```
1 ; Shamus+ Patch Utility
2 ; Siegfried Lenz, 2017
3 ;
4 ; Version 1.0
5 ;
6 ; for MAD-Assembler 1.9.x
7 ;
8 ; Shamus (C) Synapse Software, 1982 // Copyright assumed to be with
  Cathryn Mataga, 1982-2017
9 ;
10 ; Special thanks to the authors of all the utilities used in the
  creation of this patch,
11 ; especially phaeron for Altirra, David Firth for Atari800MacX,
  Peter Dell for WUDSN,
12 ; Rob McMullen for Omnivore, the VICE team members for x64, Clinton
  Parker & JAC! for Action!,
13 ; samar productions for C64Debugger.
14 ;
15 ; throughout this program map segments are referred to as follows
16 ; the address refers to the storage address of the Atari original
  map in the
17 ; original program
18 ; map segment 1: $23F2 (horizontal bars)
19 ; map segment 1: $2472 (vertical bars)
20 ; map segment 1: $1D43 (objects)
21 ; map segment 1: $1DC3 (colours / keyhole room door location)
22 ; map segment 5: $1EE3 (pod rooms)
23 ; map segment 6: $24F8 (level boundaries)
24 ;
25 ; System equates:
26 ;
27 CONSOL      equ $D01F
28 SETVBV      equ $E45C
29 XITVBV      equ $E462
30 NMIEN       equ $D40E
31 DLISTL      equ $D402
32 DLISTH      equ $D403
33 KBCODE      equ $D209
34 SKSTAT      equ $D20F
35 SKCTL       equ SKSTAT
36 SKRES       equ $D20A
37 AUDF1       equ $D200
38 HITCLR      equ $D01E
39 TRIG0       equ $D010
40 VVBLKI      equ $0208
```

```
41 VCOUNT       equ $D40B
42
43              org $3480
44              ins "Shamus.com game code.dat"
45 ;insert original Atari game code
46 ;
47 ;original code needs to be run from $7000
48 ;
49 ; this code is executed after the end of the orignal game init
   code, replacing the last instruction
50 ; of the original game init code.
51 ;
52              org $701C
53
54 TITLEPATCH   LDA #$0C
55              STA $0552
56              STA $0566
57              LDA #$3F
58              STA $055C
59 ; add blue plus sign to right of Shamus on title screen
60
61 DLISTPATCH1 LDA #$01
62              STA $0C5F
63              STA $35D7
64 ; patch jump instructions to ends of menu and in-game display lists
65
66              LDA <MENUDLIST
67              STA $0C60
68              LDA >MENUDLIST
69              STA $0C61
70 ;patch end of menu display list to point to new list with extra
   line for selected level
71
72              LDA <GAMEDLIST
73              STA $35D8
74              LDA >GAMEDLIST
75              STA $35D9
76 ;patch end of in-game display list to point to new list with extra
   line for selected level
77
78              LDA #(SCRLTXTWRAP-SCRLTXTBEG)
79              STA $182E
80 ;set $182E to length of new title string
81 ;$182E is checked to see if scrolling needs to reset
82
```

```
 83              LDA <SCRLTXTBEG
 84              STA $1838
 85              LDA >SCRLTXTBEG
 86              STA $183D
 87 ;change address of moving title string to modded data
 88
 89              LDA #$20
 90              STA $260A
 91              LDA <MAPSELECT
 92              STA $260B
 93              LDA >MAPSELECT
 94              STA $260C
 95 ;patch start of CONSOL read routine with a JSR to a new subroutine
    checking OPTION key
 96 ; by replacing the LDA CONSOL instruction with a JSR.
 97 ;
 98              LDY #$06
 99 LOOP0        LDA COLORSHIFTCODE,Y
100              STA $4979,Y
101              DEY
102              BNE LOOP0
103 ;this replaces the original color shift routine with a JSR to the
    new colorshift routine
104 ;
105              LDA #$4C
106              STA $18AF
107              LDA <MOVEBONUS
108              STA $18B0
109              LDA >MOVEBONUS
110              STA $18B1
111              LDA #$60
112              STA $18F3
113 ;this redirects the object shuffle routine to modified code
    executing it only for the Atari maze
114 ;
115              LDA #$20
116              STA $2B85
117              LDA <ADVANCETNMT
118              STA $2B86
119              LDA >ADVANCETNMT
120              STA $2B87
121 ;patch routine used when finishing maze to execute code to check
    for tournament level
122
123              LDA #$EA
```

```
124             LDX #12
125 LOOP0A      STA $1F0B,X
126             DEX
127             BNE LOOP0A
128
129 ; patch allowing the "speed increase" routine to decrease the speed
    control variable at $0206
130 ; below zero, thus allowing correct speed adjustments when
    returning to lower levels
131
132             LDA #$30
133             STA $2FA4
134             LDA #$02
135             STA $2FA5
136 ; patch main game delay loop to ignore negative speed numbers, thus
    avoiding game slowdown.
