

# Kissenschlacht, grundlegend – Technisch, nicht Ethisch.

<http://worgtsone.scienceontheweb.net/worgtsone/> - [mailto: worgtsone @ hush.com](mailto:worgtsone@hush.com)

13. Oktober 2011

## Inhaltsverzeichnis

<b>1</b>	<b>Kissenschlacht 27.10.2007</b>	<b>3</b>
<b>2</b>	<b>Definition – 2002</b>	<b>3</b>
<b>3</b>	<b>Oberflächlich – 2004</b>	<b>4</b>
3.1	Variablen . . . . .	4
3.2	Server in php – 2007 . . . . .	4
3.3	Datenspeicher Datenbank? (optional) – 2005 . . . . .	5
3.4	Datenspeicher Datei! – 2007 . . . . .	5
<b>4</b>	<b>Kommunikation im PAP (auch grob falsch)</b>	<b>6</b>
4.1	Hat sich etwas geändert? . . . . .	6
<b>5</b>	<b>Nachgearbeitet am 23.10.2007</b>	<b>7</b>
5.1	GetUrl.java . . . . .	7
5.2	kissens.php3 . . . . .	7
5.3	kissens.sh . . . . .	7
5.4	Ausgabe . . . . .	7
<b>6</b>	<b>Java-Server (28.10.2007)</b>	<b>8</b>
<b>7</b>	<b>Ein Webserver auf Port 8008</b>	<b>9</b>
<b>8</b>	<b>HttpServer02</b>	<b>10</b>
<b>9</b>	<b>Projektwoche 15-19.06.2008</b>	<b>11</b>
9.1	Screenshot . . . . .	11
9.2	Vorteile . . . . .	11
9.3	Nachteile . . . . .	11
9.4	TODO . . . . .	12
9.5	Quelltext Server . . . . .	13
9.6	Quelltext Client . . . . .	17
<b>10</b>	<b>Unmaintainable Code – 2009-07-08</b>	<b>23</b>
10.1	Kurzer Blick auf den Client . . . . .	23
10.2	Kurzer Blick auf den Server – hoppla . . . . .	24

### **Disclaimer**

Wissen ist zum Teilen da. Ich teile mein Wissen mit Ihnen, lieber Kollege.  
Ich bin aber nicht perfekt. Unter [worgtsone@hush.com](mailto:worgtsone@hush.com)  
nehme ich dankbar Ihre Verbesserungsvorschläge entgegen.

\*

**Legal Blurb:** Alle Informationen in diesem Dokument sind falsch, unvollständig,  
irreführend, irrelevant und / oder funktionieren einfach nicht.  
Wenn Sie es trotzdem benutzen, und es geht dabei etwas kaputt, ist das Ihr  
Problem, nicht meins.

\*

**Bitte teilen Sie meine Web-Adresse nicht Ihren Schülern mit.**

## 1 Kissenschlacht 27.10.2007

Ein voller Erfolg.

Ich konnte alle Einser-Kandidaten der 11, 12 und 13 für das Projekt gewinnen. Außer einem.

Einige wenige wußten, was Server, Clients, http, http-Clients, Ports, Apachen, php-Module, Datenbanken, verschnackste Tabellen und jdbc-Datenquellen sind.

Einer ders nicht wußte, konnte immerhin XAMPP installieren und eine kissens/index.php erzeugen, die antwortete und Ergebnisse zurückgab.

Ein anderer 12er bohrte sie auf, so da sie Spieler und Kissen in einer Datenbank verwaltete.

## 2 Definition – 2002

Browserspiele sind Spiele, wo mehrere User (Menschen oder auch Rechner) über einen Browser gegeneinander antreten.

Jeder User bekommt dabei ein eigenes Browserfenster, in dem er Statusmeldungen erhält und Aktionen ausführen kann.

Ich denke bei der Ausführung ausschließlich in Java und PHP 3.0, auch wenn es andere Möglichkeiten geben mag.

**Nachtrag 2008-06-20:** Jetzt nur noch in Java. Punkt.

### 3 Oberflächlich – 2004

Schauen wir das Ganze zunächst von außen an:

Ein Spieler geht zur URL. Wenn er noch keine Session-ID hat, bekommt er eine und darf dann mitspielen.

(Spiel-ID, Username, Paort, Lieblings-Charakter, Fähigkeits-Punkte etc. sind äußerst wichtig. Wir werden später zeigen, wie man sie modelliert.)

Zum Spielen braucht er

- Mitspieler;
- ein Spielfeld;
- eine Kommunikations-Plattform.

Schleich will er (möglichst in Echtzeit) ber Aktionen der anderen Spieler informiert werden und selber aktiv werden.

Visualisierung, schätze ich, nach Wahl, als Applet oder Java-Anwendung.

(2007:) Kommunikation übers http-Protokoll.

#### 3.1 Variablen

So. Das bedeutet, daß wir Informationen ber den Spielstand irgendwo vorhalten müssen: Lebens-Punkte, Zeitpunkt der letzten Änderung, etc. Für alle Spieler. Für alle Spiele. Unabhängig von der Laufzeit der Skripte. Spieler, die nicht mehr mitspielen, müssen behandelbar sein.

#### 3.2 Server in php – 2007

Die lokalen Änderungen, sprich: die Spielzüge des lokalen Spielers müssen dem Server mitgeteilt werden. Ich schlage dazu eine total verrückte URL vor:

```
http://mein.server.org/php/kissens.php3?user=anton&xpos=23&ypos=999&richtung=3.33&werfen=ja&speed=-1:akissen=3:msecsOfNoActivity=888
```

Sie ruft eine php-Datei auf `mein.server` auf und übergibt ihm die Variablen

- Username;
- x- und y-Position auf dem Spielfeld;
- Blickrichtung in Radianen.  
Warum nicht. Java rechnet intern sowieso damit.
- ob der User gerade ein Kissen geworfen hat;
- ob der User läuft (speed=1) oder steht (0) oder rückwärts läuft (-1);
- wieviele Kissen der User noch zur Hand hat;
- wieviele msec der User inaktiv war. Nach zB 5000 msec wird er rausgeworfen.

