Kunstmatige Intelligentie 2008 — opdracht 1

simpleBridge

Rick van der Zwet
<hvdzwet@liacs.nl>

LIACS
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

18 maart 2008

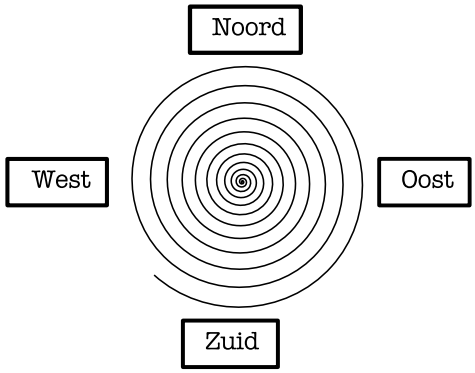
1 Inleiding

Dit verslag gaat over de eerste programmeer-opgave van het vak kunstmatige intelligentie[1]. Welke als opdracht heeft het kaartspel *simpleBridge* te onderzoeken. Tijdens dit onderzoek zal uitgezocht worden of met simpele regels te voorspellen is wat de uitslag wordt van een spelletje *simpleBridge*. Als tweede wordt gekeken of het mogelijk is om met behulp van een aantal simpele regels een zeer eenvoudige reflex-agent te construeren die winnend speelt tegenover zijn willekeurige tegenhangers.

2 Uitleg probleem

simpleBridge is zeer versimpelde versie bridge[1] met 4 spelers die in 2 teams tegen elkaar spelen. Er wordt gespeeld met een normaal kaartspel, dit bestaat uit 52 unieke kaarten, welke een kleur hebben (Schoppen, Harten, Klaveren, Ruiten) en een van de 13 unieke oplopende waardes. Er zijn van elke kleur dus 13 uniek oplopende kaarten in het spel. [4]

[1] Kijk voor de volledige spelregels van bridge op Bridge pagina van Wikipedia[3]



Figuur 1: Voor de duidelijkheid zijn in het verslag heten de spelers Noord, Oost, Zuid en West. En speelt Noord samen met Zuid in team *NZ* en bevat team *OW* de spelers Oost en West.

Elke speler krijgt willekeurig 13 kaarten toegewezen en heeft geen zicht op de kaarten van de andere spelers, nu wordt er willekeurig een team toegewezen die de *troefkleur* mag kiezen en het aantal slagen dat ze denken te gaan halen, dit wordt het *contract* genoemd. Tijdens deze keuze hebben ze zicht op alle kaarten van het team, maar niet op de kaarten van het andere team. Na het maken van het contract wordt er afgespeeld, hierbij is voor elke speler niet meer bekend wat de andere spelers voor kaarten in handen hebben.

Het *afspelen* gaat als volgt:

Er wordt willekeurig een speler gekozen die een kaart naar keuze opgegooid. De rest van de spelers gooien om de beurt (met de klok mee) een kaart mee, welke van dezelfde kleur is als de eerste kaart die opgegooid is. Als deze niet in bezit is volstaat een het willekeurige andere kaart.

Nadat alle vier de spelers een kaart hebben gespeeld wordt er gekeken welke kaart de winnaar is. De hoogst gespeelde troef wint en als deze niet present zijn dan is de hoogste kaart van de kleur die als eerste kaart gespeeld is de winnaar en heeft het team van de speler die deze kaart gespeeld heeft een *slag*

De speler van de winnende kaart is nu aan de beurt en mag een kaart naar keuze opgooien en wordt het spel weer hetzelfde afgespeeld als hierboven. Dit gaat door tot alle kaarten gespeeld zijn en er dus 13 slagen verdeeld zijn tussen de 2 teams.

Het aantal verwachte slagen (het contract) wordt nu vergeleken met het aantal daadwerkelijk gehaalde slagen. Als deze gelijk zijn levert dit 20 punten op, elk slag te weinig kost 20 punten en elk slag teveel kost 10 punten.

