

**FRANCESCO CLEMENTONI**

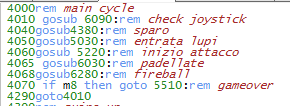
**POOY4K**: ovvero *Pooyan 4k*

Cronaca della stesura del programma in basic e commento

# Considerazioni preliminari

Il programma qui presentato rientra , per il contest omonimo , nella categoria 4k, con macchina target Commodore 64 e linguaggio Basic V2. Ide utilizzato: CBM Prg Studio.

In assenza di costrutti moderni, come chiamate a funzioni e multithread, e dati i limiti imposti dal contest (linguaggio e dimensioni) il gioco non rispecchia fedelmente l’originale, tuttavia si è cercato di mantenere quanto più gli elementi salienti e si è personalizzato qualcos’altro con un po’ di creatività.

Il codice è strutturato essenzialmente in un ciclo principale che esegue gli opportuni gosub alle varie funzionalità (animazioni, collision detection, ecc), e a un insieme di variabili che – per lo più – servono a gestire gli stati in cui il gioco si trova di volta in volta.

Ogni chiamata a una routine, infatti, non esaurisce il compito della stessa ma la porta avanti di uno step, in modo tale che al suo return può essere mandata avanti la successiva, e così via.

Il codice finale, poi, è stato ottimizzato usando le abbreviazioni fornite dal basic, per le quali il suddetto IDE fornisce lo strumento adatto per la conversione in massa dei comandi. I Rem sono stati poi eliminati tramite espressioni regolari su Notepad++ (*rem.\*$* per le righe che iniziano con rem, e *:rem.\*$* per i rem a fine riga) . Lo spazio guadagnato è stato così riallocato, diminuendo il numero di righe, e ottenendo infine un programma con dimensioni entro i limiti richiesti.

E’ infine stato possibile inserire una sorta di colonna sonora all’inizio usando pochissimo spazio, come verrà illustrato nel commento del codice.

# Il listato non ottimizzato

Di seguito, il listato del gioco non ottimizzato, verrà analizzato nei paragrafi successivi.

1gosub6230

10print*"{clear}"*:lvl=3:v=53248:pa$=*"{black}F-{light green}"*

20dimwf$(3):rem *vettori variabili interi1 e stringhe lupi*

40i2=80:rem *posizione iniziale y suino*

50i3=8:rem *delta y suino*

60yf=(i2/8)-5:rem *posizione iniziale freccia*

62m4=0:rem *paracadutisti scesi*

70wf$(1)=*"{brown} r{light green}"*:wf$(2)=*"{brown} U8K{light green}"*:wf$(3)=*"{brown} K8U{light green}"*

80m1=20:m2=190:m3=50:rem *m1=x char lupo,m2=x sprite lupo,m3=y sprite lupo*

100 rem *pokes per i colori in multicolor*

110poke53281,6:poke53282,9:poke53283,9:poke53270,peek(53270)or16

140rem *sprites*

150pokev+21,255:pokev+37,10:pokev+38,1:pokev+39,0:pokev+28,3

185poke2040,13:poke2041,14:poke2042,15:poke2043,11:rem *banks*

190sz=832:rem *Sprite block 13*

200forx=0to62:ready:pokesz+x,y:nextx

205sz=896:forx=0to62:ready:pokesz+x,y:nextx

208forx=0to62:poke960+x,0:next:rem *sprite freccia senza read*

209poke1000,255:pokev+41,0:pokev+4,0:pokev+5,0:rem *sprite freccia*

210pokev,255:pokev+1,i2:rem *posizione iniziale suino*

220pokev+40,2:pokev+2,m2:pokev+3,0:rem *colore e posizione iniziale lupo*

230forx=0to22:ready:poke728+x,y:next:pokev+6,0:pokev+7,0:pokev+42,13:rem *palla*

240 hs=10\*peek(727)

2100rem *SCREEN 1 -*

2120?spc(26)*" {reverse on}{light green}{169}{160}{127}{reverse off} "*;

2130?spc(26)*" {reverse on}{169}{reverse off}{169} {127}{reverse on}{127}{reverse off} "*;

2140?*"{reverse on}{169}{reverse off}"*+v$+*"VVV i {127}{reverse on}{127}{reverse off} "*;

2150?v$+*"{190} i hhhhhh"*;

2160?*"VVvVvvVVVVVVVVVVV{190}"*spc(15)*"hhhhhhh"*;

2170?*"V{reverse on}h{reverse off}vVvvvVV{190}{183}{183}{183}{183}{183}{183}"*spc(17)*"hhhhhhh"*;

2180?*" {reverse on}hh{reverse off}vvvvV{190}"*spc(24)*"{167}{164}{164}{165}hhh"*;

2190?*" {reverse on}hhh{reverse off}dv{reverse on}e{reverse off}{190}"*spc(25)*"{167}{164}{164}{165}hhh"*;

2200?*" vv{reverse on}h{reverse off}vh{reverse on}e{reverse off}"*spc(26)*"{167} {165}hhh"*;

2205fOt=1to3

2210?*" vd{reverse on}h{reverse off}vhv"*spc(26)*"hhhhhhh"*;

2220fOz=1to2:?*" vd{reverse on}h{reverse off}vhv"*spc(26)*"{167}{164}{164}{165}hhh"*;:nE

2240?*" Y{reverse on}hh{reverse off}vvv"*spc(26)*"{167} {165}hhh"*;:nE

2330?*" vv{reverse on}h{reverse off}v6v"*spc(26)*"hhhhhhh"*;

2340?*" vvvv6v"*spc(26)*"\*\*\*\*hhh"*;

2350?*" vvvvvvVVVVVVVVVV"*+v$+*"hhh{home}"*;

