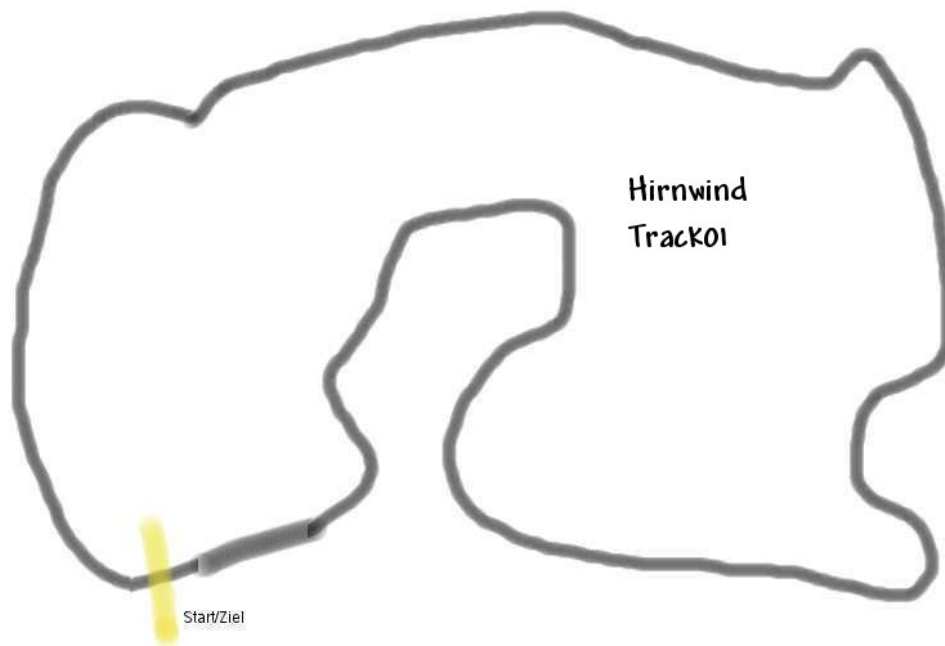


Projekt Hirnwind

Selbstprogrammierte Autos fahren und lösen Probleme

<http://worgtsone.ostermann-rodgau.de/> - [mailto: worgtsone @ hush.com](mailto:worgtsone@hush.com)

Mi 12. Nov 22:28:07 CET 2008 – 6. November 2011



Inhaltsverzeichnis

1	Intro	3
1.1	Single	3
1.2	Paar	3
1.3	Multi	3
2	Regeln	4
3	Besprechung Renn-Tech	6
3.1	Bandbreite	6
3.2	Rennverlauf	6
3.3	Identitätsklau	6
3.4	ver 0.01	6

Disclaimer

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.

1 Intro

Habe 6 Sekunden einen gelben Plastik-Quader mit 3 Sensor-Eingängen und 3 Motor-Ausgängen in der Hand gehalten. Wird durch 6 Batterien AA betrieben und üblicherweise in NQC (not quite C) programmiert.

Die Dinger sind teuer, und die Sensoren haben – err – interessante Kalibrierungskurven.

Also baun wir das nach in Java.

Nun soll das Ganze ja auch üben. Die Programmierung soll jedem Mitspieler überlassen bleiben (vielleicht in eine Klasse `FahrAlgorithmus`?).

Zuschauer sind willkommen, die gucken in einem Browser-Applet mit 10 fps zu.

Fahrer und Zuschauer brauchen Username und Pass – paperlapapp, bei 5 Fahrern und 10 Zuschauern ist Schluß. Basta.

Die Autos, Sensoren etc. sind bei allen gleich. Es gibt 3 Standard-Matten und Standard-Spiele:

1.1 Single

Hockenheim: Rase 10 Runden auf der grau-schwarzen Linie.

LostInCyberspace: Finde durch das Labyrinth.

Hansel: Sammle möglichst schnell alle 40 Brotkrumen (die bleiben liegen) oder Käfer (die krabbeln herum) oder Schrauben (mit einem kleinen Greifer) auf.

1.2 Paar

HaseUndIgel: Der Igel fährt möglichst lange dem Hasen davon.

1.3 Multi

DeathMatch: Finde durch das Labyrinth. Lege Minen, die beim Drüberfahren oder nach 10 sec explodieren.

