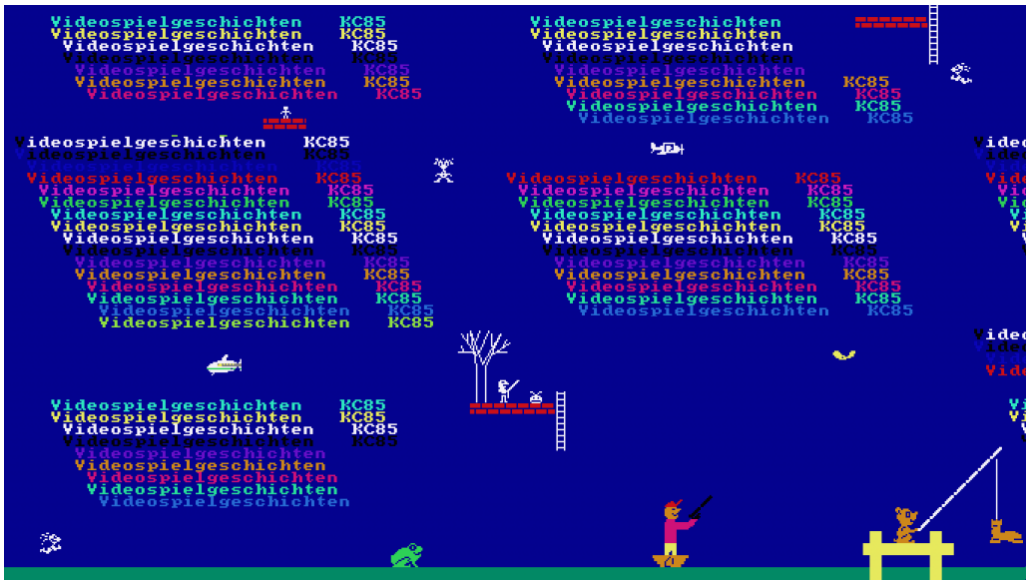


Retro-Programmieren auf dem KC85

Mario Donick am Samstag, dem 31. Juli 2021

Auch „im Osten“ gab es Computer und Computerspiele – und alle paar Jahre wieder sind sie Thema in den Medien: Dieser Autor hat darüber 2014 einen langen Bericht in der Retro Gamer geschrieben, 2017 gab es eine zweiteilige Serie von Denis Gießler in der GameStar, und 2019 veröffentlichte René Meyer das Buch „Computer in der DDR“.

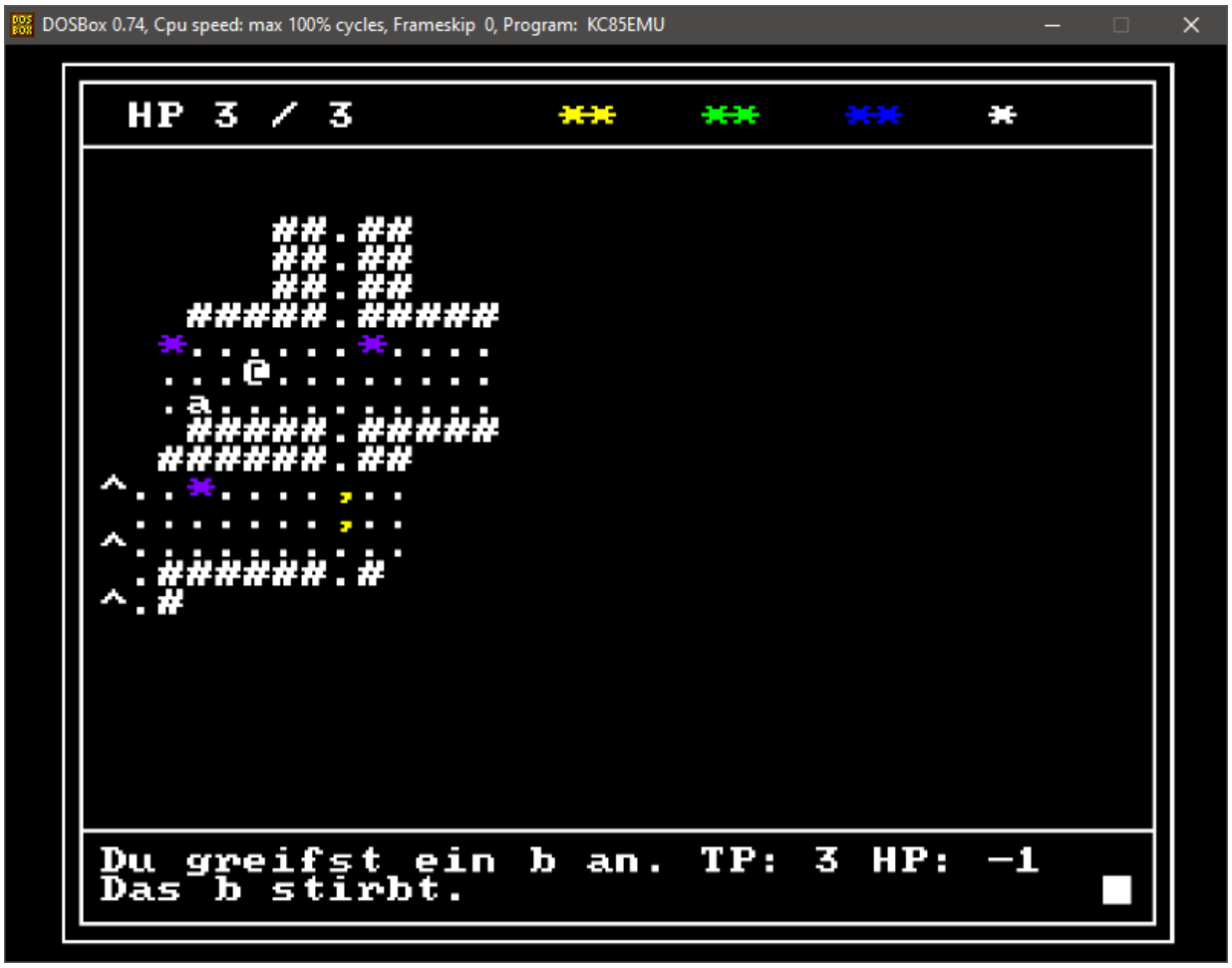
Doch ehrlich gesagt waren die meisten DDR-Spiele eher kleine Spielereien – und mit den alten Systemen zu programmieren, sorgt zumindest bei mir für intensiveres Retrofeeling als das Spielen. Weihnachten 2020 machte ich eine entsprechende Zeitreise.

