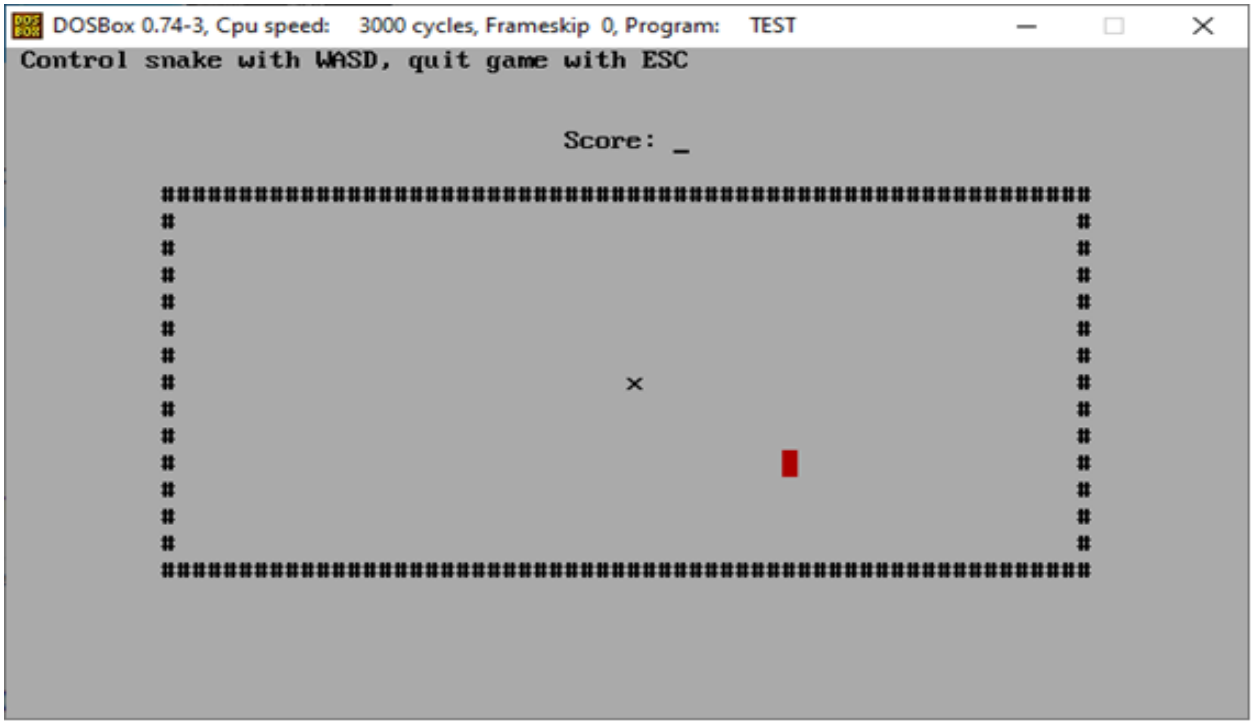
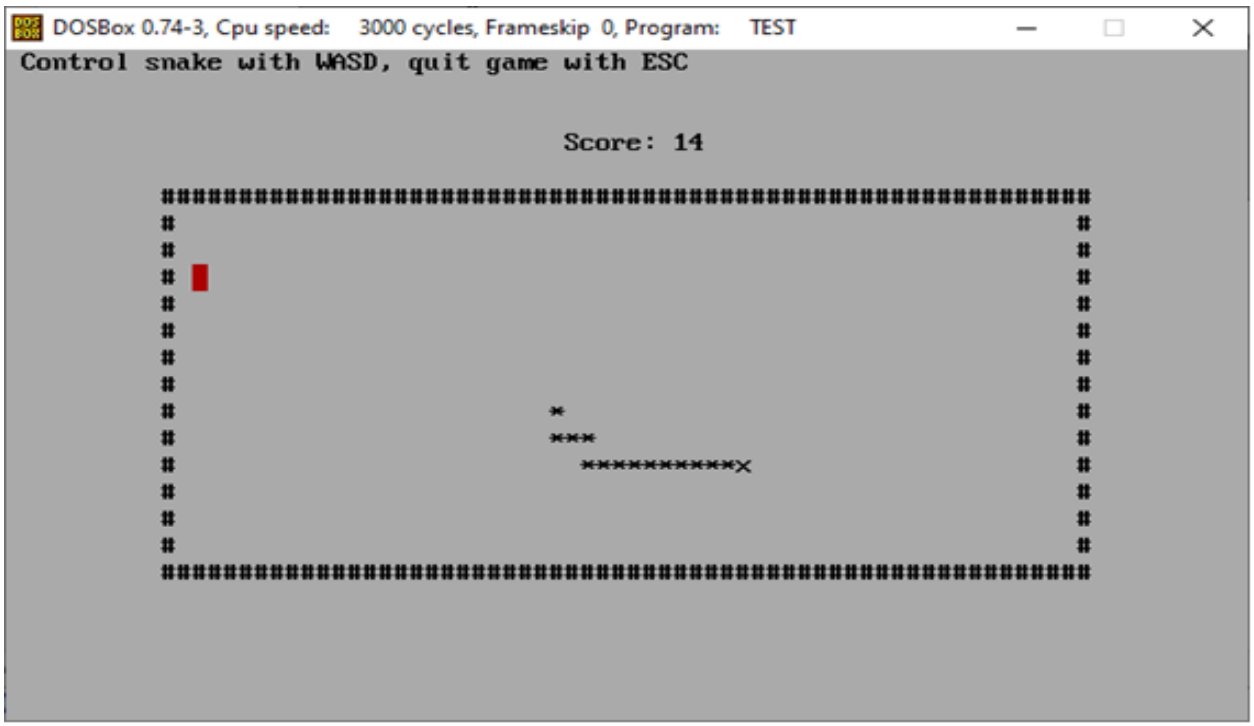
Andreas Meier, 18 INB-2 SnakeDoku, ASSEMBLER
Snake: alle Vorgaben, außer Highscore speichern



