```
################################
# BASIC Tenliners Contest 2016 #
#                              #
# FireFighter64                #
# (c) 2016 Roman Werner @romwer #
# email: roman.werner@gmail.com #
################################
```

Das Abenteuer wartet! Erlebe als FireFighter, wie Wasser mit 17 bar Betriebsdruck auf Mauern peitscht und mit jedem Einsatz dein Heldenstatus wächst. Schärfe und nutze dabei deine Reflexe, um das drohende Flammeninferno zu verhindern.

Features

- Rasante Action und Geschicklichkeit
- Ausgewogene Spielbalance
- Hunderte von unterschiedlichen Locations
- Übersichtliche Statusanzeigen für Rang, Kontostand und Löschwasserreservoir
- Verblüffende Tiefenwirkung durch 3D-Effekt
- Mehrfarbige Feueranimation
- Zeitlupenmodus à la Matrix zur besseren Kontrolle des Wasserstrahls
- Epische Endsequenz ;)

Aufgabe und Ziel

Stelle dich der Herausforderung und durchlaufe eine erfolgreiche FireFighter-Karriere, vom einfachen Rookie (Anfänger) bis hin zu einem Chief Fire Officer - und verdiene gutes Geld dabei.

Es gilt pro Einsatzort 10 Brandherde zu löschen. Folge dem Suchscheinwerfer, der sich der Fassade entlang bewegt, und drücke im richtigen Moment den Feuerknopf, um den Wasserstrahl einzusetzen. Für jeden gelöschten Brand werden dir 500$ auf dein Konto gutgeschrieben. 50$ Bonus gibt es zusätzlich für jede verbleibende Wassereinheit, wenn alle Flammen gelöscht sind. Dein Rang erhöht sich bei erfolgreichem Abschluss eines Einsatzes automatisch.

Wichtig: Achte stets auf deinen Wasservorrat am unteren Bildrand, sonst kann es passieren, dass Du Deinen Job und Heldenstatus schnell wieder los bist.

Rank/Rang Laufbahn-Übersicht:

```
 0  - Rookie (Anfänger)
 1  - Firefighter Trainee
 2  - Firefighter Development
 3  - Firefighter
 4  - Crew Manager
 5  - Watch Manager
 6  - Station Manager
 7  - Group Manager
 8  - Area Manager
 9  - Deputy Chief Fire Officer
10  - Chief Fire Officer
```

Erreichst Du den höchsten Rang, erwartet Dich eine nette Überraschung!

Steuerung:

```
Wasserstrahl einsetzen:      Feuerknopf (Joystick Port 2)
Spiel nach Ende neu starten: Beliebige Taste drücken
```

Spielanforderungen:

- C64/C128* mit BASIC 2.0 kompatibler Laufzeitumgebung
- Joystick Port 2 (Feuerknopf)
- 1 Spieler

*Ebenfalls spielbar auf Windows/MacOS/iOS/Android (C64 Emulator vorausgesetzt)

Starten im VICE: C64 emulator (http://vice-emu.sourceforge.net/):
==========
Emulator Starten und Datei "firefighter64.prg" in das VICE Fenster ziehen
-oder-
Lade das Programm mit der Anweisung: load "firefighter64.prg",8
Anschliessend starte das Programm mit: run

FireFighter64 BASIC Programm-Code:
==========
```
0 clr:poke53280,0:poke53281,0:h=1984:p=1304:c=55576:n=111:d=102:m=599:g$="{home}{reverse
off}{down}{down}{down}{light blue}u r: "
1 g=10:r=15:s=s+w*50:print"{light
blue}{clear}rank"ltab(9)"$"s"{down}{down}{down}{down}{down}{down}":fori=0to39:s$=s$+"{black}
":pokeh+i,247:next
2 deffnr(x)=int(rnd(1)*x):l$=left$(s$,fnr(33)*2):y=81:v=10:a$="{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{light
gray}P"
3 b$=left$(a$,v+(fnr(5)*2)):l$=left$(l$+b$,78)+"{dark
gray}P":l$=l$+right$(s$,80-len(l$)):j=56320
4 printl$;:s$=l$:r=r-1:on-(r>0)goto2:w=39:l=l+1:s$="":r$="{red}f":fori=0to9:print"{reverse
on}{green}        ";
5
o=fnr(m):x(i)=o:on-(peek(p+o)<>80)goto5:pokep+o,d:pokec+o,2:next:ifl>vthend=83:r$="{green}h":g
oto9
6
fori=0tom:q=p+i:f=peek(q):pokeq,y:on-(peek(j)=n)goto7:pokec+x(iand7),iandvor2:pokeq,f:next:got
o6
7 pokeh+w,32:w=w-1:fork=0to9:on-(x(k)=i)goto8:next:pokeq,f:pokec+i,14:on-(w<0)goto9:next:goto6
8 x(k)=-1:s=s+500:pokeq,f:pokec+i,14:print"{home}{reverse off}{light
blue}"tab(v)s:g=g-1:on-(g=0)goto1:next:next:goto6
9
x=fnr(m):on-(peek(p+x)=32)goto9:pokep+x,d:pokec+x,7:printg$r$"ired!":on-(peek(197)=64)goto9:go
to0
```