2355gosub 5400

2360goto4000

2500rem *sprite suino*

2510data*0,0,0,0,60,0*

2520data*0,195,0,3,0,192*

2530data*3,20,192,12,85,192*

2540data*12,117,192,13,85,192*

2550data*13,85,192,12,85,192*

2560data*12,85,192,8,20,192*

2570data*44,101,192,170,149,192*

2580data*44,85,192,8,85,192*

2590data*15,215,192,3,255,0*

2600data*0,252,0,0,0,0*

2610data*0,0,0*

2620 rem *sprite lupo*

2720 DATA *0,20,0,0,85,0*

2730 DATA *1,215,64,7,125,208*

2740 DATA *31,125,244,29,255,116*

2750 DATA *125,85,125,85,0,85*

2760 DATA *74,130,161,66,170,129*

2770 DATA *18,170,132,18,235,132*

2780 DATA *4,170,16,4,40,16*

2790 DATA *2,150,128,10,150,160*

2800 DATA *42,150,168,40,150,40*

2810 DATA *0,150,0,0,170,0*

2820 DATA *2,130,128*

2825 rem *palla*

2830 DATA *0,255,0,3,0,192,4,255,32,8,0,16,8,255,16,4,0,32,3,0,192,0,255,0*

4000rem *main cycle*

4010 gosub 6090:rem *check joystick*

4040gosub4380:rem *sparo*

4050gosub5030:rem *entrata lupi*

4060gosub 5220:rem *inizio attacco*

4065 gosub6030:rem *padellate*

4068gosub6280:rem *fireball*

4070 if m8 then goto 5510:rem *gameover*

4290goto4010

4300rem *suino up*

4310ifi1=126thengoto4316

4314i4=-1:ifi7=0theni5=29:rem *if here, fired*

4316ifi7=0thenyf=(i2/8)-5

4319ifi2=80thenreturn

4322at$=*" "*:yy=(i2/8)-5:xx=30:gosub6000

4325i2=i2-i3:pokev+1,i2:rem *muovi suino*

4330return

4350rem *suino down*

4354ifi1=125thengoto4359

4356i4=-1:ifi7=0theni5=29:rem *if here, fired*

4359ifi7=0thenyf=(i2/8)-5

4360ifi2=216thenreturn

4362at$=*"{light green}i"*:yy=(i2/8)-6:xx=30:gosub6000

4365i2=i2+i3:pokev+1,i2:rem *muovi suino*

4370return

4380rem *freccia*

4390ifi4=0andi7=0thenreturn

4395if noti7 then pokes+1,16:pokes,195:pokes+4,129:rem *beep*

4400i6=yf:i5=i5+15:i7=-1

4402 gosub 5270

4405ifi5>200 or m5theni5=260:i7=0:if m5thenpokev+2,peek(v+2)-8:poke53277,2

4412 pokev+4,260-i5:pokev+5,(i6+5)\*8

4415i4=0:if i7thenpokes+4,16:rem *beep off*

4420return

5030rem *entrata lupi*

5035ifi8<>-1thenreturn

5036b=notb

5037 fort=0to4

5038ifi9-(15-5\*t)<0thengoto5060

5040 ?*"{home}"*spc(i9-(15-5\*t))wf$(1)

5050?*"{home}{down}"*spc(i9-(15-5\*t))wf$(2-b)

5060nextt

5070 i9=i9+1:ifi9>14theni8=1:i9=0

5210return

5220 rem *inizio attacco*

5225 if i8<>1 thenreturn

5228 if m3>50 then goto5245

5230 at$=*" "*:yy=0:xx=m1:gosub6000

5240 at$=*" "*:yy=1:xx=m1:gosub6000

5245 pokev+2,m2+3\*sin(m3):pokev+3,m3

5246 poke53277,0

5247 gosub4380:rem *sparo*

5248 m3=m3+int(rnd(1)\*3)+lvl:gosub6090:rem *oscillazione e vel.variabili+joystick*

5249 if m3<210 then if notm5then goto 5254

5250 if m3>=210 then m7=m7+1:gosub5430

5251 m3=50:m1=m1-5:m2=m2-38:m4=m4+1:pokev+3,0:rem *wolf gone*

5254 m5=0:ifm4=4 then gosub5450

5260 return

5270 rem *collision detection*

5280 fy=20+(i6+5)\*8:fx=280-i5:m5=0

5285 if fx>m2theniffx<m2+24theniffy>m3theniffy<m3+28then m5=-1:m6=m6+10

5288 if not m5 then return

5290 pokes+1,16:pokes,195:pokes+4,33:rem *beep*

5291 gosub 5400:pokes+4,16:rem *update score and stop beeping*

5292 return

5400 rem *update score*

5410 at$=*"{white}hiscore:"*+stR(hs)+*" lvl:"*+stR(lvl-2)+*" score: "*+stR(m6)+*"{light green}"*

5415 yy=24:xx=6:gosub6000:reT

5420 rem *lasciato x sicurezza return*

5430 rem *update the right wolf*

5432 at$=wf$(1)+*"{light green}"*:xx=32:yy=23-4\*m7:gosub6000

5433 at$=wf$(2-b)+*"{light green}"*:xx=32:yy=24-4\*m7:gosub6000

5434 fort=32to16step-1:pokes+1,t:pokes,179+t:pokes+4,33:next:rem *beep*

5436 if m7=4 then m8=-1

5438 pokes+4,16

5440 return

5450 rem *lvl up*

5455 gosub5480:gosub5400

5470 return

5480 rem *azzeramento parziale*

5490 i8=0:i9=0

5500 lvl=lvl+1:m1=20:m2=190:m3=50:m5=0:m4=0:return

5510 rem *end*

5512 poke53271,1:fort=i2to225:pokev+1,t:nextt:poke53271,0

5515 pokes+1,16:pokes,195:pokes+4,33:rem *beep*

5520 ?*"{clear}"*:pokev+21,0:?*"{white}game over"*:?:?*"punteggio: "*+str$(m6)