Startposition: Spiel startet sobald WASD gedrückt wurde



- Rote Pixel sind Äpfel
- Einsammeln der Äpfel erhöht den Score um eins und verlängert die Größe der Schlange um eins
- Symbol für den Rand: #
- Symbol für den Schlangenschwanz: $*$

- Sobald Rand oder Schwanz berührt wurde oder ESC gedrückt wird beendet sich das Spiel

```
.model small
.stack 100h

.data
keysmsg db 'Control snake with WASD, quit game with ESC$'
scoremsg db 'Score:   $'
deathmsg db 'Thx for playing!$'
delaytime db 5

Head_X db 40
Head_Y db 12
Snake_Size dw 1

snaketail_x db 300 dup(0)
snaketail_y db 300 dup(0)

scorenum db 4 dup(0)
scorepos db 42

hit db 0
wert1 db 33
wert2 db 213
wert3 db 251
random db 1
random2 db 1
drei db 3
acht db 8
zehn db 10
```

- Delaytime für den delay benutzt
- Head_X und Head_Y Koordinaten für den Schlangenkopf
- Snake_Size ist die Länge der Schlange (dw weil diese zwischendurch in SI zwischengespeichert wird, sodass selbe Bytelänge wegen Fehlermeldung)
- Snaketail sind Arrays mit jeder Position eines Körperteils
- Scorenum ist der Score und Scorepos ist die Position, die verwendet wird bei int_to_string Ausgabe
- Hit ist ein Flag ob die Schlange gegen den Rand oder Körper gegangen ist
- Der Rest sind Variablen um eine Pseudo-random Position für den nächsten Apfel zu berechnen

Procedures:

Main:
Als erstes ausgeführt über "*end main*", speichert Daten ab und macht Funktionsaufrufe für das Setup und Keyboardeingaben.

Keyboard_input:
Wartet zunächst auf erste Eingabe eines Keys um das Spiel zu starten, danach wird immer diese Richtung geloopt bis eine Key-Eingabe kommt, falls diese WASD oder ESC ist wird die Richtung der Schlange geändert (Falls Schlange z.B. nach rechts gerichtet ist bringen D und A logischerweise nichts und wird vernachlässigt).

Background:
Setup für die Spielebene, d.h. Hintergrundfarbe, Spielfeld, 1. Apfel und Stringausgaben.

Snake_start:
Schlangenkopf wird gesetzt, sowie die Position dessen in array gespeichert (Cursor wird immer wieder am Ende zum Score gesetzt, damit dort das Blinken des Cursors stattfindet).