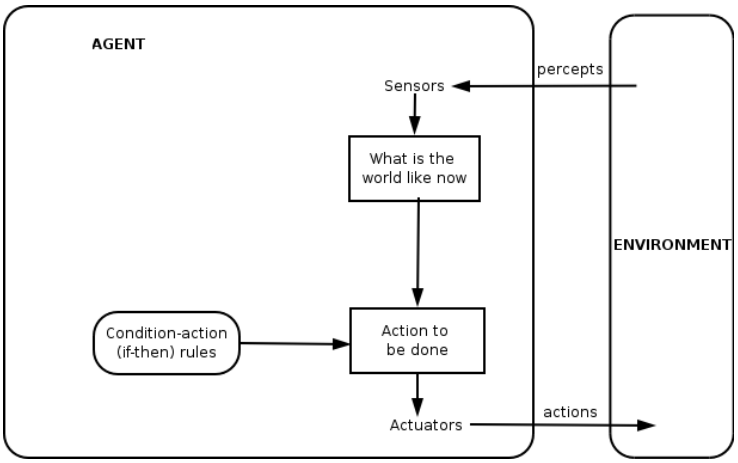
Het meeste aantal punten zijn dus te halen door het beste te bieden en

daarop het slimste te spelen. Als extra randvoorwaarde zal er geprobeerd worden zoveel mogelijk slagen te halen, dit komt echter niet terug in de punten aantallen.

3 Theorie

Als het kiezen van het bod en het afspelen volledig willekeurig gebeurt zal het contract zelden behaald worden door de grote hoeveelheid kaart mogelijkheden.

Tijdens de keuze van het contract word gebruik gemaakt van de informatie die beschikbaar is en van de regels die gesteld zijn. Er wordt niet in het het 'geheim' het spel afgespeeld om ervoor te zorgen dat een goed contract gemaakt wordt. Aan ditzelfde gedrag voordoet de kaartspeel agent. Dit gedrag is ook te zien bij een simpele reflex agent (zie Figuur 2, bron figuur Wikipedia [2]).



Figuur 2: De simpele reflex agenten kan je vergelijken met een simpele actie en reactie mechanisme als het menselijke instinct. Zonder dat er bij nagedacht wordt over het doel/nut wordt er overgegaan tot actie na invoer van de buitenwereld

[2] `http://en.wikipedia.org/wiki/Image:IntelligentAgent-SimpleReflex.png`

4 Aanpak

Voor zowel de contract agent en de afspeel agent zijn simpele regels verzonnen die respectievelijk het bieden of het afspelen proberen te voorspellen.

4.1 Contract agent

Troef levert de meeste slagen op, dus de kleur waar de meeste kaarten van beschikbaar zijn wordt troef en levert een slag op. Als volgende zijn de hoogste kaarten aan de beurt. Eerst wordt er bijvoorbeeld gekeken of de aas present is, dan de heer, enzovoort. Dit wordt gekeken per bepaalde kleur. Alle aansluitende hoge kaarten leveren ook een slag op.

4.2 Afspeel agent

De tactiek wordt om zo snel mogelijk de hoge kaarten uit het spel te krijgen. Dit wordt tewerkstelligd door op moment dat de speler uit mag komen de hoogste kaart uit zijn kaarten te spelen, bij gelijke waarden heeft een troef de voorkeur. Als er bijgespeeld moet worden (dus niet als eerste aan de beurt) dan de hoogste kaart van de kleur meespelen, of de laagste (troef)kaart bij het niet in handen hebben van de meespeel kleur.

5 Implementatie

Er is gekozen om als basis het aangeboden skelet programma[2] te gebruiken welke in de programmeertaal C++ versie 4.0.1 geschreven is. De structuur/werking van de code is niet veranderd, maar de opzet van bijvoorbeeld de uitvoer code wel.

6 Experimenten

Om de experimenten zo eerlijk mogelijk te laten verlopen is ervoor gekozen om de randvoorwaarden gelijk te houden. Onder de randvoorwaarden valt het aantal kaarten dat zichtbaar is tijdens contract keuze en het niet kunnen zijn van de kaarten op tafel. Verder zijn de opstellingen 10000 keer gespeeld om uitersten te elimineren. De opstellingen speelden met dezelfde random begin seed om zo de kaarten die gegeven worden hetzelfde te laten zijn.

