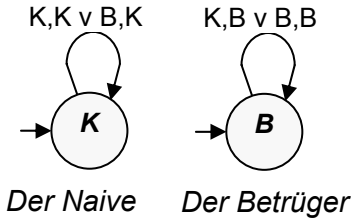
6.2 Zweidimensionale Gitterautomaten

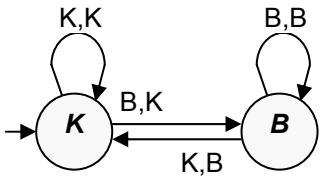
Wir wollen einen zellulären Automaten bauen, der auf dem *Gefangenendilemma* aufbaut, aber etwas abgewandelt auf den *Handel im Internet*. Das Verhalten der Handelspartner wird durch endliche Automaten simuliert, die auf einem in beiden Dimensionen abgeschlossenen Gitter sitzen und innerhalb einer *Moore-Nachbarschaft* (den acht Gitterzellen rundherum) Handel mit den Partnern treiben. Sie tauschen – wie im Internet üblich – Waren gegen Geld. Dabei gibt es unterschiedliche Arten von Geschäftpartnern:

- *Naive* kooperieren immer, liefern also den korrekten Gegenwert.
- *Betrüger* kooperieren nie.
- *Gewitzte* kooperieren anfangs und reagieren danach so, wie der Partner beim letzten Mal.

Wir können das Verhalten der Handelspartner durch Zustandsdiagramme beschreiben:

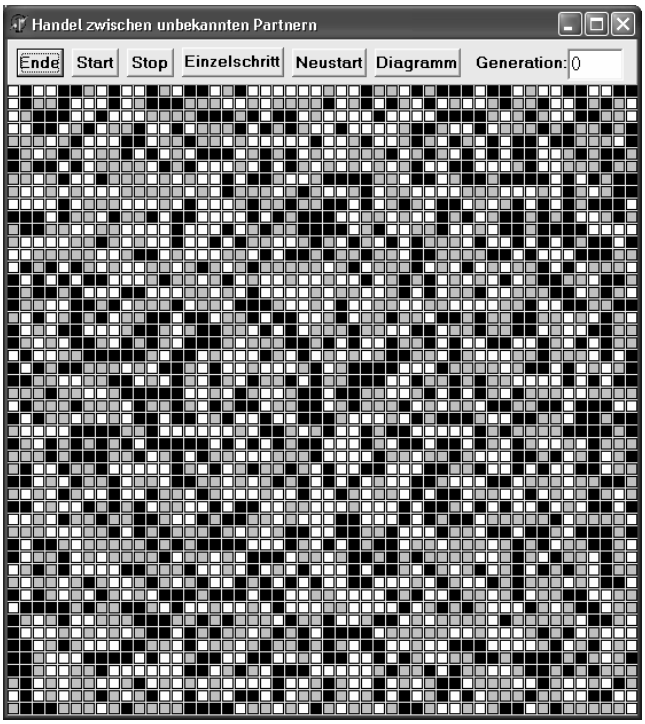


Der Naive *Der Betrüger*



Der Gewitzte

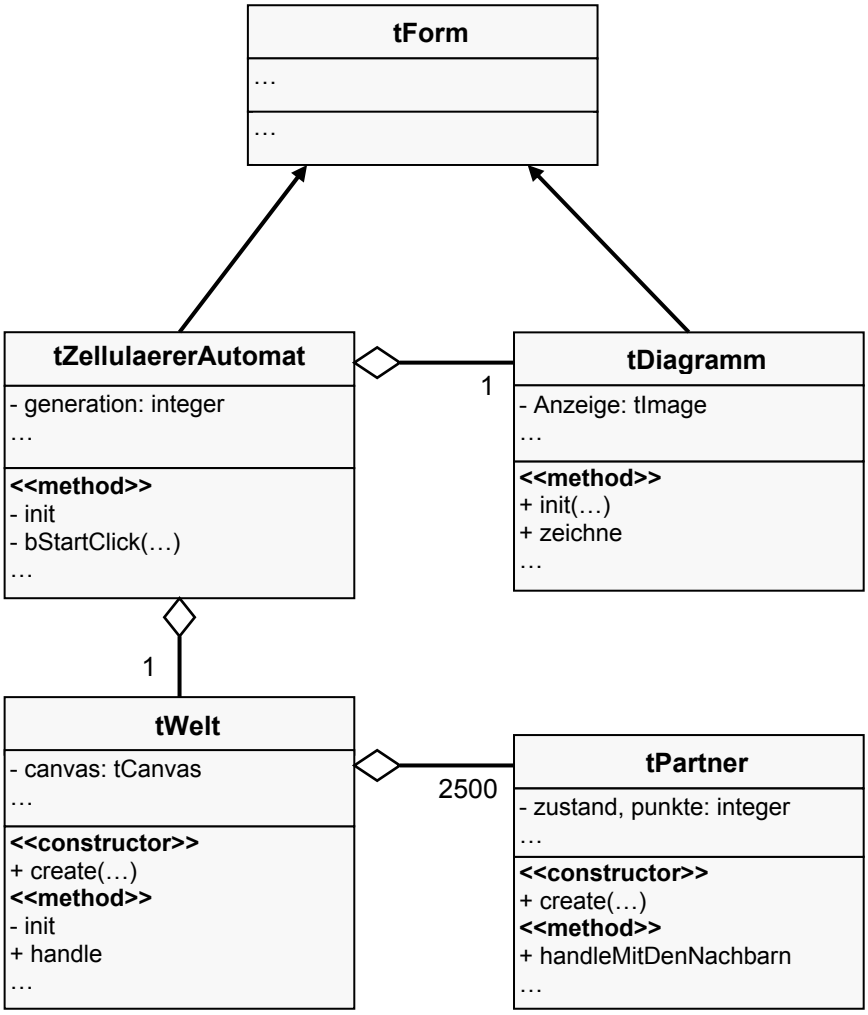
Dabei bedeutet *K*: kooperieren und *B*: betrügen. Die Zustände wurden entsprechend benannt.



Ordnen wir solche Automaten in einem Gitter an, verteilen sie zufällig und färben sie entsprechend ihrem Zustand (weiß als „Naiver“, schwarz als „Betrüger“ oder grau als „Gewitzter“) ein, dann erhalten wir ein Bild ähnlich dem oberen.

Der weitere Ablauf ist einfach: Zuerst handeln alle Partner einmal mit ihren Nachbarn aus der Moore-Nachbarschaft. Danach bewerten alle Partner den Erfolg ihrer Nachbarn. Als Opportunisten übernehmen sie den Zustand des erfolgreichsten Nachbarn oder behalten ihren Zustand bei, wenn sie selbst besser waren.

Zur Realisierung des Gitterautomaten wird eine Automatenklasse – die *Partner* – in einer eigenen Unit vereinbart und darauf aufbauend eine *Welt* (auch in einer Unit), die mit einem Gitter aus Automaten hantiert. Beides wird von der Programmoberfläche des *zellulären Automaten* gesteuert. Zusätzlich wollen wir die zeitliche Entwicklung des Systems in einem *Diagramm* darstellen. Zusammenhänge werden als *ENTHÄLT*-Beziehungen realisiert. Der Automat und das Diagramm sind Tochterklassen von Formularen.