Tip: Programm-Code im CBM prg Studio (http://www.ajordison.co.uk/) ansehen (kopieren und als BASIC-Programm einfügen). Dort werden die Kennwörter schön farbig markiert und der Code kann von dort auch direkt im Vice: C64 emulator ausgeführt werden.

Hier derselbe Code (PUR-80) unter Verwendung von erlaubten Abkürzungen - relevant für den Contest:

```
0cL:pO53280,0:pO53281,0:h=1984:p=1304:c=55576:n=111:d=102:m=599:g$="hrdddbu r: "
1g=10:r=15:s=s+w*50:?"bcrank"ltA9)"$"s"dddddd":fOi=0to39:s$=s$+"b ":pOh+i,247:nE
2dEfnr(x)=int(rN(1)*x):l$=leF(s$,fnr(33)*2):y=81:v=10:a$="gmgPgPgPgPgPgPgPgP"
3b$=leF(a$,v+(fnr(5)*2)):l$=leF(l$+b$,78)+"gP":l$=l$+rI(s$,80-len(l$)):j=56320
4?l$;:s$=l$:r=r-1:on-(r>0)gO2:w=39:l=l+1:s$="":r$="rf":fOi=0to9:?"rg        ";
5o=fnr(m):x(i)=o:on-(pE(p+o)<>80)gO5:pOp+o,d:pOc+o,2:nE:ifl>vtHd=83:r$="gh":gO9
6fOi=0tom:q=p+i:f=pE(q):pOq,y:on-(pE(j)=n)gO7:pOc+x(iand7),iaNvor2:pOq,f:nE:gO6
7pOh+w,32:w=w-1:fOk=0to9:on-(x(k)=i)gO8:nE:pOq,f:pOc+i,14:on-(w<0)gO9:nE:gO6
8x(k)=-1:s=s+500:pOq,f:pOc+i,14:?"hrb"tAv)s:g=g-1:on-(g=0)gO1:nE:nE:gO6
9x=fnr(m):on-(pE(p+x)=32)gO9:pOp+x,d:pOc+x,7:?g$r$"ired!":on-(pE(197)=64)gO9:gO0

         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

FireFighter64 Tenliner erklärt (000=Zeile 0/100=Zeile 1/...):
==========
```
000 clr: rem Zuruecksetzen aller Variablen auf 0 (viel sparsamer als l=0:s=0:w=0:)
010 poke53280,0: rem Rahmenfarbe auf schwarz
020 poke53281,0: rem Hintegrundfarbe auf schwarz
030 h=1984: rem Startzeile Screen-RAM Wasserstandsanzeige
040 p=1304: rem Startzeile Screen-RAM der Haeuserfront
050 c=55576: rem Startzeile Farb-RAM der Haeuserfront
060 n=111: rem Konstante fuer Feuerknopf gedrueckt Wert
070 d=102: rem Konstante fuer den PETSCII-Wert vom Karomuster (=Feuer)
```

```
080 m=599: rem Konstante fuer die maximale Anzahl zu durchlaufenden Zeichen (15 Zeilen mit
je 40 Zeichen startend bei 0)
090 g$="{home}{reverse off}{down}{down}{down}{light blue}u r: ": rem Teilstring fuer "You
are F/Hired"-Game Over Text