Der Witz ist, daß diese Sachen sehr leicht in `php` zu parsen sind.

### 3.3 Datenspeicher Datenbank? (optional) – 2005

Das alles schreit nach einer Datenbank. Ich dachte dabei an folgende Spalten:

Datenbank Spiel ( Tabelle Spiel (time timestamp, int spielerID, int spielID, string aktion ) ).

**Beispiel:** Auf einem Server laufen 13 brandheiße Partien TicTacToe. In Partie 11 setzt Spieler02 ein Kreuz auf Feld 5 (das in der Mitte). Das paßt alles in die obige Datenbank.

Nachtrag: Die SpielID beschränkt die mölichen SpielerIDs, also braucht es mehr Tabellen: Spiel (mit allen Spielen drin), Spieler (ordnet den Spielen Spieler zu) und Aktion (ordnet Spielern Aktionen mit Timestamp zu).

### 3.4 Datenspeicher Datei! – 2007

Um die grundsätzliche Funktion sicherzustellen, reicht ja wohl *ein* Spielfeld.

Und eine Datei. Die kann man nämlich mit einem Shell-Script quälen. Auch wenn das heißt, daß die Lernenden mit `bash`-Syntax gequält werden.

Die Server-Antwort, also die Rückgabe-Datei, die die Clients alle – sagen wir – 100msec vom Server fordern, könnte grundsätzlich so aussehen:

```
SpielfeldID:Spieler:xPos:yPos:Blickrichtung:speed
```

(ja, mit den Doppelpunkten – dann ist sie leicht in Java zu parsen) zum Beispiel:

```
2387456:anton:34.5:2.3:0.993:-1
```

Und dann brauchen wir noch die Kissen

```
SpielfeldID:KissenID:xPos:yPos:Flugrichtung:speed:geflogeneStrecke
```

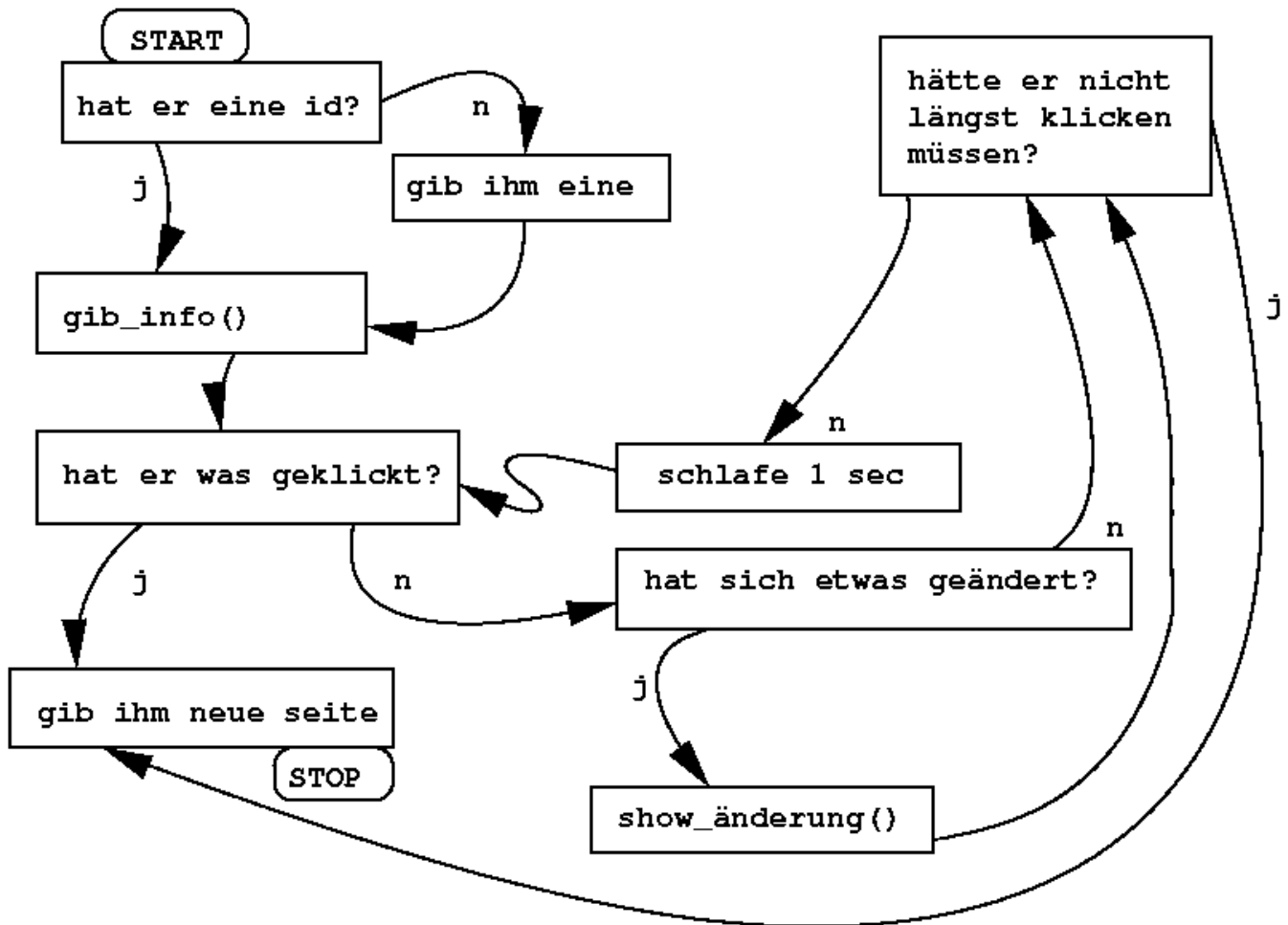
zum Beispiel:

```
2387456:Kissen78:3.5:27.3:0.993:2:19.8
```

Die geflogene Strecke wird gebraucht, weil ein Kissen aufhört zu existieren (oder zu Boden fällt?) wenn es `streckeMax` geflogen ist.

