

```

0 clr: rem Reset all variables to 0 (better than l=0:s=0:w=0:)
1 x=rnd(-1): rem use any negative number as initial seed value for pseudo randomizer/procedural generation
2 poke53280,0: rem Set frame color to black
3 poke53281,0: rem Set background color to black
4 h=1984: rem Startrow of Screen-RAM for water tank
5 p=1304: rem Startrow of Screen-RAM for facade
6 c=55576: rem Startrow nColor-RAM for facade
7 n=111: rem Constant for fire pressed value
8 d=102: rem Constant for PETSCII-value of the checkers symbol (=Fire)
9 m=599: rem Constant for max. number of characters to step through (15 rows of 40 characters, starting at 0)
10 g$="{home}{reverse off}{down}{down}{down}{light blue}u r ": rem Partial string for "You are F/Hired"-game over text

11 rem *** definition of some city names known for skyscrapers (replace with your own) ***
12 c$(0)="{red}hong kong"
13 c$(1)="{blue}new york"
14 c$(2)="{orange}dubai"
15 c$(3)="{purple}tokyo"
16 c$(4)="{gray}shanghai"
17 c$(5)="{cyan}chicago"
18 c$(6)="{pink}singapore"
19 c$(7)="{green}toronto"
20 c$(8)="{brown}miami"
21 c$(9)="{light green}las vegas"
22 c$(10)="{yellow}congratulations"

23 g=10: rem Number of fires to extinguish per level
24 r=15: rem Row counter for facade building (r=rows)
25 s=s-w*50: rem Add the remaining water units to the score when going to the next level
26 print "{light blue}{clear}rank"tab(9)"$stab(24)c$(1)"{down}{down}{down}{down}
{down}": rem Display status information at the top
27 for i=0 to 39: rem In this 40x loop there are 2 things happening... :)
28 s$=s$+"{black} ": rem Fill base string with black spaces
29 pokeh+i,247: rem Build lower water level status bar based on PETSCII-value 247
30 next

31 rem *** Creating the group of buildings (procedural generation) ***
32 rem The facade look & feel is being assembled in rows 200-430 (completing the base string x15).
33 rem For each line a house element will be added to the existing string at a position provided by the randomizer function.
34 rem The string in row 42 contains the actual graphic for one element.
35 rem Note: Maybe it makes you wonder why {light gray} is being repeated after each character.
36 rem The color/character-pairing is required so that the string always ends up exactly with 40 printable characters.
37 rem It would have been much simpler without the dark gray shadow, but it's a much nicer look & feel that way. :)

38 def fnr(x)=int(rnd(1)*x): rem Function definition of r(x) to return a random value for range x. Used a couple of times.
39 l$=left$(s$,fnr(33)*2): rem Get the leftmost n (1-32) characters from the last base string.
40 y=81: rem Constant for PETSCII-value circle (=searchlight)
41 v=10: rem General constant for value 10 (fires, length of house element, no of levels or color pink).
42 a$="{light gray}{cm m}{light gray}P{light gray}P{light gray}P{light gray}P{light gray}P{light gray}P{light gray}P"
43 b$=left$(a$,v+(fnr(5)*2)): rem Build a house element between 10-15 characters
44 l$=left$(l$+b$,78)+"{dark gray}
P": rem Add the house element to the left part of the string and shorten it if bigger than 39 characters (color/char pairs).
45 l$=l$+right$(s$,80-len(l$)): rem Fill the rest with rightmost part of the previous string
46 j=56320: rem Constant for Joystick Port 2 address
47 print l$: rem Print the 40-character-row
48 s$=l$: rem Replace base string with base string plus new element
49 r=r-1: rem Decrease row counter
50 on (r>0) goto 38: rem Repeat as long as there are still rows to be drawn.
51 w=39: rem Set water variable to maximum
52 l=l+1: rem Increase level variable by 1
53 s$="": rem Set base string back to empty. Only relevant for next level.