Die programmtechnischen Details des Problems bieten keine Neuigkeiten und werden deshalb hier nicht aufgeführt. Interessant ist eigentlich nur, wie die Bitte um „Handel mit den Nachbarn“ zu interpretieren ist. Der Automat muss aufpassen, dass er nicht über die Grenzen der Welt hinausgreift. In unserem Fall sollen die Randautomaten ihre Kollegen auf der anderen Seite als Partner betrachten. Wir beschreiben also eine „Toruswelt“.

```
procedure tPartner.handelMitDenNachbarn;
var xlinks,xrechts,yoben,yunten: integer;
begin
if xpos = 0 then xlinks  := uwelt.xmax else xlinks := xpos-1;
if xpos = uwelt.xmax then xrechts  := 0 else xrechts := xpos+1;
if ypos = 0 then yoben  := uwelt.ymax else yoben := ypos-1;
if ypos = uwelt.ymax then yunten  := 0 else yunten := ypos+1;
punkte := ergebnisMit(welt.elemente[xlinks,ypos],1);
punkte := punkte+ergebnisMit(welt.elemente[xlinks,yoben],2);
punkte := punkte+ergebnisMit(welt.elemente[xpos,yoben],3);
punkte := punkte+ergebnisMit(welt.elemente[xrechts,yoben],4);
punkte := punkte+ergebnisMit(welt.elemente[xrechts,ypos],5);
punkte := punkte+ergebnisMit(welt.elemente[xrechts,yunten],6);
punkte := punkte+sergebnisMit(welt.elemente[xpos,yunten],7);
punkte := punkte+ergebnisMit(welt.elemente[xlinks,yunten],8);
end;
```

die Toruswelt

2	3	4
1		5
8	7	6

Natürlich wissen wir jetzt noch nicht, wie das „Handelsergebnis“ bestimmt wird. Dafür müssen die Automaten ihre Partner befragen, wie die denn zu reagieren gedenken – und diese Reaktion ergibt sich aus der Vorgeschichte, die wiederum bei jedem Automaten z. B. in einem Feld von Wahrheitswerten (*wasDerNachbarTat*) gespeichert sein muss. Der Feldindex ergibt sich aus der Lage der Automaten zueinander, die wir durch Zahlen beschreiben. Der befragte Automat berechnet zuerst die „inverse“ Position – denn dort befindet sich der Fragesteller, und beantwortet dann die Frage, was er zu übergeben gedenkt.

```
function tPartner.ergebnisMit(p:tPartner;lage:integer): integer;
var inverseLage        : integer;
    derAndereKooperiert: boolean;
    ichKooperiere      : boolean;
begin
case lage of
  1..4: inverseLage := 4+lage;
  5..8: inverseLage := lage-4;
  end;
derAndereKooperiert    := p.kooperiertMit(inverseLage);
wasDerNachbarTat[lage] := derAndereKooperiert;
ichKooperiere          := kooperiertMit(lage);
```

inverse Lage berechnen

Nachsehen, was der Nachbar tut

```
if ichKooperiere
  then if derAndereKooperiert
         then result := random(9)+2
         else result := 0
  else if derAndereKooperiert
         then result := random(20)
         else result := random(2);
end;
```

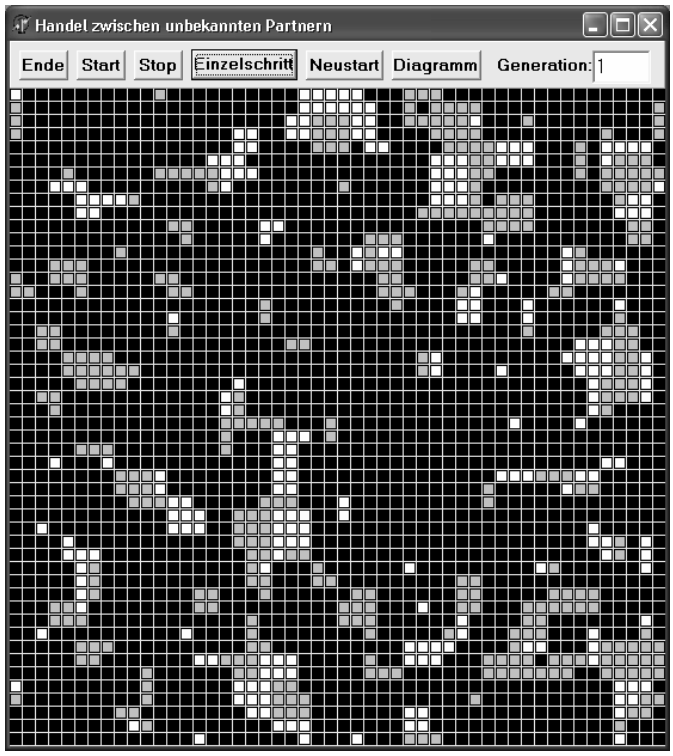
jetzt reagieren: hier findet man die Gewichtsfaktoren der vier möglichen Situationen (mit etwas Zufall)

Kooperiert der Partner? Das hängt von dessen Zustand ab!