**2008-06-20:** Na zu Boden natürlich.

## 4 Kommunikation im PAP (auch grob falsch)



PAP der Kommunikation

### 4.1 Hat sich etwas geändert?

Wenn einer der Mitspieler es sich mittendrin anders überlegt und wegsurft, sollte das Spiel nicht hängen (schließlich wartet da noch jemand). Zweckmäßigerweise definiert man eine Variable `timeOutSpieler` (in sec). Wenn jemand so lange nichts getan hat, wird er automatisch rausgeworfen.

So. Fehlt nur noch der Nachweis, da das funktioniert. Ich liefere eine abgespeckte Version, nur 1 Spiel, nur 2 Spieler, nur 2 Aktionen, davon ist eine Logout.

(2005) Später.

(2007) Kann noch dauern. Bin gerade mal wieder umgezogen und hab so ein paar Ideen über den Server...

## 5 Nachgearbeitet am 23.10.2007

Wer glaubt schon, da man in php zuerst feinfühlig Daten an Java-Programme ausliefern kann? Niemand. Deshalb liefere ich hier ein wenig Quelltext.

### 5.1 GetUrl.java

Bastelt eine total verrückte Url mit Datenübergabe, ruft sie auf und zeigt das Ergebnis an.

```
import java.net.*;
import java.io.*;

class GetUrl {
    public static void main (String args[]) throws Exception {
        System.out.println ("GetUrl by GRM, Oct 23 2007, does GetUrl");
        URL url = new URL ("http://localhost/php/kissens.php3?user=doofi05");
        BufferedReader in =
            new BufferedReader (new InputStreamReader (url.openStream ()));
        String inputLine;
        while ((inputLine = in.readLine ()) != null)
            System.out.println (inputLine);
        in.close ();
    }
}
```

### 5.2 kissens.php3

Überprüft übergebene Daten und zeigt das Ergebnis an.

```
***hallo from kissens***
<?php
    if (isset($user))
        print "User ist ".$user;
    if (isset($user2))
        print ", UserZwo=".$user2;
?>
```

### 5.3 kissens.sh

Shell für Faule.

```
lynx 'localhost/php/kissens.php3?user2=mastro&user=werAuchImmer'
```

### 5.4 Ausgabe

```
*** hallo from kissens *** User ist werAuchImmer, UserZwo=mastro
```

## 6 Java-Server (28.10.2007)

Wer möchte schon immer XAMPP auf seinen Rechner spielen... ist ne tolle software, aber braucht root-Rechte und plattenplatz usw... — ... und ist einfach nicht selbst geschrieben.

Der Server darf doch im Prinzip laufen wo er will, zB auf Port 6666. Da darf jeder User lesen und schreiben, auch auf Un\*x-Systemen. Viel zu tun hat er auch nicht:

```

init (port 6666)
while (run) {
  warte auf request
  pruefe auf erlaubtheit: wenn seit dem letzten request zu wenig
    zeit vergangen ist, schicke leere antwort :>)
  pruefe request auf plausibilitaet, zb
    wenn speed > 1 dann speed = 1
    wenn speed < -1 dann speed = -1
    wenn xpos oder ypos ausserhalb spielfeld
      dann stutz sie zurecht
  berechne die vergangene zeit deltatee
  hol die datei mit allen kissen
  hol die datei mit allen spielern
  schmeiss alle spieler mit timeout>5000 raus
  und deren Kissen
  anhand deltatee:
    sind kissen zu boden gefallen?
      ja: verwandle sie in liegende kissen
    haben kissen das spielfeld verlassen?
      ja: leg sie so hin, dakein spieler getroffen wird
  such den spieler in der datei
    wenn es ihn nicht gibt:
      erzeug ihn und leg ein paar kissen dazu
  veraendere alle spielerpositionen
    anhand pos, richtung, speed und deltatee
  wurde ein spieler von einem fliegenden kissen erwischt?
    ja : nimm ihm alle kissen weg und leg sie irgendwohin
      gib ihm neue position
      erhoehe seine abwrfe und die treffer des schuetzen
  sind zwei spieler zusammengetroffen?
    ja : nimm jedem ein kissen weg.
  hat einer ein herumliegendes kissen erwischt?
    ja: gib dem spieler und loeschs vom boden
  konnte er werfen?
    ja : hat er geworfen?
      ja : instanziiere ein neues fliegendes kissen
        mit dem namen des spielers drauf
  bastel die spieler-datei und speicher sie
  bastel die kissen-datei und speicher sie
  schick dem spieler die antwort
}

```

## 7 Ein Webserver auf Port 8008

Nachtrag vom 4.4.2008

Es ist mühsam, ein LAMP zu installieren und dann php zu programmieren.  
Es ist einfacher, einen Webserver unter Java selber zu schreiben:

```
# public class SimpleHTTPServer {
#     private ServerSocket serverSocket = null;
#
#     public void runServer( int port ) {
#         try {
#             // create a new ServerSocket and bind it to the specified port
#             serverSocket = new ServerSocket( port );
#             // now do the same thing as a listen().
#             //this method will block until a client connects to us.
#             Socket nextClientSocket = serverSocket.accept();
#             // print a little bit out about the client
#             System.out.println( "got a connection from " +
#                 nextClientSocket.getInetAddress().toString() );
#             // close the socket for now as we're just testing
#             nextClientSocket.close();
#         } catch( Exception e ) {
#             System.err.println( "cannot create new server socket on port " + port );
#             e.printStackTrace( System.err );
#             return;
#         }
#     }
#
#     public static void main( String[] argv ) {
#         int port = 0;
#
#         if( argv.length != 1 ) {
#             System.err.println( "usage: java SimpleHTTPServer <port>" );
#             System.exit( 1 );
#         }
#
#         try {
#             port = Integer.parseInt( argv[0] );
#         } catch( NumberFormatException nfe ) {
#             System.err.println( "please use a valid integer for the port" );
#             System.exit( 1 );
#         }
#
#         SimpleHTTPServer server = new SimpleHTTPServer();
#         server.runServer( port );
#     }
# }
```

Nun muß nur noch die Spielelogik hinein.

