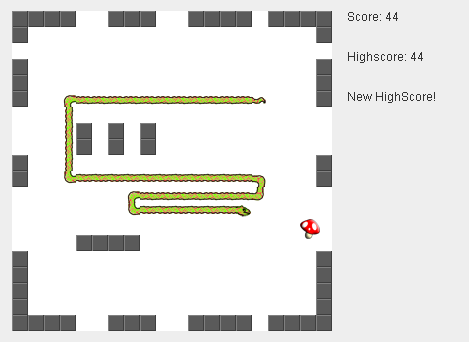
Systemdokumentation „Snake“



Von Marco Wettstein

Juli 2010

Inhaltsverzeichnis

[1.1. Vorgabe 3](#_Toc266454769)

[1.2. Erreichte Lösung 3](#_Toc266454770)

[2. Erfüllte Kriterien 4](#_Toc266454771)

[2.1. Musskriterien 4](#_Toc266454772)

[2.2. Zusätzliche „Features" 4](#_Toc266454773)

[3. Zeitlicher Aufwand 4](#_Toc266454774)

[4. Analyse 5](#_Toc266454775)

[4.1. Use-Case-Diagramme 5](#_Toc266454776)

[4.1.1. Einfaches Use-Case-Diagramm 5](#_Toc266454777)

[4.4.2. Erweitertes Use-Case-Diagramm 6](#_Toc266454778)

[4.2.3. Erweiterter Use-Case Entwurf 7](#_Toc266454779)

[4.2. Use-Case-Szenarios 8](#_Toc266454780)

[4.3. Klassendiagramm ohne Felder und Methoden 9](#_Toc266454781)

[4.4. Ablauf-Diagramm 10](#_Toc266454782)

[4.5. CRC-Cards 11](#_Toc266454783)

[4.6. Erweitertes Klassendiagramm mit Funktionen und Feldern 12](#_Toc266454784)

[4.7. Sequenzdiagramm 13](#_Toc266454785)

[5. Implementierung 14](#_Toc266454786)

[5.1. Klassendiagramm der Implementierung 14](#_Toc266454787)

[6. Systemtest 15](#_Toc266454788)

[7. User-Dokumentation 16](#_Toc266454789)

[7.1. Spielprinzip 16](#_Toc266454790)

[7.2. Spiel starten 16](#_Toc266454791)

[7.3. Steuerung 16](#_Toc266454792)

[7.4. Spielende 16](#_Toc266454793)

[7.5. Hinweise 16](#_Toc266454794)

[7.6. Der Karten-Editor 16](#_Toc266454795)

# Aufgabenstellung

## Vorgabe

Beim Start der Anwendung wird ein Fenster geöffnet. Das Fenster dient der Darstellung und Interaktion des Spiels. Sie werden Code erhalten der dieses Fenster für Sie erstellt. Ihre Aufgabe beschränkt sich auf die Spiellogik und die Darstellung der Szene auf dem Spielfeld.

Das Spielfeld hat eine Grösse von 320x320Pixel und ist in 'Zellen' von jeweils 16x16Pixeln unterteilt. Das sind also 20X20 Zellen.

Beim Start eines neuen Spiels soll eine Kurze Schlange (kopf, ein Zwischenelement, und Schwanz) erzeugt und an einer zufälligen Position mit einer zufälligen Richtung im Spielfeld platziert werden. Zusätzlich wird eine Futterportion an einer zufälligen Position im Spielfeld platziert. Sobald das Spiel gestartet wird, muss die Schlange periodisch weiterbewegt werden. Die Schlange bewegt sich dabei in einem Raster. Pro Zeitintervall bewegt sich die Schlange um jeweils eine Zelle weiter. Der Benutzer kann durch drücken der Tasten die Bewegungsrichtung der Schlange ändern. Die gewählte Richtung wird beibehalten bis eine erneute Richtungsänderung vorgenommen wird.

Kollidiert die Schlange mit einem Hindernis (Z.B. der Spielfeldgrenze) oder sich selber, so ist das Spiel beendet. Kollidiert die Schlange mit einer Futterportion, so wächst sie um eine Einheit und eine neue Futterportion muss auf dem Spielfeld Platziert werden.