137
138 ;             LDA #$EA
139 ;            STA $2514
140 ; change CLI instruction to NOP to allow keyboard interrupts for
    pause function
141
142
143             LDA <PATCHVBI
144             STA $25D8
145             LDA >PATCHVBI
146             STA $25D9
147             LDA #$02
148             STA SKCTL
149 ; as VBI is used on the game title screen, the VBI routine for the
    pause (and possibly map) function
150 ; needs to be inserted at a point that executes after the title
    screen this is done by redirecting
151 ; a subroutine call
152
153 PATCHVVBLKI LDX #$47
154             LDY #$00
155             STY $E0
156             LDA #$14
157             STA $E1
158 NEXT3       CPY #$FF
159             BEQ NEXT2
160             LDA ($E0),Y
161             CMP #$22
162             BNE NEXT1
```

```
163             INY
164             LDA ($E0),Y
165             CMP #$02
166             BNE NEXT3
167             LDA >VVBLKI0
168             STA ($E0),Y
169             DEY
170             LDA <VVBLKI0
171             STA ($E0),Y
172             INY
173             BNE NEXT2       ;cannot be 0 as no matches are on an
    $xxFF address
174
175
176 NEXT1       CMP #$23
177             BNE NEXT2
178             INY
179             LDA ($E0),Y
180             CMP #$02
181             BNE NEXT3
182             LDA >VVBLKI1
183             STA ($E0),Y
184             DEY
185             LDA <VVBLKI1
186             STA ($E0),Y
187
188 NEXT2       INY
189             BNE NEXT3
190             INC $E1
191             DEX
192             BNE NEXT3
193
194 ; this changes all instructions referring to $222 and $223 which
    are needed for the
195 ; VBI to a "safe" location at VVBLKI0/1
196 ; As no other IRQs are used, the IRQ vector instead of the keyboard
    vector is used
197 ;
198
199 ;now copy original map data
200 ;
201 ORIGMAP     LDA MAPTABLELO
202             STA $E0
203             LDA MAPTABLEHI
204             STA $E1
```

```
205             LDY #$00
206 LOOP1       LDA $23F2,Y
207             STA ($E0),Y
208             INY
209             BNE LOOP1
210 ;copy FF bytes - segment 1 and 2 of map data - from $23F2 to map
    data table
211             INC $E1
212 ;advance map data table by FF bytes for next segment
213 LOOP2       LDA $1D43,Y
214             STA ($E0),Y
215             INY
216             CPY #$80
217             BNE LOOP2
218 ;copy 80 bytes - segment 3 of map data - from $1D43 to map data
    table
219 LOOP2A      LDA $1D43,Y
220             BEQ SKIPLUMA
221             ORA #$60
222 SKIPLUMA    STA ($E0),Y
223             INY
224             BNE LOOP2A
225 ;copy 80 bytes - segment 4 of map data - from $1DC3 to map data
    table
226 ;if a colour is stored add generic Atari luminance value of 6 to
    high nibble to allow
227 ;use of modified original segment 4 data with modified color
    routine
228
229             INC $E1
230 ;advance map data table "pointer" by FF bytes for next segment
231             LDA #$08
232             STA ($E0),Y
233             INY
234 ;store length of pod room table to first byte
235 LOOP3       LDA $1EE2,Y
236             STA ($E0),Y
237             INY
238             CPY #$09
239             BNE LOOP3
240 ;copy list of 8 pod rooms - segment 5 of map - to map data table
241             LDY #$7D
242 LOOP4       LDA $24F8-$7D,Y
243             STA ($E0),Y
244             INY
```

```
245             CPY #$80
246             BNE LOOP4
247 ;copy list of level boundaries - segment 6 of map - to map data
    table
248 ;this list is always three bytes long
249             LDY #$00
250             JSR MAPCOPY
251 ;now copy modified map data back into game to have it ready for use