Elke opstelling is in de code ingevoerd en daarna is de code opnieuw gecompileerd en uitgevoerd. Tijdelijke (object) bestanden zijn tussentijds weggegooid.

De resultaten van de experimenten zijn te vinden in Tabel 1.

Tabel 1: Opstelling is gecodeerd als Noord,Oost,Zuid,West welke Random of Slim zijn. De contract agent is Random of Slim. Tekort, Correct en Teveel en Score zijn de uitkomsten na 10000 spel rondes, met random seed getal 1203969568. Hoe hoger de Score hoe beter

Opstelling	Contract agent	Tekort	Correct	Teveel	Score
R, R, R, R	R	5020	793	4187	-545350
S, R, R, R	R	5072	741	4187	-554260
S, R, S, R	R	5020	770	4210	-551230
S, S, S, S	R	5039	756	4205	-567600
R, R, R, R	S	6075	1751	2174	-297840
S, R, R, R	S	6127	1686	2187	-307010
S, R, S, R	S	6031	1651	2318	-319270
S, S, S, S	S	5889	1497	2614	-339580

7 Conclusie

De score is enkel afhankelijk van het type contract agent gebruikt. De random agent levert een score van -56760 t/m -545350, de random afspeel agent een betere score dan de slimme agent. De slimme agent levert een score van -339580 t/m -297840 en ook hier geld weer hoe de random afspeel agent een betere score dan de slimme agent.

Nader onderzoek aan de code wijst uit dat er twee factoren aangewezen kunnen worden die dit gedrag kunnen verklaren. Tijdens het maken van het contract wordt het paar volledig random gekozen en hiervan wordt de score bepaald. Het kan dus zijn dat tijdens de combinatie S, R, S, R waarbij de NZ dus slim spelen een contract keuze R toegewezen krijgen. Deze mogelijkheid is onderzocht door het slimme contract altijd voor paar NZ te laten gelden en leverde de waardes op $6215, 1549, 2236, -342270$. De score is zelfs nog slechter.

Of de slimme speler is te 'dom' of het slimme contract is te hoog ingeschat. Dit laatste is het meest aannemelijk want tijdens het bepalen van het contract word er geen rekening gehouden met verschillende feiten. Als speler A een hoge kaart heeft van een bepaalde kleur en speler B ook. Dan zullen ze tijdens het spelen van dit slag beiden hun hoge kaart tegelijk moeten opgooien en zal er maar een slag gehaald worden. Verder wordt een hoge troefkaart twee keer geteld als slag. Zowel in de troef ronde als in de hoge kaarten ronde.

Verder heeft de slimme speelagent geen gebruikt gemaakt van het contract wat geboden is in het geval van de random contract agent en zijn er in bijna 40% van de gevallen teveel slagen gehaald.

Als laatste is de score niet geheel betrouwbaar, omdat elk gemaakt contract even zwaar telt, terwijl beide agenten van het meeste aantal slagen uitgaan. Een score factor bij een gehaald contract van $aantalgehaaldeslagen*10$ zou dit beter reflecteren.

Referenties

[1] W.A. Kosters, Kunstmatige intelligentie Programmeer-opgave 1 van 2008 – Bridge, `http://www.liacs.nl/~kosters/AI/bridge.html`

[2] W.A. Kosters, Kunstmatige intelligentie Programmeer-opgave 1 van 2008 – Bridge – skelet programma, `http://www.liacs.nl/~kosters/AI/bridge.cc`

[3] Wikipedia, the free Encyclopedia, Bridge, latest version `http://nl.wikipedia.org/wiki/Bridge`

[4] Wikipedia, the free Encyclopedia, Speelkaart, as of 25 Februari 2008, `http://nl.wikipedia.org/wiki/Speelkaart`

[5] S.J. Russell en P. Norvig, Artificial Intelligence, A Modern Approach, Second ediion, Prentice Hall, 2003.