## 8 HttpServer02

Das geht besser: vom 24.04.2008.

```
import java.net.*;
import java.io.*;
import java.util.*;

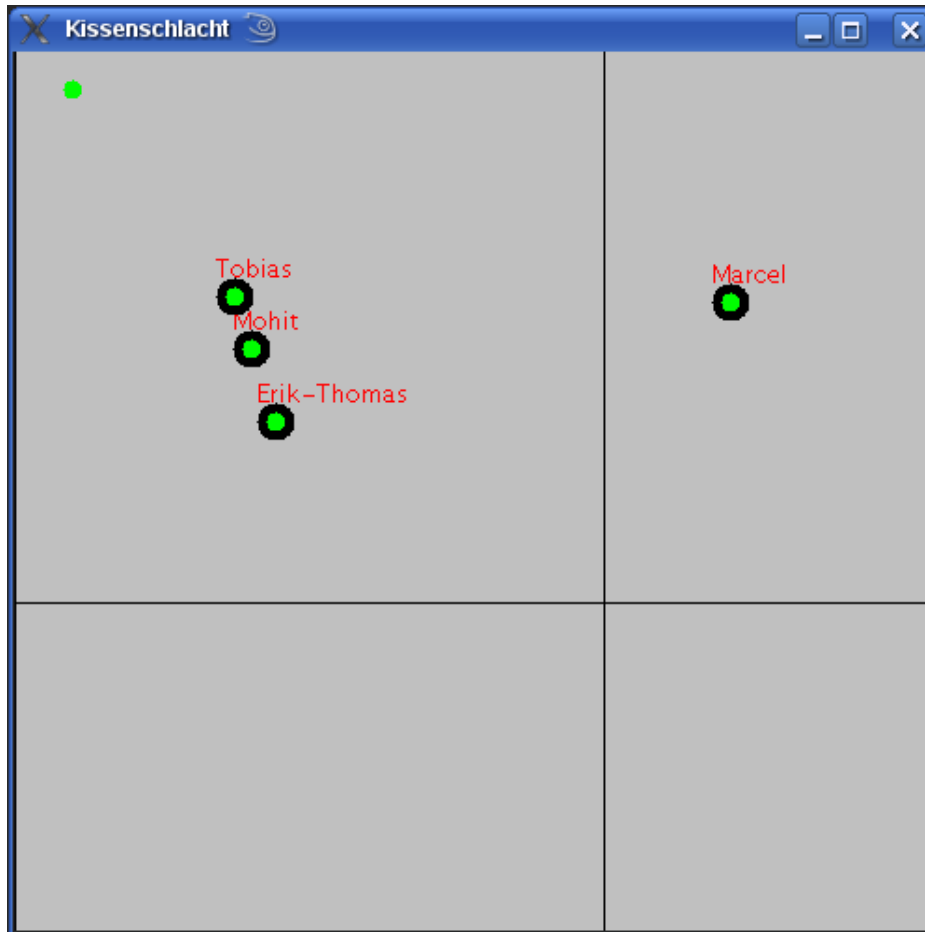
class HttpServer02 {
    public static void main (String args[]) {
        System.out.print ("HttpServer01 ");
        try {
            ServerSocket listen = new ServerSocket (8008);
            System.out.println (" on port " + listen.getLocalPort ());
            while (true) {
                String data =
                    "<head><meta http-equiv=\"refresh\" content=\"2\"></head>";

                data += "<body><h2>Willkommen auf ostersrv versuch 2!</h2><hr>";
                long mmss = System.currentTimeMillis ();
                data += "Zeit in japanisch und ohne Monate : ";
                long mil = mmss % 1000;
                mmss = mmss / 1000;
                long sec = mmss % 60;
                mmss = mmss / 60;
                long min = mmss % 60;
                mmss = mmss / 60;
                long hou = mmss % 24;
                mmss = mmss / 24;
                long day = mmss % 365;
                mmss = mmss / 365;
                long yea = mmss;

                data += (1970 + yea) + " years " + day + " tage " +
                    hou + " stunden " + min + " minuten " + sec + "." + mil +
                    "sekunden.<hr>";
                Socket client = listen.accept ();
                System.out.println ("connection from " + client.toString ());
                PrintStream ps = new PrintStream (client.getOutputStream ());
                ps.print ("HTTP/1.0 200 OK\r\n");
                ps.print ("Content-length: " + data.length () + "\r\n\r\n");
                ps.print (data);
                client.close ();
            }
        }
        catch (Exception e) { }
    }
}
```

## 9 Projektwoche 15-19.06.2008

### 9.1 Screenshot



### 9.2 Vorteile

Ich habe wohl die stärksten Java-Programmierer der ganzen Schule versammeln können. Und das, obwohl ich zum Zeitpunkt der Projektvorstellung gar nicht da war.

Wir haben uns drei Tage lang auf folgendes Ablaufschema geeinigt:

Client : init - while !exit anfrage(); antwortVerstehen(); zeichnen(); schlafen() .

Server : init - while !exit anfrageVerstehenUndBeantworten(); schlafen() .

Die einzelnen aufgetauchten Problemchen (zB: wie binde ich einen KeyListener ein - wie male ich einen Kreis dahin wo er hin soll - wer überprüft hier die Regeln - wie schickt / versteht man eine http-anfrage - etc) konnten die Schler aus real existierenden Java-Quellen im Internet zusammenklauen.

Jede Zeile ist handgeschrieben.

Natürlich hatte ich die Aufgaben in Sub-Probleme aufgeteilt und verteilt. Die Schüler waren schlaue genug, die Probleme auf separate Java-Klassen zu verstreuen und die getrennt zu ändern und zu kompilieren. Ist ja egal, wie oft man eine Klasse ändert, solange die Schnittstelle gleich bleibt.

Ich bin stolz auf sie.

### 9.3 Nachteile

Keine Kissen.

Ich will nur zwei Klassen: Client und Server. Parser-Gerümpel teilen sie durch CutUndPaste, basta. Den Client gerne auch als Applet.

Hups! Drei Klassen : Server, Client, Robot. Robot? Robot!