54 r$="{red}f": rem set "F"ired value as default - will be used in line 9
55 for i=0 to 9: rem Also in the following 10x loop there are 2 things happening :)
56 print "{reverse on}{green} "; rem 1. Print part of grass and 2. ...
57 if l>v then goto 63: rem do not place fires when end screen (10) is reached
58 o=fnr(m): rem Get a random position for a fire in the screen area
59 x(i)=o: rem Assign this position to a fire
60 on (peek(p+o)<>80) goto 58: rem No wall element at this position? Then try again with another value
61 pokep+o,d: rem Draw at this position the PETSCII checker symbol (=fire)
62 pokec+o,2: rem Write the color value light red to the same relative position but to the color-RAM
63 next: rem After this loop we have 10 relative screen positions stored in x()
64 if l>v then d=83: r$="{green}
h": goto 98: rem Reached highest level/rank? Then replace fire with heart symbol, set "H" for "Hired" and jump to end sequence

65 rem *** Here starts the main loop ***
66 for i=0 to m: rem Loop 600x (40 characters x 15 rows) and do the following...
67 q=p+i: rem p+i is being used more than once. Makes the final code smaller.
68 f=peek(q): rem Save current character at this screen position.
69 pokeq,y: rem Write the PETSCII-symbol for circle (searchlight)
70 on (peek(j)=n) goto 76: rem Joystick button pressed? Then water on!
71 pokec+x(i and 7),i and v or 2: rem Set the fire color red(2) or light red(10) at one of the 10 fires.
72 pokeq,f: rem Replace the searchlight with the saved character.
73 next: rem Repeat until the bottom right corner is reached.
74 goto 66: rem Then start again at the top left screen position.

75 rem *** Logic for when the Joystick button is being pressed ***
76 pokeh+w,32: rem Draw a space character at the current water level position.
77 w=w-1: rem decrease remaining water units by one.
78 for k=0 to 9: rem Check all 10 relative fire positions.
79 on (x(k)=i) goto 87: rem identical with current Position? Then extinguish fire!
80 next
81 pokeq,f: rem Replace the searchlight with the saved character.
82 pokec+i,14: rem Mark this position with light blue (=color of water).
83 on (w<0) goto 98: rem Water units all used up? Then go to "F"ired end sequence!
84 next: rem repeat until the bottom right corner is reached.
85 goto 66: rem Then start again at the top left screen position.

86 rem: *** Logic for when a fire is being extinguished ***
87 x(k)=-1: rem Set the relative screen position of the fire to -1 (make it "out of scope").
88 s=s+500: rem Increase account by $500.
89 pokeq,f: rem Replace the searchlight with the saved character.
90 pokec+i,14: rem Mark this position with light blue (=color of water).
91 print "{home}{reverse off}{light blue}stab(v)s: rem Update $ status on screen.
92 g=g-1: rem Extinguished another fire (minus 1).
93 on (g=0) goto 23: rem Extinguished all fires? Then go to another location (next level).
94 next: rem Do this until all 10 fire positions were checked.

```

```

95 next: rem repeat until the bottom right corner is reached.
96 goto66: rem Then start again at the top left screen position.

97 rem *** Logic for when water is 0 (=Game Over) or Rank 10 has been reached (=Success) ***
98 ifpeek(j)=11goto98: rem wait for button press
99 x=fnr(m): rem Get random value for relative screen position.
100 on-(peek(p+x)=32)goto99: rem If there is a space character at this position then try again.
101 pokep+x,d: rem Draw a checker (fire) or a heart symbol (success) to that Screen-RAM position.
102 pokec+x,7: rem At the same relative position set the color yellow in the color RAM.
103 print$sr$"ired!": rem Print "Hired" or "Fired!" text.
104 ifpeek(j)<>11goto99: rem While no key is being pressed spread fire (or hearts) across the building blocks.

