Projekt Assembler-Programmierung
SNAKE

Marc Uxa & Benjamin Huber

Jahrgang: 18-INB2 & 19-INB1

HTWK Leipzig

2. August 2021

Inhaltsverzeichnis

1 Spiel

1.1 Anleitung

Bei diesem Spiel handelt es sich um das klassische Snake-Spiel. Es muss Futter eingesammelt werden, bei jedem Einsammeln steigt der Score um einen Punkt.
Es kann zwischen verschiedenen Schwierigkeitsgraden gewählt werden. Je nach Modus ändert sich die Geschwindigkeit und die Punkteanzahl, die benötigt wird, um das Spiel erfolgreich zu beenden. Die Stufen können mit einem Mausklick auf des entsprechende Wort ausgewählt werden.

easy (mode = 1):

- speed = 4
- max. 30 Punkte erreichbar
- ab 15 Punkten erhöht sich die Geschwindigkeit

normal (mode = 2):

- speed = 3
- max. 40 Punkte erreichbar
- ab 20 Punkten erhöht sich die Geschwindigkeit

hard (mode = 3):

- speed = 2
- max. 50 Punkte erreichbar
- ab 35 Punkten erhöht sich die Geschwindigkeit

Die Schlange kann mit den folgenden Tasten gesteuert werden:

- W nach oben
- A nach links
- S nach unten
- D nach rechts

Mit ESC kann das Spiel sofort beendet und geschlossen werden, nachdem eine Schwierigkeitsgrad ausgewählt wurde.

1.2 Besonderheiten

In dem Programm sind diese Teile enthalten:

- eigene Interrupt Service Routine für ISR1Ch
- eigene Maus Unterroutine (AH = 0Ch)
- Videomodus 3 (VGA-Grafik)
- Soundeffekte

2 Entwurfsskizzen

2.1 Programmlogik

Die Prozedur ”resetSnake” aktualisiert unser Schlangen-Array und wird vor jedem Zeichnen aufgerufen. Dabei werden die Werte in dem Array nach links geshiftet.
Bei ”snakeInc” werden die Werte des Array nach rechts geshiftet und eine weitere Stelle, die zuvor mit einer 0 initialisiert war bekommt nun den Wert des Kopfes.

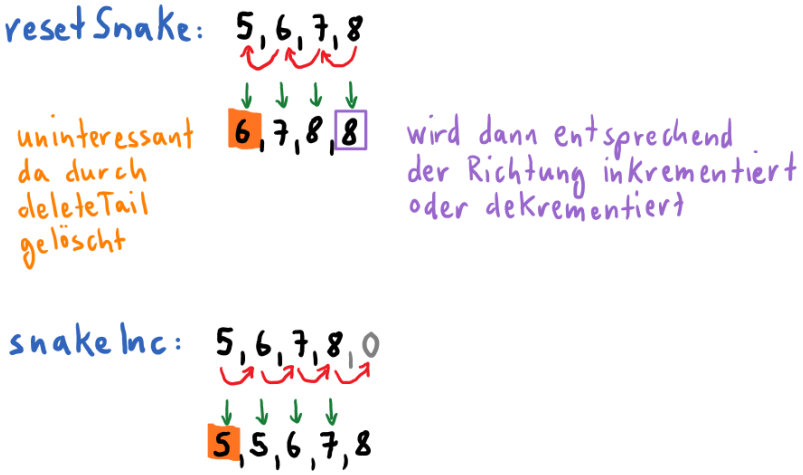

Abbildung 1: Prozeduren resetSnake und snakeInc

Das nachfolgende Bild zeigt, wie die Koordinaten angepasst werden, je nach Richtung in der sich die Schlange bewegen soll.

