33. Bundesweiter Informatikwettbewerb – Aufgabe 5

Team „ByteSector“

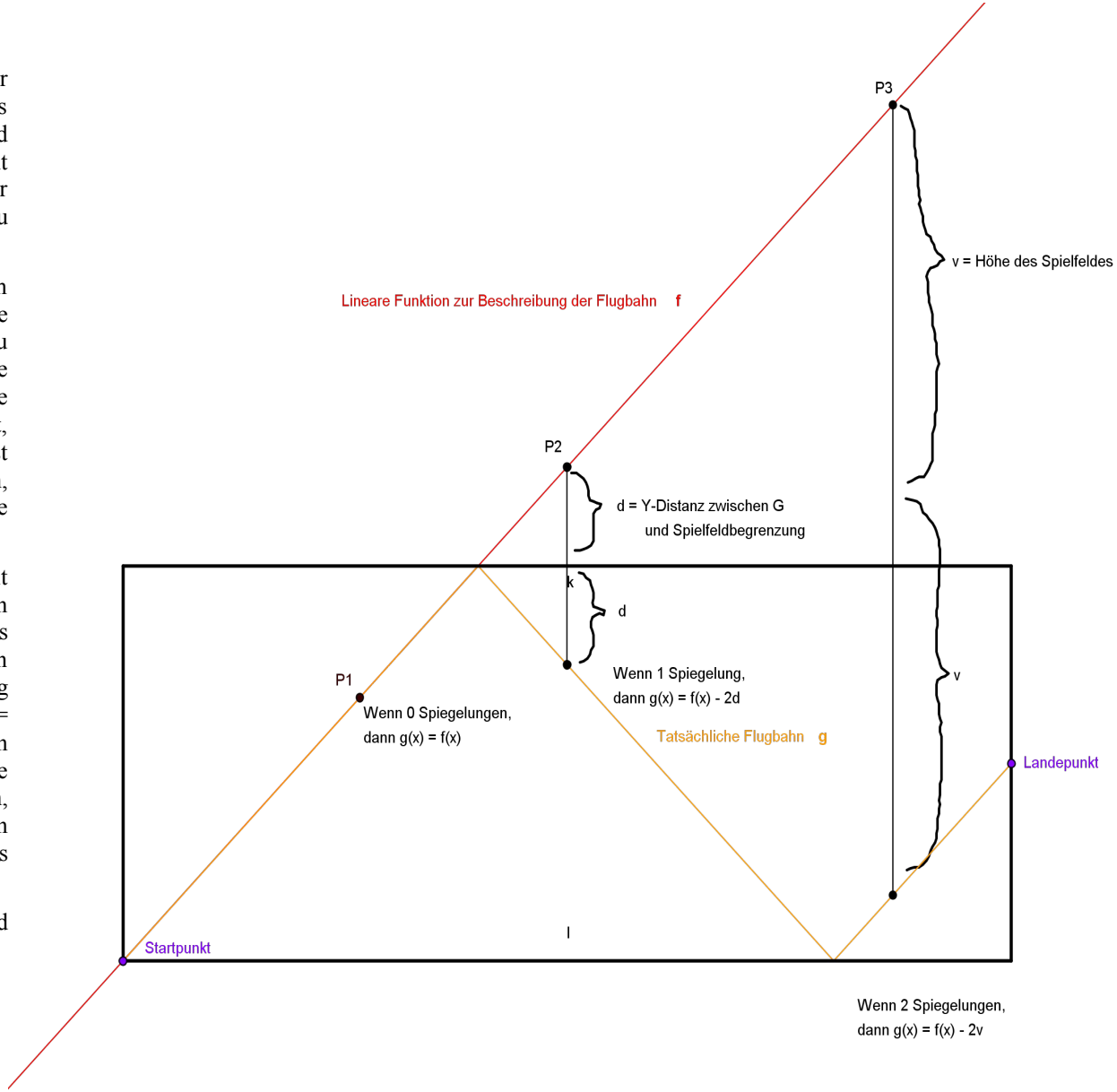
Lösungsidee:

Um ein Pong-Spiel zu gewinnen, ist es erforderlich, mehr Schläge als der Gegner zu treffen. Um dies zu erreichen muss einerseits der eigene Schlag möglichst oft erfolgreich sein und gleichzeitig der gegnerische Schlag möglichst schwer gemacht werden. Ein Schlag wird dann als schwer angesehen, wenn der Schläger eine möglichst große Distanz bewältigen muss, um zu der Position, von der er den Ball erfolgreich trifft, zu gelangen.