Appendix

Het programma gebruikt voor de experimenten ziet er als volgt uit:

```
001: //
002: // bridge.cc
003: // Eerste programmeeropgave voor AI, 28 januari 2008, Bridge
004: // Walter Kosters
005: //
006:
007: #include <iostream>
008: #include <iomanip>
009: #include <cstdlib>
010: #include <ctime>
011: using namespace std;
012:
013: /* Maak array verwijzingen expliciet en uniform */
014: #define NOORD 0
015: #define OOST  1
016: #define ZUID  2
017: #define WEST  3
018:
019: /* Kaartdefinities */
020: #define KLAVEREN 0
021: #define RUITEN   1
022: #define HARTEN   2
023: #define SCHOPPEN 3
024:
025: /* Contract mogelijkheden */
026: #define CONTRACT_RANDOM 0 /* Agent EEN */
027: #define CONTRACT_SLIM 1   /* Agent DRIE */
028:
029: /* Niveau's */
030: #define NIVEAU_RANDOM 0 /* Agent TWEE */
031: #define NIVEAU_SLIM 1 /* Agent VIER */
032:
033: /* Speeltype */
034: #define SPEEL_KOSTER_BRIDGE 0
035:
036: #define VERBOSE 1
037: #define SILENT  0
038:
039: /* Afsluit codes, teveel is positief, teweinig negetief */
040: #define GEHAALD = 0
041:
042: //sorteer array A (met n elementen) via bubblesort
043: void bubblesort (int A[ ], int n) {
044:   int i, j, temp;
045:   for ( j = 1; j < n; j++ )
046:     for ( i = 0; i < n-j; i++ )
047:       if ( A[i] > A[i+1] ) {
048:         temp = A[i];
049:   A[i] = A[i+1];
050:   A[i+1] = temp;
051:       }//if
052: }//bubblesort
053:
054: //welke getalswaarde heeft kaart i?
055: //1..13 zijn Klaver1..13, 14..26 zijn Ruiten1..13,
056: //27..39 zijn Harten1..13 en 40..52 zijn Schoppen1..13
057: //eigenlijk is 13 de aas, dus 1 is een 2 ;-)
058: int dewaarde (int i) {
059:   if ( i <= 13 )
060:     return i;
061:   else if ( i <= 26 )
062:     return i-13;
063:   else if ( i <= 39 )
064:     return i-26;
065:   else
066:     return i-39;
067: }//dewaarde
068:
069: //welke kleurwaarde heeft kaart i?
070: char dekleurwaarde (int i) {
071:   if ( i <= 13 )
072:     return KLAVEREN;
073:   else if ( i <= 26 )
074:     return RUITEN;
075:   else if ( i <= 39 )
076:     return HARTEN;
077:   else
078:     return SCHOPPEN;
079: }//dekleur
080:
081: //welke kleur heeft kaart i?
082: char dekleur (int i) {
083:     switch ( dekleurwaarde (i) ) {
084:     case KLAVEREN:
085:         return 'K';
086:     case RUITEN:
087:         return 'R';
088:     case HARTEN:
089:         return 'H';
090:     case SCHOPPEN:
091:         return 'S';
092:     }
093: }//dekleur
094:
095:
096: //print kaart i netjes
097: void printkaart (int i) {
098:   cout << dekleur (i) << dewaarde (i);
099:   if ( dewaarde (i) < 10 )
100:     cout << " ";
101: }//printkaart
102:
103: //welke speler is i?
104: char despeler (int i) {
105:   switch ( i ) {
106:     case NOORD: return 'N';
107:     case OOST: return 'O';
108:     case ZUID: return 'Z';
109:     case WEST: return 'W';
110:   }//switch
111:   return 'X';
112: }//despeler
113:
114: //===
115:
116: class spel {
117:   public:
118:     int aantalkaarten;     //hoeveel kaarten heeft iedere speler?
119:     char troef;            //KRHS
120:     int hoogte;            //1..13
121:     bool NZ;               //true betekent NZ, false betekent OW
122:     int dekaarten[4][13];  //N:0,O:1,Z:2,W:3
123:     int slagenNZ;          //aantal slagen voor NZ
124:     int slagenOW;          //idem voor OW
125:     int spelerniveau[3];   /* Kwaliteit van speler */
126:     void schud ( );
127:     void print ( );
