use joystick in port 2",0
startscrtext6
!tx "< and > .... rotate pallo",0
startscrtext7
!tx "fire ....... explode bomb",0
startscrtext8
!tx "collect food, avoid collisions.",0
startscrtext9
!tx "a bomb is given every 15 points.",0
startscrtext10
!tx "high score: 000",0
startscrtext11
```

```
!tx "by hannu nuotio for c64cgc 2006",0


; --- Bitmaps

pallologo   ; 144x88, 1bpp logo
!bin "logo.b"


; --- Killer data

; A killer occupies 2x2 chars.
; Initial data organisation: (read by drawkiller)
; l     r      left    right
;              7654321076543210
; 0     16     ...#####........
; 1     17     ..#######.......
; 2     18     ###########.....
; 3     19     ###########.....
; 4     20     ###########.....
; 5     21     ###########.....
; 6     22     ###########.....
; 7     23  -> ..#######....... top
; 8     24     ...#####........ bottom
; 9     25     ................
; 10    26     ................
; 11    27     ................
; 12    28     ................
; 13    29     ................
; 14    30     ................
; 15    31     ................

kdata ;76543210
!by   %...##### ; left
!by   %..###### ; 1
!by   %######## ; 2
!by   %######## ; 3
!by   %######## ; 4
!by   %######## ; 5
!by   %######## ; 6
!by   %..###### ; 7
!by   %...##### ; 8
!by   %........ ; 9
!by   %........ ; 10
!by   %........ ; 11
!by   %........ ; 12
!by   %........ ; 13
!by   %........ ; 14
!by   %........ ; 15
!by   %........ ; right
!by   %#.......
```

```
!by   %###.....
!by   %###.....
!by   %###.....
!by   %###.....
!by   %###.....
!by   %#.......
!by   %........
!by   %........
!by   %........
!by   %........
!by   %........
!by   %........
!by   %........

; A shifted killer occupies at most 3x2 chars.
; Final data organisation: (by drawkiller)
; l  m  r      left    middle  right
;              765432107654321076543210
; 0  8  16     ...#####................
; 1  9  17     ..#######...............
; 2  10 18     ###########.............
; 3  11 19     ###########.............
; 4  12 20     ###########.............
; 5  13 21     ###########.............
; 6  14 22     ###########.............
; 7  15 23  -> ..#######............... top
; 24 32 40     ...#####................ bottom
; 25 33 41     ........................
; 26 34 42     ........................
; 27 35 43     ........................
; 28 36 44     ........................
; 29 37 45     ........................
; 30 38 46     ........................
; 31 39 47     ........................
;
; shift down by  (y mod 8) steps (apply offset)
; shift right by (x mod 8) steps (ror loop)
; draw with drawkiller (or-version of printchar)

ktemp = *   ; shifted version (48 bytes)
```