105 rem *** Show hi-score table ***
106 gosub158: rem initialize hi-score list (get from file if available)
107 print"{clear}{red}": rem clear screen and set character color to red
108 printtab(12)"*** hi-score ***": rem print title in center of screen
109 print: rem print empty row
110 hp=0: rem hi score pointer = index where new hi score goes in
111 t=9: rem initial tab position for printing the hiscore row
112 n$="": rem initialize name variable
113 cs$="{cm @} {left}{left}": rem cursor (underscore) including blanking of the rightmost character
114 fori=1to10: rem loop through the 10 score elements/rows...
115 ifhp=0ands>=hs(i)thengosub127:hs(i)=s:hs$(i)="":hp=i: rem if current score is higher than listed score: shift and replace
116 hs$=str$(hs(i)): rem convert number to string
117 hs$=right$(hs$,len(hs$)-1): rem remove leading blank
118 ifi=10thent=t-1:rem for the last row start at tab position 9 instead of 10.
119 printtab(t); rem place cursor position at value of t(ab)
120 printi"{left}. "right$("000000"+hs$,5)" "hs$(i): rem print name and score with leading zeros
121 print: rem print empty row
122 next
123 ifhp>0thengosub133: rem if hi score pointer was set in the loop above then let player enter name
124 ifpeek(j)<>11thengoto124: rem wait for fire button on Joystick Port 2
125 goto0: rem restart game

126 rem *** Shift remaining score elements one element up ***
127 fork=10to1step-1: rem loop backwards from index 10 to the player's hi score position
128 hs(k)=hs(k-1): rem score at index 9 moves to 10, 8 moves to 9, etc.
129 hs$(k)=hs$(k-1): rem same for player name
130 next
131 return: rem return to next statement after gosub call

132 rem *** enter name ***
133 poke214,hp*2+1: rem set row where the high score pointer was found (Calculating row position from top based on hp index)
134 sys58732: rem place cursor at this row through kernal routine
135 printtab(19);cs$; rem print cursor string
136 geta$: rem get last keypress value
137 ifa$=""thengoto136: rem if no key was pressed then try again
138 a=asc(a$): rem get ascii value of that key
139 iflen(n$)=0thengoto142: rem skip the following checks if entered name string is blank
140 ifa=13thengoto147: rem if return key was pressed then continue with save
141 ifa=20thenn$=left$(n$,len(n$)-1):printa$;cs$;goto136: rem if delete key was pressed then delete last character and wait for next key press
142 if(a<35anda<>32)ora>90orlen(n$)>10thengoto136: rem if invalid key was pressed or name is over 10 characters long then wait for next key press
143 n$=n$a$: rem append current character to name string
144 printa$;cs$; rem print pressed key together with cursor string that also includes cursor positioning for next key
145 goto136: rem wait for next key press

146 rem *** save hi-score list ***
147 hs$(hp)=n$: rem assign entered name to hiscore table at hiscore pointer position
148 open1,8,15,"s:ff64hiscore":close1: rem use SCRATCH (s:) command to delete an existing hiscore file on disk
149 open1,8,1,"ff64hiscore,s": rem open disk channel to write (,1) to this filename sequentially (,s)
150 fori=1to10: rem write 10x...
151 print#1,hs(i): rem write high score value
152 print#1,hs$(i): rem write high score name
153 next
154 close1: rem close channel
155 s=0: rem reset score variable so enter name subroutine is not called again
156 goto107: rem show high score list again (this time without entering name)

157 rem *** initialize/load high score list ***
158 hs=10000: rem default top score is 10000
159 fori=1to10: rem for all 10 high score entries...
160 hs(i)=hs: rem assign a value
161 hs=hs-1000: rem next value will be 1000 units less
162 hs$(i)="firefighter": rem default name is firefighter
163 next
164 open1,8,0,"ff64hiscore,s": rem open disk channel to read (,0) from high score filename
165 fori=1to10: rem for all 10 high score entries...
166 input#1,hs(i): rem read high score value
167 input#1,hs$(i): rem read high score name
168 next
169 close1: rem close channel
170 return: rem return to next statement after gosub call
171 rem (c) 2017 roman werner / @romwwer / roman.werner@gmail.com

```