Da der Ball immer schneller wird, wird das Zeitintervall, in dem der Ball eine der Spielfeldseiten erreicht, immer kürzer. Die Schläger haben folglich immer weniger Zeit sich neu zu positionieren, d.h. es verringert sich die maximale Distanz, die ein Schläger bis zum Aufprall des Balles zurücklegen kann. Die KI, welche als erste den Ball nicht mehr rechtzeitig erwischt, verliert den Ballwechsel. Damit eine KI sich also möglichst lange im Duell halten kann, muss sie möglichst früh anfangen, ihren Schläger richtig zu positionieren. Um dies zu tun, muss die KI wissen, auf welcher Höhe der Ball aufkommen wird.

Die Flugbahn des Balles verläuft zwischen Kollisionen mit Schlägern und Spielfeldbegrenzung linear, lässt sich also durch eine lineare Funktion beschreiben. Da der Ball während eines Fluges in eine horizontale Richtung seine Flugbahn nur durch eine Kollision mit der oberen oder unteren Spielfeldbegrenzung ändert und diese Änderung der Regel „Einfallswinkel = Ausfallswinkel“ folgt, also eine Achsenspiegelung der Flugbahn ist, lässt sich die Flugbahn des Balles von einer Spielfeldseite zur anderen ebenfalls durch eine lineare Gleichung beschreiben, wenn man beachtet, dass man bei der Konvertierung zwischen Funktion und tatsächlicher Flugbahn ggf. die Punkte etwas verschieben muss.

Wie genau diese Verschiebung verläuft, veranschaulicht das Bild rechts:



Da die tatsächliche Flugbahn an der Spielfeldbegrenzung gespiegelt wird, sind die Punkte auf der tatsächlichen Flugbahn hinter dem Punkt der Kollision mit der Spielfeldbegrenzung an dieser gespiegelt, also 2d vom Wert der linearen Funktion f(x), wenn d die Differenz aus dem Y-Wert des Punktes und dem Y-Wert der Spielfeldbegrenzung ist, entfernt.

Wird die Flugbahn ein zweites Mal gespiegelt, hebt sich die Achsenspiegelung auf und es ist nur noch eine Verschiebung vorhanden. Diese beträgt genau zwei Spielfeldhöhen. Nennt man diese Spielfeldhöhe v, lässt sich für die Übersetzung für die Werte der linearen Funktion auf die Werte der tatsächlichen Flugbahn folgendes formulieren:

Bei gerader Anzahl n an Spiegelungen:

$$g(x)=f(x)-2\mathrm{v}*n/2$$

Bei ungerader Anzahl n an Spiegelungen:

$$g(x)=f(x)-2\mathrm{d}-2\mathrm{v}*(n-1)/2$$

Nun ist das Verhältnis zwischen linearer Funktion zur Beschreibung der Flugbahn und tatsächlicher Flugbahn bekannt, wie diese Funktion aber berechnet wird, steht noch offen. Würden die angezeigten Positionen des Balles der tatsächlichen Position des Balles entsprechen, würde man die Funktion der Form $f(x)=mx+b$ simpel mit den Formeln

$m=\frac{\Delta y}{\Delta x}$ und $b=y-mx$ durch Einsetzen von erfassten Positionen bestimmen können.

Da die angezeigten Positionen allerdings nur eine Annäherung an die tatsächliche Position des Balles ist, muss eine lineare Regression mit den gesammelten Punkten durchgeführt werden. Mit der Methode der kleinsten Quadrate kann eine solche Regression bewältigt werden. Die gesuchte Methode ist die, in der die Quadrate der Residuen der Punkte, am geringsten ist, sprich der Fehler E minimal. Ist ein tatsächlicher Wert y_i , ein errechneter $\tilde{y}_i$, die Anzahl an Werten n und hat der erste Wert den Index 1, wird der Fehler E mit der Formel $E=\sum_{i=1}^{n}(y_i-\tilde{y}_i)^2$ berechnet. Gesucht ist nun also das Minimum der Funktion

$$E(m;b)=\sum_{i=1}^{n}(y_i-\tilde{y}_i)^2$$

$$E(m;b)=\sum_{i=1}^{n}(y_i-(mx_i+b))^2$$

$$E(m;b)=\sum_{i=1}^{n}(y_i-mx_i-b)^2$$

Um dieses Minimum herauszufinden, muss die Funktion nun abgeleitet und mit null gleichgesetzt werden. Die beiden Ableitungen sehen wie folgt aus:

$$E'_m(m;b)=\sum_{i=1}^{n}\left(2(y_i-mx_i-b)(-x_i)\right) \qquad E'_b(m;b)=\sum_{i=1}^{n}\left(2(y_i-mx_i-b)(-1)\right)$$