Diesen Artikel schreibe ich mit WordPro – dem besten Textverarbeitungsprogramm der DDR. Ja, es gab für die Büroarbeit nicht nur Raubkopien aus dem westlichen Ausland (wie WordStar, das in der DDR als „TP“ vertrieben wurde), sondern auch Eigenentwicklungen, die oft zuerst in Form von Quelltexten in Buchform vertrieben wurden.

„Tips und Tricks für kleine Computer“ nannte das Vater-Sohn-Gespann Klaus und Stefan Schlenzig ihr Buch, das zeigte, dass mit dem KC85/3 (einem Z80-kompatiblen 8-Bit-Computer mit nur 32 Kilobyte RAM und 320×256 Pixel Bildschirmauflösung) ernsthaftes

Arbeiten möglich war.

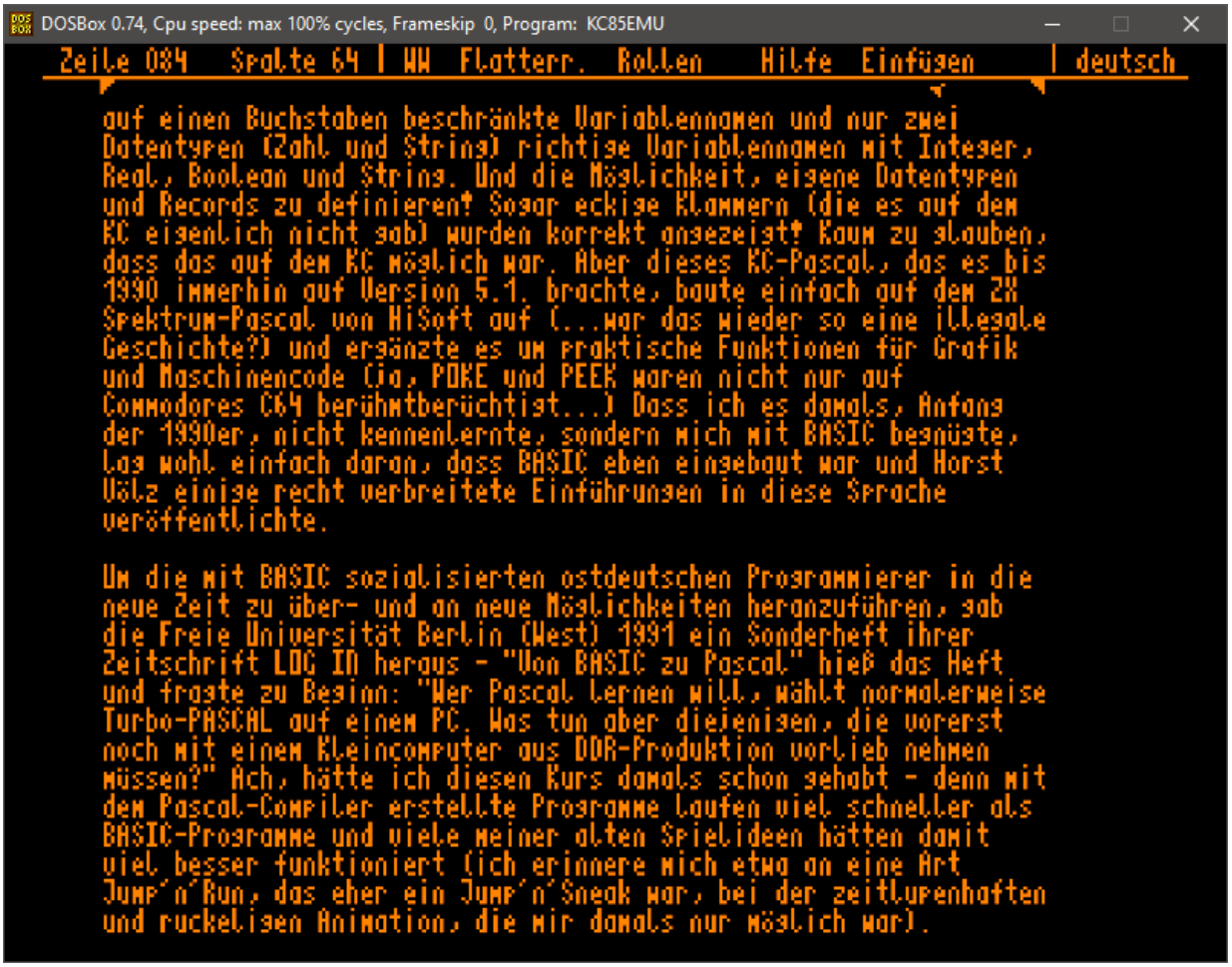
Kurz nach der Wende bekam ich so einen alten Computer von einem Verwandten geschenkt. Statt an Arbeit war ich im Alter von 10, 11 Jahren natürlich eher am Spielen interessiert, und ironischerweise habe ich in den letzten acht Monaten mehr Text mit WordPro auf einem KC85-Emulator geschrieben als damals als ich noch einen echten „Kleincomputer“ hatte.



