Gomoku

Ahmad Jamalzada en Erik de Vos, Universiteit Leiden

6 december 2013

1 Beschrijving programma

Ons programma is een interactief Gomoku-spel. Dit programma stelt de gebruiker in staat om zowel zelf Gomoku tegen de computer te spelen, als de computer tegen zichzelf te laten spelen. Verder kan de gebruiker ook zijn eigen kleur kiezen, de grootte van het speelbord wijzigen en "valsspelen". Dit doet de gebruiker d.m.v. het terughalen van zijn eigen en de computers eerdere zetten. Wanneer de gebruiker tegen de computer speelt wordt onderwijl constant de huidige stand in beeld getoond.

2 Opmerkingen bij programma

Allereerst hebben wij in dit programma niet de optie toegevoegd om het aantal mogelijke vervolgpartijen te berekenen/geven. Ook zal onze opdracht drie headerfiles bevatten (te weten: gobord.h, bordvakje.h en stapel.h). Dit omdat wij de keuze hebben gemaakt om "de stapel"niet met integers te vullen, maar met pointers naar de posities die de gebruiker eventueel terug wilt halen. Hierom hebben wij de stapel moeten declareren in de klasse gobord wat problemen oplevert wanneer men slechts gobord.h en stapel.h als headerfiles gebruikt.

3 Werktijd

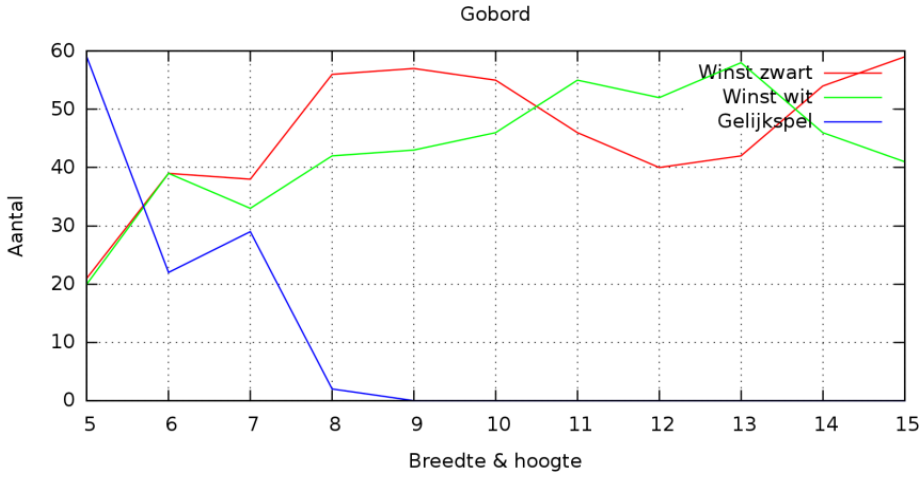
Hoewel een programmeeropdracht altijd veel van je tijd vergt, viel deze opdracht ons beide alleszinds mee, vergeleken met opdracht 3.

Hieronder is aangegeven in tabelvorm hoe lang wij hieraan hebben gewerkt.

Week	Ahmad	Erik
48	12 uur	11 uur
49	8 uur	6 uur

4 Onderzoek

Onderzoek van L. Victor Allis heeft uitgewezen dat bij "perfect play"(een concept waarbij een speler exact de juiste zetten doet, en daarmee altijd wint onafhankelijk van de zetten van de tegenstander) in Gomoku, zwart wint.(zie [1]) Aangezien ons programma verre van perfecte zetten doet wanneer wij de computer tegen de computer laten spelen zal dit bij ons niet de uitkomst worden. Hierom hebben wij een grafiek geplot via gnuplot, waarin wij de hoeveelheid winstpartijen van zwart/wit en het aantal gelijke spelen hebben vergeleken met de grootte van het bord. Hierin hebben wij alleen vierkante borden vergeleken, waarbij de hoogte (hier dus gelijk aan de breedte) op de x-as zijn geplot. Hieruit bleek vooral dat wanneer het bord groter wordt dan 10, het aantal gelijke spelen vrijwel altijd 0 zal zijn. (zie figuur 1.)