100 g=10: rem Anzahl zu loeschende Feuer pro Level (g=Gefahrenherd? Passt)
110 r=15: rem Zeilencounter fuer Haeuserzeilen (r=rows)
120 s=s+w*50: rem Addiere verbleibende Wassereinheiten zu Score hinzu
130 print"{light blue}{clear}rank"ltab(9)"$"s"{down}{down}{down}{down}{down}{down}": rem
Ausgabe der oberen Statusanzeige
140 fori=0to39: rem In diesem 40x Loop geschehen 2 Sachen auf's Mal :)
150 s$=s$+"{black} ": rem 1. Haeuserzeile mit schwarzen Leereichen fuellen
160 pokeh+i,247: rem Untere Wasserstandsanzeige mit PETSCII-Wert 247 aufbauen
170 next

190 rem *** Es folgt der Levelaufbau ***
191 rem In den Zeilen 200-430 wird die Fassade gebildet (15x Basiszeile ergaenzt).
192 rem Dabei wird immer zum letzten String ein weiteres Hauselement hinzugefuegt.
193 rem Der String in Zeile 240 enthaelt die eigentliche Haeusergrafik.
194 rem Hinweis: Vielleicht wundert es, dass {light gray} wiederholt wird.
195 rem Die Farb-/Zeichen-Paare sind aber wichtig, damit beim Zusammensetzen
196 rem mit ungeradem Left$-Anteil keine sichtbaren Zeichen bei der Ausgabe
197 rem verloren gehen, da der String mit Printl$; ausgegeben wird. Haette ich
198 rem die Haeuserfront nur einfarbig darstellen wollen - ohne Schatten - dann
199 rem haette man die Farbe weglassen und 40 statt 80 Zeichen nehmen koennen.

200 deffnr(x)=int(rnd(1)*x): rem Funktionsdefinition r(x) fuer Rueckgabe eines Zufallswertes
im Bereich x. Spare damit einige Zeichen.
210 l$=left$(s$,fnr(33)*2): rem Per Zufall die ersten 1-32 Zeichen vom letzten String
uebernehmen
220 y=81: rem Konstante fuer PETSCII-Wert Kreis (=Suchscheinwerfer)
230 v=10: rem Allgemeine Konstante fuer Anzahl Feuer bzw. Farbe Hellrot
240 a$="{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{light gray}P"

297 rem in Zeile 300 wird vom 10 Zeichen langen Hauselement-String je nach Zufall
298 rem ein 5 bis 10 Zeichen langes Stueck herausgenommen und dann noch ein
299 rem Schattenzeichen (in dunkelgrau) hinzugefuegt

300 b$=left$(a$,v+(fnr(5)*2)): rem Zufaellig langes Bauteil bauen (10-15 Zeichen)
310 l$=left$(l$+b$,78)+"{dark gray}P": rem das Bauteil wird am linken String-Teil angehaengt
und auf max 39 Zeichenpaare (Farbe/Zeichen) gestutzt
320 l$=l$+right$(s$,80-len(l$)): rem Rest mit Zeichen des Vor-Strings auffuellen
330 j=56320: rem Konstante fuer Joystick Port 2 Adresse

400 printl$;: rem 40-Zeichen-Zeile ausgeben
410 s$=l$: rem Basis-String ersetzen mit Basis-String plus neuem Teil
420 r=r-1: rem Zeilencounter reduzieren
430 on-(r>0)goto200: rem Solange noch Zeilen zu zeichnen sind, das Ganze nochmals wiederholen
440 w=39: rem Wasser-variable auf Maximum setzen
450 l=l+1: rem Level-variable um eins erhoehen
460 s$="": rem Basis-String auf leer zuruecksetzen. Erst relevant bei Neuaufbau
470 r$="{red}f": rem setze als default den "F"ired Wert, in Zeile 9 verwendet wird
480 fori=0to9: rem Auch in diesem 10x Loop geschehen 2 Sachen auf's Mal :)
490 print"{reverse on}{green}        ";: rem 1. Wiesenfarbe einschalten und Wiesen-Element
zeichnen und 2. ...

500 o=fnr(m): rem Ermittle zufaellige eine relative Position im Screen-Bereich
510 x(i)=o: rem Weise diesen Wert einem Feuer zu
520 on-(peek(p+o)<>80)goto500: rem Kein Mauerzeichen? Dann ermittle nochmals
530 pokep+o,d: rem Zeichne an dieser Position das Karomuster (=Feuer)
540 pokec+o,2: rem An derselben Stelle Farbe rot ins Farb-RAM schreiben
550 next: rem Nach diesem Loop haben wir 10 relative Screen-Positionen in x()
560 ifl>vthend=83:r$="{green}h":goto900 rem Hoechsten Level bzw. Rang erreicht? Dann setze
Herz-Zeichen, "H"ired-Vorzeichen und springe zu Endsequenz