    when original
252 ;atari map is played without using SELECT
253
254 ORIGINIT    JMP $0486
255 ; continue to original init routine
256
257 PATCHVBI    LDX >CHECKKEYS
258             LDY <CHECKKEYS
259             LDA #$06
260             JSR SETVBV
261             LDA #$C0
262             STA NMIEN
263             JMP $30E9
264 ; turn on VBI and continue to the "hijacked" routine
265
266 MAPSELECT   LDA CONSOL
267             AND #$04
268             BNE MAPSELDONE
269 ;if bit 2 is cleared, OPTION has been pressed
270             LDY CURRENTMAP
271             INY
272 ;so increase current level
273             CPY MAXMAP
274 ;check if it it is already above the maximum number of maps
275             BNE INCMAP
276 ;if not, change the map
277 RESETMAP    LDA <MAPNAMES
278             STA MAPNAME
279             LDA >MAPNAMES
280             STA MAPNAME+1
281             LDY #$00
282             BEQ MAPNAMEDONE
283 ;if it is, reset the level to zero and store base address of map
    names to display list
284 ;
285 INCMAP      CLC
286             LDA MAPNAME
```

```
287 ;read address of map name data from display list
288             ADC #20
289 ;add 20 (DEC) as all map names are 20 bytes long
290             STA MAPNAME
291 ;store it back to display list
292             BCC MAPNAMEDONE
293             INC MAPNAME+1
294 ;if carry is set, increase MSB in display list as well
295 MAPNAMEDONE STY CURRENTMAP
296             LDA #$00
297             CPY TOURNAMENT
298             BNE SKIPTOURNMT
299             TAY
300             LDA #$02
301 SKIPTOURNMT STA ISTOURNMT
302 ; set actual level to zero and set tournament flag byte to 2 (next
    map in tournament)
303 ; tournament skips Original C64 maze as it is almost identical to
    Original Atari maze
304 CALLMAPCOPY JSR MAPCOPY
305             JSR WAIT
306             JSR WAIT
307 MAPSELDONE  LDA CONSOL
308 ;this is the LDA CONSOL instruction we replaced when patching the
    CONSOL read routine in
309 ;the original program.
310             RTS
311 ;return to original CONSOL read routine
312
313 CHECKKEYS   LDA SKSTAT
314             AND #$04
315             BEQ WHICHKEY
316 ; check if a key is still pressed
317 EXIT        JMP XITVBV
318 ; if not, exit VBLANK
319 WHICHKEY    LDA KBCODE
320             CMP #$21
321             BNE EXIT
322 ; check if it is SPACE, if not exit to VBLANK
323 PAUSE       LDA #$00
324             STA NMIEN
325 ; disable all non-maskable interrupts (i.e. further VBIs and DLIs)
326
327 WAITVBI     LDA VCOUNT
328             BPL WAITVBI
```

```
329 ; wait for vertical scan to reach bottom of screen
330 
331             LDA <PAUSEDLIST
332             STA DLISTL
333             LDA >PAUSEDLIST
334             STA DLISTH
335 ; change display to pause display
336 
337             LDA #$00
338             LDY #$03
339 KILLAUDIO   STA AUDF1,Y
340             DEY
341             BPL KILLAUDIO
342 ; and silence all four audio channels
343 
344 PAUSING     LDA TRIG0
345             BNE PAUSING
346 ; now wait for trigger press
347 
348 ENDPAUSE    LDA #$B8
349             STA DLISTL
350             LDA #$35
351             STA DLISTH
352 ; change back to game display list
353 
354             LDA #$C0
355             STA NMIEN
356 ; enable interrupts
357 
358             STA HITCLR
359 ; clear collision detection (to avoid spurious collisions of player
    with Pause text that may
360 ; occur because player remains on-screeen during pause (as I have
    no way of determining it's position
361 ; and therefore could only make it disappear by copying all player
    data to a safe location and back).