Figuur 1: Grafiek geplot met gnuplot

Referenties

[1] L. Victor Allis (1994), "Searching for Solutions in Games and Artificial Intelligence", proefschrift, Rijksuniversiteit Limburg, ISBN 90-900748-8-0

Programma code

Er is voor ons gekozen voor de programmeertaal C++. De code van het programma is als volgt:

JamalzadaDeVoshoofd4.cc:

```
1  //Dit is een programma geschreven door Ahmad Jamalzada en Erik de Vos.
2  //Onze code is opgesplitst in 6 bestanden, waaronder drie headerfiles (zie
3  //lateX verslag voor onze uitleg voor deze keuze). De namen van deze bestanden
4  //zijn: JamalzadaDeVoshoofd4.cc, JamalzadaDeVosgobord4.cc,
5  //JamalzadaDeVosgobord4.h, JamalzadaDeVosstapel4.cc, JamalzadaDeVosstapel4.h,
6  //JamalzadaDeVosbordvakje4.h.
7  //Onze code genereert een interactief Gomoku-spel in de terminal, waarin
8  //allerlei wijzigingen kunnen worden aangebracht. De compiler die wij hebben
9  //gebruikt is g++(GCC 4.8.1). Als editor hebben wij gedit gebruikt.
10 //Ons project hebben wij op 6 December 2013 afgerond.
11 //Onze studentennummers zijn: Erik: 1430750  Ahmad: 1145657
12
13 #include <iostream>
```

```
#include "JamalzadaDeVosgobord4.h"

using namespace std;

void infoblokje( ){
    cout << endl << "Dit programma is geschreven door Ahmad Jamalzada en Erik "
         << "de Vos\nWij zijn eerstejaars Natuurkunde studenten.\n"
         << "Dit programma is gemaakt voor de vierde opdracht van het vak\n"
         << "Programmeermethoden. Het is afgerond op 6 December 2013.\n"
         << "U kunt met dit programma het spel Gomoku spelen tegen een"
         << " computer.\nOok kunt u de computer tegen zichzelf laten spelen."
         << endl << endl;
}

int main( ){
    infoblokje( );
    gobord Gobord;
    Gobord.menu( );

    return 0;
}
```

JamalzadaDeVosbordvakje4.h:

```
#ifndef __JamalzadaDeVosbordvakje4_h__
#define __JamalzadaDeVosbordvakje4_h__

class bordvakje{
  public:
    char kleur;             //     7 0 1
    bordvakje* buren[8];    //     6   2
    bordvakje( );           //     5 4 3
};

#endif
```

JamalzadaDeVosstapel4.h:

```
#ifndef __JamalzadaDeVosstapel4_h__
#define __JamalzadaDeVosstapel4_h__
#include "JamalzadaDeVosbordvakje4.h"

class vakje{
    public:
        vakje( );
        bordvakje* positie;
        vakje* volgende;
};

class stapel{
    public:
        stapel( );
        ~stapel( );
        void zetOpStapel(bordvakje* p);
        void haalVanStapel(bordvakje* & hulp);
        bool isStapelLeeg( );
    private:
        vakje* bovenste;
};

#endif
```

JamalzadaDeVosgobord4.h:

```
#ifndef __JamalzadaDeVosgobord4_h__
#define __JamalzadaDeVosgobord4_h__
#include "JamalzadaDeVosstapel4.h"

int leesgetal( );

class gobord{
    private:
        stapel Stapel;
        bordvakje* ingang;
        int hoogte, breedte;
        char kl1;
        char kl2;
        int telSpeler;
        int telPC;
        int rijNummer;
        int kolomNummer;
        int nZet;
        bool gedaan;
        void rits(bordvakje* boven, bordvakje* onder);
        bordvakje* maakRij(int aantal);

    public:
        gobord( );
        gobord(int hoogte, int breedte);
        ~gobord( );
        bordvakje* gaNaar(int i, int j);
        int richting(char & kleur, int richting);
        long double mogelijkePartijen(int getal);
        void pcSpelDuizend( );
        void mensSpel( );
        void terughalen( );
        void pcSpel( );
        void schoon( );
        void menu( );
        void kiesKleur( );
        void setDimensies( );
        void bouwBord( );
        void randomZet(char kleur);
        void mensZet( );
        void drukAf( );
        bool klaar( );
        bool gewonnen(char & kleur);
        void doeZet(int i, int j, char kl);
};
#endif
```