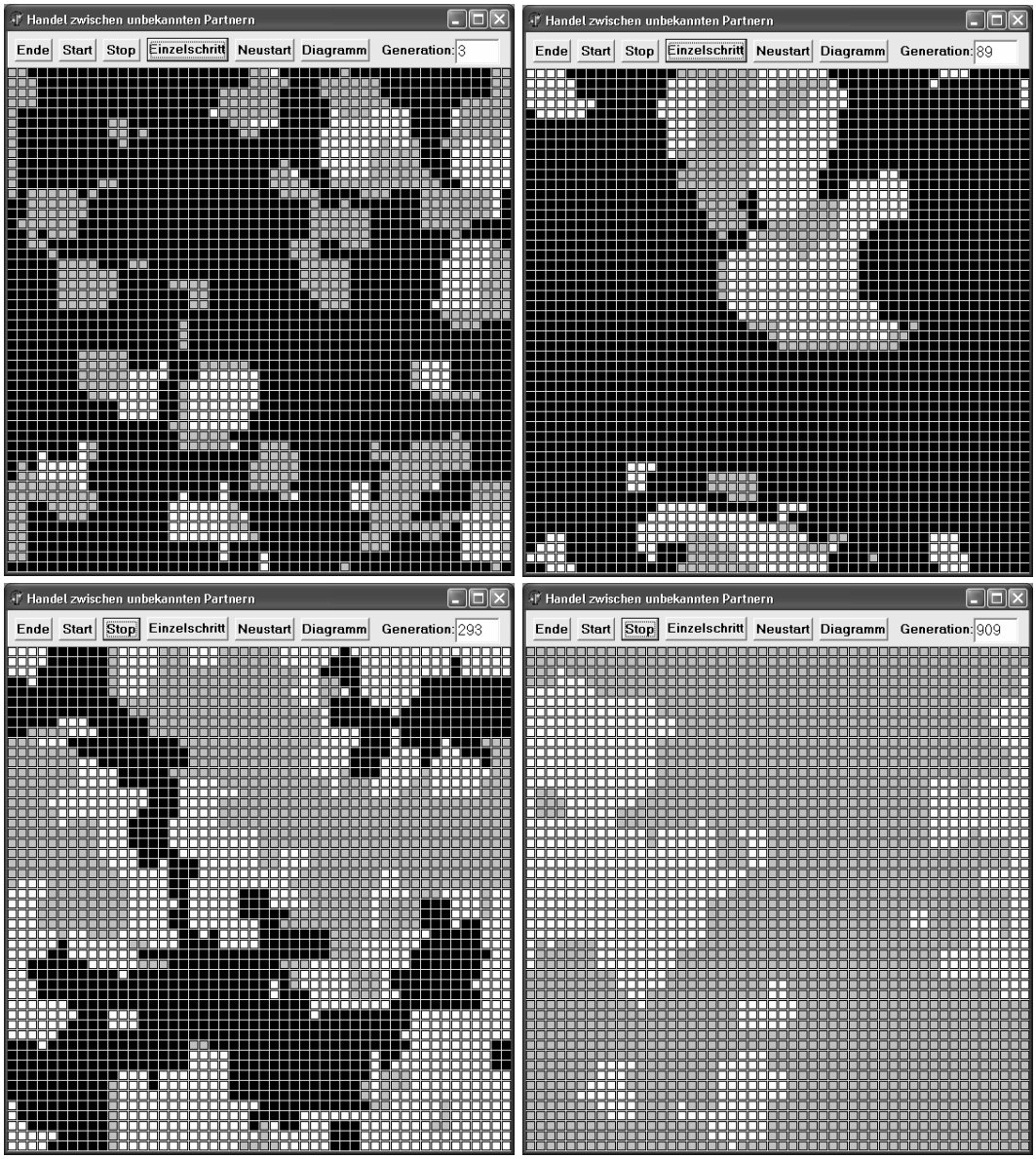
```
function tPartner.kooperiertMit( lage: integer): boolean;
begin
case zustand of
  1: result := true;                          //der Naive: kooperiert immer
  2: result := false;                         //der Betrüger: kooperiert nie
  3: result := wasDerNachbarTat[lage]; //der Gewitzte: TitForTat
  end
end;
```