362 
363             STA SKRES
364 ; reset keyboard scan
365             LDX #$05
366             JSR WAIT1
367             BEQ EXIT
368 ; wait a bit before game resumes
369 
370 ; MAPCOPY copies the map for the selected game (Y register) into
```

```
   game
371 MAPCOPY     TYA
372             PHA
373             LDA MAPTABLELO,Y
374             STA $E0
375             LDA MAPTABLEHI,Y
376             STA $E1
377             LDY #$00
378 LOOP5       LDA ($E0),Y
379             STA $23F2,Y
380             INY
381             BNE LOOP5
382 ;copy FF bytes - segment 1 and 2 of map data - to $23F2
383             INC $E1
384 ;advance map data table "pointer" by FF bytes for next segment
385 LOOP6       LDA ($E0),Y
386             STA $1D43,Y
387             INY
388             BNE LOOP6
389 ;copy FF bytes - segment 3 and 4 of map data - to $1D43
390             INC $E1
391 ;advance map data table "pointer" by FF bytes for next segment
392             LDY #$00
393             LDA ($E0),Y
394             STA $1FEF,Y
395 ;store length of pod room table into code that loops through pod
   room table
396             LDY #$7D
397 LOOP7       LDA ($E0),Y
398             STA $24F8-$7D,Y
399             INY
400             CPY #$80
401             BNE LOOP7
402 ;copy three bytes of segment 5 (level boundaries) to $24F8
403             LDA $E1
404             CLC
405             INC $E0
406             ADC #$00
407             STA $1FF7
408             LDA $E0
409             STA $1FF6
410 ;store address of maptable pod room data + 1 to code that loops
   through pod room table
411 ;
412             PLA
```

```
413             ASL
414             ASL
415             ASL
416             ASL
417             ORA #$0E
418             TAY
419 ;Y now contains number of maze x 16 which is equal to start of game
    screen map name
420             LDX #$0F
421 LOOP9       LDA MAPNAME0,Y
422             STA MAPLEGEND+4,X
423             DEY
424             DEX
425             BNE LOOP9
426 ;copy 16 characters from table of game screen map display names to
    screen memory for
427 ;last line of screen
428             RTS
429
430 WAIT        LDX #$FF
431 WAIT1       LDY #$FF
432 WAIT2       DEY
433             BNE WAIT2
434             DEX
435             BNE WAIT1
436             RTS
437 ;this routine counts to FFFF
438 ;it can be used for shorter waits by loading X with a lower value
    and jumping to WAIT1
439 ;
440 SHIFTCOLOR  CLC
441             ASL
442             ADC #$00
443             ASL
444             ADC #$00
445             ASL
446             ADC #$00
447             ASL
448             ADC #$00
449             RTS
450 ; the original Shamus routine uses just the lower nibble of map
    segment 3 data
451
452 MOVEBONUS   LDA CURRENTMAP
453             BEQ SHIFTBONUS
```

```
454             LDA ISTOURNMT
455             CMP #$02
456             BEQ SHIFTBONUS
457 EXITSHIFT   JMP $18FE
458 ; The Atari game shifts the contents of some chambers around whenever a new game is selected.
459 ; This does not work for some C64 mazes as it shifts objects into corridor rooms.
460 ; This code executes the object shift routine when playing the original Atari maze and bypasses
461 ; it when playing a C64 maze.
462
463 SHIFTBONUS  LDY #$04
464 LOOP8       DEY
465             CPY #$FF
466             BEQ EXITSHIFT
467             LDA $D20A       ;RANDOM
468             AND #$01
469             BEQ LOOP8
470             JSR $18BD
471             JMP LOOP8
472 ; this is a functional copy of the code originally at $18AF using the original map shift code
473 ; as a subroutine.
474
475 ADVANCETNMT LDY ISTOURNMT
476             BEQ EXITADVANCE
477 ;if tournament flag is zero, use speed value already in ACC and return to game
478 ;
479             INC $0202
480 ;add one life
481             LDA #$05
482 ;this is the default speed for the next maze, one "step" faster than normal game start,
483 ; same as second time round for normal Atari game
484             CPY TOURNAMENT
485 ;check if next level actually exists. If not, player has made it through all mazes.
486             BNE ADVANCEMAP
487 ; if finished with last maze, extra effect?
488             LDX #$01
489             STX ISTOURNMT
490 ;set tournament flag to 1 so it will be increased to two during normal ADVANCEMAP routine
```

```
491
492             TAY
493             DEY
494             DEY
495             TYA
496 ; if the player has made it through all mazes, increase the speed
    for the next attempt to $04
497             LDY #$00
498 ; and set the next level to be loaded to the first level
499 ADVANCEMAP  PHA
500 ; save the speed to be played next
501             INC ISTOURNMT
502 ;advance tournament flag to next level in tournament
503             JSR MAPCOPY
504 ; Y already contains next level to be played in tournament mode,
    (or zero if starting at first
505 ; maze agein. MAPCOPY puts maze data in place
506 ; if finished with last maze, extra effect?