5525 if(m6>hs)tH?*"{return}\*\*nuovo hiscore\*\*{return}"*:poke727,m6/10

5530 ?*"ancora? (s/n) "*:pokes+4,16

5540getz$:if(z$)=*""*thengoto5540

5550 ifz$=*"n"* then end

5560 run

6000 rem *printat - at$,xx,yy*

6010 poke780,0:poke781,yy:poke782,xx:sys65520:printat$;

6020 return

6030 rem *padelle variabili*

6040 if lvl<5 or m7=0thenreturn

6060 at$=*" "*:if int(rnd(1)\*5)=3thenat$=pa$

6070 xx=30:yy=24-4\*m7:gosub6000

6075 if at$=*" "* then return

6078 ifi2>=232-32\*m7thenm8=-1

6080 return

6090rem *check joystick*

6095i1=peek(56320)

6100ifi7=0thenyf=(i2/8)-5

6110ifi1=126ori1=110thengosub4300

6120ifi1=125ori1=109thengosub4350

6130ifi1=111theni4=-1:

6135ifi7=0theni5=29

6140ifi8=0theni8=int(rnd(1)\*2)-1

6150 return

6160 rem *music*

6180 i=val(mid$(mu$,no,1))

6190 pokes+1,hi(i):pokes,lo(i):pokes+4,17:fort=1to100:next: pokes+4,16:fort=1to50:next

6220 no=no+1:ifno>len(mu$)thenno=1

6222 return

6230 rem *home*

6231 s=54272:dim hi(4):dim lo(4):pokes,15:mu$=*"1323132314342414243"*:no=1

6232 hi(1)=34:hi(2)=43:hi(3)=51:hi(4)=61:lo(1)=75:lo(2)=52:lo(3)=97:lo(4)=126

6234 pokes+24,15:pokes+5,100:pokes+6,215:rem *volume,attack,sustain*

6235 fort=stos+24:pokes,0:next:rem *sid init*

6238 fort=1to20:v$=v$+*"V"*:next:rem *variabile mattoni ripetuti*

6240 ?*"{clear}"*:at$=*"{white}press a key"*

6250 xx=15:yy=12:gosub6000:bb=rnd(-ti):xx=1:yy=1

6255 at$=*" {red}p{green}o{white}o{yellow}y{purple}4{green}k "*:gosub6000:xx=xx+1:ifxx>100thengoto6240

6260 gets$:ifs$=*""*thengosub6180:goto6255

6270 return

6280 rem *fireball*

6290 ifm9=-1thengoto6400

6300 ifint(rnd(1)\*8)=3thenm9=-1:n1=m3:n2=m2:return

6400 n2=n2+10:n1=n1+0.5:ifn2>=255orn1>=255thenpokev+7,0:pokev+6,0:m9=0:return

6410 pokev+6,n2:pokev+7,n1:ifn1>i2-12thenifn1<i2+12thenifn2>=242thenm8=-1

6420 return

# Spiegazione e commento delle sezioni di codice

## La schermata iniziale: inizializzazioni e musica

La prima riga (1gosub6230) rimanda alla 6230 dove viene impostata la variabile s come base per le locazioni di memoria relative al SID, il quale viene qui anche resettato e impostato a valori di volume, attack e sustain standard. Vengono anche definiti un array di 4 locazioni per l’alta frequenza e uno di altrettante locazioni per la bassa , in vista del successivo step di generazione di note, per il quale servirà anche la variabile mu$="4342414243".

La variabile V$ servirà in seguito per ridurre lo spazio necessario a stampare sequenze di “V” che compongono – in multicolor – buona parte dello sfondo di gioco.

La scritta scrollante POY4K verrà visualizzata attraverso la chiamata alla routine che si occupa di fare il “print at”. Tale routine, che verrà utilizzata spesso, si baserà sempre sui valori di xx, yy e at$, rispettivamente posizione x, posizione y e variabile stringa da stampare. Non essendoci funzioni – come detto in prefazione – è comunque possibile simularle attraverso gosub e attraverso l’assegnazione di variabili dedicate solo a tale scopo, un gruppo di esse per ogni “funzione” .

### La musichetta nella schermata iniziale

La riga 6260, che attende la pressione di un tasto per iniziare il gioco, richiama ciclicamente la 6180 per l’esecuzione dell’ennesima nota del motivetto iniziale.

Le note utilizzate sono le 4 che compongono l’accordo di sol 7 (do, mi, sol, sib) e sono memorizzate, per l’alta e la bassa frequenza, nei rispettivi array hi() e lo(). La variabile mu$="1323132314342414243" viene così scandita ad ogni passaggio per recuperare l’indice degli array corrente, corrispondente alla nota da suonare, tramite i=val(mid$(mu$,no,1)).

## Altre operazioni preliminari prima dell’inizio del gioco

Dopo la schermata di presentazione il codice ritorna alla riga 10, alla quale seguono una serie di inizializzazioni di variabili e di poke, il cui significato è ben comprensibile dai REM che corredano le varie istruzioni. Uniche segnalazioni degne di nota:

1. il disegno dello sfondo di gioco si avvale della variabile V$ già introdotta, nonché di un uso capillare della funzione SPC(n), che ha consentito di risparmiare parecchi bytes.
2. Lo sprite della freccia non è generato con dei read giacchè è quasi tutto 0, è sufficiente che tutti i bit della locazione 1000 sono impostati a 1 tramite il valore 255.
3. Infine, la variabile lvl, che sta per “livello”, è inizializzata a 3 perché a valori più bassi fa partire il gioco troppo lentamente (si illustrerà in seguito in che modo condiziona la velocità e, quindi, la difficoltà), tuttavia per motivi estetici verrà mostrata al giocatore come lvl-2, in modo da partire da 1.