JamalzadaDeVosstapel4.cc:

```
#include <iostream>
#include "JamalzadaDeVosstapel4.h"

using namespace std;

//Constructor voor class vakje.
vakje::vakje( ){
    positie = 0;
    volgende = NULL;
}
```

```
11
12 //Constructor voor class stapel.
13 stapel::stapel( ){
14     bovenste = NULL;
15 }
16
17 //Controleert of de stapel leeg is.
18 bool stapel::isStapelLeeg( ){
19     if (bovenste == NULL)
20         return true;
21     else
22         return false;
23 }
24
25 //Plaatst de laatst gedane zet in de stapel.
26 void stapel::zetOpStapel(bordvakje* p){
27     vakje* temp = new vakje;
28     temp->positie = p;
29     temp->volgende = bovenste;
30     bovenste = temp;
31 }
32
33 //Verwijdert de laatst gedane zet.
34 void stapel::haalVanStapel(bordvakje* & hulp){
35     vakje* temp = bovenste;
36     hulp = bovenste->positie;
37     bovenste = bovenste->volgende;
38     delete temp;
39 }
40
41 //Destructor voor class stapel.
42 stapel::~stapel( ){
43     bordvakje* p;
44     while (!isStapelLeeg( ))
45         haalVanStapel (p);
46 }
```