128:     void maakcontract (int hoe, bool printdetails);
129:     void afspelen (int speeltype, bool printdetails);
130:     int trekkaart (int hoe, char kleur, int A[ ], int aantal);
131:     int uitslag (bool printdetails);
132: };//spel
133:
134: //verdeel de kaarten, en schud ze over de 4 spelers
135: void spel::schud ( ) {
136:   int kaarten[52];
137:   int i, j, temp;
138:   aantalkaarten = 13;
139:   for ( i = 0; i < 52; i++ )
140:     kaarten[i]= i+1;
141:   for ( i = 52-1; i > 0; i-- ) {
142:     j = rand ( ) % (i+1);
143:     temp = kaarten[i];
144:     kaarten[i] = kaarten[j];
145:     kaarten[j] = temp;
146:   }//for
147:   for ( i = 0; i < 13; i++ ) {
148:     dekaarten[NOORD][i] = kaarten[i];
149:     dekaarten[OOST][i] = kaarten[i+13];
150:     dekaarten[ZUID][i] = kaarten[i+26];
151:     dekaarten[WEST][i] = kaarten[i+39];
152:   }//for
153:   bubblesort (dekaarten[NOORD],13);
154:   bubblesort (dekaarten[OOST],13);
155:   bubblesort (dekaarten[ZUID],13);
156:   bubblesort (dekaarten[WEST],13);
157:   slagenNZ = slagenOW = 0;
158: }//spel::schud
159:
160: //print huidige kaarten
161: void spel::print ( ) {
162:   int i;
163:   cout << "=Kaart configuratie=" <<endl;
164:   cout << endl << "Noord : ";
165:   for ( i = 0; i < aantalkaarten; i++ ) {
166:     printkaart (dekaarten[NOORD][i]);
167:     cout << " | ";
168:   }//for
169:   cout << endl;
170:   cout << "Oost  : ";
171:   for ( i = 0; i < aantalkaarten; i++ ) {
172:     printkaart (dekaarten[OOST][i]);
173:     cout << " | ";
174:   }//for
175:   cout << endl;
176:   cout << "Zuid  : ";
177:   for ( i = 0; i < aantalkaarten; i++ ) {
178:     printkaart (dekaarten[ZUID][i]);
179:     cout << " | ";
180:   }//for
181:   cout << endl;
182:   cout << "West  : ";
```

```
183:   for ( i = 0; i < aantalkaarten; i++ ) {
184:     printkaart (dekaarten[WEST][i]);
185:     cout << " | ";
186:   }//for
187:   cout << endl;
188: }//spel::print
189:
190: //maak een contract
191: void spel::maakcontract (int hoe, bool printdetails) {
192:   int spelerA, spelerB;
193:   int zoekpuntA, zoekpuntB;
194:   int slagen;
195:   int kleurAantal[] = {0, 0, 0, 0};
196:
197:   /* Speler volledig random */
198:   NZ = ( rand ( ) % 2 == 1 );
199:
200:   switch (hoe) {
201:   case CONTRACT_RANDOM:
202:     troef = dekleur (rand ( ) % 52);
203:     hoogte = 1 + rand ( ) % 13;
204:     break;
205:   case CONTRACT_SLIM:
206:     /* Variable om de juiste array rijen te selecteren */
207:     if ( NZ ) {
208:         spelerA = NOORD;
209:         spelerB = ZUID;
210:     } else {
211:         spelerA = OOST;
212:         spelerB = WEST;
213:     }
214:
215:     /* Bepaal aantal kaarten per kleur */
216:     for ( int i = 0; i < 13; i++ ) {
217:         kleurAantal[dekleurwaarde (dekaarten[spelerA][i])]++;
218:         kleurAantal[dekleurwaarde (dekaarten[spelerB][i])]++;
219:     }
220:
221:     /* Troef is de meeste kaarten in handen */
222:     troef = 0;
223:     for ( int i = 1; i < 4; i++ )
224:         if ( kleurAantal[i] > kleurAantal[i - 1] )
225:             troef = i;
226:
227:     /* Elke troef levert een slag op */
228:     hoogte = kleurAantal[troef];
229:
230:     /* Hoogste kaart(en) leveren ook slagen op */
231:     slagen = 0;
232:     /* i is de zoekkaart */
233:     for (int i = 52; i >= 0;) {
234:         while (zoekpuntA != -1 && dekaarten[spelerA][zoekpuntA] > i)