La riga di print che rende a video quest’ultima informazione, unitamente al punteggio, è contenuta nella sezione di codice che inizia a 5400 e che non fa altro che richiamare la già citata “print at” tramite gosub 6000, e viene eseguita una tantum prima del main cycle.

## Inizio del gioco – analisi del ciclo principale

Alla riga 4000 inizia il ciclo principale, composto di 6 gosub e di un check sulla variabile m(8) per rilevare se si è in stato di gameover.

4000rem *main cycle*

4010 gosub 6090:rem *check joystick*

4040gosub4380:rem *sparo*

4050gosub5030:rem *entrata lupi*

4060gosub 5220:rem *inizio attacco*

4065 gosub6030:rem *padellate*

4068gosub6280:rem *fireball*

4070 if m8 then goto 5510:rem *gameover*

4290goto4010

Analizziamo i singoli gosub del main cycle

### Check segnali del joystick (gosub 6090)

Il codice che viene eseguito in questa subroutine è il seguente:

6090rem *check joystick*

6095i1=peek(56320)

6100ifi7=0thenyf=(i2/8)-5

6110ifi1=126ori1=110thengosub4300

6120ifi1=125ori1=109thengosub4350

6130ifi1=111theni4=-1:

6135ifi7=0theni5=29

6140ifi8=0theni8=int(rnd(1)\*2)-1

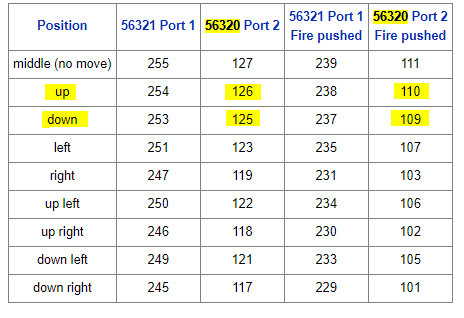
6150 return

Prima di commentarlo, occorre definire il senso delle variabili qui in gioco.

* La variabile i1 è utilizzata per leggere il valore della locazione 56320 (joystick porta 2).
* La variabile i2 memorizza la y dello sprite protagonista.
* La variabile i4 è pari a true (cioè -1) se si è premuto fire.
* La variabile i5 memorizza la x della freccia.
* La variabile i7 è pari a false (cioè 0) se la freccia non sta viaggiando.
* Infine, i8=-1 se è in atto l’animazione di entrata dei lupi, 0 se neutro, 1 se si entra in stato inizio attacco, 2 se è in uno stato idle

La riga 6100 aggiorna la posizione y della freccia solo se questa non sta viaggiando. Questo per evitare che, una volta sparata, continui a seguire i movimenti y dello sprite protagonista. Quando è sparata, infatti, la sua ordinata non deve più cambiare.

Le righe 6110 e 6120 richiamano le subroutine di spostamento dello sprite protagonista a seconda che si stia andando su o giù. Vengono rilevati i movimenti su e giù con e senza sparo, ecco perché entrambi le righe fanno il check sull’or di due valori.



La riga 6130 verifica se è stato premuto fire. In tal caso mette a true la variabile i4 (in modo che il programma sia informato di ciò anche in altri punti) , mantenendo così vivo l’evento anche al successivo variare del peek(56320)

La riga 6135 fa un check ridondante, nel senso che si ritroverà anche altrove, non è correlata allo scopo della routine, e verifica se la freccia – per qualche motivo- ha smesso di viaggiare, e in tal caso riporta la sua x al valore iniziale tramite i5=29.

La riga 6140 attraverso i8 verifica se ci si trova nello stato neutro, quello che precede l’uscita dei lupi. L’azione che segue è tale che se i8=0 (quindi stato neutro), o lo stato resta neutro oppure riparte l’animazione dei lupi ( i8=-1 ). Poiché ciclicamente si tornerà in questa istruzione, prima o poi i nemici inizieranno la loro sequenza di entrata. Questo solo per dare un minimo di randomizzazione alla loro uscita che, comunque, avverrà quasi subito. Se si vuole un tempo maggiore, basta aumentare il ventaglio dei valori random restituiti.

### Sparo della freccia (gosub 4380)

4380rem *freccia*

4390ifi4=0andi7=0thenreturn

4395if noti7 then pokes+1,16:pokes,195:pokes+4,129:rem *beep*

4400i6=yf:i5=i5+15:i7=-1

4402 gosub 5270

4405ifi5>200 or m5theni5=260:i7=0:if m5thenpokev+2,peek(v+2)-8:poke53277,2

4412 pokev+4,260-i5:pokev+5,(i6+5)\*8

4415i4=0:if i7thenpokes+4,16:rem *beep off*

4420return

La prima cosa che viene verificata è se non sia in corso un lancio precedente, alla riga 4390.

Lo scoccare della freccia inizia quindi con il play del suono (riga 4395), l’aggiornamento della posizione y della stessa a partire da quella del suino, l’incremento della posizione x e infine lo switch di i7 a true (riga 4400), perché a questo punto la freccia sta viaggiando.

E infatti segue immediatamente il collision detection con il gosub 5270. La riga successiva controlla se la corsa del proiettile è terminata per vie “naturali” o perché è stato colpito un lupo (m5=true). In entrambi casi la freccia viene fatta sparire. Se il lupo è stato colpito (m5=true) allora lo allarga come a simularne l’esplosione, prima che venga fatto sparire.

La riga 4412 esegue i poke necessari per visualizzarla nella posizione corrente, la 4415 infine decreta che fire non è più in stato di “premuto” e termina il play del suono qualora esso sia stato iniziato in questo ciclo.

Riguardo al collision detection il codice è facilmente comprensibile. Ci limitiamo a dire che non si è approcciato il problema tramite opportuni peek perché era più difficile da gestire rispetto a un controllo su un intorno, scelta qui effettuata.