#### 9.4 TODO

- Jeder kann sich für jeden ausgeben. Schlüssel vereinbaren.
- Kissen.
- Code umstricken, so daß er den o.g. Einzeilern folgt.
- Grafik aufmotzen.
- Sound.
- 3-d-Darstellung.
- Roboter. Doofe, zufallsgesteuerte. Und Schlaue, mit Strategie und Taktik.  
Selbstgeschriebene Künstliche Intelligenz in meiner selbstgeschriebenen kleinen Welt...

Und schon ist die nächste Projektwoche gerettet.

## 9.5 Quelltext Server

```

###file config.cfg los ###
#Config Server Kissenschlacht
#Spielfeld
Spielfeld_in_Pixel      = 400
#Maximale Spieler Anzahl
Max_Spieler             = 10
#erlaubte Anfragen vom Client an den Server in ms
Anfragen                = 70
#Zeit, nach der ein Spieler vom Server geschmissen wird, wenn er abwesend ist
#(in ms , Standart 60000ms = 1 Minute)
Anticamper              = 60000
### file config.cfg end ###

###file ConfigLesen.java los ###
import java.io.*;
import java.util.*;

class ConfigLesen {

static int maxspieler, spielfeld, anticamper, fps;

static void config () {
    try {
        Properties properties = new Properties ();
        properties.load (new FileInputStream ("config.cfg"));
        spielfeld =
            Integer.parseInt (properties.getProperty ("Spielfeld_in_Pixel"));
        fps = Integer.parseInt (properties.getProperty ("Anfragen"));
        anticamper = Integer.parseInt (properties.getProperty ("Anticamper"));
        maxspieler = Integer.parseInt (properties.getProperty ("Max_Spieler"));
        /* System.out.println(spielfeld);
           System.out.println(fps);
           System.out.println(anticamper); */
        //fileProp.setProperty("Name","test");
    } catch (FileNotFoundException e) {
        System.out.println ("File Not Found");
    } catch (IOException e2) {
        System.out.println ("Konnte Stream nicht laden");
    }
}

}

### file ConfigLesen.java end ###

###file Parsen.java los ###
import java.util.StringTokenizer;
public class Parsen {

public static String name="";
public static int speed,dir, xpos, ypos;

static void parseClient(String s){
    StringTokenizer st = new StringTokenizer(s, ":");
    int i=0;
    while (st.hasMoreTokens()) {
        switch(i){
            case 0: name= st.nextToken();
                break;
            case 1: speed=Integer.parseInt(st.nextToken());
                break;
            case 2: dir=Integer.parseInt(st.nextToken());
                break;
            default: st.nextToken();
        }
        i++;
    }
}

static void parseServer(String s){
    StringTokenizer st = new StringTokenizer(s, ":");
    int i=0;
    while (st.hasMoreTokens()) {
        switch(i){
case 0: name= st.nextToken();
break;
case 1: xpos=Integer.parseInt(st.nextToken());
break;

```



```

        flame = false;

//      System.out.println ("debug: relevantLine = " + relevantLine);
if (length < 5)
{
    System.out.println ("length.relevantLine< 5!!!");
    relevantLine = "fehler!!!";
}
if (relevantLine.substring (0, 5).equals ("GET ?") ||
    relevantLine.substring (0, 6).equals ("GET /?"))
{
    String s = relevantLine.replaceAll ("\\?", "");
    s = s.replaceAll (" HTTP/1.1", "");
    s = s.replaceAll ("/", "");
    s = s.replaceAll ("GET ", "");
//anfrage parsen
    Parsen.parseClient (s);
    name = Parsen.name;
//suchen ob spieler vorhanden, daten aktualisieren
    while (a < maxspieler) {
        if (name.equals (Spieler.name[i])) {
            zeit = System.currentTimeMillis () - Spieler.zeit[i];
            if (zeit > fps) {
                if (Parsen.speed == 666) {          //Spieler löschen
                    Spieler.name[i] = "";
                    flame = true;
                } else {
                    Spieler.name[i] = Parsen.name;
                    if (Parsen.speed > 5) {
                        Parsen.speed = 5;
                    }
                    if (Parsen.speed < -5) {
                        Parsen.speed = -5;
                    }
                    Spieler.speed[i] = Parsen.speed / 2.0;
                    Spieler.dir[i] = Parsen.dir;
                    if (Parsen.speed != 0) {
                        Spieler.zeit[i] = System.currentTimeMillis ();
                    }
                }
                Spieler.zeit (anticamper);
            } else {
                flame = true;
            }
            leer = true;
            a = maxspieler;
        } else {
            i++;
            a++;
        }
    }
}
if (Parsen.speed == 666) {          //Spieler löschen
    flame = true;
    leer = true;
}
//wenn spieler nicht gefunden neuen erstellen, wenn maxspieler vorhanden server=voll
if (leer == false) {
    i = 0;
    a = 0;
    while (a < maxspieler) {
        name = Spieler.name[i];
        if (name.equals ("")) {
            Spieler.name[i] = Parsen.name;
            Spieler.speed[i] = Parsen.speed / 5.0;
            Spieler.dir[i] = Parsen.dir;
            Spieler.xpos[i] = Math.round (Math.random () * 999) + 1;
            Spieler.ypos[i] = Math.round (Math.random () * 999) + 1;
            Spieler.zeit[i] = System.currentTimeMillis ();
            leer = true;
            a = maxspieler;
        } else {
            a++;
            i++;
        }
    }
    if (a == maxspieler && leer == false) {
        voll = true;
        System.out.println ("error: Server voll!");
    }
}

