# Pflichtenheft LaserChess

## Spielbeschreibung

### Herkunft

Die Idee des Projekts LaserChess stammt vom Brettspiel *KHET* (<http://www.khet.com/>).

Die Regeln und die Komplexität des Spiels werden geringfügig an unser C Projekt angepasst.

### Regeln

<definieren>

Das Spielfeld ist 6x6 Felder gross…

Es existieren folgende Figuren:

* König
  + Kann verschoben werden, besitzt jedoch keine Orientierung
  + Wird er getroffen, verliert der Spieler die Runde
* Spiegel
  + Kann verschoben werden, besitzt vier Orientierungen
  + Wird er von einer Seite getroffen, die nicht 45° schräg ist, wird er zerstört

## Grafikumfang

Der Umfang der Grafiken kann bei genügen vorhandener Zeit noch erweitert werden. Siehe dazu das Kapitel Erweiterungs- und Anpassungsmöglichkeiten.

### Figuren

Der König wird am Anfang einfach als Rechteck in der Farbe des jeweiligen Spielers gezeichnet.

Der Spiegel wird als rechtwinkliges, gleichschenkliges Dreieck in der Farbe des jeweiligen Spielers gezeichnet.

### SPielfeld

Das Spielfeld wird als Gitternetz dargestellt.

### Laser

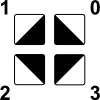
Der Laser wird als rote Linie gezeichnet und zwar bewegt er sich langsam vorwärts.

## Umsetzung / Definitionen

### FIguren

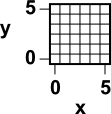
Die Figuren werden in Structs gespeichert:

* player:
  + Enum player\_A, player\_B
* type:
  + Enum sphinx, mirror
* pos:
  + Struct x, y
* orientation:
  + enum TR, TL, BL, BR



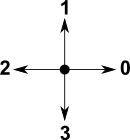
### Map

Das Spielfeld wird in einem 6x6 Figuren-struct-pointer-array gespeichert. Dort hat man Zugriff auf alle Spielfiguren. Man muss ja nicht wissen, wo eine bestimmte Figur ist. Man hat immer die Situation, dass man wissen will, welche Figur an einer bestimmten Position ist.



### Laser

Der Laser besitzt eine Richtung: enum right, up, left, down.



Ändert sich die Richtung im Uhrzeigersinn, wird die Zahl dekrementiert. Im Gegenuhrzeigersinn wird sie inkrementiert. Die Zahl muss innerhalb des Intervals 0 bis 3 bleiben (dies erledigt eine „Normalisierungsfunktion“ anch jeder Richtungsänderung des Lasers).

Trifft der Laser auf einen Spiegel, wird die Differenz der Laser-Richtung und der Spiegelorientierung berechnet. Daraus kann dann folgendes abgeleitet werden:

* Resultat = 0 oder 1
  + Zerstörung des Spiegels, da er von „hinten“ getroffen wird
* Resultat = 2
  + Ablenkung nach rechts (Richtung des Lasers wird dekrementiert)
* Resultat = 3
  + Ablenkung nach links (Richtung des Lasers wird inkrementiert)

D:\Programming\C\LaserChess\docs\img\deflection.png

### Ablauf des Laserschusses

Beim Laserschuss wird eine Funktion laser(pos, dir) aufgerufen, welche den gesamten Laserpfad zeichnet und auch gleich weitere Ereignisse auslöst.

Sie ist so aufgebaut, dass immer nur ein Feld verarbeitet wird. Die Position ist die Startposition des Laserstrahls auf dem Spielfeld. Die Richtung entspricht der Richtung des Lasers. Diese beiden Argumente legen dann auch gerade das Feld fest, welches der Laser als nächstes berührt / durchläuft.

Die Funktion ruft sich selbst (nach jedem durchlaufenen Feld ) solange auf, bis entweder die Sphinx oder eine Wand getroffen wird.

## Erweiterungs- und Anpassungsmöglichkeiten

### Spiel

Es könnte sein dass das Spiel etwas schwerfällig und langweilig wird – Wir haben im Vorfeld das Originalspiel als iOS Game heruntergeladen und damit gespielt. Dabei kam heraus, dass man den Spielschluss extrem herauszögen kann indem man eine defensive Spielstrategie wählt und somit dem Gegner alles in den Weg stellt.

Dieser Umstand könnte durch verschiedene Regelanpassungen verbessert werden:

* Ein Spieler darf mehr als nur einen Zug zugleich machen (z.B. drehen UND verschieben)
* Die Laserquelle darf verschoben werden
* Splitter einbauen, wie im richtigen Game.  
  Diese ‚splitten‘ den Laserstrahl auf: Einer geht geradeaus hindurch, der andere wird um 90° abgewinkelt abgestrahlt. 🡪 Es sind mehr Laserstrahlen im Spiel, somit geht mehr kabutt.

### Grafik

Folgende Grafikverbesserungen können bei vorhandener Zeit noch gemacht werden:

* Alle Grafiken durch Bilder ersetzen
* Beim „draggen“ mit der Maus Figur unter der Maus anzeigen