Setzt man diese Ableitungen gleich null, kommt man nach einiger Umformung und Einsetzung zu den Termen

$$m=\frac{(\sum_{i=1}^{n}(x_i y_i)n-\sum_{i=1}^{n}x_i\sum_{i=1}^{n}y_i)}{(\sum_{i=1}^{n}x^2{}_i n-(\sum_{i=1}^{n}x_i)^2)} \qquad b=\frac{(\sum_{i=1}^{n}y_i\sum_{i=1}^{n}x^2{}_i-\sum_{i=1}^{n}(y_i x_i)\sum_{i=1}^{n}x_i)}{(\sum_{i=1}^{n}x^2{}_i n-(\sum_{i=1}^{n}x_i)^2)}$$

Wenn wir nun unsere Sammlung an Punkten, die wir während der Beobachtung des Balles erstellt haben, benutzen, um diese Terme zu lösen, erhalten wir die Steigung m und die Verschiebung auf der Y-Achse b, mit denen wir den Term der Ausgleichsgeraden bilden können.

Die Erstellung dieses Funktionsterms wird in jedem Spielzug wiederholt, da eine neue Position des Balles bekannt ist und die Regression einen weiteren Punkt einbeziehen kann, was die Annäherung präziser macht.

Ist nun schließlich ungefähr bekannt, wo der Ball auftreffen wird, bewegt die KI den eigenen Schläger in Richtung dieser Position, sofern es sich um einen ankommenden Ball handelt. Handelt es sich um einen Ball, der in Richtung der gegnerischen Hälfte fliegt, wäre es intelligent, anhand der aktuellen Schlägerposition des Gegners und dem ungefähr bekannten Landepunkt des Balles auf der gegnerischen Seite, die aus dem Schlag des Gegners resultierende Flugbahn vorauszuberechnen und den eigenen Schläger entsprechend zu positionieren. Für den Fall, dass der Gegner eine solche Strategie ebenfalls anwendet, kann versucht werden, den Teil des Schlägers, der den Ball trifft, per Zufall zu variieren, um dadurch die resultierende Flugbahn drastisch zu ändern. Eine Vorbereitung während eines entfernenden Balles wurde in dieser Lösung aus Zeitgründen allerdings nicht umgesetzt und der Schläger wird während des gegnerischen Zuges auf die mittlere Höhe zurückgefahren.

Umsetzung:

Die Umsetzung erfolgt mithilfe des Turniersystems des BwInfs in der Sprache Java. Die KI wurde „**Wysb**“ getauft.

Am Anfang jedes Zuges wird eine Routine zum Empfang der Parameter der *zug*-Methode durchlaufen, welche hier *setup* genannt ist. In dieser wird aus der erhaltenen **id**, dem erhaltenen **Zustand** und dem erhaltenen **Zug** die beiden Schläger und der Ball ausgelesen, um sie später im Programm benutzen zu können.

Ist die *setup*-Routine abgeschlossen, folgt die Berechnung der Flugbahn, genannt *flugbahnUpdate*, und schließlich die Anweisung an den Schläger mit der *bewegung*-Methode.