```

```

    }
  }
} else {
    fehler = true;
    System.out.println ("line fängt nicht mit get /? an.");
}
if (voll || fehler) {
    PrintStream ps = new PrintStream (client.getOutputStream ());
    if (fehler) {
        System.out.println ("Fehler!");
    } else
        ps.print ("Server voll ");
} else {
    if (flame) {
    } else {
        double dir_in_rad = -Math.PI / 180 * Spieler.dir[i] + Math.PI / 2;
        Spieler.xpos[i] =
            Spieler.xpos[i] + Spieler.speed[i] * Math.cos (dir_in_rad);
        Spieler.xpos[i] = Math.round (Spieler.xpos[i] * 10000) / 10000.0;
        //      ypos = ypos + speed * Math.sin(dir_in_rad);
        //      in java ist y von oben nach unten
        Spieler.ypos[i] =
            Spieler.ypos[i] - Spieler.speed[i] * Math.sin (dir_in_rad);
        Spieler.ypos[i] = Math.round (Spieler.ypos[i] * 10000) / 10000.0;
        // sanity check
        if (Spieler.xpos[i] > spielfeld)
            Spieler.xpos[i] = spielfeld;
        if (Spieler.xpos[i] < 20)
            Spieler.xpos[i] = 20;

        if (Spieler.ypos[i] > spielfeld)
            Spieler.ypos[i] = spielfeld;
        if (Spieler.ypos[i] < 20)
            Spieler.ypos[i] = 20;

        PrintStream ps = new PrintStream (client.getOutputStream ());
        String data = "";
        for (int dat = 0; dat < maxspieler; dat++) {
            if (Spieler.name[dat].equals ("")) {
// do nothing
            } else {
                if (dat == (maxspieler - 1)) {
                    data +=
                        Spieler.name[dat] + ":" + Math.round (Spieler.xpos[dat]) +
                        ":" + Math.round (Spieler.ypos[dat]) + ":" +
                        Spieler.dir[dat];
                } else {
                    data +=
                        Spieler.name[dat] + ":" + Math.round (Spieler.xpos[dat]) +
                        ":" + Math.round (Spieler.ypos[dat]) + ":" +
                        Spieler.dir[dat] + "\n";
                }
            }
        }

        int mx = (int) (99 +
            79 * Math.cos (System.currentTimeMillis () / 1000.0));
        int my = (int) (99 +
            79 * Math.sin (System.currentTimeMillis () / 1000.0));
        data += "mohit:" + mx + ":" + my + ":30\n";
        mx = (int) (99 +
            69 * Math.cos (System.currentTimeMillis () / 1000.0));
        my = (int) (99 +
            69 * Math.sin (System.currentTimeMillis () / 1000.0));
        data += "mohit:" + mx + ":" + my + ":30\n";
        mx = (int) (99 +
            49 * Math.cos (System.currentTimeMillis () / 1000.0));
        my = (int) (99 +
            49 * Math.sin (System.currentTimeMillis () / 1000.0));
        data += "mohit:" + mx + ":" + my + ":30\n";
        System.out.print ("data = " + data);
        ps.print (data);
    }
}
client.close ();
}
}
catch (Exception e) {

```

```

        // System.out.println("exception nummer " + e.getN());
        System.out.println (e.getMessage ());
    }
}
}
### file Server07.java end ###

####file Spieler.java los ###
class Spieler{
//{"","","","","","","","","",""};
    static String [] name = new String[ConfigLesen.maxspieler];

    static void name(){
        for(int i = 0; i < ConfigLesen.maxspieler ; i++){
            name[i]="";
        }
    }
    static double [] speed = new double [ConfigLesen.maxspieler];
    static int [] dir = new int [ConfigLesen.maxspieler];
    static double [] xpos = new double [ConfigLesen.maxspieler];
    static double [] ypos = new double [ConfigLesen.maxspieler];
    static long [] zeit = new long [ConfigLesen.maxspieler];
    static void zeit(int kick){
        long current=System.currentTimeMillis();
        for(int i=0 ; i< 10 ; i++ ){
            if( (current - zeit[i]) > kick){
                name[i]="";
            }
        }
    }
}
### file Spieler.java end ###

```

## 9.6 Quelltext Client

```

####file config.cfg los ###
#Config Client Kissenschlacht
#Spielername
name = Marcel
#Ip Adresse
url=localhost
### file config.cfg end ###

####file ConfigLesen.java los ###
import java.io.*;
import java.util.*;

class ConfigLesen {

    static String name;
    static String url;

    public static String name () {
        try {
            Properties properties = new Properties ();
            properties.load (new FileInputStream ("config.cfg"));
            name = properties.getProperty ("name");
        } catch (FileNotFoundException e) {
            System.out.println ("File Not Found");
        } catch (IOException e2) {
            System.out.println ("Konnte Stream nicht laden");
        }
        return name;
    }

    public static String url () {
        try {
            Properties properties = new Properties ();
            properties.load (new FileInputStream ("config.cfg"));
            name = properties.getProperty ("url");
        } catch (FileNotFoundException e) {
            System.out.println ("File Not Found");
        } catch (IOException e2) {
            System.out.println ("Konnte Stream nicht laden");
        }
    }
}

```

```

    return name;
}
}
### file ConfigLesen.java end ###

###file kissen.java los ###
public class kissen {
    int x, y;
    boolean show;

    void set_koords (int x, int y) {
        this.x = x;
        this.y = y;
    }
}
### file kissen.java end ###

###file Parsen.java los ###
import java.util.StringTokenizer;

public class Parsen {

    public static String name = "";
    public static int dir, speed, xpos, ypos;

    static void parseClient (String s) {
        StringTokenizer st = new StringTokenizer (s, ":");
        int i = 0;
        while (st.hasMoreTokens ()) {
            switch (i) {
                case 0: name = st.nextToken (); break;
                case 1: speed = Integer.parseInt (st.nextToken ()); break;
                case 2: dir = Integer.parseInt (st.nextToken ()); break;
                default: st.nextToken ();
            }
            i++;
        }
    }

    static void parseServer (String s) {

        StringTokenizer st = new StringTokenizer (s, ":");
        int i = 0;
        while (st.hasMoreTokens ()) {
            switch (i) {
                case 0: name = st.nextToken (); break;
                case 1: xpos = Integer.parseInt (st.nextToken ()); break;
                case 2: ypos = Integer.parseInt (st.nextToken ()); break;
                case 3: dir = Integer.parseInt (st.nextToken ()); break;
                default: st.nextToken ();
            }
            i++;
        }
    }
}
### file Parsen.java end ###

###file screen.java los ###
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;

public class screen extends Frame implements KeyListener {

    private int user_nrs = 10;
    private int fieldX, fieldY;
    String s = " ";
    user[] user = new user[user_nrs];
    kissen[] kissen = new kissen[user_nrs * 5];
    int my_speed = 0, dir = 90;
    boolean exit = false;
    static String name = ConfigLesen.name ();
    static String url = ConfigLesen.url ();

    public screen (int fieldX, int fieldY, int user_nrs) {
        super ("Kissenschlacht");
        // Festlegen einiger Variablen