## Erreichte Lösung

Die Vorgaben wurden eingehalten, mit der unten aufgeführten Ausnahme

Das es nach Aufgabenstellung erlaubt war, weitere „Features“ einzubauen, habe ich meiner Phantasie ein wenig freien Lauf gelassen.

* Für die Darstellung von Schlange und Essen wurden Grafiken verwendet, die Schlange und das Essen, welches durch einen Pilz dargestellt wird, sind dabei Zeichnungen von Stefan Bakocs, welche ich eingefärbt habe.
* Das Essen, welches zufällig auf dem Spielfeld plaziert wird, hat eine zufälligen „Nährwert“ zwischen 1 und 5 Einheiten. Je grösser der Nährwert, desto grösser wird das Essen, resp. der Pilz dargestellt, desto mehr Punkte gibt der Pilz und desto mehr wächst auch die Schlange.
* Das verlassen des eigentlichen Spielfeldes führt nicht zum „Game Over“. Statt dessen erscheint die Schlange auf der anderen Seite des Spielfeldes.
* Statt des vorgegeben Labyrinthes kann der Spieler ein eigens Labyrinth laden. Die Labyrinthe sind in einem Texteditor erstellbar. Der Einfachheit halber wird die Schlange nicht wie in der Vorgabe verlangt zufällig plaziert, sondern immer an der gleichen Position gesetzt.
* Die erreichten Punkte werden gespeichert, falls es sich um einen neuen „Highscore“ handelt.

# Erfüllte Kriterien

## Musskriterien

|  |  |
| --- | --- |
| **Kriterien** | **Entscheid** |
| Spiel startet | Ok |
| Essen erscheint zufällig | Ok |
| Die Karte hat Hindernisse | Ok |
| Kurze Schlange bei Spielstart | Ok |
| Position und Richtung der Schlange zufällig | Nicht erfüllt, wegen Karten-Feature |
| Schlange bewegt sich automatisch | Ok |
| Schlange ist über Tastatur steuerbar | Ok |
| Kollision mit Hindernissen wird überprüft | Ok |
| Kollision mit Schlange wird überprüft | Ok |
| Schlange wächst bei Nahrungsaufnahme | Ok |
| Bei Kollsion mit Hindernis oder Schlange ist das Spiel vorbei | Ok |

## Zusätzliche „Features"

|  |
| --- |
| **Features** |
| Punkte werden gezählt bei Futteraufnahme |
| Highscore wird angezeigt und in Textdatei gespeichert |
| eigene Grafiken für Schlange |
| eigene Grafiken für Essen |
| Verschiedene Nahrungstypen |
| Eigenes Labyrinth verwendbar |

# Zeitlicher Aufwand

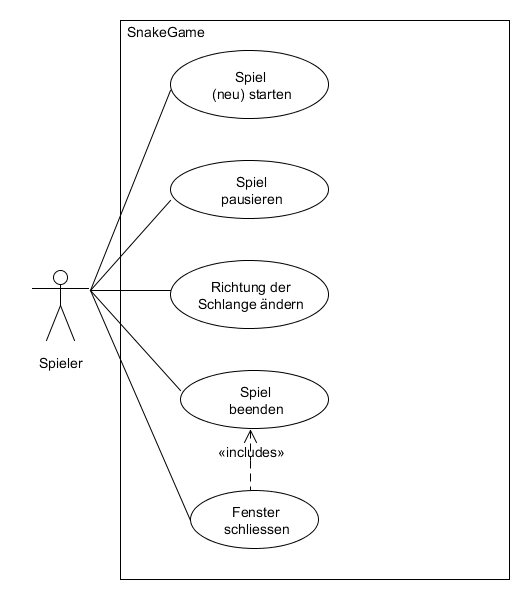
|  |  |  |
| --- | --- | --- |
| **Arbeit** | **Geplant** | **Tatsächlich** |
| Analyse & Planung | 1 Tage | 1 Tage |
| Implementierung | 1 Tag | 1.5 Tage |
| Tests | 0.5 Tage | 0.5 Tage |
| Dokumentation | 1 Tag | 1 Tage |
|  |  |  |
| **Total** | **3.5 Tage** | **4 Tage** |