JamalzadaDeVosgobord4.cc

```
1 #include <iostream>
2 #include <cstdlib>
3 #include "JamalzadaDeVosgobord4.h"
4
5 using namespace std;
6
7 // Constructor voor het gobord, geeft startwaarden van verscheidene
8 // membervariabelen aan. Ook wordt hier het bord alvast gebouwd.
9 gobord::gobord( ){
10     nZet = 0;
11     ingang = NULL;
12     hoogte = 6;
13     breedte = 7;
14     kl1 = 'Z';
15     kl2 = 'W';
16     rijNummer = 1;
17     kolomNummer = 1;
18     gedaan = false;
19     telSpeler = 0;
20     telPC = 0;
21     bouwBord( );
22 }
```

```
23 // Constructor voor de klasse bordvakje.
24 bordvakje::bordvakje( ){
25     for (int i = 0; i < 8; i++)
26         buren[i] = NULL;
27 }
28 // Haalt laatst gedane zet (van de gebruiker of de pc) terug van het gobord.
29 void gobord::terughalen( ){
30     bool leeg = Stapel.isStapelLeeg( );
31     bordvakje* hulp = NULL;
32 
33     if (!leeg){
34         Stapel.haalVanStapel(hulp);
35         hulp->kleur = '_';
36     }
37     nZet--;
38 }
39 //Verwerkt de getallen die de gebruiker invoert.
40 int leesgetal(int bovengrens){
41     char numb = cin.get( );
42     int getal = 0;
43 
44     while (numb == '\n')
45         numb = cin.get( );
46 
47     while (numb != '\n'){
48         if (numb <= '9' && numb >= '0'){
49             getal = 10 * getal + (numb - '0');
50             if (getal > bovengrens)
51                 getal = getal / 10;
52         }
53     numb = cin.get( );
54     }
55     return getal;
56 }
57 //Maakt het gobord leeg.
58 void gobord::schoon( ){
59     bordvakje* p = ingang;
60     for (int i = 0; i < hoogte; i++){
61         bordvakje* q = p;
62         while (p != NULL){
63             p->kleur = '_';
64             p = p->buren[2];
65         }
66         p = q->buren[4];
67     }
68     nZet = 0;
69 }
70 //Maakt een rij van bordvakjes.
71 bordvakje* gobord::maakRij(int aantal){
72     bordvakje* p = new bordvakje;
73     bordvakje* eerste = p;
74     p->kleur = '_';
75     aantal--;
76 
77     while (aantal != 0){
78         p->buren[2] = new bordvakje;
79         p->buren[2]->buren[6] = p;
80         p = p->buren[2];
81         p->kleur = '_';
82         aantal--;
83     }
```

```
84
85      return eerste;
86  }
87  //Verandert de "speelkleur" in de door de gebruiker gekozen kleur. En past
88  //de computer kleur aan op basis van zijn keuze (met behulp van de
89  //membervariabele kl1 & kl2).
90  void gobord::kiesKleur( ){
91      cout << "Welke kleur wilt u zijn? [Z]wart of [W]it" << endl;
92      cin >> kl1;
93
94      if (kl1 == 'w')
95          kl1 = 'W';
96      if (kl1 == 'z')
97          kl1 = 'Z';
98      if (kl1 == 'W') {
99          kl2 = 'Z';
100         randomZet(kl2);
101     }
102     else if (kl1 == 'Z')
103         kl2 = 'W';
104 }
105 //Verandert de grootte van het gobord, in de grootte als ingevoerd door de
106 //gebruiker (met behulp van de membervariabelen hoogte & breedte).
107 void gobord::setDimensies( ){
108     cout << "Wat is uw gewenste hoogte?" << endl;
109     hoogte = leesgetal(50);
110     while (hoogte == 0){
111         cout << "Wat is uw gewenste hoogte?" << endl;
112         hoogte = leesgetal(50);
113     }
114     cout << "Wat is uw gewenste breedte?" << endl;
115     breedte = leesgetal(50);
116     while (breedte == 0){
117         cout << "Wat is uw gewenste breedte?" << endl;
118         breedte = leesgetal(50);
119     }
120     bouwBord( );
121 }
122 //Returnt een pointer naar een positie op het gobord, ingevoerd met 2 integers.
123 bordvakje* gobord::gaNaar(int i, int j){
124     bordvakje* p = ingang;
125     for (int n = 1; n < i; n++)
126         p = p->buren[4];
127     for (int n = 1; n < j; n++)
128         p = p->buren[2];
129     return p;
130 }
131 //Voert, indien mogelijk, de aangegeven zet uit op het gobord.
132 void gobord::doeZet(int i, int j, char kl1){
133     bordvakje* p = gaNaar(rijNummer, kolomNummer);
134     if (p->kleur == '_'){
135         p->kleur = kl1;
136         gedaan = true;
137         Stapel.zetOpStapel(p);
138         nZet++;
139     }
140     else {
141         cout << "Dit vakje is al in gebruik.\n";
142         gedaan = false;
143     }
144 }
```

```
145 //De gebruiker geeft aan welke zet hij wilt doen, wanneer deze zet is gedaan
146 //wordt gecontroleerd of de persoon gewonnen heeft. Zo niet, dan zet de computer
147 //en wordt zijn zet gecontrolleerd op winst. Ook wordt gecontroleerd op een vol
148 //bord (gelijkspel).
149 void gobord::mensZet( ){
150     bool controle = false;
151 