235:             zoekpuntA--;
236:         while (zoekpuntB != -1 && dekaarten[spelerB][zoekpuntB] > i)
237:             zoekpuntB--;
238:
239:         /* Hebben we de kaart in bezit of moeten we naar een aas van
240:          * een andere kleur gaan zoeken */
241:         if (zoekpuntA == -1 && zoekpuntB == -1) {
242:             break;
243:         } else if (zoekpuntA != -1 && dekaarten[spelerA][zoekpuntA] == i) {
244:             slagen++;
245:             i--;
246:         } else if (zoekpuntB != -1 && dekaarten[spelerB][zoekpuntB] == i) {
247:             slagen++;
248:             i--;
249:         } else {
250:             if ( i <= 13 ) {
251:                 /* KLAVEREN aas is niet in bezit */
252:                 break;
253:             } else {
254:                 /* Spring naar de volgende aas */
255:                 if ((i % 13) == 0)
256:                     i = i - 13;
257:                 else
258:                     i = i - (i % 13);
259:             }
260:         }
261:     }
262:     hoogte = hoogte + slagen;
263:
264:     /* Set troef naar ASCII waarde om */
265:     troef = dekleur ( (troef + 1) * 13);
266:     break;
267:   }
268:
269:   if (printdetails) {
270:     cout << endl << "=Contract=" << endl;
271:     cout << "Paar   : " << ( NZ ? "NZ " : "OW " ) << endl;
272:     cout << "Slagen : " << hoogte <<endl;
273:     cout << "Troef  : " << troef << endl << endl;
274:   }
275: }//spel::maakcontract
276:
277: //trek een random kaart uit A (die er aantal heeft)
278: //voorkeur voor kleur, anders random
279: int spel::trekkaart (int hoe, char kleur, int A[ ], int aantal) {
280:   int j, i = rand ( ) % aantal, vanaf, totenmet, dekaart;
281:   int hoogstekaart = 0;
282:   bool heeftkleur = false;
283:
284:   /* Bepaal of gevraagde kleur in bezit is */
285:   for ( j = 0; j < aantal; j++ ) {
286:     if ( dekleur (A[j]) == kleur ) {
287:       if ( heeftkleur )
288:         totenmet = j;
289:       else
290:         totenmet = vanaf = j;
291:       heeftkleur = true;
292:     }//if
293:   }//for
294:
295:   switch (hoe) {
296:   case NIVEAU_RANDOM:
297:     if ( heeftkleur )
298:       i = vanaf + rand ( ) % ( totenmet - vanaf + 1 );
299:     break;
300:   case NIVEAU_SLIM:
301:     /* Als eerste kiezen, levert bepaalde privileges */
302:     if ( kleur == 'X' ) {
303:         /* bepalen beste kaart, hoogste troef, dan hoogste andere kaart */
304:         for ( j = 1; j < aantal; j++ ) {
305:             if ( dewaarde (A[j]) > dewaarde (hoogstekaart) ) {
306:                 hoogstekaart = j;
307:             } else if ( dewaarde (A[j]) == dewaarde (hoogstekaart) ) {
308:                 /* Troef is beter */
309:                 if ( dekleur (A[j]) == troef ) {
310:                     hoogstekaart = j;
311:                 }
312:             }
313:         }
314:         i = hoogstekaart;
315:         break;
316:     }
317:
318:     /* Troef spelen, levert aparte regels */
319:     if ( kleur == troef ) {
320:         /* Hoogste troef als present, anders laagste kaart */
321:         if ( heeftkleur )
322:             i = totenmet;
323:         else
324:             i = 0;
325:         break;
326:     }
327:
328:     /* Als kleur present dan meespelen met hoogste kaart */
329:     if ( heeftkleur ) {
330:         i = totenmet;
331:         break;
332:     }
333:
334:     /* Bij ontbreken van kleur, bepaal of troef present is  */