# Analyse

## Use-Case-Diagramme

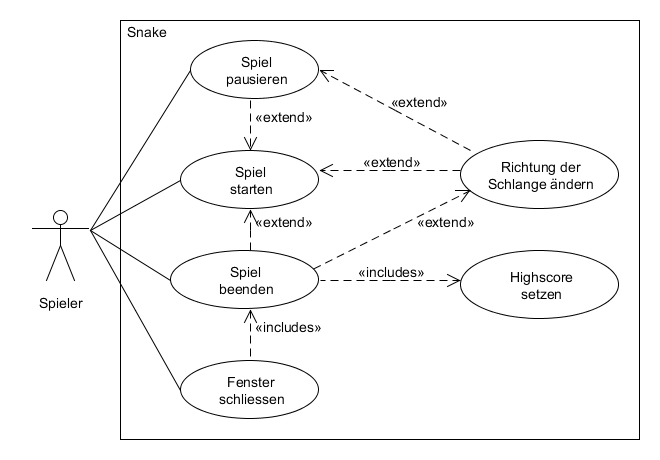
### 4.1.1. Einfaches Use-Case-Diagramm

Die Anforderungen ergeben für die Use-Case-Szenarien im groben folgendes Diagramm:



### 4.4.2. Erweitertes Use-Case-Diagramm

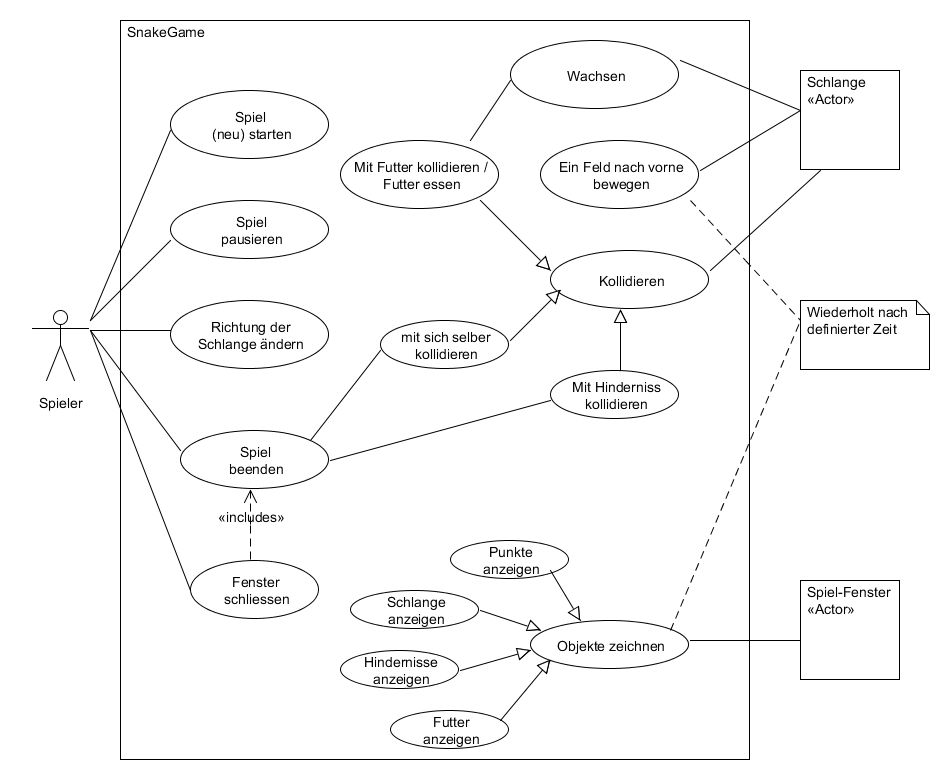
Eine Erweiterung des Obigen Use-Case-Diagramm stellt folgendes Diagramm dar:



### 4.2.3. Erweiterter Use-Case Entwurf

Wir haben einen weiteren Entwurf erstellt, allerdings werden innere Systemzusammenhänge ebenfalls gezeigt, was dem Use-Case-Prinzip widerspricht.