152     cout << "Voer uw gewenste rijnummer in\n";
153     rijNummer = leesgetal(hoogte);
154     while (rijNummer == 0){
155         cout << "Voer uw gewenste rijnummer in\n";
156         rijNummer = leesgetal(hoogte);
157     }
158     cout << endl << "Voer uw gewenste kolomnummer in\n";
159 
160     kolomNummer = leesgetal(breedte);
161     while (kolomNummer == 0){
162         cout << "Voer uw gewenste kolomnummer in\n";
163         kolomNummer = leesgetal(hoogte);
164     }
165     cout << endl;
166 
167     if (nZet < hoogte*breedte){
168         doeZet(rijNummer, kolomNummer, kl1);
169         controle = gewonnen(kl1);
170         if (controle){
171             drukAf( );
172             cout << "U heeft vijf op een rij gewonnen!" << endl;
173             telSpeler++;
174             schoon( );
175         }
176         if (gedaan && nZet < hoogte * breedte && !controle){
177             randomZet(kl2);
178             controle = gewonnen(kl2);
179             if (controle){
180                 cout << "De computer heeft vijf op een rij gewonnen!" << endl;
181                 drukAf( );
182                 telPC++;
183                 schoon( );
184             }
185         }
186         else if (nZet == hoogte * breedte){
187             cout << "Het bord is vol. Het spel is voorbij." << endl;
188             drukAf( );
189             schoon( );
190         }
191     }
192 }
193 //Simuleert een spel tussen twee tegenstanders die elk 'random' zetten. Oftewel
194 //de computer tegen de computer. Ook hier wordt gecontroleerd op
195 //winst/gelijkspel.
196 void gobord::pcSpel( ){
197     bool controle = false;
198     kl1 = 'Z';
199     kl2 = 'W';
200     schoon( );
201 
202     while (nZet < hoogte * breedte && !controle){
203         randomZet(kl1);
204         controle = gewonnen(kl1);
205         if (controle){
```

```
206             drukAf( );
207             cout << "Zwart" << " heeft gewonnen" << endl;
208         }
209         else if (nZet < hoogte * breedte && !controle) {
210             randomZet(kl2);
211             controle = gewonnen(kl2);
212             if (controle){
213                 drukAf( );
214                 cout << "Wit" << " heeft gewonnen" << endl;
215             }
216         }
217     }
218     if (nZet == hoogte * breedte && !controle){
219         drukAf( );
220         cout << "Gelijkspel..." << endl;
221     }
222     schoon( );
223 }
224 //Telt het aantal opeenvolgende kleuren in een bepaalde richting
225 //vanaf het laatst gedane zet.
226 int gobord::richting(char & kleur, int richting){
227     int teller = 0;
228     bordvakje* p = gaNaar(rijNummer, kolomNummer);
229     bordvakje* q = p;
230     q = q->buren[richting];
231     for (int l = 0; l < 5 && q != NULL && q->kleur == kleur; l++){
232         teller++;
233         q = q->buren[richting];
234     }
235     return teller;
236 }
237 //Controleert of vanaf de laatste zet er vijf opeenvolgende zelfde kleuren zijn
238 //gevonden (zichzelf meegerekend).
239 bool gobord::gewonnen(char& kleur){
240     int tl0, tl1, tl2, tl3, tl4, tl5, tl6, tl7;
241
242     tl0 = richting(kleur, 0) + 1;
243     tl1 = richting(kleur, 1) + 1;
244     tl2 = richting(kleur, 2) + 1;
245     tl3 = richting(kleur, 3) + 1;
246     tl4 = richting(kleur, 4);
247     tl5 = richting(kleur, 5);
248     tl6 = richting(kleur, 6);
249     tl7 = richting(kleur, 7);
250
251     if (tl0 + tl4 >= 5 || tl2 + tl6 >= 5 || tl1 + tl5 >= 5 || tl3 + tl7 >= 5)
252         return true;
253     else
254         return false;
255 }
256 //Met behulp van een randomgenerator wordt een 'willekeurige zet' gedaan.
257 void gobord::randomZet(char kleur){
258     srand (time(NULL));
259     rijNummer = (rand( ) % hoogte) + 1;
260     kolomNummer = (rand( ) % breedte) + 1;
261     bordvakje* p = gaNaar(rijNummer, kolomNummer);
262     while (p->kleur != '_'){
263         rijNummer = (rand( ) % hoogte) + 1;
264         kolomNummer = (rand( ) % breedte) + 1;
265         p = gaNaar(rijNummer, kolomNummer);
266     }
```

```
267     Stapel.zetOpStapel(p);
268     p->kleur = kleur;
269     nZet++;
270 }
271 //Het submenu wanneer de gebruiker heeft gekozen om zelf tegen de PC te spelen.
272 //Hier kiest de gebruiker uit de aangegeven opties door de desbetreffende letter
273 //in te voeren.
274 void gobord::mensSpel( ){
275     char kar = '&';
276 
277     while (kar != 'H' && kar != 'h'){
278         drukAf( );
279         cout << "[Z]etten\n[K]leur kiezen\n"
280              << "[H]oofdmenu\n[T]erughalen zet\n"