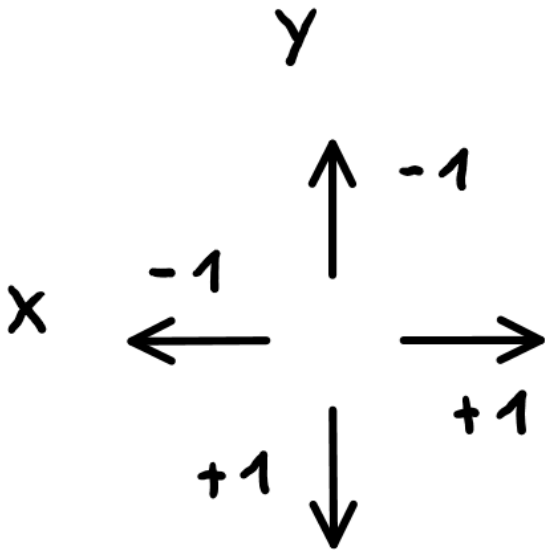

Abbildung 2: Bewegung der Schlange

2.2 GUI Skizzen

Die grafische Oberfläche wurde nach den nachfolgenden Skizzen erstellt:

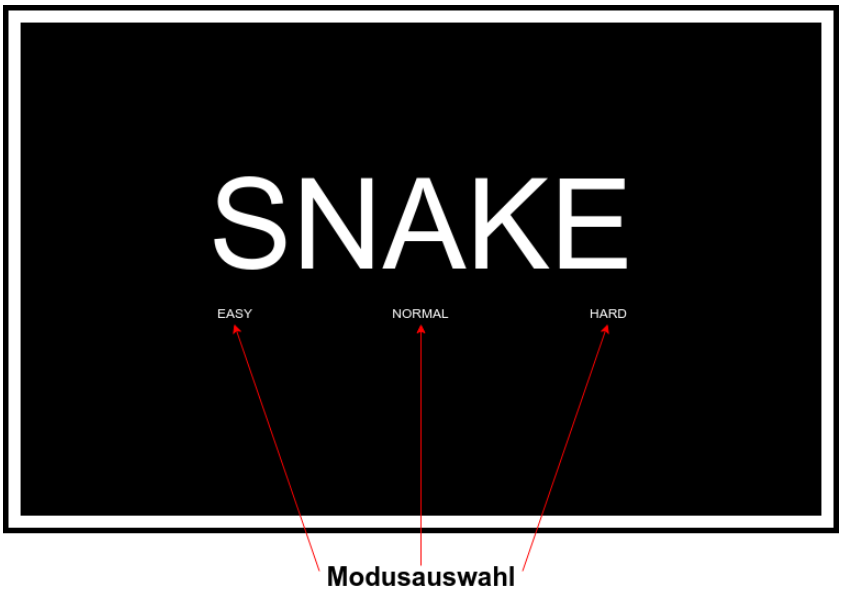

Abbildung 3: Startbildschirm

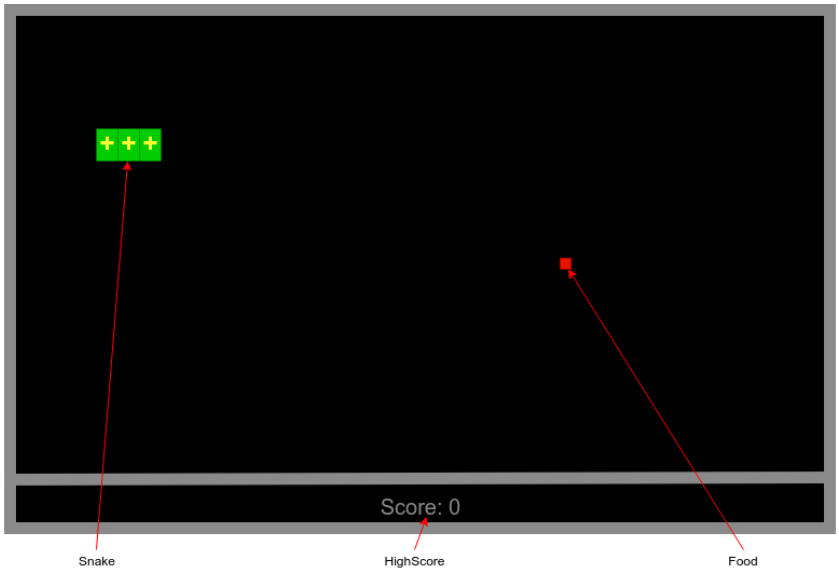

Abbildung 4: Spielbildschirm

Abbildung 5: Bildschirm bei erfolgreichem Spielende

Abbildung 6: Bildschirm bei verlorener Runde

3 Programmarchitektur

Nach dem untenstehenden Diagramm ist die Architektur aufgebaut:

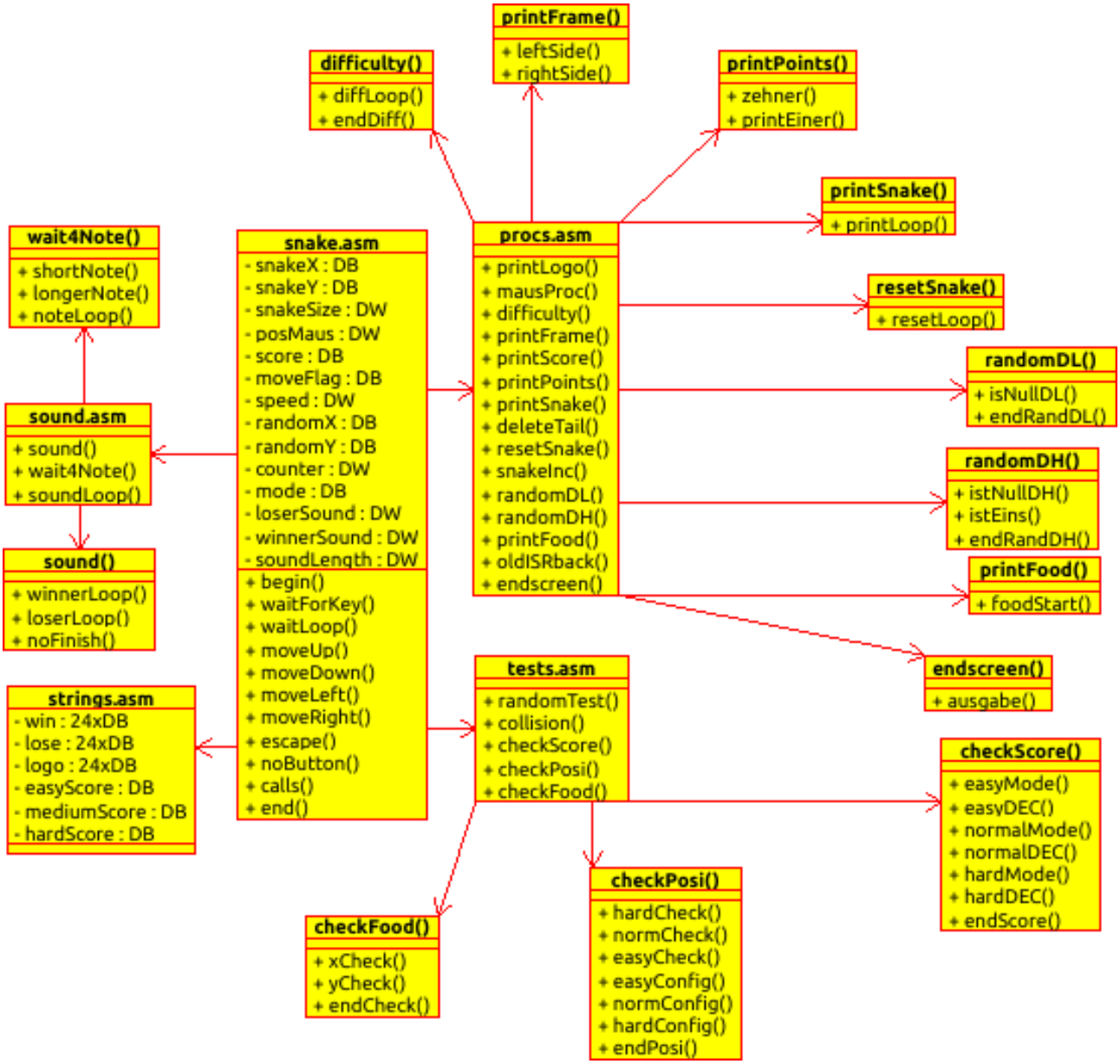

Abbildung 7: Programmstruktur

Es gibt eine snake.asm Datei, in der die Steuerung des Rundenablaufs und die Steuerung des Programms an sich. Dabei nutzt das Hauptprogramm andere ASM-Dateien, in denen verschiedene Prozeduren für die Tonausgabe und die einzelnen Bauteile für den Spielablauf enthalten sind. In einer weiteren Datei sind die Zeichenketten, für den Startbildschirm und für die zwei möglichen Endbildschirme gespeichert.

4 Tests

4.1 Random Numbergenerator Test 1

Die Test der Zufallszahlen würde in einer Tabelle festgehalten.
weshalb wir uns entscheiden Wie man an der Tabelle erkennen kann wiederholen sich immer wieder diesel-

Difficulty	Testnummer	Testergebnis	Vorkommen
Easy	1	55	1
	2	78	6
	3	45	4
	4	56	4
	5	89	4
	6	56	
	7	67	3
	8	92	1
	9	12	4
	10	78	
Normal	11	33	
	12	45	
	13	78	
	14	67	
	15	12	
	16	67	
	17	78	
	18	45	
	19	56	
	20	12	
Hard	21	22	1
	22	89	
	23	78	
	24	45	
	25	78	
	26	89	
	27	56	
	28	89	
	29	12	
	30	23	1

Abbildung 8: RNG Test 1

ben Zahlen in anderer Reihenfolge. Es handelt sich also hierbei nur um einen Pseudozufallszahlengenerator (PRNG). Am Anfang kommen (meist) die selben beiden Zahlen raus was daran liegt, dass es durch die ”sound” - Prozedur keine Verzögerung gibt, welche man aber brauch, damit andere Zahlen rauskommen. Der Unterschied durch die ”sound” - Prozedur ist auch sehr gering und berechenbar. Im Allgemeinen funktionieren die Zahlen, aber da wir bei 30 Tests 10 Unterschiedliche Zahlen bekommen reicht dies aus, dass das Zeichnen des Futters einigermaßen zufällig wirkt.