Weihnachten 2020 tauchte ich in eine Retro-Programmiererfahrung ein und schrieb ein rogulike für den DDR-„Kleincomputer KC85/4“. Dazu nutzte ich KC85EMU als Emulator, der wiederum in einer DOSBox läuft.

Aber es soll hier nicht um Textverarbeitung in der DDR gehen; das soll nur den Rahmen setzen – den engen Rahmen aus 320 mal 256 Pixeln in 16 Farben, einen zweistimmigen Piezosummer für die Tonausgabe, optional an einen Kassettenrecorder angeschlossen als Verstärker.

Der KC85 war mein erster Computer, auf ihm habe ich Programmieren gelernt und ihn so lange benutzt, bis er nach und nach kaputtging. Ich lieh mir damals Computerbücher aus der Stadtbibliothek aus, kaufte mir Spielezeitschriften und versuchte, meinem KC die ganzen tollen Dinge zu entlocken, die „Westcomputer“ und Konsolen jener Zeit längst konnten.



Dieser Artikel entstand mit WordPro auf einem emulierten KC85.

Manche Genies wie Raimo Bunsen („Bennion Geppy“, „Mad Breakin“) oder Stefan Scholz („Karate“) erschufen auf dem KC echte Wunder – flüssige Animationen, gute Sounds und Gameplay, das schon fast an West-Spiele erinnerte (siehe zum Beispiel dieses Video zu „Bennion Geppy“) und sogar heute noch Spaß macht. Das ging nur, weil sich solche Entwickler in die Tiefen von Assembler einarbeiteten.

Manche Spiele erinnerten mit bunten Hintergründen und flüssigen Animationen an „richtige“ Spiele, wie man sie sonst nur von Westcomputern kannte. Hier „Karate“ von Stefan Scholz

Ich hingegen blieb stets auf den sehr langsamen BASIC-Interpreter beschränkt und anfangs auf nur 15 Kilobyte nutzbaren RAM. Später erhielt ich immerhin eine 64K-Erweiterung und sogar das seltene Diskettenlaufwerk. Letzteres eignete sich u.a. dazu, Pixelgrafiken abzuspeichern und die dann als Illustrationen für Spiele zu nutzen.

Mein komplexestes Spiel dieser Art war ein rundenbasiertes Wirtschaftsspiel auf einem fremden Wüstenplaneten, das lose an Hamurabi erinnerte und das ich sogar selbst mit Freude spielte – das obere Bildschirmdrittel zeigte Statistiken und Menüs, der größere untere Bereich mehr oder wenige gut gezeichnete Standbilder vom Planeten oder von der zwielichtigen Bar, wo man irgendwas Illegales machen konnte. Leider habe ich vergessen, wie das genau war.

Überhaupt ist meine ganze KC-Geschichte verloren gegangen. Denn mit 16, 17 war meine KC-Zeit vorbei. Da bekam ich meinen ersten PC und ich dachte nicht daran, dass ich mit fast 40 einmal wehmütig daran zurückdenken würde.

Mein BASIC-Wissen übertrug ich mit Hilfe des schulischen Informatikunterrichts auf das zeitgemäßere Turbo Pascal (in der Schule für DOS, zu Hause für Windows, wo es umständliche Objektorientierung gab). Und tatsächlich war Pascal dann die Sprache, bei der ich hängenblieb, zuletzt in der Form von FreePascal, mit dem ich mein roguelike-RPG „LambdaRogue – The Book of Stars“ entwickelte (erhältlich bei itch.io).

Späte Freude

Professionell entwickelt heute wohl niemand mehr mit Pascal (oder?), aber ich mag diese zwar traditionelle und etwas geschwätzige, aber dadurch klare und gut lesbare Sprache.

Und ich war sehr überrascht, als ich Weihnachten 2020 (!) feststellte, dass es auch auf dem KC85 einen gut nutzbaren Pascal-Compiler gab (kann man bei kc85.info runterladen). Das führte dazu, dass ich eine sehr schöne Woche lang in ein Retrofieber verfiel und das meines Wissens nach erste KC-roguelike programmierte.