Der Vollständigkeit halber wird dieser hier gezeigt



## Use-Case-Szenarios

Spiel starten

Beschreibung:

- Spielfeld wird generiert mit 20 x 20 Feldern

- Hindernisse werden platziert

- Punkte werden auf 0 gesetzt

- Schlange wird erzeugt bestehend aus 3 glieder

- Schlange wird auf ein zufälligs Spielfeld gesetzt

- Schlange blickt in eine zufällige Richtung

- eine Futterportion wird zufällig auf dem Feld platziert

- Zeitticker wird gesetzt

- Zeitticker wird gestartet

Auslöser:

- Spiel starten wird aufgerufen

Vorbedingung:

- Anwendung ist gestartet

Akteur:

- Spieler

Nachbedingunen:

- Schlange bewegt sich

Spiel starten

Beschreibung:

- Zeitticker wird gestoppt

Auslöser:

- Spieler drückt auf Pause

Vorbedingung:

- Anwendung ist gestartet, Spiel ist gestartet

Akteur:

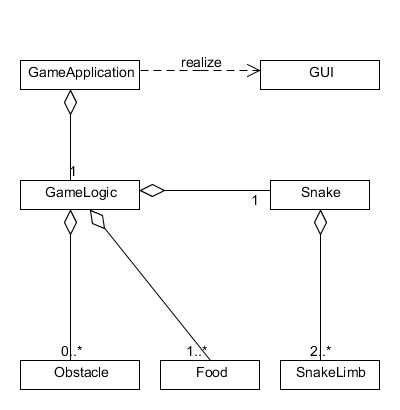
- Spieler

Nachbedingunen:

- Spiel ist pausiert

## Klassendiagramm ohne Felder und Methoden

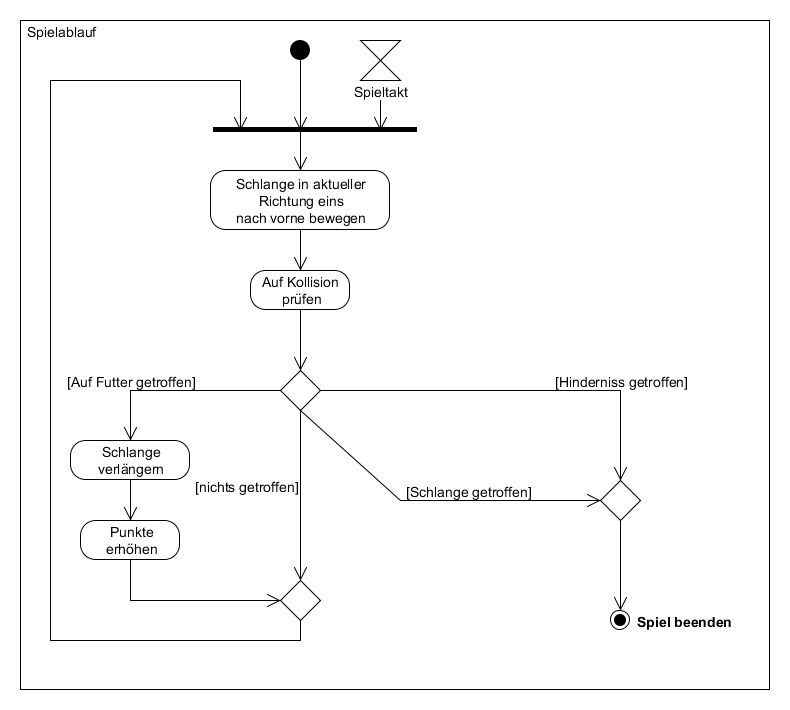
Durch Noun-Extraction der Use-Cases und Grundlegenden Funktionsüberlegungen ergibt sich folgendes Klassendiagramm:



Das Interface „GUI“ ist hierbei das GUI, welches uns zur Verfügung gestellt wird.