Wie arbeitet der Gitterautomat nun?

In den ersten Generationen setzen sich meist die Betrüger durch.



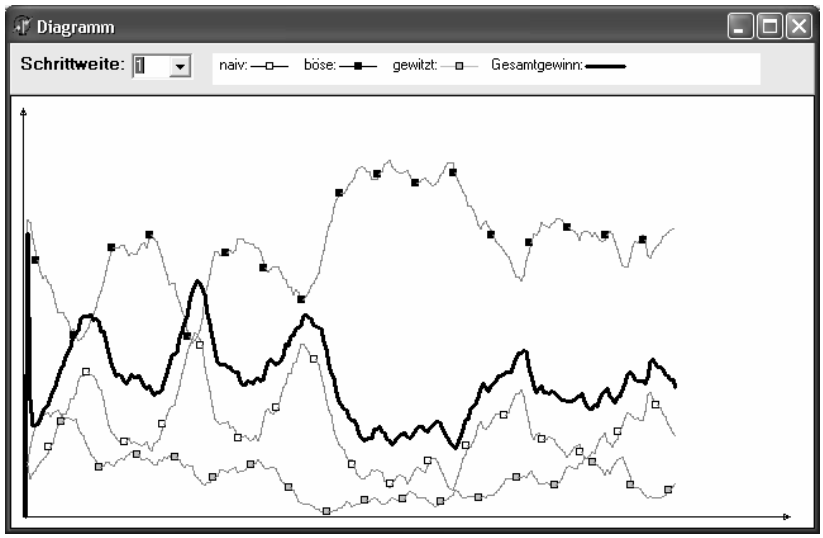
Doch danach bilden sich Cluster aus Naiven und Gewitzten, und dann beginnt eine wilde Schlacht.



Zwar werden die Naiven hart von den Betrügern bedrängt. Sie halten sich aber ganz gut in Gruppen. Die Gewitzten setzen sich gegenüber den Betrügern – je nach Konfiguration – meist durch und kooperieren mit den Naiven. Am Ende siegen meist die Gewitzten – aber eben nicht immer.

Die Beobachtung der manchmal überraschenden Abläufe liefert Ansatzpunkte zur Diskussion ethischer Fragen. Auch wenn das Beispiel natürlich nicht direkt auf gesellschaftliche Systeme übertragbar ist, so haben wir doch ein neuartiges Argument für kooperatives, soziales Verhalten gefunden, das nicht aus transzendenten oder philosophischen Überlegungen gewonnen wird, sondern aus Effizienzbetrachtungen. Als Beispiel mag ein Diagramm dienen, in dem einerseits die Gesamtzahlen der drei Automatentypen (Naiver, Betrüger, Gewitzter) aufgetragen wurden, und dazu die Summe der insgesamt von allen Typen erreichten Handelspunkte, also eine Art „Bruttosozialprodukt“.

Man sieht sehr schön, dass der „gesellschaftliche Wohlstand“ gegenläufig zur Anzahl der Egoisten ist – natürlich bei den eingestellten Bedingungen.



Die Naiven harmonieren prächtig mit den Gewitzten – wenn sie denn unter sich sind. Da dann alle kooperieren, ist das Handelsvolumen maximal.