**Projektarbeit Informatik**

Projektdokumentation   
LaserChess V1.2



Website: [stocyr.github.com/LaserChess](http://stocyr.github.com/LaserChess/)

Dozent: Ivo Oesch

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

2 Semester 2012

**Inhaltsverzeichnis**

[1 Einleitung 1-1](#_Toc328148524)

[1.1 Aufgabenstellung: 1-1](#_Toc328148525)

[1.2 LaserChess 1-1](#_Toc328148526)

[2 Planungsvorgehen 2-2](#_Toc328148527)

[3 Realisation 3-3](#_Toc328148528)

[3.1 Team 3-3](#_Toc328148529)

[3.2 Versionskontrollsystem Git 3-3](#_Toc328148530)

[3.3 Main (LaserChess) 3-4](#_Toc328148531)

[3.4 Spiel 3-5](#_Toc328148532)

[3.5 Logik 3-5](#_Toc328148533)

[3.6 Grafik 3-5](#_Toc328148534)

[4 Testvorgehen 4-6](#_Toc328148535)

[5 Dokumentationsvorgehen 5-7](#_Toc328148536)

[5.1 Arbeitsjournal/Laborjournal: 5-7](#_Toc328148537)

[5.2 Zeitplan: 5-7](#_Toc328148538)

[6 Spielanleitung 6-8](#_Toc328148539)

[6.1 Hauptmenü 6-8](#_Toc328148540)

[6.2 Normaler Spielablauf 6-9](#_Toc328148541)

[6.3 Platziermodus 6-11](#_Toc328148542)

[6.4 Easter Eggs 6-11](#_Toc328148543)

[7 Schlusswort 7-12](#_Toc328148544)

[8 Anhang 8-13](#_Toc328148545)

# Einleitung

## Aufgabenstellung:

Innerhalb von 8 Wochen planen und realisierten wir ein ‘kleines‘ C-Programm. Am Ende der 8 Wochen hielten wir eine kleine Kurzpräsentation um unsere Vorgehensweise und Resultate aufzuzeigen.

Da wir das von uns definierten Pflichtenheft schon früh erfüllt hatten, fügten wir mehrere kleine Verbesserungen und Ergänzungen (unter anderem auch Easter Eggs) über das ganze Semester hinweg hinzu.

Dieses Projekt zählt 25% zur Modulqualifikation des 2. Semesters. Der Abgabetermin ist der 25.06.2012.

## LaserChess

Die Idee *LaserChess* entstand ursprünglich von dem Brettspiel KHET 2.0 (Abbildung 1). Wir haben aber lediglich das Spielprinzip übernommen. Die Regeln, sowie Grundaufstellungen und Figuren passten wir unseren Bedürfnissen an.

In dieser Dokumentation gehen wir näher auf die Entwicklung von LaserChess ein. Wir betrachten die einzelnen Planungs-und Realisationsschritte, sowie die Resultate. Näheres zur Software von LaserChess ist in der Softwaredokumentation zu finden.



Abbildung 1: KHET 2.0

# Planungsvorgehen

Als die Spielidee nach einer erstaunlich kurzen Suche gefunden war und das Spielprinzip von KHET verinnerlicht wurde, machten wir uns als erstes an das Pflichtenheft (Anhang 1) ran. Im Pflichtenheft definierten wir unsere Spielregeln, die Spielumgebung, den Spielablauf und den Grafikumfang der Beta-Version. Mit diesem Pflichtenheft konnten wir eine ungefähre Zeitplanung (Anhang 2) erstellen.

