Entwicklerdokumentation

Hauptprogrammerklärung („main()“)

In „main()“ sind Lokal die wichtigsten Daten des Spiels (z.B. posTail, posFruit und Score...) gespeichert. Beim Start wird der Info-Screen („informationGame()“) ausgegeben. Nachdem der Benutzer eine Taste gedrückt hat, wird das Spielfeld gezeichnet („gameField(…)“) und mit der Funktion setup() das Spiel initialisiert. Sobald der Benutzer eine Richtungstaste drückt (WASD/Pfeiltasten) bewegt sich die Schlange, ab da befindet sich das Spiel im Game Loop, bis der Spieler stirbt. Als Letztes wird der aktuelle Score mit den gespeicherten Scores verglichen sollte es ein neuer Highscore sein, landet man im Highscorescreen („highScoreScreen(…)“), ansonsten im endScreen („endScreen(…)“).

Modul Beschreibung:

Modul/Funktionsname	Modulzusammenfassung:	Hauptaufgabe:
->gotoxy() ->gameField() ->startScreen() ->informationGame() ->endScreen() ->highScoreScreen()	Anzeige/Konsolensteuerung	Spielfeld zeichnen, Textanzeigen verwalten,
setup()	Spielinitialisierung	Startwerte setzen (Schlange, Frucht, Score)
input() backgroundProcess()	Spielsteuerung & Logik	Bewegung der Schlange, Kollisionserkennung
compareScore() getHighestScore()	Highscore-Management	Score speichern und vergleichen

Hauptmodule:

Funktion	Funktionsbeschreibung
input()	Mit _kbhit() wird überprüft, ob eine Taste gedrückt wurde, mit _getch() wird dann bestimmt welche Taste gedrückt worden ist. Das ermöglicht die Richtung der Schlange durch die Variable direction zu verändern
backgroundProcess()	DIe Schlangenposition wird in dem zweidimensionalen Array posTail gespeichert, welches der Funktion

	übergeben wird. Die Funktion ändert je nach übergebener Richtung die Position der Schlange und gibt sie dann mithilfe gotoxy() aus. Zudem überprüft sie, ob die Schlange eine Frucht berührt, wenn ja, erhöht sie den score, die Schlangenlänge und generiert eine neue Frucht.
gotoxy()	Setzt den Cursor auf die gesetzte Position in der Konsole. Das ermöglicht einzelne Teile in der Konsole zu bearbeiten, ohne die gesamte Konsole zu löschen und wieder neuauszugeben zu müssen.
`CompareScore()`	Schreibt den aktuellen Score in ein Textdokument (snakeScores.txt). Liest alle Scores aus und übergibt sie getHighestScore() um den höchsten Score wiederzugeben.
`getHighestScore()`	Gibt den Höchsten Score aus dem Array der gespeichert Scores aus.

Verwendete Bibliotheken

benötigte Bibliotheken:	Nutzen
time.h	Um Zufallszahlen zu erzeugen mit der Funktion „rand()“. Wird benötigten um eine zufällige Position für die Frucht zu Erzeugen.
stdio.h conio.h	Benötig man, um zu prüfen, ob der Benutzer eine Taste gedrückt hat (mit der Funktion „_kbhit()“). Zudem beinhaltet die Bibliothek „conio.h“ die Funktion „_getch()“, in der man sehen kann welche Taste gedrückt wurde.
windows.h	Benötigt man, um die Position in der Konsole zu ändern, um kein neues Spielfeld auszugeben, sondern aktiv die Position der Schlange im Spielfeld zu ändern. Zudem benötigen wir den Befehl „Sleep(...)“ um mit einer zeitlichen Verzögerung die Position der Schlange zu verändern. Damit kann man dann auch die Geschwindigkeit der Schlange steuern.

- Die angestrebte Lösung ist speziell für Windows ausgelegt, da die verwendeten Bibliotheken (windows.h und conio.h) Windows-spezifisch sind. Als Compiler wird von uns der MinGW (GCC für Windows) verwendet.