Und was für verpasste Gelegenheiten schienen da auf. Statt GOTO und GOSUB richtige Prozeduren mit PROCEDURE und FUNCTION. Statt auf einen Buchstaben beschränkte Variablennamen und nur zwei Datentypen (‚Zahl' und String) richtige Variablennamen mit Integer, Real, Boolean und Char. Und die Möglichkeit, eigene Datentypen zu definieren! Sogar eckige Klammern (die es auf dem KC eigentlich nicht gab) wurden korrekt angezeigt.

Kaum zu glauben, dass das auf dem KC möglich war. Aber dieses KC-Pascal, das es bis 1990 immerhin auf Version 5.1. brachte, baute einfach auf dem ZX Spectrum-Pascal von HiSoft auf (… war das wieder so eine illegale Geschichte?) und ergänzte es um praktische Funktionen für Grafik und Maschinencode (ja, POKE gab es nicht nur zum Cheaten auf Commodores C64 …)

Dass ich Pascal damals, Anfang der 1990er, nicht kennenlernte, sondern mich mit BASIC begnügte, lag wohl einfach daran, dass BASIC eben eingebaut war und Horst Völz einige recht verbreitete Einführungen in diese Sprache veröffentlichte.

Um die mit BASIC sozialisierten ostdeutschen Programmierer in die neue Zeit zu über- und an neue Möglichkeiten heranzuführen, gab die Freie Universität Berlin (West) 1991 ein Sonderheft ihrer Zeitschrift LOG IN heraus – „Von BASIC zu Pascal" hieß das Heft und fragte zu Beginn:

> „Wer Pascal lernen will, wählt normalerweise Turbo-PASCAL auf einem PC. Was tun aber diejenigen, die vorerst noch mit einem Kleincomputer aus DDR-Produktion vorlieb nehmen müssen?"
>
> *LOG IN 11 (1991) Sonderdruck KC-PASCAL, S. 1.*

Ach, hätte ich diesen Kurs (Download als PDF bei kc85.info) nur damals schon gehabt – denn mit dem Pascal-Compiler erstellte Programme laufen viel schneller als BASIC-Programme und viele meiner alten Spielideen hätten damit viel besser funktioniert.

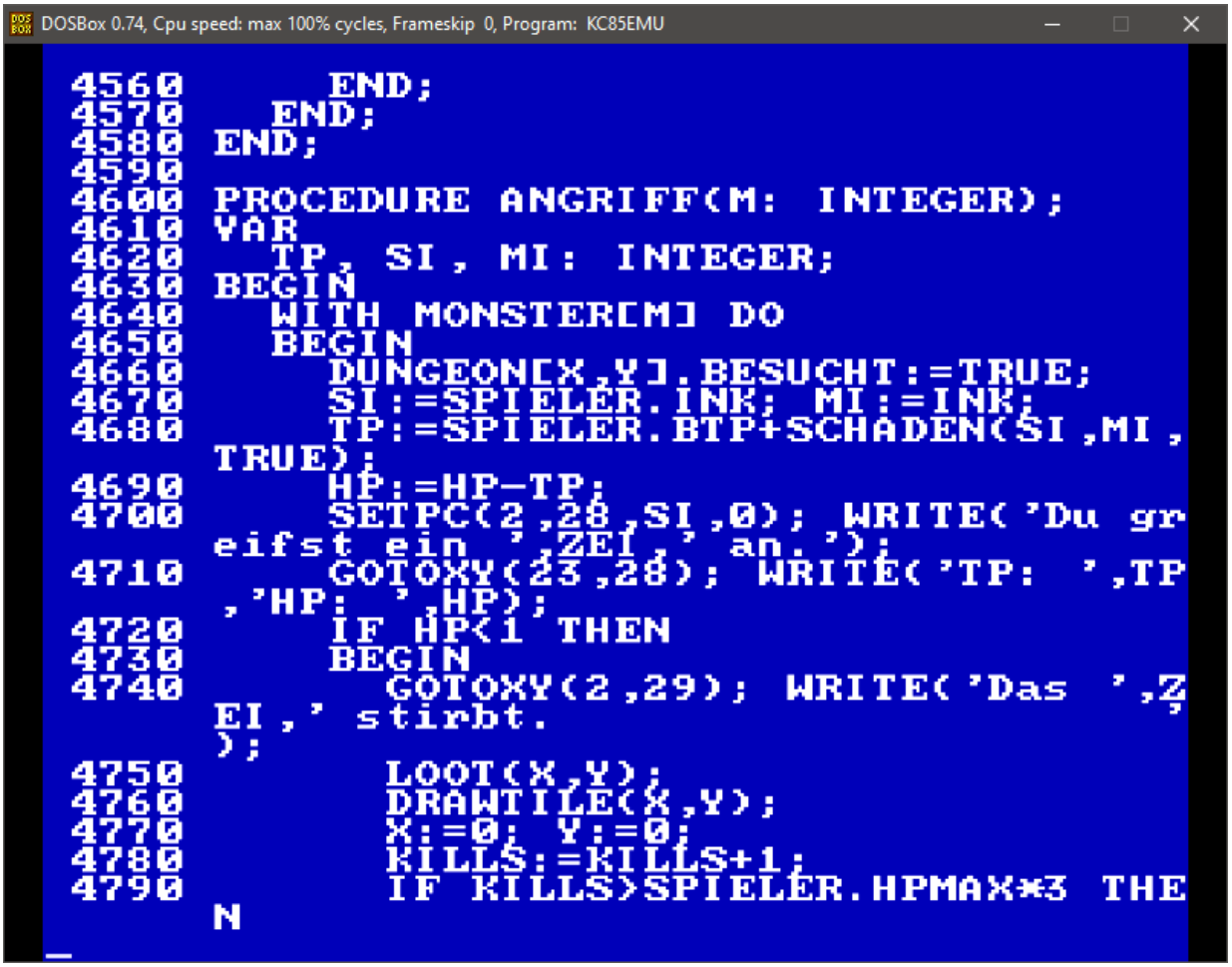
Ich erinnere mich etwa an den Versuch eines Jump'n'Runs, das eher ein Jump'n'Sneak war, angesichts der zeitlupenhaften und ruckeligen Animation, die mir damals nur möglich war (wie gesagt, andere Leute konnten das alles besser …)