#### Movimento su e giù del suino protagonista

Come scritto più su, le righe 6110 e 6120 richiamano le subroutine di spostamento dello sprite protagonista (4300 e 4350)a seconda che si stia andando su o giù.

4300rem *suino up*

4310ifi1=126thengoto4316

4314i4=-1:ifi7=0theni5=29:rem *if here, fired*

4316ifi7=0thenyf=(i2/8)-5

4319ifi2=80thenreturn

4322at$=*" "*:yy=(i2/8)-5:xx=30:gosub6000

4325i2=i2-i3:pokev+1,i2:rem *muovi suino*

4330return

4350rem *suino down*

4354ifi1=125thengoto4359

4356i4=-1:ifi7=0theni5=29:rem *if here, fired*

4359ifi7=0thenyf=(i2/8)-5

4360ifi2=216thenreturn

4362at$=*"{light green}i"*:yy=(i2/8)-6:xx=30:gosub6000

4365i2=i2+i3:pokev+1,i2:rem *muovi suino*

4370return

Entrambi le subroutine di direzione verificano per prima cosa se si tratti di direzione + sparo o solo direzione. Il gioco, infatti, consente di sparare mentre ci si muove. In caso di direzione più sparo, viene comunque impostato lo stato “inizio sparo“ (i4=-1), e successivamente viene verificato che la freccia non sia già precedentemente in viaggio (check su i7). Se la freccia deve partire, allora la sua x viene impostata a 29 e la sua y viene aggiornata in base alla posizione del suino. Infine, se il suino sta scendendo, viene disegnato un pezzo di corda sopra di lui tramite print at, se invece sta salendo viene cancellato l’ultimo pezzo allo stesso modo, per poi aggiornare la y dello sprite e disegnarlo.

### Animazione di entrata dei lupi (gosub 5030)

5030rem *entrata lupi*

5035ifi8<>-1thenreturn

5036b=notb

5037 fort=0to4

5038ifi9-(15-5\*t)<0thengoto5060

5040 ?*"{home}"*spc(i9-(15-5\*t))wf$(1)

5050?*"{home}{down}"*spc(i9-(15-5\*t))wf$(2-b)

5060nextt

5070 i9=i9+1:ifi9>14theni8=1:i9=0

5210return

Si tratta di una semplice animazione fatta coi caratteri in multicolor. Unica nota da segnalare: la parte inferiore dei lupi viene switchata ciclicamente tra due variabili stringa wf$(2) e wf$(3) tramite wf$(2-b) con b=notb ad ogni ciclo , in modo da far sembrare che stiano camminando.

### Attacco paracadutisti (gosub 5220)

5220 rem *inizio attacco*

5225 if i8<>1 thenreturn

5228 if m3>50 then goto5245

5230 at$=*" "*:yy=0:xx=m1:gosub6000

5240 at$=*" "*:yy=1:xx=m1:gosub6000

5245 pokev+2,m2+3\*sin(m3):pokev+3,m3

5246 poke53277,0

5247 gosub4380:rem *sparo*

5248 m3=m3+int(rnd(1)\*3)+lvl:gosub6090:rem *oscillazione e vel.variabili+joystick*

5249 if m3<210 then if notm5then goto 5254

5250 if m3>=210 then m7=m7+1:gosub5430

5251 m3=50:m1=m1-5:m2=m2-38:m4=m4+1:pokev+3,0:rem *wolf gone*

5254 m5=0:ifm4=4 then gosub5450

5260 return

Dopo aver verificato, tramite i8, se ci troviamo nel posto giusto, controlla se l’ascissa del lupo paracadutista è all’inizio, e in caso sovrascrive degli spazi per cancellare il lupo non paracadutista.

Dopodiché si tratta di far scendere lo sprite secondo lo stato attuale delle variabili m2 e m3 che ne descrivono le coordinate. Per dare un effetto di ondulazione si è introdotto un delta x tramite la funzione seno, dandole come argomento la y, in modo tale che all’aumentare della discesa il suddetto delta si muove tra -1 e 1. Viene poi moltiplicato per 3 per rendere più apprezzabile lo scostamento.

pokev+2,m2+3\*sin(m3):pokev+3,m3

All’interno della routine viene anche eseguito un ulteriore controllo dello stato-sparo rispetto a quello del ciclo principale (riga 5247 gosub 4380), onde evitare lag di rilevazione dello stesso, date le diverse righe che interessano la sezione attuale. Similmente la riga successiva ripete il polling del joystick, dopo aver incrementato la y del paracadutista (variabile m3) di un delta casuale e in funzione del livello di gioco:

m3=m3+int(rnd(1)\*3)+lvl

Successivamente, la riga 5249 verifica se il paracadutista non abbia ancora raggiunto terra e non sia stato colpito. Se è così, salta le successive e va all’ultima prima del return, che esegue un controllo su quanti paracadutisti sono partiti. Nel caso siano 4, allora il livello viene incrementato.

Se invece ci troviamo alla riga 5250, allora il lupo ha raggiunto il terreno e, tramite gosub 5430, viene disegnato a destra dello schermo tramite gli stessi caratteri che lo definiscono nell’entrata in scena all’inizio di ogni livello. La riga successiva reinizializza alcune variabili e cancella lo sprite del paracadutista, mentre – infine – la 5254 controlla se i lupi scesi sono quattro, e in tal caso fa avanzare di livello tramite gosub 5450, che esegue a sua volta due gosub, uno per l’azzeramento parziale delle variabili, e uno per la stampa a video del livello aggiornato. Ovviamente, questa ridondanza è mantenuta qui solo per facilità di lettura, ma nel codice minimizzato è eliminata, portando i due gosub direttamente nella chiamata del primo.