335:     for ( j = 0; j < aantal; j++ )
336:       if ( dekleur (A[j]) == troef ) {
337:         if ( heeftkleur )
338:           totenmet = j;
339:         else
340:           totenmet = vanaf = j;
341:         heeftkleur = true;
342:       }//if
343:     /* 'Steel' het slag door kleinste troefje te spelen */
344:     if ( heeftkleur ) {
345:         i = vanaf;
346:         break;
347:     }
348:
349:     /* Niets kunnen betekenen hier, laagste kaart dan maar ;-) */
350:     i = 0;
351:     break;
352:   }
353:
354:   /* Doe de boekhouding en werk het de eigen hand bij */
355:   dekaart = A[i];
356:   for ( j = i; j < aantal-1; j++ )
357:     A[j] = A[j+1];
358:   A[aantal-1] = 0;
359:   return dekaart;
360: }//spel::trekkaart
361:
362: //speel af; printdetails geeft meer informatie
363: void spel::afspelen (int speeltype, bool printdetails) {
364:   int speler, winnaar, hoogste, winkleur, vraagkleur;
```

```
  int kaart[ ] = {0, 0, 0, 0};
  int buurt = 1;

  /* Laat random speler  beginnen */
  speler = rand ( ) % 4; // 0123

  if (printdetails) {
    cout << "=Spel=" <<endl;
    cout << "S = Speelt, Kx = Kaart x" << endl;
    cout << "Noord -> Oost -> Zuid -> West -> Noord --> Oost -> Zuid" <<endl;
    cout << "Buurt | S | K1  | K2  | K3  | K4  | Winnaar |" << endl;
    cout << "------|---|-----|-----|-----|-----|---------|" << endl;
  }

  switch (speeltype) {
  case SPEEL_KOSTER_BRIDGE:
    while ( aantalkaarten > 0 ) {
      if (printdetails) {
        if ( buurt < 10 )
          cout << " ";
        cout << buurt << "     | ";
      }

      /* Initiele kaart, geen voorkeur voor kleur */
      kaart[0] = trekkaart (spelerniveau[speler],'X',dekaarten[speler],
                            aantalkaarten);
      vraagkleur = dekleur (kaart[0]);
      if ( printdetails ) {
        cout << despeler (speler) << " | ";
        printkaart (kaart[0]);
        cout << " | ";
      }//if

      /* Laat de rest bijspelen */
      for (int i = 1; i <= 3; i++) {
        kaart[i] = trekkaart (spelerniveau[i], vraagkleur,
                              dekaarten[(speler+i)%4],aantalkaarten);
        if ( printdetails ) {
          printkaart (kaart[i]);
          cout << " | ";
        }//if
      }

      aantalkaarten--;

      /* controleer troef in het spel */
      if ( dekleur(kaart[0]) == troef || dekleur(kaart[1]) == troef
           || dekleur(kaart[2]) == troef || dekleur(kaart[3]) == troef )
        winkleur = troef;
      else
        winkleur = dekleur (kaart[0]);

      /* bepaal de winnende spelers */
      hoogste = -1;
      if ( dekleur (kaart[0]) == winkleur && kaart[0] > hoogste ) {
        winnaar = speler;
        hoogste = kaart[0];
      }//if
      if ( dekleur (kaart[1]) == winkleur && kaart[1] > hoogste ) {
        winnaar = (speler+1) % 4;
        hoogste = kaart[1];
      }//if
      if ( dekleur (kaart[2]) == winkleur && kaart[2] > hoogste ) {
        winnaar = (speler+2) % 4;
        hoogste = kaart[2];
      }//if
      if ( dekleur (kaart[3]) == winkleur && kaart[3] > hoogste ) {
        winnaar = (speler+3) % 4;
        hoogste = kaart[3];
      }//if

      if ( printdetails ) {
        cout << despeler (winnaar) << "        |" << endl;
      }//if
      if ( winnaar == 0 || winnaar == 2 )
        slagenNZ++;
      else
        slagenOW++;
      speler = winnaar;

      buurt++;
    }//while
    break;
  }//switch