Weihnachten 2020 jedenfalls holte ich diese verpasste frühe Programmiererfahrung nach und fing an, ein traditionelles roguelike zu programmieren.

Die Sprache selbst war mir dank viel FreePascal-Erfahrung immer noch vertraut. Eher gewöhnungsbedürftig war die Bedienung des zeilenbasierten Editors, mit dem der Quellcode eingegeben wurde.

Zeilenbasiert meint: Kein freies Bewegen des Cursors durch den Quelltext, kein einfachen Löschen und Einfügen. Stattdessen hat jede Zeile eine Nummer, und mit Eingabe des Kommandos „E“ plus Nummer holt man die Zeile in den Editierpuffer. In diesem Modus gibt es dann eine Menge weiterer Eintasten-Befehle, wie „K“ („Kill“), um das Zeichen an der Cursorposition zu löschen, oder „C“ („Change“), um das Zeichen an der Cursorposition durch ein anderes zu ersetzen.

Um größere Abschnitte in den bestehenden Quelltext einzufügen, kann man die nicht genutzten Zeilen (z.B. zwischen Zeile 700 und 710) verwenden und dann mit einem Befehl umnummerieren (damit aus 701 bis 709 dann die neuen 710 bis 790 werden, was wiederum Platz dazwischen schafft).



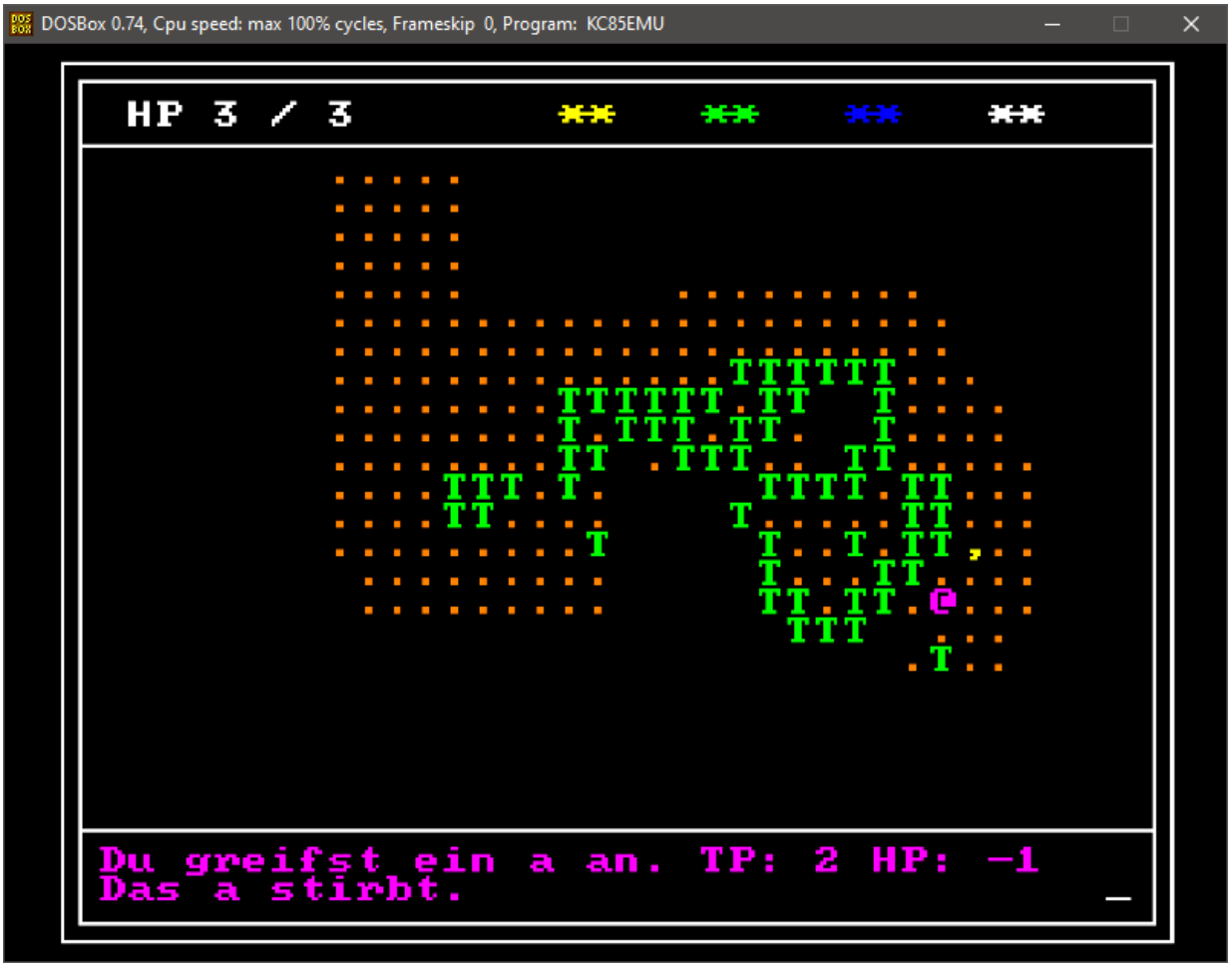
Quelltextausschnitt aus meinem KC roguelike. Aus heutiger Sicht ist die Bearbeitung sehr ineffizient, aber man gewöhnt sich an alles und nach einer Woche fühlt es sich an, als hätte man nie was anderes gemacht.