Dann folgte die erste Softwareanalyse. Um uns einen Überblick über den Spieleablauf zu schaffen, erstellten wir ein Flowchart (Abbildung 2) und aus diesem eine grobe Modulaufteilung: *Grafik*, *Spiel*, *Logik* und *Main* (LaserChess). Um die Module dann in seine Funktionen aufzuspalten und diese unter uns aufzuteilen, verwendeten wir das Prinzip mit den CRC und erstellten daraus ein ‘Structured Design‘ (Anhang 3). Dieses Prinzip funktionierte aussergewöhnlich gut - innerhalb von 2 Lektionen hatte jeder von uns mehrere Kärtchen mit Funktionen und deren Schnittstellendefinitionen in den Händen, welche man individuell bearbeiten konnte.

Als unser Spiel mit der Planung erstmals etwas Gestalt annahm, bemerkten wir auch diverse Kleinigkeiten, welche wir im Pflichtenheft ergänzen mussten. Etwas verspätet, aber zum Glück nicht vergessen, haben wir dann den Styleguideline (Anhang 4) definiert.

Abbildung 2: Flowchart

# Realisation

## Team

Das Team bestand aus den folgenden Mitgliedern:

* Cyril Stoller (Projektleiter)
* Marcel Bärtschi
* Jascha Haldemann
* Nicolas Käser

Die Teammitglieder beschäftigten sich jeweils mit unterschiedlichen Aspekten der Software, wie z.B. Grafik, Logik, Hauptapplikation etc. und entwickelten sich im Laufe der Zeit zu Spezialisten auf ihrem Terrain. Nichts desto trotz konnten und wurden auch kleine Änderungen in „fremden“ Modulen gemacht. Dadurch wurden das Verständnis und die Sicht der Zusammenhänge untereinander verbessert.

## Versionskontrollsystem Git

Wir haben uns entschieden, das Versionskontrollsystem *Git* ([git-scm.com](http://git-scm.com)) zu verwenden, da ein Softwareprojekt in dieser Grössenordnung und mit vier Teammitgliedern ohne Versionskontrollsystem kaum realisierbar ist.

Das gesamte Projekt inklusive der Dokumentation ist auf dem Online-Projekt-Host *GitHub* ([github.com](https://github.com/)) gehostet. Dieser Server stellt Open-Source Softwareprojekten gratis eine Plattform zur Verfügung, welche perfekt mit dem Versionskontrollsystem *Git* harmonisiert. Die Git-Client Applikation ist in Abbildung 3 zu sehen. Dort hatte dann jedes Teammitglied rund um die Uhr Zugriff auf die aktuellste Version und sämtlichen bisherigen Änderungen.

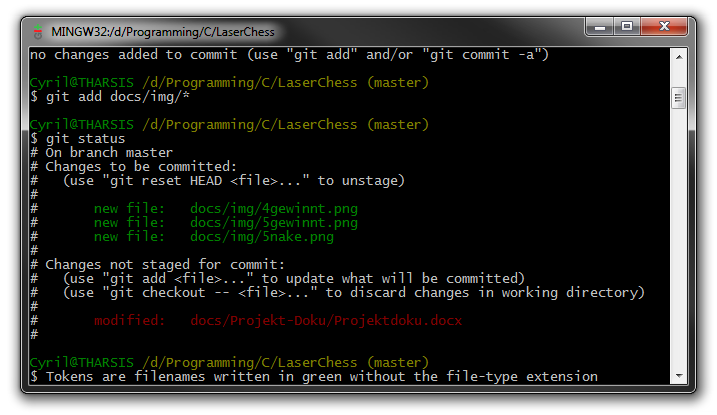


Abbildung 3: Git-Client Software als Windows-Konsolenapplikation

Auf dieser Online-Plattform konnten wir ebenfalls so genannte Issues erfassen, also Aufgaben, welche noch zu erledigen sind. Diese wurden dann jeweils einer Person zugewiesen und nach getaner Arbeit geschlossen. Dieses Issue-System arbeitet sehr eng mit Git zusammen, so dass man in Commit-Messages auf Issues verweisen und solche sogar mit einer entsprechenden Formulierung schliessen kann (z.B. „**... closes #13 ...**“, um den Issue mit der Nummer 13 zu schliessen).

