**Projektarbeit Informatik**

Softwaredokumentation   
LaserChess V1.2



Website: <http://stocyr.github.com/LaserChess/>

Dozent: Ivo Oesch

Autoren: Marcel Bärtschi, Jascha Haldemann, Nicola Käser, Cyril Stoller

2 Semester 2012

**Inhaltsverzeichnis**

1 Aufgabenstellung 1-1

2 Richtlinien 2-2

3 Analyse 3-3

4 Design 4-4

4.1 Grobdesign 4-4

4.2 Detaildesign 4-4

4.3 Modulbeschreibungen 4-4

4.4 Datenstrukturen 4-4

5 Implementation 5-5

6 Test 6-6

7 Anhang 7-7

# Einleitung

Dieses Dokument dokumentiert das Softwareprojekt „LaserChess“. Die genaue Beschreibung der Aufgabenstellung ist im Dokument „Projektdokumentation“ enhalten. Das Pflichtenheft, welches im Voraus für dieses Projekt erstellt wurde und alle Grundideen enthält, ist im Anhang enthalten.

# Richtlinien

Für dieses Projekt gelten folgende Richtlinien:

Jedes Modul besteht grundsätzlich aus zwei Dateien, einer Schnittstellendefinitionsdatei (Headerdatei .h-File) und einer Implementationsdatei (.c-File). Die **main()**-Fuktion ist in der Haupt-C-Datei LaserChess.C enthalten. Alle projektweite Typendefinitionen sind in der Datei LaserChess.H enthalten. Die Namen der Header und der Implementationsdatei entsprechen jeweils dem Modulnamen, soweit möglich und sinnvoll.

Der Selbstschutz in den Headerdateien erfolgt in der Form

**#ifndef NAME**

**#define NAME**

**<Schnittstellendefinition>**

**#endif**

Wobei NAME wie folgt aufgebaut ist **DATEINAME\_H**. Jedes Modul ist mit einem Modulheader ausgestattet, jeweils in der Schnittstellen und der Implementationsdatei. Der Header enthält den Modulnamen, den Dateinamen, einen kurzen Funktionsbeschrieb zum Modul, die Namen aller zur Verfügung gestellten Funktionen und den Namen des ursprünglichen Autors. Die History jedes Files kann auf <https://github.com/stocyr/LaserChess/tree/master/src> angesehen werden. Zuerst klickt man auf den Namen des gewünschten Files, dann auf den Button **Blame** oben rechts:



Jede Funktion ist mit einem Funktionsheader ausgestattet. Der Header enthält den Funktionsnamen, einen kurzen Funktionsbeschrieb, die Namen und Funktion der Argumente, eine Beschreibung des Rückgabewertes und den Namen des Autors. Funktionen sollen klar definierte Aufgaben haben. Namen von Bezeichnern sollen Aussagekräftig sein, und die Funktion des entsprechenden Objektes erklären oder andeuten.

Modul- und Funktionsheader werden in englischer Sprache geschrieben. Kommentare im Code drin jedoch sind auf Deutsch verfasst und immer von der Art **// <Kommentar>** damit ganze Codeblöcke zu debug-Zwecken mit **/\* ... \*/** auskommentiert werden können. Der Code ist grundsätzlich mit sinnvollem und aussagekräftigem Kommentar zu versehen.

Makros (**#define**) werden grundsätzlich in Grossbuchstaben geschrieben, zu Strukturierung können Underscores verwendet werden. Beispiele: **PI**, **MAXIMAL\_FIELD\_WIDTH**

Enums werden grundsätzlich mit einem grossen Anfangsbuchstaben geschrieben. Structs werden grundsätzlich kleingeschrieben. Variablen werden grundsätzlich kleingeschriebenen und nötigenfalls mit Underscore "\_" in Wörter aufgeteilt. Funktionen werden grundsätzlich kleingeschriebenen und nötigenfalls mit Underscore "\_" in Wörter aufgeteilt.

Für jedes Modul ist grundsätzlich ein Programmierer verantwortlich. Dieser ist für die Sauberkeit und das Nachführen des Modulheaders zuständig. Andere Programmierer dürfen jedoch ebenfalls Änderungen in diesem Modul vornehmen.

# Analyse

## Pflichtenheft

Im Pflichtenheft, welches im Anhang vorliegt, wurde schon relativ viel Analysearbeit gemacht:

* die Regeln des Spiels wurden definiert,
* das Spielfeld festgelegt,
* die Spielfiguren inklusive einer möglichen Datenstruktur beschrieben,
* eine Menu-Führung definiert,
* die möglichen Spielzüge definiert,
* das Spielfeld als map-Datenbank definiert und
* Details der Laserfunktionalität vorgestellt.