All das dies und die geringe Textauflösung des KC (pro Zeile nur 40 Zeichen – u.a. deshalb war das eingangs erwähnte WordPro mit ganzen 80 Zeichen/Zeile so innovativ) machen die Programmeingabe und das Bearbeiten ziemlich qualvoll. Aber das war in meinen jungen Jahren mit BASIC nicht anders, man gewöhnt sich an alles und irgendwann findet man Zeileneditor wieder voll effizient …

Ein roguelike auf dem KC85

Jedenfalls produzierte ich Weihnachten gut 800 Zeilen Quelltext, der tatsächlich ein funktionierendes roguelike in genretypischer ASCII-Darstellung ergab – und voller Klischees steckt. Genau das war mein Ziel: Ein klischeehaftes Fantasy-roguelike, das so in den 1980ern wirklich hätte erscheinen können.

Als in Purpur gewandete*r Nachwuchs-Elementarmagier*in @ steigen wir im „KC Roguelike“ (KCRL; einen echten Namen hat das Spiel noch nicht) immer tiefer in die Erde hinab, quasi als Abschlussprüfung unserer Ausbildung (wieso muss ich eigentlich gerade an Final Fantasy VIII denken…?) Die 26 bildschirmgroßen Levels sind zufallsgeneriert, Elementarmonster (a bis z) wollen uns an der Erfüllung unserer Aufgabe hindern.