## Ablauf-Diagramm

Aus Überlegungen zum Grundlegenden Ablauf eines Spiel-Ticks ergibt sich folgendes Sequenzdiagramm:



## CRC-Cards

Um die Grundlegenden Funktionen der Klassen zu erhalten, haben eine CRC-Session gemacht. Dies ergab folgende CR-Cards:

|  |  |
| --- | --- |
| **Klasse „Snake“** | |
| Verantwortlich für:   * Eigene Bewegung * Eigenes Wachstum * Prüfen, ob die sich Glieder der Schlange an einer bestimmten Position befinden * Eigenes Wachstum * Darstellung von sich selbst | Zusammenarbeit mit:   * GameLogic * SnakeLimb |

|  |  |
| --- | --- |
| **Klasse „Obstacle“** | |
| Verantwortlich für:   * Prüfen, ob sich das Hinderniss an einer bestimmten Position befindet * Darstellung von sich selbst | Zusammenarbeit mit:   * GameLogic |

|  |  |
| --- | --- |
| **Klasse „Food“** | |
| Verantwortlich für:   * Wissen, wie viele Punkte das Essen wert ist * Prüfen, ob sich das Essen an einer bestimmten Position befindet * Darstellung von sich selbst | Zusammenarbeit mit:   * GameLogic * Snake |

|  |  |
| --- | --- |
| **Klasse „SnakeLimb“** | |
| Verantwortlich für:   * Prüfen ob sich das Schlangensegement an einer bestimmten Position befindet * Darstellung von sich selbst | Zusammenarbeit mit:   * Snake |

|  |  |
| --- | --- |
| **Klasse „Snake“** | |
| Verantwortlich für:   * Eigene Bewegung * Eigenes Wachstum * Prüfen, ob sie sich selbst beisst * Eigenes Wachstum * Darstellung von sich selbst | Zusammenarbeit mit:   * GameLogic |

|  |  |
| --- | --- |
| **Klasse „GameApplication“** | |
| Verantwortlich für:   * Tastendrücke abfangen * Menu abfragen * Zeittick an GameLogic geben * Spiel initialisieren * Punkte von GameLogic entgegen nehmen * Die GameLogic über das GUI zeichnen lassen | Zusammenarbeit mit:   * GameLogic * GUI-Interface |

|  |  |
| --- | --- |
| **Klasse „GameLogic“** | |
| Verantwortlich für:   * Hindernisse setzten * Schlange setzten * Schlange bewegen * Schlange auf Kollision mit sich selbst, den Hindernissen und der Nahrung überprüfen * Pro Zeittick Zug ausführen * Punkte festhalten * Schlange, Essen und Hindernisse zeichnen | Zusammenarbeit mit:   * GameApplication * Snake * Obstacle * Food |

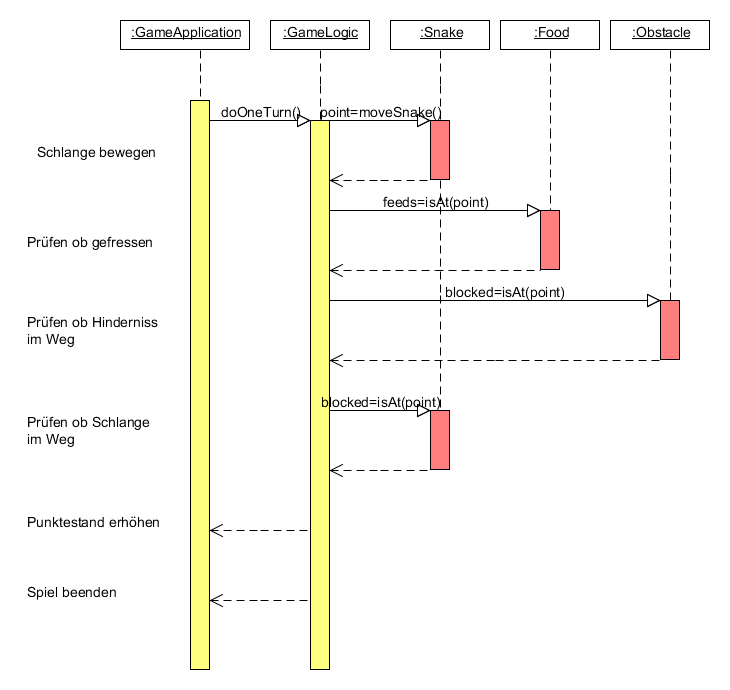
## Erweitertes Klassendiagramm mit Funktionen und Feldern

Aus den CRC-Cards hat sich folgendes, provisorisches UML-Klassendiagramm ergeben:

