Carme-Snake

# Übersicht

[bild, wie carme kit und computer zusammenarbeiten]

Der Computer ist für das einlesen der Pfeiltasten zuständig. Er sendet diese Befehle über ein eigen dafür entwickeltes Protokoll über EIA232 an das Carme Kit.

Das Carme Kit empfängt diese Daten und kontrolliert damit das Spiel. Es ist ausserdem auch für die graphische Darstellung zuständig.

## Computer

Die Software auf dem PC läuft in C mit der GUI Library von Ivo Oesch. Es wird jeweils eine Pfeiltaste gesendet, wenn sie unterschiedlich ist als die vorherige. Sobald die EXE gestartet wird, kann mit der ersten gedrückten Pfeiltaste das Spiel auf dem (laufenden) Carme Kit gestartet werden.

## Carme Kit

Die Software auf dem Carme Kit läuft grösstenteils in C. Das einzige Modul in Assembler wird der Interrupt Handler sein und die dazu notwendigen Initialisierungen. Die Software kontrolliert das Grafik-Display, empfängt die Pfeiltasten vom PC und steuert den Spielablauf (streut Futter, kontrolliert die Schlangenlänge, prüft auf Schlangen-Wandkollision und Schlangen-Schlangenkollision).

# Datenstrukturen

## Enumerations

### Direction dir

right = 0, up = 1, left = 2, down = 3

## Strutures

### Position pos

Diese Struktur enthält einen X- und einen Y-Wert, der eine Position auf dem Spielfeld repräsentiert.

## Globale Variablen

### Direction (dir)

Ist die momentane Richtung, in der sich die Schlange bewegt

### Food (pos)

Ist die momentane Position des Futters

### Snake

Ist ein Ringbuffer, welcher die Schlage repräsentiert. Dazu gehören folgende Variablen:

#### snake (pos[])

#### head (int)

#### tail (int)

#### size (int)

# Modulaufteilung

Ein Modul wird das PC-Programm sein, welches Pfeiltasten sammelt und weiterleitet. Dieses Modul ist gleichzeitig auch eine Ausführbare Datei.

Ein weiteres Modul ist der hardwarenahe UART-Interrupt Handler in Assembler. Er empfängt Pfeiltasten und aktualisiert die entsprechende Richtung in einer globalen Variablen (definiert in einer C-Datei).

Dann gibt es noch drei Module, welche sich um das eigentliche Spiel kümmern: Game-Logic, Snake-Controller und Graphics.

## PC\_Controller

## Uart\_Handler

## main

blabla

|  |  |  |
| --- | --- | --- |
| **init\_game** | | **main.c** |
| **Beschreibung** | Initializes the game with the game playground, the snake and a food piece. Also initializes all runtime variables | |
| **Input** | initial direction of the snake (direction) | |
| **Output** | - | |

|  |  |  |
| --- | --- | --- |
| **main** | | **main.c** |
| **Beschreibung** | main function | |
| **Input** | - | |
| **Output** | returns 0 | |

|  |  |  |
| --- | --- | --- |
| **randomize\_food** | | **game\_logic.c** |
| **Beschreibung** | generates a random position | |
| **Input** | - | |
| **Output** | the new food location (location) | |

|  |  |  |
| --- | --- | --- |
| **delay()** | | **game\_logic.c** |
| **Beschreibung** | delays the processing by a certain milliseconds | |
| **Input** | delay time (int) | |
| **Output** | - | |

## Snake\_Controller

|  |  |  |
| --- | --- | --- |
| **step\_forward** | | **snake\_controller.c** |
| **Beschreibung** | steps the snake one step forward. tells the executing function, what has happened while stepping forward: Enum “result” | |
| **Input** | - | |
| **Output** | result of stepping forward (enum result) | |

|  |  |  |
| --- | --- | --- |
| **eat** | | **snake\_controller.c** |
| **Beschreibung** | eats the food and increases the length of the snake | |
| **Input** | food (location) | |
| **Output** | - | |

|  |  |  |
| --- | --- | --- |
| **check\_wall\_collision** | | **snake\_controller.c** |
| **Beschreibung** | checks if the snake collided with the wall | |
| **Input** | field\_size (location) | |
| **Output** | true / false | |

|  |  |  |
| --- | --- | --- |
| **check\_snake\_collision** | | **snake\_controller.c** |
| **Beschreibung** | checks if the snake collided with itself | |
| **Input** | - | |
| **Output** | true / false | |

|  |  |  |
| --- | --- | --- |
| **check\_food\_collision** | | **snake\_controller.c** |
| **Beschreibung** | checks if the snake collided with a food piece | |
| **Input** | food (location) | |
| **Output** | true / false | |

## Graphics

|  |  |  |
| --- | --- | --- |
| **draw\_snake\_head** | | **graphics.c** |
| **Beschreibung** | draws a new snake piece on the head of the snake | |
| **Input** | head location (location) | |
| **Output** | - | |

|  |  |  |
| --- | --- | --- |
| **clear\_snake\_tail** | | **graphics.c** |
| **Beschreibung** | clears the snakes tail at the end of the snake | |
| **Input** | tail location (location) | |
| **Output** | - | |

|  |  |  |
| --- | --- | --- |
| **draw\_field** | | **graphics.c** |
| **Beschreibung** | draws a rectangle representing the playing field | |
| **Input** | - | |
| **Output** | - | |

|  |  |  |
| --- | --- | --- |
| **draw\_food** | | **graphics.c** |
| **Beschreibung** | draws a new food piece | |
| **Input** | food location (location) | |
| **Output** | - | |