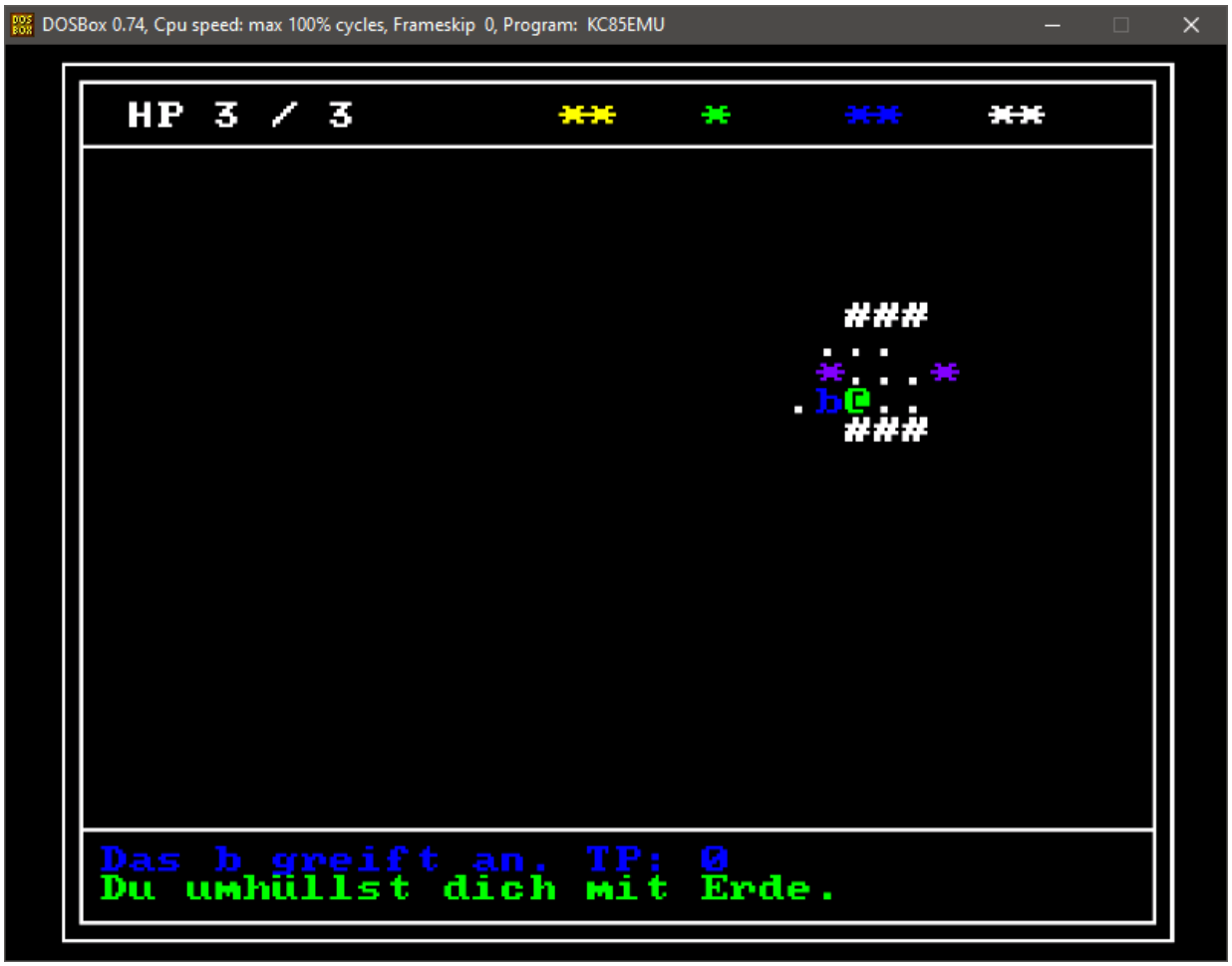


Kurz nach Spielbeginn in meinem „KC roguelike“

Wir hüllen uns in Wasser, Feuer, Erde oder Luft, um so den Angriffen der Monster zu entgehen oder sie zu bekämpfen: Wasser löscht Feuer, Luft aka Wind verweht Wasser, Feuer verbrennt Erde, Wasser nährt Erde, usw. Entsprechend sind wir selbst auch verwundbar, wenn wir gerade im falschen Element stecken. Andererseits greifen uns Monster nicht an, wenn wir dasselbe Element wie sie haben.

Der Wechsel in ein Element verbraucht Elementarenergie. Mit im Dungeon verteilten Energiekugeln laden wir die Energie wieder auf, was uns aber im Austausch einen Gesundheitspunkt kostet (als ich das programmierte, hatte ich gerade die „Witcher“-Folge gesehen, wo Magie auch nur im Austausch für Lebensenergie funktioniert).

Gesundheit regeneriert sich beim Betreten des nächsten Levels (da gibt es außerdem einen Punkt auf die Maximum-HP dazu), oder indem wir von getöteten Monstern hinterlassene … tja, keine Ahnung – ich habe vergessen, wofür das gelbe Komma steht, das Monster als Loot hinterlassen, aber es aufzuheben, heilt. Die Monster werden mit zunehmendem Level stärker, und manchmal gibt es sie in Großbuchstaben als besonders starke Varianten.



Wasser nährt Erde, daher kann mir das b nichts anhaben. Zu Beginn starten wir mit je zwei Energiepunkten pro Element, aber in Dungeons können wir im Austausch für Lebenspunkte weitere erhalten.

Es gibt auch einen einfachen Line-of-Sight-Algorithmus. Sowas ist für roguelikes essenziell, denn schließlich wird so erst das Gefühl erzeugt, dunkle Dungeons oder Höhlen zu erkunden, ohne zu wissen, was um die Ecke lauert.

Dafür klaute ich den bei GitHub verfügbaren LoS-Quelltext des alten VMS Moria – das war eine frühe Variante des Moria-roguelikes, die 1983 in VMS Pascal geschrieben wurde. Sie lief auf der alten VAX-11. Dieser Algorithmus hat einige Nachteile im Vergleich zu modernen LoS-Algorithmen (u.a. ist er etwas unpräzise), aber er ist kurz und läuft auch auf dem KC in annehmbarer Geschwindigkeit.

Schnell stieß ich jedoch auf ein Problem: Den Speicherplatzbedarf des Quelltexts. Der wird komplett in den KC geladen, und wenn er zu lang ist, steht für den Kompiliervorgang nicht mehr genug RAM zur Verfügung.