Move:
Wird von Keyboard_input aufgerufen, um die Position der Schlange zu verändern und verändert die Position des Schlangenkopfes.

Move_tail:
Wird von Move aufgerufen und loopt bis der ganze Körper der Schlange neue Positionen hat in den Arrays. Die letzte Position der Schlange wird als erstes mit Leerzeichen überschrieben und dann jeweils die Vorgängerposition abgespeichert, sodass die Schlange sich im Prinzip nur am Kopf und am Schwanzende verändert.

Compare:
Überprüft, ob Schlangenkopf Rand, Körper oder Apfel getroffen hat.

Fruit_eat:
Erhöht Snake_Size und speichert eine weitere Position in den snaketail Arrays ab. Danach wird neuer Apfel erstellt durch Funktionsaufrufe.

Createfruit:
Erstellt Apfel als roten Pixel an Position, die durch randomnum procedure erstellt wurde.

Setscore:
Bestimmt die Position sowie die Stelle im Array scorenum welche Zahl als Score ausgegeben wird (über Int_to_string).

Int_to_string:
Wandelt Snake_Size − 1 über Rechenoperationen (div 10 der Zahl, Remainder in AH +48 für ASCII, dann in scorenum abspeichern, solange der Quotient in AL ungleich 0 ist) so um, dass die ASCII Digits der Zahl in einem Array eingespeichert werden können und dann von hinten nach vorne im Array die ASCII ausgegeben werden können um die Zahl auf den Display dar zustellen.

Death:
Wird aufgerufen, wen ESC oder Kollision und cleart den Screen gibt Ausgabe "$Thx\ for\ playing$" und gibt Kontrolle an DOS-Box zurück.

Randomnum:
Generiert Pseudo-random Zahl an Funktion des Linear congruential generator (LCG) angelehnt (SEED ist System Zeit) und Divisionsoperationen, um die Zahl zu verkleinern, sodass Wahrscheinlichkeit größer ist, dass die Zahlen innerhalb der Grenzen des Spielfelds liegen. Falls Zahlen out of bounds sind oder innerhalb der Schlange liegen werden neue Koordinaten erstellt.

Checkbody:
Prüft ob Koordinaten für nächsten Apfel in der Schlange liegen würden.

Delay:
Erstellt delay über 1A Interrupt.

Clear_screen:
Reinigt den Bildschirm von allen setup-Einstellungen.

Genutzte Quellen:

- Vorlesungsfolien
- https://en.wikipedia.org/wiki/INT_10H
- http://spike.scu.edu.au/~barry/interrupts.html
- https://www.reddit.com/r/learnprogramming/comments/2xaz1p/converting_an_integer_into_a_string_in_masm32/
- https://www.cs.uaf.edu/2015/fall/cs301/lecture/09_16_stack.html
- https://stackoverflow.com/questions/41897233/real-time-keypress-event-assembly-x86-tasm
- http://site.iugaza.edu.ps/ahaniya/files/Assembly-Language-Lab11.pdf
- https://stackoverflow.com/questions/46431682/printing-individual-characters-from-a-string-to-standard-output-in-ms-dos-assemb
- http://www.c-jump.com/CIS77/ASM/Instructions/lecture.html
- https://stackoverflow.com/questions/24532514/assembly-how-to-put-letters-in-grahic-mode
- https://forum.allaboutcircuits.com/threads/how-to-program-a-delay-in-assembly.13710/
- https://www.cs.virginia.edu/~evans/cs216/guides/x86.html#:~:text=The%20least%20significant%20byte%20of,to%20the%20same%20physical%20register.
- http://www.c-jump.com/CIS77/ASM/Instructions/lecture.html
- https://stackoverflow.com/questions/36219498/change-the-background-color-of-dosbox-console-when-executing-a-tasm-program
- https://stackoverflow.com/questions/54256179/assembly-masm-x86-16-bit-text-mode-change-color
- https://en.wikipedia.org/wiki/BIOS_color_attributes
- https://stackoverflow.com/questions/35994699/assembly-masm-array-manipulation
- https://en.wikipedia.org/wiki/Linear_congruential_generator
- https://www.csie.ntu.edu.tw/~acpang/course/asm_2004/slides/chapt_07_PartIISolve.pdf
- https://stackoverflow.com/questions/45270168/routine-that-causes-delay-in-assembly-using-1ah