4.2 Random Numbergenerator Test 2

Nachdem wir in diesem Test das kleine Delay durch die ”sound” - Prozedur nicht mehr ausnutzen kommen immer 2-mal dieselben Zahlen raus, jedoch im Gesamten viel mehr unterschiedliche Zahlen, dadurch haben wir uns entschieden diese Methode beizubehalten, weil es so auch optisch schöner im Gesamtbild ist.

Difficulty	Testnummer	Testergebnis	Vorkommen
Easy	1	12	5
	2	99	5
	3	88	1
	4	99	
	5	66	3
	6	77	2
	7	12	
	8	33	4
	9	12	
	10	55	4
Normal	11	99	
	12	33	
	13	12	
	14	55	
	15	44	2
	16	12	
	17	66	
	18	55	
	19	22	4
	20	33	
Hard	21	99	
	22	22	
	23	33	
	24	22	
	25	55	
	26	22	
	27	99	
	28	44	
	29	66	
	30	77	

Abbildung 9: RNG Test 2

4.3 Collision Test

In dieser Prozedur wird überprüft ob der Schlangenkopf mit einem anderen ”+” Zeichen kollidiert und ist das der Fall, berührt sich die Schlange selber und das Spiel ist vorbei.

4.4 Check Score

Hier wird überprüft ob die vorgegebene Punktezahl erreicht ist mit der ein Spiel gewonnen wird oder die Geschwindigkeit erhöht wird.

4.5 Check Posi

Hier wird überprüft ob ein Buchstabe der Strings ”easy”, normal” oder ”hard” angeklickt wurde und die entsprechende Spielkonfiguration wird gesetzt.

4.6 Check Food

Hier wird getestet ob die Position des Schlangenkopfs mit der Position des Futters übereinstimmt. Sind die Koordinaten die selben, ertönt ein Ton, die Schlange wird verlängert und das Futter wird an einer neuen Stelle gezeichnet