2 Regeln

1. Das Spielfeld ist so groß wie die darunterliegende png-Grafik, vorzugsweise 800x600 Pixel.
Der Server präsentiert zunächst die beliebtesten Spielfelder und Matten, man kann aber (nach Anmeldung) seine eigenen Lieblinge setzen.
2. Ein Auto kann nicht das Spielfeld verlassen. Wenn es das tut, wird es senkrecht zur Spielfeldwand hineingeschoben.
3. Ein Auto kann max. 8 Sensoren haben.
4. Es gibt Lichtsensoren. Sie tasten die Helligkeit des Bodens +- 15 Pixel ab und geben einen Wert von 0 (schwarz) bis 255 (weiß) zurück.
Der Sensoren-Wert wechselt manchmal erst nach 100msec.
5. Es gibt Tastsensoren. Wenn sie gegen ein Hindernis stoßen, sind sie ON, sonst OFF. Sie haben die Methode `boolean isOn()`.
Der Sensoren-Wert wechselt manchmal erst nach 100msec.
6. Es gibt max. eine Videokamera. Sie produziert pro Sekunde ein Bild 320x240 Pixel, Graustufen von 0..255, und ist irgendwo fest (nicht drehbar) am Auto montiert.
(Armer Rechner, der die Bilder rendern muß...)
7. Das Auto kann `vor()`, `back()`, `right()`, `left()`.
8. `vor()` bewegt das auto ein pixel vor und braucht dafür 1/10 sec.
9. `back()` bewegt das auto ein pixel rückwärts und braucht dafür 1/10 sec.
10. `right()` dreht das Auto um seinen Mittelpunkt 5 Grad nach rechts.
11. `left()` dreht das Auto um seinen Mittelpunkt 5 Grad nach links.
12. Alle Sensoren können nur an den vier Fahrzeugecken montiert werden.
13. An einer Fahrzeugecke können beliebig viele Sensoren montiert werden.
14. Die Fahrzeugecken heißen: 1 li vo, 2 re vo, 3 li hi, 4 re hi.
15. Der Programmierer schreibt seinen `FahrAlgorithmus`. Der hat die Methoden `init()` und `go()`.
`init()` wird zu Anfang ausgeführt und montiert die Sensoren ans Auto. Es sagt also, welcher Sensor an welcher Ecke mit welchem Eingang verbunden wird.
`init()` darf max. 1000 msec dauern – wenn nicht, wird das Auto zerstört.
`go()` läuft anschließend. Es darf die Sensoren pollen.
16. Um Server-Ressourcen zu schonen, laufen die Autos als ClosedSource auf den Clients. Sie empfangen ihre Befehle von `go()` und melden Position und Richtung an den Server (vergl. "Kissenschlacht" , ebenfalls von worgtsone.ostermann-rodgau.de).
17. Menschliche Eingriffe sind nicht geplant, können aber von spielwütigen Programmierern in `FahrAlgorithmus` eingebaut werden.

18. Spielablauf: Zunächst wird das Spiel geladen, zB LinieFahren. Es lädt dann das Spielfeld und die Autos und berichtet an den Server und bekommt eine Spiel-ID.
Anschließend fahren die Autos los. Bis jemand gewonnen hat oder das Spiel sonstwie zu Ende ist.
Hier sollten die Highscores etc. vom Server festgehalten werden. Anschließend ist das Spiel vorbei, und alles wird zerstört.
19. Paar-Spiele und Multi-Spiele kennen folgende Zustände: 1 init, 2 waitForPartner (wantedPartner), play(), destroyAll().

3 Besprechung Renn-Tech

3.1 Bandbreite

Rennen fahren macht am meisten Spaß mit fernwirkenden Sensoren. Ein vollständiges Bild verschlingt zu viel Bandbreite – also wird der Server 6 Zeilen zurückliefern:

position und richtung des autos

minimale entfernung zum hindernis in 0° sowie +-10 und +-20°.

*

Ein möglicher Fahralgorithmus wäre dann : lenk in die Richtung wo das Hindernis am weitesten ist, und zwar mit Bleifuß.