599 rem *** Es folgt der Spielstart (Hauptschleife) ***

600 fori=0tom: rem Durchlaufe 600x (40 Zeichen x 15 Zeilen) und mache folgendes:
```

```
610 q=p+i: rem p+i wird mehrmals benoetig. Spare damit ein paar Zeichen.
620 f=peek(q): rem Rette aktuelles Zeichen an dieser Screen-Position
630 pokeq,y: rem Setzte das PETSCII-Zeichen fuer Suchscheinwerfer (Kreis)
640 on-(peek(j)=n)goto700: rem Joystick gedrueckt? Dann Wasser marsch!
650 pokec+x(iand7),iandvor2: rem setze eine Feuerfarbe rot(2) oder hellrot(10) an einem der
10 Feuer
660 pokeq,f: rem Ersetze den Suchscheinwerfer wieder mit gerettetem Zeichen
670 next: rem und das solange, bis unten rechts agenkommen
680 goto600: rem Dann starte wieder an Screen-Position links oben

699 rem *** Es folgt die Logik, wenn der Joystick-Knopf gedrueckt wurde ***

700 pokeh+w,32: rem Zeichne an aktueller Wasserstandsanzeige ein Leerzeichen
710 w=w-1: rem reduziere Wassereinheit um eins
720 fork=0to9: rem Durchlaufe alle 10 relative Feuerpositionen
730 on-(x(k)=i)goto800: rem identisch mit aktueller Position? Dann Feuer loeschen!
740 next
750 pokeq,f: rem Ersetze den Suchscheinwerfer wieder mit gerettetem Zeichen
760 pokec+i,14: rem Markiere die Stelle mit hellblau (=Wasserfarbe)
770 on-(w<0)goto900: rem Wasser verbraucht? Dann Endsequenz mit "F"ired-Ausgabe!
780 next: rem und das solange, bis unten rechts angekommen
790 goto600: rem Dann starte wieder an Screen-Position links oben

799 rem: *** Es folgt die Logik, wenn ein Feuer geloescht wird ***

800 x(k)=-1: rem Setze relative Screen-Position auf -1 (damit "out of scope")
810 s=s+500: rem Erhoehe Kontostand um $500
820 pokeq,f: rem Ersetze den Suchscheinwerfer wieder mit gerettetem Zeichen
830 pokec+i,14: rem Markiere die Stelle mit hellblau (=Wasserfarbe)
840 print"{home}{reverse off}{light blue}"tab(v)s: rem Aktualisiere Kontostand
850 g=g-1: rem Ein weiteres Feuer geloescht
860 on-(g=0)goto100: rem Saemtliche Feuer geloescht? Dann naechster Einsatzort
870 next: rem und das solange, bis alle Feuerpositionen durchgechecked
880 next: rem und das solange, bis unten rechts agenkommen
890 goto600: rem Dann starte wieder an Screen-Position links oben

899 rem *** Es folgt die Logik wenn der Wasserstand auf 0 ist (=Game Over) ***

900 x=fnr(m): rem Zufallswert fuer relative Position
910 on-(peek(p+x)=32)goto900: rem Falls Position=Leerzeichen dann andere Position
920 pokep+x,d: rem Setze im Screen-RAM an diesre Stelle ein Karomuster (=Feuer)
930 pokec+x,7: rem Setze im entsprechenden Farb-RAM die Farbe Gelb
940 printg$r$"ired!": rem Ausgabe von Game Over oder Erfolg text
950 on-(peek(197)=64)goto900: rem Falls keine Taste gedrueckt, dann Feuerinferno ausweiten
960 goto000: rem Neustart