# Design

## Grobdesign

Es werden vier verschiedene Module erstellt:

* Hauptmodul „LaserChess“
* Logikmodul „Logik“
* Spielmodul „Spiel“ und
* Grafikmodul „Grafik“

## Detaildesign



## Modulbeschreibungen

### Grafikfunktionen

Das Grafikmodul stellt der Software die Funktionen für die grafische Umsetzung zur Verfügung. Alles was grafisch ausgegeben wird, wird von diesem Modul erledigt. Für das Modul war vor allem Nicolas Käser zuständig, Jascha Haldemann hat ihn gegen Ende des Projekts noch unterstützt.

Im Folgenden werden die Schnittstellen der Funktionen in diesem Modul mittles CRC-Karten aufgelistet:

|  |  |  |
| --- | --- | --- |
| **DrawTransformedImage** | | **Grafik.c** |
| **Beschreibung** | Draws the given image scaled and rotated at the given position into the current image. | |
| **Input** | imageId: Handle of image to draw x, y position to draw image at…  scalex: Scalingfactor for x axis (float value)  scaley: Scalingfactor for y axis (float value)  Angle: Angle to rotate Image (in rad) | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **draw\_sharp\_empty\_rectangle** | | **Grafik.c** |
| **Beschreibung** | Draws empty rectangle with sharp edges. | |
| **Input** | x and y koord. as windowskoord.,  int Width, int Height, ColorType Color, int LineWidth | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **pixel\_to\_map** | | **Grafik.c** |
| **Beschreibung** | Convert windowskoord. to map position. | |
| **Input** | x and y as windowskoord. | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **map\_to\_pixel** | | **Grafik.c** |
| **Beschreibung** | Convert mappositon to windowskoord. (Point upper left) | |
| **Input** | x and y as map position | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **draw\_playground** | | **Grafik.c** |
| **Beschreibung** | Draws a playground (field and lines) | |
| **Input** |  | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **scale\_handler** | | **Grafik.c** |
| **Beschreibung** | Returns the percentage for scaling the image to field size. | |
| **Input** | image\_ID, a valid ID of a loaded image file | |
| **Output** | size scale,  x- and y-scale factor in percentage of the field size | |

|  |  |  |
| --- | --- | --- |
| **draw\_focus** | | **Grafik.c** |
| **Beschreibung** | Draws a focus at the selected field. | |
| **Input** | x and y as map position | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **draw\_rot\_focus** | | **Grafik.c** |
| **Beschreibung** | Draws a rotation-image on the selected field | |
| **Input** | x and y as map position | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **draw\_empty\_field** | | **Modul** |
| **Beschreibung** | Deletes the selected field (reset). | |
| **Input** | x and y as map position | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **draw\_half\_laser** | | **Grafik.c** |
| **Beschreibung** | Help-function for draw\_laser and draw\_angled\_laser.  Draws half of the laser in the selected field  (v1.1: Laser glows). | |
| **Input** | x and y as map position, direction | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **draw\_laser** | | **Grafik.c** |
| **Beschreibung** | Draws the laser in the selected field (2x draw\_half\_laser). | |
| **Input** | x and y as map position, direction | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **draw\_angled\_laser** | | **Grafik.c** |
| **Beschreibung** | Draws the angled laser in the selected field (2x draw\_half\_laser). | |
| **Input** | x and y as map position, direction, angle (-1 right, 1 left) | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **destroy\_images** | | **Grafik.c** |
| **Beschreibung** | Deletes with init\_images() loaded images from memory | |
| **Input** |  | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **init\_images** | | **Grafik.c** |
| **Beschreibung** | Loads images from files into memory | |
| **Input** | Blup… | |
| **Output** | Blup… | |

|  |  |  |
| --- | --- | --- |
| **draw\_figure** | | **Grafik.c** |
| **Beschreibung** | Draws figure at its location with its rotation/direction | |
| **Input** | pawn \*figure (figurepointer) | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **draw\_figure\_destroyed** | | **Grafik.c** |
| **Beschreibung** | Draws/animates the destruction of a mirror.  (V1.0, it only draws an empty field  (V1.1, "Melting"-animation with rectangles  (V1.2, offset increases always 1 pixel, not laser width  (V1.3, New animation, with glow) | |
| **Input** | pawn \*figure | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **draw\_invert\_colors** | | **Grafik.c** |
| **Beschreibung** | Inverts the colours of the defined part. | |
| **Input** | x and y as windowskoord. for start position; width and height for the size | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **draw\_winner\_text** | | **Grafik.c** |
| **Beschreibung** | Writes winner text on screen. | |
| **Input** | pawn \*hit\_king | |
| **Output** |  | |

### Logikfunktionen

Das Logikmodul enthält vor allem die Logik des Laserabschusses. Ausserdem enthält sie verschiede Helper-Tools und Konvertierungsroutinen welche von den anderen Modulen genutzt werden. Für das Modul war vor allem Cyril Stoller zuständig.