Das Projekt inklusive der gesamten Versions-History, den Issues und verschiedenen Diagrammen zur Entwicklung des Projekts ist hier zu finden: [github.com/stocyr/LaserChess](https://github.com/stocyr/LaserChess). Die Issue-Liste findet man hier: [github.com/stocyr/LaserChess/issues](https://github.com/stocyr/LaserChess/issues).

## Main (LaserChess)

Im Main werden die Figuren und das gesamte Spiel initialisiert. Ausserdem ist hier das Hauptmenü zu finden. Dazu kommen der Argumenthandler und mehrere Ergänzungen.

## Spiel

Im Spielmodul wird das gesamte Spiel oder genauer der Spielablauf gehandhabt. Dazu gehören das Berechnen der zu zeichnenden Fokusfelder und deren Löschung.

## Logik

Im Logikmodul wird vor allem der Laser über das ganze Spielfeld gezeichnet und andere Funktionen aufgerufen, die das Verhalten der Figuren beeinflusst. Dazu kommen diverse ‘Helpertools‘ wie die Abfrage ob gegebene Koordinaten noch im Spielfeld sind, Handhabung der Mausklicks und Andere. Ausserdem ist hier die Handhabung der Pfade und des Sounds zu finden.

## Grafik

Im Grafikmodul werden hauptsächlich alle Grafischen Elemente gezeichnet oder eingefügt (draw\_playground, draw\_figure, draw\_laser usw…). Auch diverse Umrechnungen, welche auch andere Module benutzen wie z.B. die Umrechnung der Pixel auf unsere definierten Spielfelder (pixel\_to\_map) und umgekehrt (map\_to\_pixel), oder die Initialisierung aller Bilder.

# Testvorgehen

Zuerst testeten wir die Stabilität des Programms und überprüften, ob keine „Magic-Numbers“ mehr vorhanden sind. Wir skalierten das gesamte Spielfeld herunter und veränderten andere Grafikparameter, wie z.B. Farben und Dicken. Abbildung 4 zeigt, dass auch nach diversen Veränderungen alles noch in Ordnung war.



Abbildung 4: Parameter-Tuning

Als zweiten Schritt froren wir die Entwicklung kurz ein und markierten den damals aktuellen Stand des Programms mit Version 1.0 (Git-Tag: **v1.0**). Zu diesem definierten Softwarestand erstellten wir dann ein Testprotokoll und führten damit einen Blackbox-Test durch. Wir notierten Fehlerquellen und Unschönheiten und fassten diese in einem Fazit zusammen.

Insgesamt führten wir drei Blackbox-Tests durch und zwar mit den Versionen: **v1.0**, **v1.1** und dem Endprodukt **v1.2**. Diese sind im Anhang 3 zu finden.

# Dokumentationsvorgehen

## Arbeitsjournal/Laborjournal:

Anfangs versuchten wir ein Laborjournal mit den täglichen Entscheidungen zu führen. Da wir aber relativ schnell einmal vorwiegend individuell bei uns Zuhause arbeiteten, führten wir bald kein gemeinsames Laborjournal mehr. Da aber der Verlauf der einzelnen Commits des Softwareversionisierungssystems den Fortschritt ähnlich gut dokumentiert, liegt die Commit-Liste dieser Dokumentation noch als Anhang 2 bei. Ausserdem ist die History jedes Files auf [github.com/stocyr/LaserChess/tree/master/src](https://github.com/stocyr/LaserChess/tree/master/src) einsehbar. Zuerst klickt man auf den Namen des gewünschten Files, dann auf den Button **Blame** oben rechts:



Abbildung 5: File-History auf GitHub.com

## Zeitplan:

Der nach dem Pflichtenheft erstellte Zeitplan (Anhang 1) wurde wöchentlich (jeden Freitag) während der Projektphase kontinuierlich mit den am Anfang eingeplanten Soll-Werten verglichen. Die Änderungen, die über die eingeplanten acht Wochen hinausgehen, wurden im Zeitplan nicht mehr vermerkt.