```

```

this.fieldX = fieldX;
this.fieldY = fieldY;
this.user_nrs = user_nrs;

// grafische Sachen
setBackground (Color.lightGray);
setForeground (Color.yellow);
setSize (fieldX, fieldY);
setFont (new Font ("Courier New", Font.PLAIN, 13));
setLocation (0, 0);
setVisible (true);
addKeyListener (this);
// Erstellen aller Objekte
for (int i = 0; i < user_nrs; i++) {
    this.user[i] = new user ();
    //this.user[i].name=randomString();
    this.user[i].x = -1;
    this.user[i].y = -1;
}

for (int i = 0; i < user_nrs; i++) {
    this.kissen[i] = new kissen ();
    this.kissen[i].show = true;
    this.kissen[i].x = -1;
    this.kissen[i].y = -1;
}

do {
    int i = 0;
    repaint ();
    sleep (50);

    Parsen parse = new Parsen ();
// Server Anfrage
    if (exit) this.my_speed = 666;

    String anfrage = "http://" + url + ":8008?" + this.name + ":" +
        this.my_speed + ":" + this.dir;
    System.out.println ("senden: " + anfrage);

    try {
        URL myurl = new URL (anfrage);
        long starttime = System.currentTimeMillis ();
        BufferedReader r =
            new BufferedReader (new InputStreamReader (myurl.openStream ()));
        do {
            i++;
            String s = r.readLine ();
            System.out.println ("server responds: " + s);
// a real question mark is a double backslashed one
            s = s.replaceAll ("\\?", "");
//            System.out.println ("after replacing1: " + s);
            s = s.replaceAll (" HTTP/1.1", "");
//            System.out.println ("after replacing2: " + s);
            s = s.replaceAll ("/", "");
//            System.out.println ("after replacing3: " + s);
            s = s.replaceAll ("GET ", "");
//            System.out.println ("after replacing4: " + s);

            parse.parseServer (s);
            String t = " ";
            sporn_player (i, parse.xpos, parse.ypos, parse.name, parse.dir);
            if (exit) {
                setVisible (false);
                dispose ();
                System.exit (0);
            }
        } while (s != null);
        long endtime = System.currentTimeMillis ();
        System.out.println ("Gesamtzeit=" + (endtime - starttime));
    }
    catch (Exception e) { }
} while (true && !exit);
}

public void paint (Graphics g) {

    g.setColor (Color.green);

```

```

g.fillOval (30, 40, 10, 10);
g.setColor (Color.black);
g.drawLine (0, 23, this.fieldX, 23);
g.drawLine (4, 0, 4, this.fieldY);
g.drawLine (0, this.fieldY - 180, this.fieldX, this.fieldY - 180);
g.drawLine (this.fieldX - 180, 0, this.fieldX - 180, this.fieldY);
int p = 0;

for (int i = 0; i < this.user_nrs; i++) {
    if (this.user[i].active == true) {
        p++;
        g.setColor (Color.black);
        g.fillOval (this.user[i].x, this.user[i].y, 20, 20);
        g.setColor (Color.green);
        g.fillOval (this.user[i].x + 5, this.user[i].y + 5, 10, 10);
        g.setColor (Color.red);
        g.drawString (this.user[i].name, this.user[i].x, this.user[i].y);
        g.drawString (this.user[i].name + ": " + i, 900, 100 + 10 * p);
    }
}
}

void sporn_kissen (int nr, int x, int y) {
    this.kissen[nr].x = x;
    this.kissen[nr].y = y;
    this.kissen[nr].show = true;
}

void sporn_player (int nr, int x, int y, String name, int dir) {
    this.user[nr].x = x;
    this.user[nr].y = y;
    this.user[nr].name = name;
    this.user[nr].active = true;
}

private static int direction = 0;
private static int speed = 5;
private static String sendUser = name;
private static int sendSpeed;
private static int sendDirection;
private static boolean sendKissen;

// here follows input processing
public void keyPressed (KeyEvent e) {

    if (e.getKeyCode () == KeyEvent.VK_ESCAPE) { this.exit = true; }

    if (e.getKeyCode () == KeyEvent.VK_Q) { this.exit = true; }

    if (e.getKeyCode () == KeyEvent.VK_1) { speed = 1; }
    if (e.getKeyCode () == KeyEvent.VK_2) { speed = 2; }
    if (e.getKeyCode () == KeyEvent.VK_3) { speed = 3; }
    if (e.getKeyCode () == KeyEvent.VK_4) { speed = 4; }
    if (e.getKeyCode () == KeyEvent.VK_5) { speed = 5; }

    if (e.getKeyCode () == KeyEvent.VK_SPACE) { sendKissen = true; }

    if (e.getKeyCode () == KeyEvent.VK_UP) { sendSpeed = speed; }

    if (e.getKeyCode () == KeyEvent.VK_DOWN) { sendSpeed = speed * -1; }

    if (e.getKeyCode () == KeyEvent.VK_LEFT) {
        direction -= 10;
        if (direction < 0) { direction = 350; }
        sendDirection = direction;
    }

    if (e.getKeyCode () == KeyEvent.VK_RIGHT) {
        direction += 10;
        if (direction > 359) { direction = 0; }
        sendDirection = direction;
    }

    if (e.getKeyCode () == KeyEvent.VK_NUMPAD8) {