2016 rem basic tenliners contest / (c) roman werner @romwer roman.werner@gmail.com
```

Making-Of bzw. Erklärungen zu einigen Design-Entscheiden

Zur Arbeitsumgebung: Ich habe durchgehend mit dem tollen CBM prg Studio gearbeitet, und von dort den BASIC Code zum Testen jeweils im VICE C64 emulator ausgeführt.
Anfangs habe ich überhaupt nicht auf die die Zeilenlimite von 10 geschaut, sondern einfach ein bisschen rumprobiert, mit einem Befehl pro Zeile.
Erst als die Logik für den Aufbau der Hausfassade funktioniert hat, habe ich mal geschaut, wie viele der Befehle in eine Zeilenlänge passen und wie viele Zeilen das schon benötigt.

Ich hatte mich beim Kürzen des Codes lange Zeit auf die Warnung "Line n is greater than 80 Bytes after generation." verlassen, zum prüfen, ob ich den PUR-80 Bereich verlasse.
Erst beim effektiven Eingeben von Hand im C64 Emulator habe ich aber gemerkt, dass 80 Bytes nicht gleich 80 Zeichen sind und musste dann als Folge den an sich fertigen Code nochmals komplett überarbeiten.
Als Folge dieser Erkenntnis musste ich Code-Änderungen dann immer erst wieder manuell auf Abkürzungsformat (z.B. pO statt poke) bringen, um dann auf 80 Zeichen prüfen zu können.
Bei jeder grösseren Änderung im Code das gleiche Spiel von vorne. Das hat mich dann insgesamt doch einige Zeit und Nerven gekostet.

Das Thema FireFighter ist erst im Laufe des Entwicklungsprozesses entstanden. Angefangen

hatte es mit der Idee, das Winkelzeichen (Shift P) für das Darstellen von Haeuserblocks einzusetzen.
Ich hatte danach mit Sprites experimentiert, z.B. 2 King Kong Würfel, die die Häuser zerstören sollten. Oder Meteoriten, die von oben herab fallen.
Ich musste mich bei PUR-80 aber schnell mit den beschränkten Möglichkeiten abfinden, nachdem ich feststellen musste, dass der Hausaufbau alleine bereits 5 Zeilen gekostet hatte.

Ich hatte dann die Eingebung mit dem Suchscheinwerfer, was mit den technischen Gegebenheiten einfach umzusetzen war. Und die Wasseranzeige, die man einfach aufbauen konnte.
Und durch den Umstand, dass der Suchscheinwerfer ohne Fassade für den Spieler nicht sichtbar ist (in Zeile 150 wird dafür im Basis-String die Zeichenfarbe {black} gesetzt), ergab sich noch eine schöne zusätzliche Herausforderung.

Das 1-Button-Spiel-Prinzip hat sich dann einfach so ergeben und ist dankbar, weil für jeden schnell zugänglich.
Den Joystick-Button (in Zeile 640) sollte man eigentlich mit peek(56320)and16=0 abfragen und nicht mit peek(56320)=111, aber solange man am Joystick nur den Knopf betätigt und in keine Richtung drückt, dann geht das ok.
Es wurden dadurch ja wieder ein paar wertvolle Zeichen gespart.

Zu Anfang hatte ich nur eine "Score"-Anzeige und habe erst später auf Rank und $ umgestellt, da dies für das Thema FireFighter-Karriere authentischer ist.
Auch gab es zu Beginn kein Ende oder Spielziel, sondern man hätte bis zum Game Over bzw. auf den nächsten Hiscore hin gespielt. Mit dem Karriere-Hintergrund (Rang 0-10) hat der Spieler nun ein klares Ziel vor Augen.

Das Feature mit "You are fired!" als Game Over-Text hat mir wegen der Doppeldeutigkeit gefallen. Eine Endsequenz konnte ich zu dem Zeitpunkt aber vergessen, weil keine Zeichen mehr übrig waren.
Erst viel später kam mir in den Sinn, dass ich bei Erreichen des 10. Levels daraus ja ganz einfach einen "You are hired!"-Text machen könnte.
Mit viel optimieren und tricksen konnte ich dann sogar doch noch eine Game-Over und End-Sequenz (Zeile 900-960) herausholen!

Die Einschränkung, dass bei einem if/then alle nachstehenden Befehle in derselben Zeile betroffen sind, war für mich immer wieder eine Knacknuss.
Auf den Trick mit dem "on-(condition)"-Befehl bin ich nicht selbst gekommen, sondern habe das in einem anderen Tenliner gesehen und gleich dankbar übernommen.

Im Spiel muss man in jedem Level 10 Flammen löschen. Das hat sich technisch so ergeben, weil...