Noch was?

# Spielanleitung

## Hauptmenü

Beim Start von LaserChess sieht man zuerst das Hauptmenü (Abbildung 6) mit dem Titel und einem Willkommensgruss. Falls das Programm nicht wie gewöhnlich über das Betriebssystem mit der „.exe“-Datei geöffnet wurde, sondern direkt mit Eclipse, folgt eine kurze Mitteilung. Es wäre hier auch möglich, die vorgegebenen Start-Argumente anzuzeigen (in V1.2 aber auskommentiert).

Zum Öffnen eines Modes oder einer anderen Funktion muss die dazugehörige Zahl (1 bis 5) eingegeben und mit Enter bestätigt werden.

**1** Öffnet ein normales Spiel mit der Standard-Aufstellung.

**2** Öffnet den Platziermodus, in welchem man die Figuren nach Wunsch selbst platzieren kann.

**3** Öffnet nach Eingabe der gewünschten Map-Datei (z.B. Aufstellung2.map) eine vordefinierte Spielaufstellung. Diese Map-Datei muss im Ordner **maps/** sein.

**4** Schaltet sowohl den Intro-sound wie auch den gesamten Spiele-Sound aus oder ein.

**5** Schliesst das Programm LaserChess (auch mit dem Schliessen-Button von Windows oben rechts möglich).

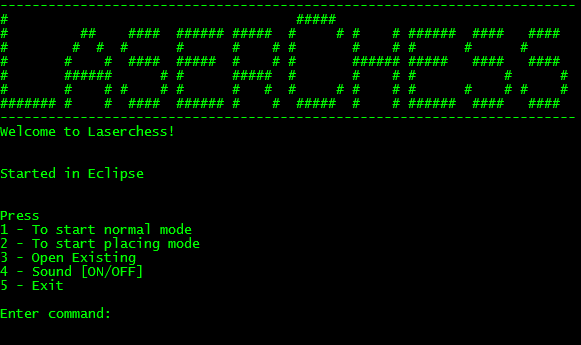


Abbildung 6: Hauptmenü LaserChess

## Normaler Spielablauf

Die vordefinierte Spielaufstellung (Abbildung 7) besteht aus einem roten und einem blauen Team mit je einer Kanone (oben links / unten rechts), einem König (Stern), einer Mauer und vier Spiegeln.

Jede Figur kann verschoben oder gedreht werden.

* König: Zerstörbar. Wird er zerstört, hat der entsprechende Spieler verloren.
* Kanone: Feuert nach jedem Zug zwingend einen Laserstrahl ab. Unzerstörbar.
* Mauer: Kann nicht zerstört werden. Dient dem Schutz des Königs.
* Spiegel: Kann den Laser auf der schrägen, weissen Fläche um 90° ablenken. Zerstörbar   
   wenn im Rücken getroffen.



Abbildung 7: Standardaufstellung

Rot beginnt jeweils den Spielzug und hat die Möglichkeit eine seiner Figuren entweder zu drehen oder um ein Feld zu verrücken. Dazu muss die gewünschte Figur angeklickt werden. Es erscheinen grüne Fokusfelder (Abbildung 8), welche die möglichen Verschiebungspositionen anzeigt. Man kann eine Figur nur auf eine der 8 umliegenden Felder (sofern diese noch nicht besetzt sind) verschieben.

Durch klicken auf eines der Fokusfelder verschiebt sich die Figur dorthin. Dreht man aber mit dem Mausrad nach oben, so dreht sich die Figur um 90° gegen den Uhrzeigersinn. Dreht man das Mausrad nach unten, ergibt das eine Drehung um 90° mit dem Uhrzeigersinn.