Im Folgenden werden die Schnittstellen der Funktionen in diesem Modul mittles CRC-Karten aufgelistet:

|  |  |  |
| --- | --- | --- |
| **laser** | | **Logik.c** |
| **Beschreibung** | Draws the laser from the cannon across the whole playground and calls all the other functions handling figure behaviour. | |
| **Input** | Receives the field from which the laser shoot is done. This field is not painted with laser anymore, but the field NEXT to it, whose direction from the field is specified with dir. | |
| **Output** | If a wall or cannon was hit, or the laser passes out of the playground, return 0. If a king was hit: -1 for player\_red, -2 for player\_blue.  If a mirror was hit: +1 for player\_red, +2 for player\_blue.  (In case of a splitter being hit: then two laser paths are generated and the return value is the one with the higher priority (descending order): king, mirror, wall / cannon) | |

|  |  |  |
| --- | --- | --- |
| **is\_inside\_map** | | **Logik.c** |
| **Beschreibung** | Checks if the given coordinates are inside the array. | |
| **Input** | Given coordinates (map position) | |
| **Output** | If inside map (means, inside the range [0 - 7][0 - 5], then it returns 1.  Otherwise it returns 0 | |

|  |  |  |
| --- | --- | --- |
| **is\_figure** | | **Logik.c** |
| **Beschreibung** | Checks if the given coordinates (map position) contains a figure. | |
| **Input** | x and y as map position | |
| **Output** | If there is a figure, return 1.  If it’s an empty field, return 0  (A wall is threatened as a figure). | |

|  |  |  |
| --- | --- | --- |
| **move\_figure** | | **Logik.c** |
| **Beschreibung** | Moves a figure to the given location. | |
| **Input** | figure pointer, new playground location | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **destroy\_figure** | | **Logik.c** |
| **Beschreibung** | Destroys a figure (deletes it from the map array). | |
| **Input** | pawn \*figure | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **mouseclick\_to\_map** | | **Logik.c** |
| **Beschreibung** | Get Mouse-Clicks and returns the map position. | |
| **Input** |  | |
| **Output** | Returns location struct, of the field who was hit or ERROR when the click was beyond the map or there was no click. | |

|  |  |  |
| --- | --- | --- |
| **path\_handler** | | **Logik.c** |
| **Beschreibung** | Combines the two strings path and file after checking if there's enough memory available. | |
| **Input** | const char path[] - String with the path of file  char file[] - String with the filename | |
| **Output** | returns string with the complete path | |

|  |  |  |
| --- | --- | --- |
| **play\_sound** | | **Logik.c** |
| **Beschreibung** | Plays the sound of chosen enumeration. | |
| **Input** | Enum: Laser, Reflexion, Destruction, Victory, Ignore, Intro, Music, Bling, Bell. | |
| **Output** |  | |

### Spielfunktionen

|  |  |  |
| --- | --- | --- |
| **create\_focus** | | **Spiel.c** |
| **Beschreibung** | Draws a green Background on all free Fields around the selected figure. | |
| **Input** | location struct (x-y-coordinates of selected figure) | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **clear\_focus** | | **Spiel.c** |
| **Beschreibung** | Draws an empty field to all marked fields. | |
| **Input** | location struct (x-y-coordinates of selected figure) | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **spiel** | | **Spiel.c** |
| **Beschreibung** | Handles the game: Treats the mouse inputs, execute laser(), displays winner, close graphics. | |
| **Input** | pawn \*figure (used for cannon-position) | |
| **Output** |  | |

### Mainfunktionen (Laserchess)

|  |  |  |
| --- | --- | --- |
| **create\_figures** | | **LaserChess.c** |
| **Beschreibung** | Initializes all figures from a received figure array.  Sets figures to the default map-position.  Currently initializes 14 figures (Optional: Splitter not defined in this version). | |
| **Input** | Pointer to the original figure array in the main-procedure | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **menu** | | **LaserChess.c** |
| **Beschreibung** | User can choose between the modes: ‘NORMAL’, ‘SETMODE’ and ‘quit the game’. | |
| **Input** |  | |
| **Output** | Enum Mode | |