281              << "[A]antal mogelijke vervolgpartijen\n"
282              << "Speler : " << telSpeler << " PC: " << telPC << endl;
283 
284         cin >> kar;
285         switch (kar){
286             case 'a':
287                 cout << "Aantal mogelijke vervolgpartijen"
288                      << " (houdt geen rekening met winst): \n"
289                      << mogelijkePartijen(hoogte * breedte - nZet) << endl; break;
290             case 'K': case 'k':
291                 kiesKleur( ); break;
292             case 'Z': case 'z':
293                 mensZet( ); break;
294             case 'T': case 't':
295                 terughalen( );
296                 terughalen( ); break;
297         }
298         cout << endl;
299     }
300 }
301 //Het hoofdmenu wat tevoorschijn komt wanneer de gebruiker het programma opstart
302 //of teruggaat vanaf het submenu (mensSpel). Hier wordt het gobord niet
303 //afgedrukt vanwege het feit dat dit hier overbodig is. Hier kiest de gebruiker
304 //uit de aangegeven opties door de desbetreffende letter in te voeren.
305 void gobord::menu( ){
306     char karakter = '$';
307 
308     while (karakter != 'S' && karakter != 's') {
309         cout << "Wat wordt uw keuze?\n" << endl
310              << "s[C]hoonmaken\n [G]rootte wijzigen\n"
311              << " [M]ens tegen PC\n [P]c tegen zichzelf laten spelen.\n"
312              << " [S]luit programma af\n";
313         cin >> karakter;
314         switch (karakter){
315             case 'G': case 'g':
316                 setDimensies( ); break;
317             case 'M': case 'm':
318                 mensSpel( ); break;
319             case 'P': case 'p':
320                 pcSpel( ); break;
321             case 'C': case 'c':
322                 schoon( ); break;
323         }
324         cout << endl;
325     }
326 }
327 //Bouwt een bord met de aangegeven hoogte en breedte. Door de eerst de rijen
```

```
328 //te maken en deze met in de goede volgorde aan elkaar te 'ritsen'.
329 void gobord::bouwBord( ){
330     ingang = maakRij(breedte);
331 
332     bordvakje* boven = ingang;
333     bordvakje* onder = NULL;
334     for (int i = 1; i < hoogte; i++){
335         onder = maakRij(breedte);
336         rits(boven, onder);
337         boven = onder;
338     }
339     nZet = 0;
340 }
341 //Verbind de bordvakjes met elkaar met behulp van pointers.
342 void gobord::rits(bordvakje* boven, bordvakje* onder){
343     while (boven != NULL){
344         boven->buren[4] = onder;
345         boven->buren[5] = onder->buren[6];
346         boven->buren[3] = onder->buren[2];
347         onder->buren[7] = boven->buren[6];
348         onder->buren[1] = boven->buren[2];
349         onder->buren[0] = boven;
350         boven = boven->buren[2];
351         onder = onder->buren[2];
352     }
353 }
354 //Drukt het gobord af, met aan de x-kant en de y-kant van het bord de kolom en
355 //rijnummers.
356 void gobord::drukAf( ){
357     bordvakje* p = ingang;
358     for (int i = 0; i < hoogte; i++){
359         bordvakje* q = p;
360         while (p != NULL){
361             cout << p->kleur << " ";
362             p = p->buren[2];
363         }
364         cout << 1 + i;
365         cout << endl;
366         p = q->buren[4];
367     }
368     for (int i = 0; i < breedte; i++){
369         if (i >= 9){
370             int k = (1 + i) / 10;
371             cout << k << " ";
372         }
373         else
374             cout << 1 + i << " ";
375     }
376     cout << endl;
377 
378     for (int i = 0; i < breedte; i++){
379         if (i >= 9){
380             int k = (i + 1) % 10;
381             cout << k << " ";
382         }
383         else
384             cout << "  ";
385     }
386     cout << endl;
387 }
388 //Berekent het aantal mogelijke partijen zonder winst meegerekend.
```

```
389 long double gobord::mogelijkePartijen(int getal){
390     if (getal == 1)
391         return 1;
392     if (getal == 0)
393         return 0;
394     else
395         return (getal * (mogelijkePartijen(getal - 1)));
396 }
397 //Destructor voor de klasse gobord. Hier wordt elk van de bordvakjes een voor
398 //een verwijdert.
399 gobord::~gobord( ){
400     bordvakje* p = ingang;
401     for (int i = 0; i < hoogte; i++){
402         bordvakje* q = p;
403         q = p->buren[4];
404         for (int j = 0; j < breedte - 1; j++){
405             p = p->buren[2];
406             delete p->buren[6];
407         }
408         delete p;
409         p = q;
410     }
411 }
```