}//spel::afspelen

//uitslag op scherm
int spel::uitslag (bool printdetails) {
  int slagen = ( NZ ? slagenNZ : slagenOW);
  if (printdetails) {
    cout << "=Uitslag=" << endl;
    cout << "Slagen NZ       : " << slagenNZ << endl;
    cout << "Slagen OW       : " << slagenOW << endl;
    cout << "Contract bod    : " << hoogte << endl;
    cout << "Contract paar   : " << ( NZ ? "NZ " : "OW " ) << endl;
    cout << "Contact gehaald : ";
  }
  if ( slagen < hoogte ) {
    if (printdetails)
      cout << "TEKORT";
  } else if ( slagen == hoogte ) {
    if (printdetails)
      cout << "CORRECT";
  } else {
    if (printdetails)
      cout << "TEVEEL";
  }
  if (printdetails)
    cout << " (" << slagen - hoogte << ")" << endl;

  return (slagen - hoogte);
}//spel::uitslag

//===

void printTotaalWaarde (string type, double waarde, double totaal) {
    cout << setiosflags(ios::left);
    cout << setw(10) << type << " : ";
    cout << resetiosflags(ios::left);
    double percent = (waarde / totaal) * 100;
    cout.precision(8);
    cout << setw(8) << waarde;
    cout << " = ";
    cout.precision(2);
    cout << setw(7) << setiosflags(ios::fixed) << percent
    <<resetiosflags(ios::fixed) << "%";
    cout << endl;

};

int main ( ) {
  int retcode = 0;
  int tekort = 0;
  int correct = 0;
  int teveel = 0;
  double eindscore = 0;
  int fout, goed;

  spel S;
  int randInit;

  /* Variabelen */
  int logNiveau = SILENT; /* print niveau SILENT|VERBOSE */
  double aantalKeer = 1000;  /* aantal rondes */

  /* initialiseer random-generator, maar print seed om debugging
   * mogelijk te maken
   */
  randInit = time(0);
  cout << "RandInit getal: " << randInit << endl;
  srand (randInit);

  /* Speel de potjes */
  for (int i = 0; i < aantalKeer; i++) {
    S.schud ( );
    if (logNiveau)
      S.print ( );

    /* Niveau contract en spelers */
    S.maakcontract (CONTRACT_RANDOM, logNiveau);
    S.spelerniveau[0] = NIVEAU_SLIM;
    S.spelerniveau[1] = NIVEAU_RANDOM;
    S.spelerniveau[2] = NIVEAU_RANDOM;
    S.spelerniveau[3] = NIVEAU_RANDOM;
    if (logNiveau)
        for (int i = 0; i <= 3; i++)
            cout << "Niveau speler " << i << ":" << S.spelerniveau[i] << endl;

    /* Afspelen bepaal type spel */
    S.afspelen (SPEEL_KOSTER_BRIDGE, logNiveau);

    if (logNiveau)
      cout << endl;
    retcode = S.uitslag (logNiveau);

    /* Boekhouding, voor de eindscore berekening */
    if (retcode > 0) {
        teveel++;
        /* elk punt teveel score: -10 */
        eindscore = eindscore + retcode * -10;
    } else if (retcode == 0) {
        correct++;
```

```
547:         /* elk goede score +20 */
548:         eindscore = eindscore + 20;
549:       } else {
550:         tekort++;
551:         /* elk punt tekort score: -20
552:           * retcode is al negatief
553:           */
554:         eindscore = eindscore + (retcode * 20);
555:       }
556:     }
557:
558:     /* Bereken eindresultaat */
559:     fout = tekort + teveel;
560:     goed = correct;
561:     printTotaalWaarde("Tekort", tekort,   aantalKeer);
562:     printTotaalWaarde("Correct", correct,   aantalKeer);
563:     printTotaalWaarde("Teveel", teveel,   aantalKeer);
564:     cout << endl;
565:     printTotaalWaarde("Goed", goed,   aantalKeer);
566:     printTotaalWaarde ("Fout", fout, aantalKeer);
567:     printTotaalWaarde ("EindScore", eindscore, aantalKeer * 20);
568: }//main
569:
```