```

```

    direction = 0;
    sendDirection = direction;
}

if (e.getKeyCode () == KeyEvent.VK_NUMPAD9) {
    direction = 45;
    sendDirection = direction;
}

if (e.getKeyCode () == KeyEvent.VK_NUMPAD6) {
    direction = 90;
    sendDirection = direction;
}

if (e.getKeyCode () == KeyEvent.VK_NUMPAD3) {
    direction = 135;
    sendDirection = direction;
}

if (e.getKeyCode () == KeyEvent.VK_NUMPAD2) {
    direction = 180;
    sendDirection = direction;
}

if (e.getKeyCode () == KeyEvent.VK_NUMPAD1) {
    direction = 225;
    sendDirection = direction;
}

if (e.getKeyCode () == KeyEvent.VK_NUMPAD4) {
    direction = 270;
    sendDirection = direction;
}

if (e.getKeyCode () == KeyEvent.VK_NUMPAD7) {
    direction = 315;
    sendDirection = direction;
}

if (e.getKeyCode () == KeyEvent.VK_RIGHT
    || e.getKeyCode () == KeyEvent.VK_LEFT
    || e.getKeyCode () == KeyEvent.VK_DOWN
    || e.getKeyCode () == KeyEvent.VK_UP
    || e.getKeyCode () == KeyEvent.VK_SPACE
    || e.getKeyCode () == KeyEvent.VK_NUMPAD8
    || e.getKeyCode () == KeyEvent.VK_NUMPAD6
    || e.getKeyCode () == KeyEvent.VK_NUMPAD2
    || e.getKeyCode () == KeyEvent.VK_NUMPAD4
    || e.getKeyCode () == KeyEvent.VK_NUMPAD1
    || e.getKeyCode () == KeyEvent.VK_NUMPAD3
    || e.getKeyCode () == KeyEvent.VK_NUMPAD9
    || e.getKeyCode () == KeyEvent.VK_NUMPAD7
    || e.getKeyCode () == KeyEvent.VK_ESCAPE) {
    System.out.println (sendUser + ":" + sendSpeed + ":" + sendDirection +
        ":" + sendKissen);
    if (exit) System.exit (0);
}
this.my_speed = sendSpeed;
this.dir = sendDirection;
}

public void keyReleased (KeyEvent e) {
    if (e.getKeyCode () == KeyEvent.VK_UP) { sendSpeed = 0; }
    if (e.getKeyCode () == KeyEvent.VK_DOWN) { sendSpeed = 0; }
    if (e.getKeyCode () == KeyEvent.VK_SPACE) { sendKissen = false; }
    this.my_speed = sendSpeed;
    this.dir = sendDirection;
}

public void keyTyped (KeyEvent event) { }

String randomString () {

    char[] vokale = { 'a', 'e', 'i', 'o', 'u' };
    String str = "";

```

```

// A-Za-z: 65-122
// A-Z   : 65-91
// a-z   : 96-122

str += (char) Math.round (Math.random () * 26 + 65);
for (int i = 0; i <= 5; i++) {
    switch (i % 2) {
        case 1:
            str += (char) Math.round (Math.random () * 26 + 96);
            break;
        case 0:
            str += vokale[(char) Math.round (Math.random () * 4)];
            break;
    }
}
return str;
}

public void sleep (int i) {
    try { Thread.sleep (i); } catch (InterruptedException e) { }
}
}
### file screen.java end ###

###file show.java los ###
public class show {

    public static void main (String[]args) {

        screen scr = new screen (500, 500, 10);

        /* Spieler und Kissen setzen/spornen
        scr.sporn_kissen(3, 4, 8);
        scr.sporn_player(4, 9, 10);
        */

        //scr.screen();

        /* alt und nicht funktionstüchtig
        Client in=new Client();
        // Mohit: "10.1.10.115", 80
        String result=in.client("10.1.10.58", 80);
        System.out.println(result);
        */
    }
}

### file show.java end ###

###file user.java los ###
public class user
{
    int x, y = 2;
    int kissen_nrs, hits;
    String name;
    boolean active;
}
### file user.java end ###

```

## 10 Unmaintainable Code – 2009-07-08

Okay.

Wir haben einen Haufen Software, die komischerweise funktioniert. Sie besteht aus einem Haufen Klassen mit mördermäßig vielen Members und soooo vielen ifs...

... und ist unwartbar. Einen Teil konnte ich selber aufräumen.

Das Feld `user` im Client gibt es auch nicht mehr, die Serverantwort wird nunmehr in der `paint`-Routine geparkt.

Die Klasse `Parse` gibt es nicht mehr; sie wurde durch `String [] strings = s.split(":");` ersetzt.

Und das funktioniert auch alles ganz vorzüglich – bis auf den Server. Der antwortet nicht mehr, außer `fps` ist abgelaufen.

Es weiß aber keiner mehr, was `fps` eigentlich ist.

### 10.1 Kurzer Blick auf den Client

```
init
finde deinen usernamen oder stirb
such den server oder stirb
while (true) {
    frag die tasten ab, ob der spieler laufen oder werfen will
    bastele die anfrage (request)
    schicke request, hole response
    lösche bildschirm
    parse response mit s.split() und male dabei
}
```

## 10.2 Kurzer Blick auf den Server – hoppla

```

init (port 6666)
spieleranzahl=0
while (true) {
  warte auf request bis eins kommt
  schlafe 20msec // und benutze nie fps
  parse request mit s.split()
  pruefe request auf plausibilitaet, zb
  wenn speed > 5 dann speed = 5
  wenn speed < -5 dann speed = -5
  wenn er werfen will
    wenn er kissen zum werfen hat
      wirf es
  laufe durch alle spieler
  wenn einer länger als 5sec schläft
    wirf ihn weg (= setz namen auf "")
    und seine kissen
    spieleranzahl--
  ist es derjenige mit dem aktuellen request?
    bewege ihn
    schreib dir auf dass du bewegt hast
  wenn du nix bewegt hast
    erzeuge neuen spieler
    und paar neue kissen
  laufe durch alle kissen
    bewege sie
  wenn eins aus dem spielfeld will
    dann lass es hier liegen
  wenn ende der flugbahn
    setze status liegendes kissen
  wenn ein spieler erwischt ist von fliegendem kissen
    zieh dem spieler punkt ab
    leg das kissen zu boden
    zähl dem schützen punkt dazu
  wenn ein spieler erwischt ist von liegendem kissen // nur ein buchstabe differenz
    zähl dem spieler kissen dazu
    leg das kissen zu boden
    zähl dem schützen punkt dazu
  schick dem spieler die antwort:
    alle mitspieler mit koordinaten, blickrichtung, kissenanzahl
    alle kissen mit koordinaten und status
}

```