## D:\Desktop\classDiagramWithFunctionAndFields.png

## Sequenzdiagramm

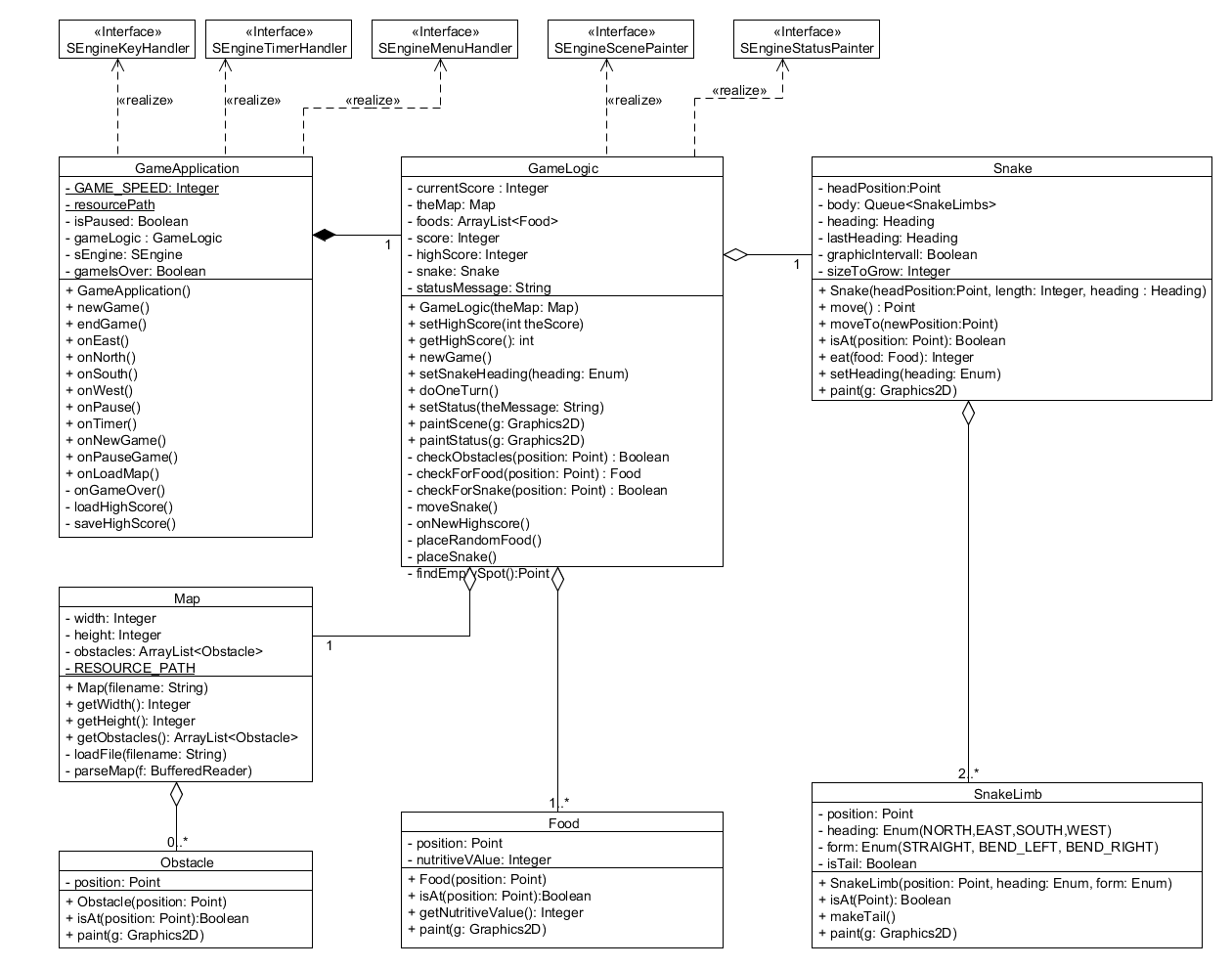
Der Grobe Spielablauf wird durch folgendes Sequenzdiagramm beschrieben:



# Implementierung

## Klassendiagramm der Implementierung

Die Implementierung wird durch folgendes Klassendiagramm beschrieben:



# Systemtest

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Vorgehen** | **Erwartung** | **Entscheid** |
| 1 | Applikation starten | Spielfenster erscheint,  Schlange erscheint,  Karte mit den Hindernissen wird geladen,  Nahrung erscheint,  Punkte werden auf 0 gesetzt,  Highscore wird geladen,  Spiel ist pausiert | ok |
| 2 | Pause drücken oder Menu „Game -> Pause Game“ | Spiel startet, Schlange bewegt sich vorwärts | ok |
| 3 | Nach oben drücken | Schlange wechselt richtung nach oben | ok |
| 4 | Nach links drücken | Schlange wechselt Richtung nach links | ok |
| 5 | Nach unten drücken | Schlange wechselt Richtung nach unten | ok |
| 6 | Nach rechts drücken | Schlange wechselt Richtung nach rechts | ok |
| 7 | Pause drücken | Spiel hält an | ok |
| 8 | nochmals Pause drücken | Spiel läuft weiter | ok |
| 9 | Auf Feld mit Nahrung stossen | Punkte werden je nach Grösse der Nahrung erhöht, Schlange wächst um entsprechend viele Glieder | ok |
| 10 | Schlange über Kartenrand hinaus bewegen | Schlange erscheint auf der anderen Seite wieder | ok |
| 11 | N drücken oder Menu  „Game->New Game“ | Neues Spiel wird gestartet,  Punkte werden zurückgesetzt, Schlange wird zurückgesetzt,  Nahrung erscheint an neuer Position, Spiel ist wieder pausiert | ok |
| 12 | Schlange auf Hinderniss bewegen | Spiel wird gestoppt, Meldung „Game Over“ erscheint.  Spiel lässt sich nur mit Taste N oder Menu „Game->New Game“ starten, oder mit Menu „Game->Quit“ beenden | ok |
| 13 | Neues Spiel starten und Schlange auf sich selbst kollidieren lassen | Spiel wird gestoppt, Meldung „Game Over“ erscheint.  Spiel lässt sich nur mit Taste N oder Menu „Game->New Game“ starten, oder mit Menu „Game->Quit“ beenden | ok |
| 14 | Neues Spiel starten und Spielen bis neuer Highscore erreicht | Punkte sind nun Highscore, Highscore wird gespeichert | ok |

# User-Dokumentation

## Spielprinzip

Der Spieler steuert Schlange in einem Labyrinth, welche sich selbstständig fortbewegt, die Richtung aber vom Spieler bestimmt werden kann.

Der Spieler soll versuchen, durch Essen von Nahrung möglichst viele Punkte zu sammeln, was jedoch mit fortschreitendem Spiel immer schwieriger wird, da die Schlange mit jeder Nahrung, die sie zu sich genommen hat länger wird.

Der Spieler kann das Labyrinth nach eigenen Wünschen abändern

## Spiel starten

Das Spiel wird durch Doppelklick auf snake.jar gestartet.

## Steuerung

Pfeiltasten: Schlange in entsprechende Richtung bewegen

Taste N: Neues Spiel starten. Achtung: der Fortschritt geht dadurch verloren!

Taste P: Spiel pausieren. Durch erneutes Drücken von P wird das Spiel fortgesetzt. Hinweis: Das Spiel startet immer pausiert.

Neues Spiel starten, sowie Spiel pausieren kann auch über das Game-Menu errreicht werden. Ebenso kann so das Programm verlassen werden.

## Spielende

Das Spiel endet wenn die Schlange entweder ein Hindernis oder sich selbst berührt.

## Hinweise

Wo kein Hindernis ist kann man die Karte am Rand verlassen, die Schlange erscheint dann an der gegenüberliegenden Stelle erneut

## Der Karten-Editor

Unter dem Menu „Game 🡪 Load Map“ kann eine eigene Karte geladen werden. Diese muss als Textdatei gespeichert werden und die Endung „.map“ haben.

In der Textdatei stellen „o“ (Kleines O) die Hindernisse dar. Es dürfen maximal 20 Zeichen pro Zeile und 20 Zeilen gesetzt sein. Abstände können mit Leerzeichen gesetzt werden.

Die Schlange startet immer auf der dritten Zeile beginnend mit der dritten Spalte und ist 3 Zeichen lang. Diese 3 Felder müssen also frei bleiben (und die nachfolgenden aus Gründen der Fairness ebenfalls.