### “Padellate” dai lupi che sono giunti a destinazione

6030 rem *padelle variabili*

6040 if lvl<5 or m7=0thenreturn

6060 at$=*" "*:if int(rnd(1)\*5)=3thenat$=pa$

6070 xx=30:yy=24-4\*m7:gosub6000

6075 if at$=*" "* then return

6078 ifi2>=232-32\*m7thenm8=-1

6080 return

Il tempo è tiranno, per cui i lupi si portano avanti con il lavoro già con la padella in mano. Se riescono ad assestare un colpo sul suino sono già pronti a cucinarlo.

Tutto ciò avviene da lvl>=5 (ovvero livello 3 per il giocatore), e non avviene se non esiste alcun lupo a destra, come suggerisce la riga 6040.

La “padella” non è altro che la stringa pa$, che viene visualizzata tramite “print at” in momenti random (il consueto gosub 6000).La posizione y della padella è funzione del numero di lupi presenti a destra (m7). Il “collision detection” è molto semplice e controlla - a “padellata in corso” - semplicemente se la y del suino è tale che la padella lo colpisca o che colpisca la corda sopra di lui.

### Hadoken! (gosub 6280)

6280 rem *fireball*

6290 ifm9=-1thengoto6400

6300 ifint(rnd(1)\*8)=3thenm9=-1:n1=m3:n2=m2:return

6400 n2=n2+10:n1=n1+0.5:ifn2>=255orn1>=255thenpokev+7,0:pokev+6,0:m9=0:return

6410 pokev+6,n2:pokev+7,n1:ifn1>i2-12thenifn1<i2+12thenifn2>=242thenm8=-1

6420 return

I paracadutisti, stufi di aspettare il proprio turno, hanno acquisito da Street Fighter la sfera energetica.

Dopo aver controllato se è già in corso un lancio di bolla energetica (e nel caso, return), il codice decide a random se effettuarne uno oppure no (riga 6300), inizializzando in quest’ultimo caso le variabili deputate a tale azione.

Il resto è normale amministrazione: incremento variabili x e y (quest’ultima ha un leggero degrado mano mano che procede, tramite n1=n1+0.5) , pokes per visualizzare lo sprite e semplice collision detection basato su intorno della posizione del suino.

### Game over (goto 5510)

L’ultima riga del ciclo principale controlla lo stato della variabile m8, la quale –se è stata impostata a true da qualche altra sezione del codice, a seguito di eventi “mortiferi” – rimanda alla riga 5510, dove inizia la fine di tutto:

5510 rem *end*

5512 poke53271,1:fort=i2to225:pokev+1,t:nextt:poke53271,0

5515 pokes+1,16:pokes,195:pokes+4,33:rem *beep*

5520 ?*"{clear}"*:pokev+21,0:?*"{white}game over"*:?:?*"punteggio: "*+str$(m6)

5525 if(m6>hs)tH?*"{return}\*\*nuovo hiscore\*\*{return}"*:poke727,m6/10

5530 ?*"ancora? (s/n) "*:pokes+4,16

5540getz$:if(z$)=*""*thengoto5540

5550 ifz$=*"n"* then end

5560 run

La riga 5512 esegue un’animazione dove il suino protagonista cade dalla fune, e per effetto della gravità si “allunga” (poke53271,1, rimesso a 0 a fine animazione).

Viene poi eseguito un beep di riscontro sonoro, per poi pulire lo schermo (anche da sprites), e mostrare il punteggio (variabile m6).

Una nota importante riguarda la memorizzazione dell’ Highest Score. Per risparmiare spazio prezioso, si è deciso di far ripartire il gioco eseguendo Run (che cancella lo stack). Così facendo, però, si azzerano tutte le variabili, compresa quella ipoteticamente dedicata al punteggio più alto. Per rimediare a ciò, si è deciso di registrare un decimo del punteggio nella locazione 727 (un decimo, perché arrivare a 255 punti non è difficile, 2550 forse è impossibile).

Tale locazione è un punto esterno ai bordi della fireball. Era possibile sicuramente trovare una locazione libera e non impegnata, tuttavia l’idea di avere una sorta di side effect sulla palla energetica, dipendentemente dal punteggio più alto precedente – mi piaceva.

E se il giocatore totalizza più di 2550 punti? Il programma va in errore, e il suino si desta e scopre di essere parte di un sistema informatico dove tutto avviene nella sua immaginazione tramite una rete che connette il suo cervello a un mondo virtuale, chiamato Suimatrix.

# Appendice

## Le variabili più importanti e il loro ruolo

Poiché il basic V2 del commodore 64 recepisce solo i primi due caratteri di una variabile come “chiavi” , onde evitare di incorrere in errori di omonimie indesiderate (ad esempio **lu**poapiedi$ e **lu**poparacadutista$ sono la stessa cosa) si è mantenuta fin da subito una lista delle variabili utilizzate, avendo scelto di usare delle serie alfanumeriche composte da lettera+numero[1-9].

Queste sono le loro definizioni:

rem *i1=joystick*

rem *i2= y suino*

rem *i3=dy suino*

rem *i4=-1 se premuto fire,0 se no*

rem *i5=x freccia*

rem *i6=y freccia*

rem *i7=-1 se la freccia sta viaggiando,else 0*

rem *i8=-1 se in atto entrata in scena lupi,0 se neutro, 1 se entra in stato inizio attacco,2 stato idle*

rem *i9=margine sinistro entrata lupi*

rem *m1=x char lupo,m2=x sprite lupo,m3=y sprite lupo*

rem *m4 contatore paracadutisti scesi*

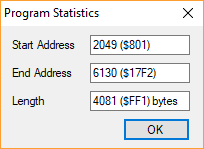
rem *m5 colpito lupo,m6 punteggio,m7 lupi arrivati a casa*

rem *m8=gameover se true*

rem *m9=-1 se iniziato lancio palla*

rem *n1=y palla, n2=x palla*

# Il codice minimizzato

Il codice minimizzato, ottenuto in parte tramite automatismi (funzione “compress” di CBM prg studio e eliminazione dei Rem tramite regex, come detto all’inizio) e in parte riorganizzando i “buchi” liberatisi per accorpare quanto più possibile pezzi di codice, è il seguente, e “pesa” 4081 bytes, come da immagine