507 ;
508             PLA
509 ; recall the  saved speed value for the next level
510
511 EXITADVANCE STA $0206
512             RTS
513 ;store either game calculated or new speed into $0206 (games speed
    variable) and return
514
515 VVBLKI0     .BY $00
516 VVBLKI1     .BY $00
517
518 MENUDLIST   .BY $46
519 MAPNAME     .BY <MAPNAMES
520             .BY >MAPNAMES
521             .BY $70
522             .BY $41 $3C $0C
523 ; adds extra line to show name of selected map to menu display
    list, then jumps back
524 ; to original display list
525
526 GAMEDLIST   .BY $46
527 MAPNAMEDISP .BY <MAPLEGEND
528             .BY >MAPLEGEND
529             .BY $41 $B8 $35
530 ; add extra mode 6 line to in-game display list to show actual map
531
```

```
532 PAUSEDLIST   .BY $70 $70 $70 $70 $70 $70 $70 $70 $70 $70 $70 $70 $70
    $70
533              .BY $47
534              .BY <PAUSETEXT
535              .BY >PAUSETEXT
536              .BY $70
537              .BY $06
538              .BY $41
539              .BY <PAUSEDLIST
540              .BY >PAUSEDLIST
541
542 PAUSETEXT    .SB "       PAUSED       "
543              .SB "  FIRE TO CONTINUE  "
544
545 SCRLTXTBEG   .SB "by william mataga    "
546              .SB "ORIGINALLY PUBLISHED BY SYNAPSE SOFTWARE  1982
    "
547              .SB "COPYRIGHT  ASSUMED  TO  BE  WITH  CATHRYN MATAGA
    2017    "
548              .SB +$80 "C64 LEVELS BY JACK L. THORNTON, JR.    "
549 SLX          .SB "ported to atari by slx"*
550              .SB "  "
551              .BY $D2 $D0 $D1 $D7
552              .SB "    "
553 SCRLTXTWRAP  .SB "by william mataga    "
554 SCRLTXTEND
555 CURRENTMAP   .BY $00
556 MAXMAP       .BY $07
557 TOURNAMENT   .BY $06
558 ;number of selection that results in tournament mode
559 ISTOURNMT    .BY $00
560 ;this is a flag that indicates if a tournament (playing one maze
    after another)
561 ;is in progress. $00 means no tournament mode, other value is next
    maze to be played
562 ;
563 LASTLEVEL    .BY $00
564 MAPTABLE
565 MAPTABLELO   .BY <(MAPDATA)
566              .BY <(MAPDATA+640)
567              .BY <(MAPDATA+1280)
568              .BY <(MAPDATA+1920)
569              .BY <(MAPDATA+2560)
570              .BY <(MAPDATA+3200)
571 MAPTABLEHI   .BY >(MAPDATA)
```

```
572             .BY >(MAPDATA+640)
573             .BY >(MAPDATA+1280)
574             .BY >(MAPDATA+1920)
575             .BY >(MAPDATA+2560)
576             .BY >(MAPDATA+3200)
577 MAPNAMTBLLO .BY <(MAP0)
578             .BY <(MAP1)
579             .BY <(MAP2)
580             .BY <(MAP3)
581             .BY <(MAP4)
582             .BY <(MAP5)
583             .BY <(MAP6)
584 MAPNAMTBLHI .BY >(MAP0)
585             .BY >(MAP1)
586             .BY >(MAP2)
587             .BY >(MAP3)
588             .BY >(MAP4)
589             .BY >(MAP5)
590             .BY >(MAP6)
591 MAPNAMES
592 MAP0        .SB " ORIGINAL ATARI MAP ";*
593 MAP1        .SB "  original C64 map  "*
594 MAP2        .SB "       holmes       "*
595 MAP3        .SB "       cluseau      "*
596 MAP4        .SB "       marlowe      "*
597 MAP5        .SB "        bond        "*
598 MAP6        .SB "     TOURNAMENT     "*
599 ;
600 ;
601 MAPLEGEND   .SB "MAP:                "
602 MAPNAME0    .SB "ORIGINAL ATARI  "
603 MAPNAME1    .SB "ORIGINAL C64    "
604 MAPNAME2    .SB "HOLMES          "
605 MAPNAME3    .SB "CLUSEAU         "
606 MAPNAME4    .SB "MARLOWE         "
607 MAPNAME5    .SB "BOND            "
608
609 COLORSHIFTCODE
610             .BY $20                     ; JSR
611             .WO SHIFTCOLOR
612             .BY $EA $EA $EA             ; 3x NOP
613
614 ;           org *+$100-<*
615 ;to ease debugging and analysis of map, start map data at next page
    boundary
```

```
616
617 MAPDATA     .ds 640
618 ; reserve 640 bytes to cover original Atari map for later re-use
619
620             ins "SHAM_A8.MAP.dat"
621 ;insert converted C64 map data
622
623             run $7000
624 ; this is the original game run address (I assume that the code at
    $7000 was added by whomever
625 ; cracked the original disc game and corrected the title music
    player routine for the Homesoft
626 ; version.)
627
628
629
```