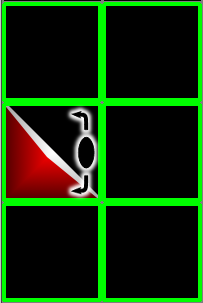


Abbildung 8: Fokusfelder

Um eine angewählte Figur wieder abzuwählen kann man entweder nochmals auf die gleiche Figur klicken oder direkt eine andere Figur anwählen.

Nach jeder Drehung oder Verschiebung ist ein Zug beendet und die Kanone des Spielers, der den Spielzug getätigt hat, feuert automatisch seinen Laser ab. Dies kann nicht verhindert werden!

Der Laser kann an den Spiegeln abprallen. Wenn er einen König (auch den Eigenen!) trifft, beendet er dadurch das Spiel. Trifft er nur eine Wand oder eine Mauer, beendet er den Zug damit ohne weitere Konsequenzen. Trifft der Laser aber einen Spiegel in den Rücken, so wird dieser zerstört und ist für das fortlaufende Spiel verloren.

Die Spieler wechseln sich nach jedem Laserabschuss ab. Ziel ist es, durch vorausschauendes und optisch planerisches Denken den gegnerischen König mit einem Laser zu treffen.

## Platziermodus

Im Platziermodus erscheint nur ein leeres Spielfeld. Es gilt nun die einzelnen Figuren nach einer vordefinierten Reihenfolge zu setzten.

Zuerst werden die Figuren mit einem Klick der linken Maustaste auf dem gewünschten Feld platziert und dann mittels Mausrad in die richtige Richtung gedreht. Schliesslich bestätigt man mit einem weiteren Klick der linken Maustaste die Figur-Position. Es muss also für jede Figur zweimal geklickt werden.

Reihenfolge:

1. roter König
2. blauer König
3. rote Kanone
4. blaue Kanone
5. rote Mauer
6. blaue Mauer
7. abwechselnd ein roter und ein blauer Spiegel (je vier Stück)

Die nächste Figur, die gesetzt wird, ist jeweils auch im Hauptmenü-Fenster zu sehen.

Nachdem alle Figuren gesetzt wurden, beginnt ein normales Spiel mit den gewohnten Regeln.

## Easter Eggs

Im Laufe der Entwicklung dieser Software erreichte die Motivation, sich endlich einmal mit der Softwaredokumentation auseinanderzusetzen, ein dramatisches Minimum. So wuchs der Anreiz, zusätzliche kleine, nicht dokumentierte Features in die Software einzubauen.

Deshalb haben wir uns erlaubt, drei kleine zusätzliche Erweiterungen einzubauen, welche im Hauptmenu mit bestimmten Tokens ausgewählt werden können. Diese Tokens sind irgendwo in diesem Dokument zu finden… Have fun!

# Schlusswort

Fazit: Wir haben alle zwingend vorausgesetzten Funktionalitäten des Pflichtenhefts erreicht. Auch einige der zusätzlich geplanten Funktionen, wie z.B. die Grafiken durch Bilder ersetzt, die Laserquelle verschiebbar gemacht, die Splitter-Figur implementiert (jedoch in Version v1.2 auskommentiert) und die selbst definierbare Spielaufstellung ermöglicht.

Dieses Informatikprojekt hat uns im grossen und ganzen sehr Spass gemacht. Wir hätten nicht gedacht, dass es uns in diesem Ausmass fesseln könnte. Natürlich machte dann das Testen und Dokumentieren weniger Spass. Nichts desto trotz; es war eine gute Erfahrung. Vor allem in Bezug auf die Teamarbeit bei einer Projektarbeit dieser Grössenordnung.

Zum Schluss hoffen wir, dass wir beim Leser mit dem Erwähnen von Easter Eggs den Appetit angeregt haben und er sich einmal kurz Zeit nimmt unsere drei Extra-Features genauer unter die Lupe zu nehmen.

# Anhang

1. Zeitplanung
2. Laborjournal
3. Testprotokolle (v1.0, v1.1, v1.2)
4. Styleguideline