10goS6230:?*"{clear}"*:lvl=3:v=53248:pa$=*"{black}F-{light green}"*:dIwf$(3):i2=80

20i3=8:yf=(i2/8)-5:m4=0:wf$(1)=*"{brown} r{light green}"*:wf$(2)=*"{brown} U8K{light green}"*:wf$(3)=*"{brown} K8U{light green}"*

80m1=20:m2=190:m3=50:pO53281,6:pO53282,9:pO53283,9:pO53270,pE(53270)or16

150pOv+21,255:pOv+37,10:pOv+38,1:pOv+39,0:pOv+28,3:pO2040,13:pO2041,14:pO2042,15

185pO2043,11:sz=832:fOx=0to62:rEy:pOsz+x,y:nEx:sz=896:fOx=0to62:rEy:pOsz+x,y:nEx

208fOx=0to62:pO960+x,0:nE:pO1000,255:pOv+41,0:pOv+4,0:pOv+5,0:pOv,255:pOv+1,i2

220pOv+40,2:pOv+2,m2:pOv+3,0:fOx=0to22:rEy:pO728+x,y:nE:pOv+6,0:pOv+7,0:pOv+42,13

2120?sP26)*" {reverse on}{light green}{169}{160}{127}{reverse off} "*;

2130?sP26)*" {reverse on}{169}{reverse off}{169} {127}{reverse on}{127}{reverse off} "*;

2140?*"{reverse on}{169}{reverse off}"*+v$+*"VVV i {127}{reverse on}{127}{reverse off} "*;

2150?v$+*"{190} i hhhhhh"*;

2160?*"VVvVvvVVVVVVVVVVV{190}"*sP15)*"hhhhhhh"*;

2170?*"V{reverse on}h{reverse off}vVvvvVV{190}{183}{183}{183}{183}{183}{183}"*sP17)*"hhhhhhh"*;

2180?*" {reverse on}hh{reverse off}vvvvV{190}"*sP24)*"{167}{164}{164}{165}hhh"*;

2190?*" {reverse on}hhh{reverse off}dv{reverse on}e{reverse off}{190}"*sP25)*"{167}{164}{164}{165}hhh"*;

2200?*" vv{reverse on}h{reverse off}vh{reverse on}e{reverse off}"*sP26)*"{167} {165}hhh"*;

2205fOt=1to3

2210?*" vd{reverse on}h{reverse off}vhv"*sP26)*"hhhhhhh"*;

2220fOz=1to2:?*" vd{reverse on}h{reverse off}vhv"*sP26)*"{167}{164}{164}{165}hhh"*;:nE

2240?*" Y{reverse on}hh{reverse off}vvv"*sP26)*"{167} {165}hhh"*;:nE

2330?*" vv{reverse on}h{reverse off}v6v"*sP26)*"hhhhhhh"*;

2340?*" vvvv6v"*sP26)*"\*\*\*\*hhh"*;

2350?*" vvvvvvVVVVVVVVVV"*+v$+*"hhh{home}"*;:hs=10\*pE(727):goS5400

2510dA0,0,0,0,60,0,0,195,0,3,0,192,3,20,192,12,85,192,12,117,192,13,85,192

2550dA13,85,192,12,85,192,12,85,192,8,20,192,44,101,192,170,149,192

2580dA44,85,192,8,85,192,15,215,192,3,255,0,0,252,0,0,0,0,0,0,0,0,20,0,0,85,0

2610dA1,215,64,7,125,208,31,125,244,29,255,116,125,85,125,85,0,85

2760dA74,130,161,66,170,129,18,170,132,18,235,132,4,170,16,4,40,16

2790dA2,150,128,10,150,160,42,150,168,40,150,40,0,150,0,0,170,0

2820dA2,130,128,0,255,0,3,0,192,4,0,32,8,0,16,8,0,16,4,0,32,3,0,192,0,255,0

4010goS6090:goS4380:goS5030:goS5220:goS6030:goS6280:ifm8tHgO5510

4290gO4010

4300ifi1=126tHgO4316

4314i4=-1:ifi7=0tHi5=29

4316ifi7=0tHyf=(i2/8)-5

4319ifi2=80tHreT

4322at$=*" "*:yy=(i2/8)-5:xx=30:goS6000:i2=i2-i3:pOv+1,i2:reT

4350ifi1=125tHgO4359

4356i4=-1:ifi7=0tHi5=29

4359ifi7=0tHyf=(i2/8)-5

4360ifi2=216tHreT

4362at$=*"{light green}i"*:yy=(i2/8)-6:xx=30:goS6000:i2=i2+i3:pOv+1,i2:reT

4380ifi4=0aNi7=0tHreT

4395ifnOi7tHpOs+1,16:pOs,195:pOs+4,129

4400i6=yf:i5=i5+15:i7=-1:goS5270

4405ifi5>200orm5tHi5=260:i7=0:ifm5tHpOv+2,pE(v+2)-8:pO53277,2

4412pOv+4,260-i5:pOv+5,(i6+5)\*8:i4=0:ifi7tHpOs+4,16

4420reT

5030ifi8<>-1tHreT

5036b=nOb:fOt=0to4:ifi9-(15-5\*t)<0tHgO5060

5040?*"{home}"*sPi9-(15-5\*t))wf$(1):?*"{home}{down}"*sPi9-(15-5\*t))wf$(2-b)