Berechnung der Flugbahn

```
private void flugbahnUpdate(){
        // Aktuelle Position eintragen, Entspiegelung erfolgt später
        int[] curKoords = {ball.xKoordinate(), ball.yKoordinate()};
        koordsUn.add(curKoords);

        // Falls dies der erste erfasste Punkt ist, kann keine Flugbahn berechnet werden, also wird übersprungen
    if (ersteMessung){
        ersteMessung = false;
    }
    else {
        // Für schnellere Eingaben aktuelle Koordinaten erneut speichern
                int x = ball.xKoordinate(),
                        y = ball.yKoordinate();

                // Falls keine Veränderung in der X-Koordinate zu sehen ist, kann keine horizontale Richtung bestimmt werden, also nicht geklärt werden,
                // ob es sich noch um die selbe Flugbahn handelt. In diesem Falle wird die Messung ignoriert.
        if (x-vorherigesX == 0){
                koordsUn.remove(koordsUn.size()-1);
                return;
        }

        // Richtung erkennen
        zuMir = vorherigesX > x;
        nachUnten = vorherigesY < y;

                // Falls sich die horizontale Richtung geändert hat, wurde der Ball entweder neu eingeworfen, oder hat einen Schläger getroffen. In diesem Falle
                // gibt es eine neue Flugbahn und die alten Werte können verworfen werden.
                // Falls sich die vertikale Richtung geändert hat, bedeutet dies, dass entweder die obere oder untere Spielfeldbegrenzung oder ein Schläger an einem der
                // äußeren
                // Teile getroffen wurde.
                // Wurde ein Schläger getroffen, ändert sich außerdem noch die horizontale Richtung. Dieser Fall wird also schon bearbeitet.
                // Wurde eine Spielfeldbegrenzung getroffen, kommt es zu der in der Dokumentation beschriebenen Spiegelung der Punkte an dieser Spielfeldgrenze.
                // In diesem Falle werden jetzt die Werte geändert, auf deren Basis die Punkte entspiegelt werden.
                //
                // Sollte dies jedoch die erste Erfassung der Richtung sein, d.h. es wurde eine neue Flugbahn durch Einwurf des Balles oder Kollision mit einem Schläger
                // erstellt,
                // ist eine Richtungsänderung nicht zu überprüfen.
                if (ersteRechnung){
                        // Aktuelle horizontale Richtung zwecks späteren Vergleichs merken
                        vorherigesZuMir = zuMir;
                        // Ist die vertikale Richtung unklar, weil der aktuelle und der vorherige Y-Wert übereinstimmen, wird die Überprüfung der vertikalen
                        // Richtungsänderung verzögert
                        if (vorherigesY == y)
                                unklarVorherigesUnten = true;
                        vorherigesNachUnten = nachUnten;
                        ersteRechnung = false;
                } else {
                        // horizontale Richtungsänderung überprüfen
                if (vorherigesZuMir != zuMir){
                        vorherigesZuMir = zuMir;

                        // Neue horiz. Richtung => Neue Flugbahn => Alle alten Punkte verwerfen => Nur aktueller Punkt (also nur einer) übrig => Flugbahnberechnung
                        // abbrechen
                        int[] aktuell = {x,y};
                        koordsUn = new ArrayList<int[]>();
                        koordsUn.add(aktuell);

                        // Spiegelung zurücksetzen, da es eine neue Flugbahn gibt
                        spiegelachse = -1;
```

```
                verschiebung = 0;
                ersteRechnung = true;

                // Neueinwurf oder Schlägerabprall => ggf. neue Y-Richtung => Alte Y-Richtungsaufzeichnungen verwerfen
                unklarVorherigesUnten = true;     // Führt dazu, dass in der nächsten Messung der alte Wert als ungültig angesehen wird
                return;
        } else {
                // vertikale Richtungsänderung überprüfen
                if (vorherigesY != y){                                                  // Falls die aktuelle Richtung unklar ist, Richtungsüberprüfung
                                                                                        // überspringen
                        if (unklarVorherigesUnten){                                     // Falls die vorherige Richtung noch unklar ist (was durch obiges
                                                                                        // geschehen kann),
                                vorherigesNachUnten = nachUnten;                        // diese Richtung, aber klar,
                                                                                        // diese Richtung als vorherige speichern und Richtungsüberprüfung
                                                                                        // überspringen
                                unklarVorherigesUnten = false;
                        } else {                                                                // Falls vorherige und aktuelle Richtung klar,
                                                                                                //Richtungsüberprüfung durchführen
                                if (vorherigesNachUnten != nachUnten){
                                        // Ist noch keine Spiegelachse gesetzt, war die Anzahl getroffener Spielfeldbegrenzungen gerade
                                        // => Jetzt wird die Spiegelachse gesetzt
                                        if (spiegelachse == -1)
                                                // Fliegt der Ball nach dem Abprall nach unten, wurde die obere Spielfeldbegrenzung (y=0) getroffen
                                                if (nachUnten)
                                                        spiegelachse = 0;
                                                else
                                                        spiegelachse = 59;
                                        // Ist bereits ein Spiegelachse gesetzt, war die Anzahl getroffener Spielfeldbegrenzungen ungerade
                                        // => Die Spiegelachse wird gelöscht und die Verschiebung erhöht
                                        else {
                                                // War die Spiegelachse vorher die obere Spielfeldbegrenzung,
                                                // muss der Punkt um 2 Spielfeldhöhen nach oben verschoben werden,
                                                // um in der Punktewolke der "übersetzten" Funktion zu liegen
                                                if (spiegelachse == 0)
                                                        verschiebung -= 118;
                                                else
                                                        verschiebung += 118;
                                                spiegelachse = -1;
                                        }

                                }
                                // Aktuelle vertikale Richtung zwecks späteren Vergleichs merken
                                        vorherigesNachUnten = nachUnten;
                        }
                }
        }
        }

        // Für die Berechnung der Steigung und der Verschiebung auf der Y-Achse wird
        // die aktuelle Y-Koordinate nun entspiegelt und neu eingetragen, wobei der wirkliche Wert überschrieben wird
        int[] curKoordsUn = {x, ungespiegelt(y)};
        koordsUn.remove(koordsUn.size()-1);
        koordsUn.add(curKoordsUn);

        // Berechnung der Steigung und der Verschiebung auf der Y-Achse
        calcMB();

        // Berechnung ders Landepunktes des Balls
    landepunkt = calcLandepunkt();
}
```

```
    // Aktuelle Punkte zwecks späteren Vergleichs merken
    vorherigesX = ball.xKoordinate();
    vorherigesY = ball.yKoordinate();
}
```