3.2 Rennverlauf

Der Server soll nacheinander drei oder vier Zustände annehmen:

– VOR dem Rennen (10sec lang)

Server nimmt Anmeldungen mit Benutzernamen entgegen. Von jeder IP nur eine. Und gibt ein Passwort heraus, eine 20stellige Zahl müßte reichen.

Max. 10 Meldungen, sonst server full, try again later.

– WÄHREND dem Rennen

Server reagiert auf GET-Kommandos der Form Benutzername – Passwort – Gas – Lenkwinkel.

Server kontrolliert Kollisionen mit Hindernissen (nicht mit gegnerischen Autos) und Zieldurchfahrt(en).

Nach dem ersten Auto gibts noch 20sec Zeit die Ziellinie zu überqueren, der Rest wird als nicht angekommen gewertet.

Bei mehr als 10 Anfragen pro Sekunde und Benutzer geht der Server 1 sec schlafen.

– NACH dem Rennen

Server zeigt 20sec Liste mit Zeiten und Benutzern.

– ZWISCHEN den Rennen

Anzeige : Nächstes Rennen in XX min YY sec.

3.3 Identitätsklau

Der User Kurt sollte den ganzen Tag so heißen dürfen...

Falls er sich als Kurt anmeldet und aus H:\secret.txt die alte Zahl wieder nennen kann, wird jeder andere Kurt herausgeschmissen ("name is occupied"), darf sich aber innerhalb der 10sec unter anderem Namen wieder anmelden.

3.4 ver 0.01

```
import java.awt.*;

class hirnwind02 {
    public static void main (String[]args) {
        System.out.println ("hirnwind02 by worgtsone.scienceontheweb.net");
        new hwserver();
        //    new hwclient();
    }}

```

```

class hwserver extends Frame{
    int N=9;
    int field=399;
    int frame=99;
    int carsize=14;
    Auto[] a = new Auto[N];
    hwserver () {
        for (int i=0; i<9; i++)
            a[i]=new Auto(field);
        this.setSize(field+2*frame,field+2*frame);
        this.setVisible(true);
        while (true) {
            repaint();
            try {Thread.sleep(29);} catch(Exception e){}
        }
    }
    public void paint(Graphics g) {
//      g.setColor(Color.BLACK);
//      g.fillRect(0,0,999,999);
//      g.setColor(Color.YELLOW);
        g.drawLine(frame,frame, field+frame, frame);
        g.drawLine(frame,frame, frame, field+frame);
        g.drawLine(frame+field,frame, frame+field, field+frame);
        g.drawLine(frame,frame+field, field+frame, frame+field);
        for (int i=0; i<9; i++){
if (Math.random(<0.1) a[i].ri=(Math.random()-0.5)/9;
a[i].move();
//      a.sprich();
double size=carsize;           // autogroesse
int x1=(int)(frame+a[i].x+size*Math.cos(a[i].wi));
int y1=(int)(frame+a[i].y+size*Math.sin(a[i].wi));
int x2=(int)(frame+a[i].x+size*Math.cos(a[i].wi+2.8));
int y2=(int)(frame+a[i].y+size*Math.sin(a[i].wi+2.8));
int x3=(int)(frame+a[i].x+size*Math.cos(a[i].wi-2.8));
int y3=(int)(frame+a[i].y+size*Math.sin(a[i].wi-2.8));
//      g.drawOval((int)a.x-4, (int)a.y-4, 9,9);
g.drawLine(x1,y1,x2,y2);
g.drawLine(x3,y3,x2,y2);
g.drawLine(x1,y1,x3,y3);
        }
    }
}

class Auto{
    public double ri; // richtung, 0=geradeaus
    public double sp; // speed, max=5
    public double x; // gehoert eingepackt
    public double y;
    public double wi;
    double field;
}

```

```
Auto(double newField) {
    this.field=newField;
    ri=0.5;
    sp=1;
    x=59; y=59; wi=0;
}
public void move(){
    wi+=ri;
    x+=sp*Math.cos(wi);
    y+=sp*Math.sin(wi);
    if (x<0) x+=field;
    if (y<0) y+=field;
    if (x>field) x-=field;
    if (y>field) y-=field;

}
public void sprich(){
    String s=" ";
    System.out.println (x+s+y+s+wi+s+sp+s+ri);
}
}
```