5060nEt

5070i9=i9+1:ifi9>14tHi8=1:i9=0

5210reT

5220ifi8<>1tHreT

5228ifm3>50tHgO5245

5230at$=*" "*:yy=0:xx=m1:goS6000:yy=1:goS6000

5245pOv+2,m2+3\*sI(m3):pOv+3,m3:pOv+29,0:goS4380

5248m3=m3+int(rN(1)\*3)+lvl:goS6090

5249ifm3<210tHifnOm5tHgO5254

5250ifm3>=210tHm7=m7+1:goS5430

5251m3=50:m1=m1-5:m2=m2-38:m4=m4+1:pOv+3,0

5254m5=0:ifm4=4tHgoS5480:goS5400

5260reT

5270fy=20+(i6+5)\*8:fx=280-i5:m5=0

5285iffx>m2tHiffx<m2+24tHiffy>m3tHiffy<m3+28tHm5=-1:m6=m6+10

5288ifnOm5tHreT

5290pOs+1,16:pOs,195:pOs+4,33:goS5400:pOs+4,16:reT

5400at$=*"{white}hiscore:"*+stR(hs)+*" lvl:"*+stR(lvl-2)+*" score: "*+stR(m6)+*"{light green}"*

5415yy=24:xx=6:goS6000:reT

5430at$=wf$(1)+*"{light green}"*:xx=32:yy=23-4\*m7:goS6000

5433at$=wf$(2-b)+*"{light green}"*:yy=24-4\*m7:goS6000

5434fOt=32to16stE-1:pOs+1,t:pOs,179+t:pOs+4,33:nE:ifm7=4tHm8=-1

5438pOs+4,16:reT

5480i8=0:i9=0:lvl=lvl+1:m1=20:m2=190:m3=50:m5=0:m4=0:reT

5510pO53271,1:fOt=i2to225:pOv+1,t:nEt:pO53271,0:pOs+1,16:pOs,195:pOs+4,33

5520?*"{clear}"*:pOv+21,0:?*"{white}game over"*:?:?*"punteggio: "*+stR(m6)

5525if(m6>hs)tH?*"{return}\*\*new hiscore\*\*{return}"*:pO727,m6/10

5530?*"ancora? (s/n) "*:pOs+4,16

5540gEz$:if(z$)=*""*tHgO5540

5550ifz$=*"n"*tHeN

5560rU

6000pO780,0:pO781,yy:pO782,xx:sY65520:?at$;:reT

6030iflvl<5orm7=0tHreT

6060at$=*" "*:ifint(rN(1)\*5)=3tHat$=pa$

6070xx=30:yy=24-4\*m7:goS6000:ifat$=*" "*tHreT

6078ifi2>=232-32\*m7tHm8=-1

6080reT

6090i1=pE(56320)

6100ifi7=0tHyf=(i2/8)-5

6110ifi1=126ori1=110tHgoS4300

6120ifi1=125ori1=109tHgoS4350

6130ifi1=111tHi4=-1:

6135ifi7=0tHi5=29

6140ifi8=0tHi8=int(rN(1)\*2)-1

6150reT

6180i=vA(mI(mu$,no,1))

6190pOs+1,hi(i):pOs,lo(i):pOs+4,17:fOt=1to100:nE:pOs+4,16:fOt=1to50:nE

6220no=no+1:ifno>len(mu$)tHno=1

6222reT

6230s=54272:dIhi(4):dIlo(4):pOs,15:mu$=*"1323132314342414243"*:no=1

6232hi(1)=34:hi(2)=43:hi(3)=51:hi(4)=61:lo(1)=75:lo(2)=52:lo(3)=97:lo(4)=126

6234pOs+24,15:pOs+5,100:pOs+6,215:fOt=stos+24:pOs,0:nE:fOt=1to20:v$=v$+*"V"*:nE

6240?*"{clear}"*:at$=*"{white}press a key"*

6250xx=15:yy=12:goS6000:bb=rN(-ti):xx=1:yy=1

6255at$=*" {red}p{green}o{white}o{yellow}y{purple}4{green}k "*:goS6000:xx=xx+1:ifxx>100tHgO6240

6260gEs$:ifs$=*""*tHgoS6180:gO6255

6270reT

6280ifm9=-1tHgO6400

6300ifint(rN(1)\*8)=3tHm9=-1:n1=m3:n2=m2:reT

6400n2=n2+10:n1=n1+0.5:ifn2>=255orn1>=255tHpOv+7,0:pOv+6,0:m9=0:reT

6410pOv+6,n2:pOv+7,n1:ifn1>i2-12tHifn1<i2+12tHifn2>242tHm8=-1

6420reT

Basic programmers often gosub but ever return

Sommario

[1. Considerazioni preliminari 2](#_Toc6176062)

[2. Il listato non ottimizzato 3](#_Toc6176063)

[3. Spiegazione e commento delle sezioni di codice 5](#_Toc6176064)

[3.1. La schermata iniziale: inizializzazioni e musica 5](#_Toc6176065)

[3.1.1. La musichetta nella schermata iniziale 6](#_Toc6176066)

[3.2. Altre operazioni preliminari prima dell’inizio del gioco 6](#_Toc6176067)

[3.3. Inizio del gioco – analisi del ciclo principale 6](#_Toc6176068)

[3.3.1. Check segnali del joystick (gosub 6090) 6](#_Toc6176069)

[3.3.2. Sparo della freccia (gosub 4380) 8](#_Toc6176070)

[3.3.3. Animazione di entrata dei lupi (gosub 5030) 9](#_Toc6176071)

[3.3.4. Attacco paracadutisti (gosub 5220) 9](#_Toc6176072)

[3.3.5. “Padellate” dai lupi che sono giunti a destinazione 10](#_Toc6176073)

[3.3.6. Hadoken! (gosub 6280) 10](#_Toc6176074)

[3.3.7. Game over (goto 5510) 11](#_Toc6176075)

[4. Appendice 11](#_Toc6176076)

[4.1. Le variabili più importanti e il loro ruolo 11](#_Toc6176077)

[5. Il codice minimizzato 12](#_Toc6176078)