Berechnung der Steigung und der Verschiebung auf der Y-Achse

```
private void calcMB(){
        // Erster Wert wird als Nullpunkt betrachtet
        // => Alle Werte werden verschoben

        // Summe Xi berechnen
        int summeX = 0;
        for (int[] curKoords : koordsUn)
                summeX += curKoords[0];

        // Summe Yi berechnen
        int summeY = 0;
        for (int[] curKoords : koordsUn)
                summeY += curKoords[1];

        // Summe XiYi berechnen
        int summeXY = 0;
        for (int[] curKoords : koordsUn)
                summeXY += curKoords[0] * curKoords[1];

        // Summe Xi² berechnen
        int summeXQuadrat = 0;
        for (int[] curKoords : koordsUn)
                summeXQuadrat += Math.pow(curKoords[0],2);


        // Erklärung siehe Dokumentation
        double c = koordsUn.size() * summeXQuadrat - Math.pow(summeX, 2);
        m = (koordsUn.size() * summeXY - summeX * summeY) / c;
        b = (summeXQuadrat * summeY - summeX * summeXY) / c;
}
```

Berechnung des Landepunktes

```
private double calcLandepunkt(){
        // Ziel-X-Koordinate aussuchen, dann Y-Koordinate an dieser Stelle ausrechnen
        int ziel;
        if (zuMir)
                ziel = 0;
        else
                ziel = 64;

        // Landepunkt erstmals berechnen
        double landepunkt = m * ziel + b;

        // Falls ausgrerechnete Y-Koordinate außerhalb des Spielfeldes (0-59) liegt, bedeutet dies, dass der Ball vor dem Erreichen der Ziel-X-Koordinate
        // noch mindestens einmal an einer horizontalen Spielfeldbegrenzung abprallt
        double tempM = m, tempB = b;
        // Solange der Landepunkt nicht innerhalb des Spielfeldes liegt
        while (landepunkt < 0 || landepunkt > 59){

                // Bestimmen, an welcher Spielfeldbegrenzung der Ball abprallen wird
                double limit;
                if (landepunkt < 0)
                        limit = 0;
```

```
                else
                        limit = 59;

                // Funktion spiegeln: Steigung mit -1 multiplizieren und Verschiebung auf der Y-Achse an entsprechender Spiegelachse spiegeln
                tempM = tempM*-1;
                double abstand = tempB - limit;
                tempB += abstand * -1 * 2;

                // Landepunkt mit gespiegelter Funktion erneut bestimmen
                landepunkt = tempM * ziel + tempB;
        }
        // Endgültigen Landepunkt zurückgeben
        return landepunkt;
}
```