spock croissant challenge

(F99's submission for the tenliners

2016 contest, cat pur-80)

the captain needs croissants. now.

it's early in the galaxy, but the captain is

already awake. now it's mr. spocks job to get

croissants for the captain.

but all federal bakeries are still closed.

what to do? the enterprise is on the parking lot,

left of the screen. to the right, there is the

dominion and after it, the home planet of mr. spock.

so he must cross the dominion to his aunt's little home bakery

where he always can get a fresh bread. mr. spock boards a

little vessel as the enterprise must stay with the captain.

no time for jealousy this morning. the captain is in deep

fever for fresh bread. so spock must hurry up!

this is no easy mission as the dominion smells the

weakness of a hungry captain and attacks the delivery

chain with poisoned neutrinos. giant neutrinos in this case.

and as time passes, the situation sharpens, as the dominion

overpowers their neutrino guns more and more.

mr. spock is in big trouble. how much croissants can

he get to feed the captain?

ah, and because it's a small vessel, every time only one

croissant fits in. this is important, when you compare

croissant-score. and no cheating, please. the score

counter did not fit in anymore.

spock has 3 lives only, even as a vulcan - since this

a classic video game - obviously, with reduced color and

animation and no sound. sorry.

for convenience, lives get counted down from C to @,

where @ means zero in C64 screen code.

after that the game reboots and you can retry

to feed the captain properly.

to compare scores globally, make a screen recording

and share it on youtube =)

(to fight capitalism, this description is designed lower case)

- technical notes -

hi, F99 of rendserv.com here.

the program can be started on win7 by dragging the

provided file croissant.d64 onto a running VICE 2.4

win32 x86 mingw emulator. kudos to the VICE guys,

it's insane effort.

it introduces the first time we know of the

romulan coding style as indirectly suggested by the

competition itself, stating no self-written asm would

be allowed. however, the original rom code is and hence

we proudly present our first rom-relocator in 3 basic lines,

together with sample gaming code:

0a$(1)="94fc025fff0463f602cbb00274e7034ca001e9fc02f1e70268a6020aa60143b11251e504

1a$(2)="59b10529b60233f30260e503e6a40183fa02a1ad020fa401e8a30112a50177ea0238ad02

2a$(3)="82a802f1bb0267e7018abb0201ea047cea0281a10119f6027aa10193bd02e4ee02c9be02

3a$(4)="86e70206a5024aa50159a402e4a60217e60148a50181a10119f60248a501dcf40207a702

4a$(5)="07e60335a60263f6027cbb0270a6012bbd02dba501abed03a9a00166e702fdbf02b5a701

5x=5:pO255,241:r=49152:q=r:fOz=1tox:fOp=1tolen(a$(z)):fOb=1to3:l=0:fOh=0to1

6b$=mid$(a$(z),p,1):p=p+1:c=asc(b$):d=c+(c>57)\*7-48:l=l\*16+d:nE:v(b)=l:?".";:nE

7fOf=0tov(3)-1:pOr,pE(v(1)+v(2)\*256+f):r=r+1:nE:p=p-1:nE:nE:pO53281,0:pO646,5

8pO51,8:?chr$(147):?" life:d - spock croissant challenge -":fOf=1to40:?"-";:nE

9fOf=1to18:?:nE:fOf=1to40:?"-";:nE:pO47,120:pO48,4:pO49,32:pO50,7:sYq:rU

we'd like to add that we decided not to use graphical characters as that would

in our opinion not look so stylish by far. please note, that by this decision,

memory space for code is roughly halved. but this only makes the challenge more

challenging. also, we denied using broken basic lines like 0"94fc02... as that

won't be basic anymore, only a container for data.

the core idea is this 5-loop construction:

(please ignore comments after "'")

for z = 1 to x ' number of strings in array

for p = 1 to len(a$(z)) ' position in string

for b = 1 to 3 ' three byte make a vector lo-hi-len

l = 0

for h = 0 to 1 ' two hex chars make one byte

b$ = mid$(a$(z),p,1) ' read char

p = p + 1 ' advance ptr

c = asc(b$)

d = c + (c>57)\*7 - 48 ' decode hex->int

l = l\*16 + d ' compile one byte

next ' h

v(b) = l

next ' b

for f = 0 to v(3) -1

poke r, peek(v(1)+v(2)\*256+f) ' write vector

r = r + 1

next ' f

p = p - 1 ' corrective backstep, no need for it?

next ' p

next ' z

it reads the obvious strings and relocates the

specified rom sequences into the address 49152ff.

then later, after some cool pokes and prints, it jumps

into there. after the gameloop is done by the death

of the player, the program restarts by "run" - by the way

proving, basic was not harmed.

you can see there was no hidden own asm code by looking

at the line poke r, peek(v(1)+v(2)\*256+f), which solely

reads from the rom addresses, provided in hex code.

for example, the first vector is 2 byte, starting from

$fc94 and the second one is 4 byte, starting from

$ff5f.