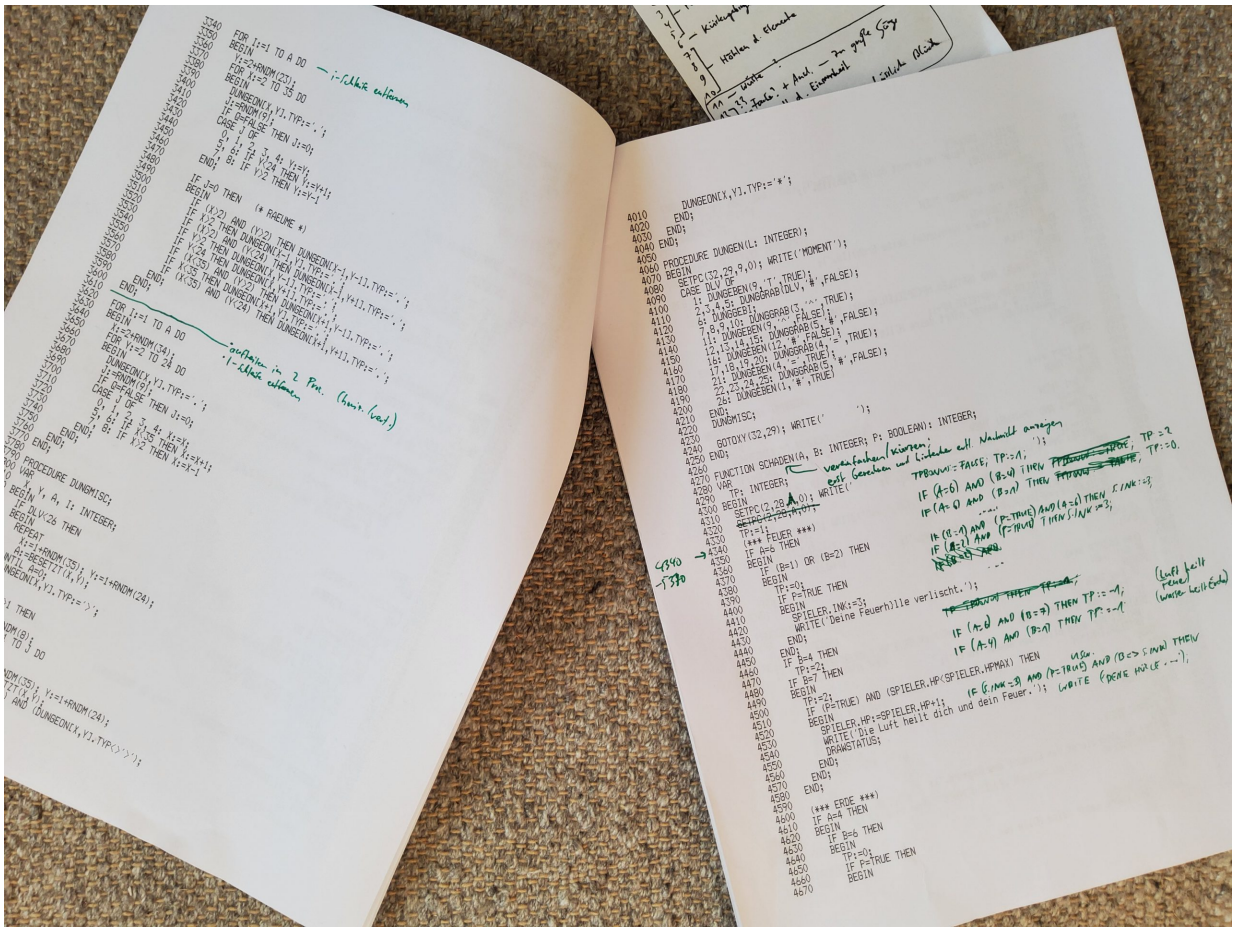
Vor allem für ein bisschen Fantasy-Atmosphäre generierenden Text ist da kaum Platz,

geschweige denn für ein Intro oder eine Kurzanleitung. Auch auf Namen für die ASCII-Monster musste ich verzichten – „Du tötest das a“ heißt es dann eben, und was das „a“ eigentlich ist, bleibt der Fantasie überlassen. Oder der Papieranleitung, die eventuell noch folgen wird.

Zwar kein Spielspaß, aber …

Insgesamt gesehen ist das KCRL also absolut nichts Besonderes, aber genau so etwas wollte ich ja haben. Doch macht das Spiel denn auch Spaß?

Ehrlich gesagt … nee. Oder nicht auf Dauer. Oder zumindest jetzt noch nicht. Im Prinzip ist das Spiel fast fertig, eigentlich fehlt nur noch Balancing und Polishing. Der permanente Instadeath lauert an jeder Ecke, was viel Konzentration verlangt. Aber es ist alles sehr repetitiv, gerade in späteren Levels gibt es nur noch wenig Abwechslung.



Einer diverse Quelltext-Ausdrucke meines KC roguelike. Der gefällt mir besser als das eigentliche Spiel.

Werde ich das noch ändern? Wird KCRL noch das beste KC-Spiel, das nie erschien? Wohl nicht, denn der Sinn der ganzen Aktion lag rückblickend woanders. Es ging um’s Eintauchen in eine frühere Zeit.

Die Arbeit an dem Spiel – immerhin eine ganze Woche von früh bis spät – hat vor allem mir selbst gut getan. „Der Weg ist das Ziel“ und so. Sich nochmal auf die alte Technik einlassen, die umständliche Bedienweise, den knappen Speicherplatz. Das Programmieren selbst war hier die Retroerfahrung, diese angestaubte Atmosphäre, die vergilbten Handbücher …

Und der Ausdruck des Spiels: Meinen Quelltext habe ich mit dem Emulator als .rtf-Datei ‚ausgedruckt', die .rtf anschließend in Word mit einem Nadeldrucker-Font formatiert und das dann auf Papier gedruckt.

Da hatte ich ihn dann in der Hand. Wie früher. Als ich noch Teenie war und die Welt viel einfacher.

Dieser Beitrag wurde publiziert am Samstag, dem 31. Juli 2021 um 09:30 Uhr in der Kategorie: Videospielgeschichten, Hardware. Kommentare können über den Kommentar (RSS) Feed verfolgt werden. Du kannst zum Ende springen und ein Kommentar abgeben. Pingen ist momentan nicht erlaubt.

Über Videospielgeschichten

Videospielgeschichten ist eine offene Plattform für Hobbyautoren und Journalisten. Die Webseite wurde 2009 gegründet, um es jedem Menschen, unabhängig von seiner Profession, zu ermöglichen, persönlich, authentisch und unabhängig über Videospiele zu schreiben

https://www.videospielgeschichten.de