|  |  |  |
| --- | --- | --- |
| **Set\_figure\_positions** | | **LaserChess.c** |
| **Beschreibung** | The player can set his pawn freely on the map.  The figures in the array are sorted by color.  To toggle the player: i/2 for red and (i/2)+7 for blue. | |
| **Input** | Pointer to Array of all the figures. | |
| **Output** | Returns -1 if exit button is pressed, otherwise 0. | |

|  |  |  |
| --- | --- | --- |
| **init\_game** | | **LaserChess.c** |
| **Beschreibung** | Initializes the game.  Calls the graphic-functions and places the figures. | |
| **Input** | Array of all the figures, play mode (to decide whether to place all the figures, to initialized state or let the users place them alternating). | |
| **Output** | If wild failure appears, returns 0, otherwise returns 1. | |

|  |  |  |
| --- | --- | --- |
| **clear\_map\_array** | | **LaserChess.c** |
| **Beschreibung** | Clears the map array (writes all positions to NULL). | |
| **Input** |  | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **argument\_handler** | | **LaserChess.c** |
| **Beschreibung** | Reads the start-arguments. If EXE was started by opening a map file, it tries to load and start a game.  If there are start-variables defined, they will be set.  Unknown arguments are printed to screen. | |
| **Input** | int argn, number of arguments; char\* args[], arguments; pawn \*figure,  Figure-array needed to start a game. | |
| **Output** |  | |

|  |  |  |
| --- | --- | --- |
| **gfxmain** | | **LaserChess.c** |
| **Beschreibung** | Uber-main function. Will be called FIRST! | |
| **Input** | System console call parameters. (OS specific) | |
| **Output** |  | |

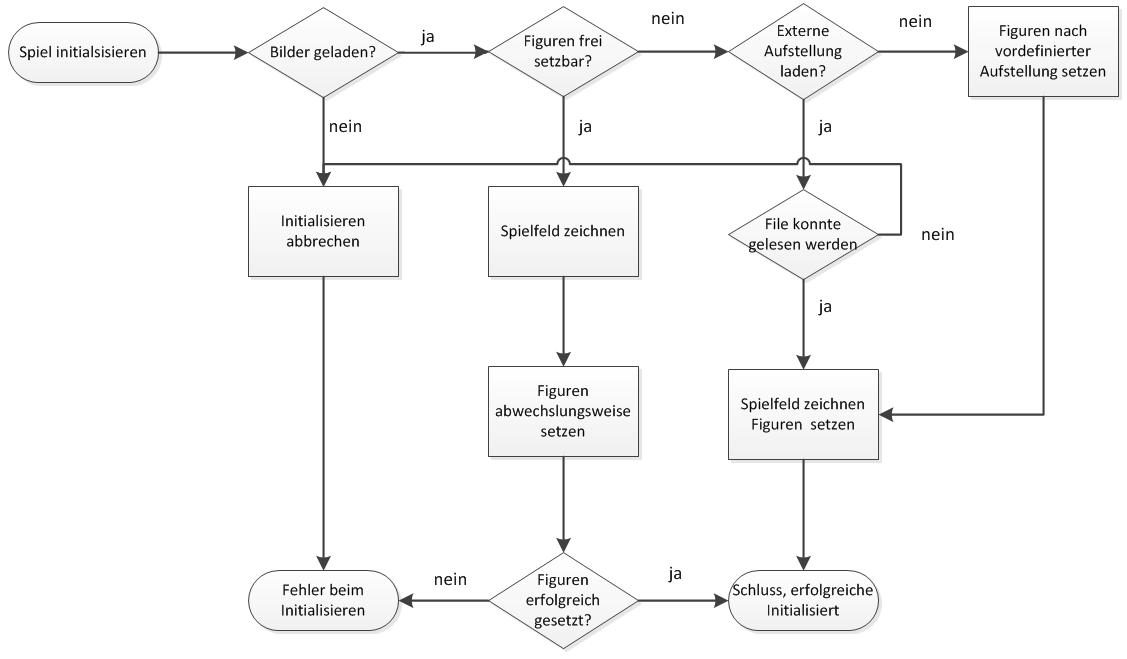
## Struktogramme und Flussdiagramme

### Modul LaserChess.c





### Modul Spiel.c



### Modul Logik.c

#### laser()



#### is\_inside\_map()



#### is\_figure()



#### move\_figure()



#### destroy\_figure()

G:\BFH\Struktogramme Logik.c\destroy_figure.png

#### mouseclick\_to\_map()



#### path\_handler()



#### map\_extension\_handler()



#### play\_sound()



### Modul Grafik.c

## Datenstrukturen

# Implementation

# Anhang

* Pflichtenheft
* Structured Design (CRC)
* Styleguideline