for your convenience, here are all the vectors listed,

along with their rom source:

v01 : $FC94 2 : $78 $AD

v02 : $FF5F 4 : $12 $D0 $D0 $FB

v03 : $F663 2 : $8D $20

v04 : $B0CB 2 : $D0 $A9

v05 : $E774 3 : $20 $91 $D1

v06 : $A04C 1 : $C6

v07 : $FCE9 2 : $FD $D0

v08 : $E7F1 2 : $37 $A4

v09 : $A668 2 : $33 $84

v10 : $A60A 1 : $FD

v11 : $B143 18 : $A5 $2F $A4 $30 $85 $5F $84 $60 $A5 $31 $A4 $32 $85 $5A $84 $5B $18 $69

v12 : $E551 4 : $28 $90 $01 $C8

v13 : $B159 5 : $85 $58 $84 $59 $20

v14 : $B629 2 : $BF $A3

v15 : $F333 2 : $A2 $03

v16 : $E560 3 : $20 $FF $E9

v17 : $A4E6 1 : $E6

v18 : $FA83 2 : $FE $D0

v19 : $ADA1 2 : $02 $C6

v20 : $A40F 1 : $33

v21 : $A3E8 1 : $B1

v22 : $A512 1 : $FE

v23 : $EA77 2 : $29 $1F

v24 : $AD38 2 : $69 $04

v25 : $A882 2 : $A8 $A9

v26 : $BBF1 2 : $62 $91

v27 : $E767 1 : $D1

v28 : $BB8A 2 : $A2 $14

v29 : $EA01 4 : $20 $F0 $E9 $20

v30 : $EA7C 2 : $87 $EA

v31 : $A181 1 : $A4

v32 : $F619 2 : $FB $A5

v33 : $A17A 1 : $CB

v34 : $BD93 2 : $C9 $0A

v35 : $EEE4 2 : $D0 $01

v36 : $BEC9 2 : $88 $C9

v37 : $E786 2 : $12 $D0

v38 : $A506 2 : $01 $C8

v39 : $A54A 1 : $98

v40 : $A459 2 : $29 $7F

v41 : $A6E4 2 : $F0 $02

v42 : $E617 1 : $84

v43 : $A548 1 : $FB

v44 : $A181 1 : $A4

v45 : $F619 2 : $FB $A5

v46 : $A548 1 : $FB

v47 : $F4DC 2 : $4A $4A

v48 : $A707 2 : $A8 $B1

v49 : $E607 3 : $D1 $C9 $20

v50 : $A635 2 : $F0 $0A

v51 : $F663 2 : $8D $20

v52 : $BB7C 2 : $D0 $CE

v53 : $A670 1 : $2F

v54 : $BD2B 2 : $04 $D0

v55 : $A5DB 1 : $02

v56 : $EDAB 3 : $58 $60 $A9

v57 : $A0A9 1 : $41

v58 : $E766 2 : $91 $D1

v59 : $BFFD 2 : $4C $00

v60 : $A7B5 1 : $C0

for even more convenience, here the resulting 131 byte

memory content in hex code:

78ad12d0d0fb8d20d0a92091d1c6fdd037a43384fda52fa430855f8460a531a432855a845b1

869289001c88558845920bfa3a20320ffe9e6fed002c633b1fe291f6904a8a96291d1a21420

f0e92087eaa4fba5cbc90ad00188c912d001c898297ff00284fba4fba5fb4a4aa8b1d1c920f

00a8d20d0ce2f04d0025860a94191d14c00c0

but convenience does not stop here,

as you might be interested in the commented mnemonics:

disable interrupt

C000 78 SEI

sync to scanline

C001 AD 12 D0 LDA $D012

C004 D0 FB BNE $C001

frame color black

C006 8D 20 D0 STA $D020

clear player

C009 A9 20 LDA #$20

C00B 91 D1 STA ($D1),Y

delay

C00D C6 FD DEC $FD

C00F D0 37 BNE $C048

C011 A4 33 LDY $33

C013 84 FD STY $FD

scroll

C015 A5 2F LDA $2F

C017 A4 30 LDY $30

C019 85 5F STA $5F

C01B 84 60 STY $60

C01D A5 31 LDA $31

C01F A4 32 LDY $32

C021 85 5A STA $5A

C023 84 5B STY $5B

C025 18 CLC

C026 69 28 ADC #$28

C028 90 01 BCC $C02B

C02A C8 INY

C02B 85 58 STA $58

C02D 84 59 STY $59

C02F 20 BF A3 JSR $A3BF

clear screen line 3

C032 A2 03 LDX #$03

C034 20 FF E9 JSR $E9FF

draw neutrinos

C037 E6 FE INC $FE

C039 D0 02 BNE $C03D

C03B C6 33 DEC $33

C03D B1 FE LDA ($FE),Y

C03F 29 1F AND #$1F

C041 69 04 ADC #$04

C043 A8 TAY

C044 A9 62 LDA #$62

C046 91 D1 STA ($D1),Y

cursor to line 20

C048 A2 14 LDX #$14

C04A 20 F0 E9 JSR $E9F0

get key

C04D 20 87 EA JSR $EA87

move player

C050 A4 FB LDY $FB

C052 A5 CB LDA $CB

C054 C9 0A CMP #$0A

C056 D0 01 BNE $C059

C058 88 DEY

C059 C9 12 CMP #$12

C05B D0 01 BNE $C05E

C05D C8 INY

C05E 98 TYA

C05F 29 7F AND #$7F

C061 F0 02 BEQ $C065

C063 84 FB STY $FB

check space

C065 A4 FB LDY $FB

C067 A5 FB LDA $FB

C069 4A LSR A

C06A 4A LSR A

C06B A8 TAY

C06C B1 D1 LDA ($D1),Y

C06E C9 20 CMP #$20

C070 F0 0A BEQ $C07C

crash

C072 8D 20 D0 STA $D020

C075 CE 2F 04 DEC $042F

C078 D0 02 BNE $C07C

C07A 58 CLI

C07B 60 RTS

draw player

C07C A9 41 LDA #$41

C07E 91 D1 STA ($D1),Y

C080 4C 00 C0 JMP $C000

as you may have noticed, a little bug could be

cleared by sys-ing to $c00d instead of $c000,

but - details...

finally we'd like thank for the great opportunity

of the contest to inspire for this awesome technique.

it is even for us quite remarkable what can be done by

only 60 rom vectors or otherwise 131 bytes of

resulting asm.

regards, F99

contact: kl.hh at web.de