...Arrays nur auf 11 Einheiten vordimensioniert sind und man für mehr einen DIM hätte verwenden müssen, was wiederum wertvolle Zeichen gekostet hätte.
...Ich beim Zuweisen-Loop der Flammen-Positionen (Zeilen 480-540) mit 10 Einheiten auch gleichzeitig den Aufbau der grünen Wiese durchführen kann.
Dass die Spiel-Balance 40 Wassereinheiten = 10 Flammen löschen gut aufgegangen ist, war schlussendlich Glück.

Das Flackern der Flammen (Zeile 650) hatte ich anfangs mit einem fixen Array mit 2 Werten f(0)=2 und f(1)=10 und dem Zuweisen von f(iand1) gelöst.
i ist dabei einfach die aufsteigende Position des Suchscheinwerfers von 0-599. Mit iand1 erhalten wir 0 oder 1 je nach Wert von i.
Mit iand10or2 habe ich das etwas anders gelöst, spare damit aber einige Zeichen (fixer Array nicht mehr notwendig) und der Effekt ist mehr oder weniger derselbe.
Das 2 Flammen nicht flackern hat einen Grund. Mit "iand7" erhält man als Index nur Zahlen von 0-7. Ich hatte diese Anweisung zwischendurch mit "fnr(10)" ersetzt, dies machte das Spiel aber etwas langsamer und damit einfacher.
Habe dann im Sinne des Spielspasses entschieden, dass das fehlende Flackern, das kleinere Übel ist.

In Zeile 30 verwende ich im Print bewusst "{light blue}{clr}" statt "{clr}{lightblue}" weil damit gleichzeitig auch die Farbe für die Wasserstand-Anzeige gesetzt wird.

Mit einer Funktionsdefinition für die Rückgabe einer Zufallszahl (Zeile 200) konnte ich einige Zeichen einsparen.
Das war zwar zuerst ein Schritt zurück hat sich aber insgesamt gerechnet und sich deshalb gelohnt. :)

Einen lustigen Bug hatte ich zwischendurch noch eingebaut, als ich den TI (den Timer) Wert

missbrauchen wollte, um alternierend eine 0 oder eine 1 zu erhalten (TIand1 statt int(rnd(0)*2)).
Nach ca. 9 Minuten spielen ist das Spiel mit einem "?Illegal Quantity Error" abgebrochen. Das rührte daher, dass TI nach 9 Minuten seit start des VICE einen Wert von über 32768 erreicht hatte.
Ich musste dann nach einer anderen Alternative suchen. Dies einfach als Beispiel, dass man bei PUR-80 vieles versucht, um Zeichen zu sparen, aber nicht immer Erfolg hat.

Warten auf eine Taste (Zeile 960) hatte ich zuallererst mit getk$:on-(k$="")goto900 gelöst. Das hatte aber dazu geführt, dass wenn man während dem Spielen bewusst oder unbewusst eine Taste gedrückt hatte, die Game Over bzw. Endsequenz übersprungen wurde.
Der Befehl "Get" liest nämlich statt der gerade gedrückten Taste den nächsten Wert aus dem Tastaturbuffer und der baut sich ohne ein get während dem Spiel nicht ab.
Peek(197) dagegen retourniert wirklich nur die aktuell gedrücke Taste. Wenn keine Taste gedrückt wird, ist der Wert immer 64.

Ich hatte zuvor noch nie ein Tenliner geschrieben und wusste nicht, auf was man sich da einlässt. :-o
Ich kann es im Nachhinein aber jedem wärmstens empfehlen. Man lernt in kürzester Zeit eine MENGE Sachen über den eigenen Computer.
Zum Beispiel habe ich beim Ausprobieren festgestellt, dass rnd(1) fast doppelt so lange braucht, eine Zufallszahl zu produzieren, wie rnd(0). Kann vielleicht mal nützlich sein.

Zum Abschluss noch ein Tip, wie man sparsam einen 0/1 Schalter machen kann: An Stelle von "if a=0 then a=1 else a=0 endif" kann man auch einfach "a=1-a" verwenden. :)

Ideen für die Weiterentwicklung ausserhalb der 10 Zeilen Grenze:
==========

- Soundeffekte (z.B. beim Betätigen des Wasserstrahls,Feuer gelöscht, Game Over etc.)
- Zusätzliche Anzeige vom Namen der Einsatzortes (z.B. Dubai, New York, etc.)
- Möglichkeit, den Suchscheinwerfer mit dem Joystick nach oben oder unten zu korrigieren.
- Zweispielermodus, bei dem der zweite Spieler mit dem Feuerknopf Wasser nachpumpen muss.
- Auch im Zweispielermodus kommt nach einem verpasstem Durchlauf ein weiteres Feuer hinzu.
- Visualisiertes Rangabzeichen und im Spiel integrierte Laufbahn-Übersicht.
- Ein action geladenes Titelbild mit passender SID Musik.
- Hiscore-Liste