Eindverslag ALGODS

Minor ECV



René Theil 12053031

Richard van Rijn 12053031

# Voorwoord

Dit verslag beschrijft de werkzaamheden voor de eindopdracht van ALGODS van de minor ECV. Voor deze eindopdracht moest een bordspel gerpogrameerd in C++, waar gebruikt gemaakt wordt van Algoritmen en/of Datastructuren. Na een middagje speuren op het internet vonden wij Hex: een bordspel waarbij het doel is om met aaneengelegde tegels de andere kant van het bord te bereiken. Hierbij speelt de ene speler van links naar rechts en de ander van boven naar beneden. Omdat de tegels de vorm hebben van een hexagon, zijn er zes aanliggende punten per tegel waardoor het spel iets ingewikkelder is voor een computerprogramma dan bijvoorbeeld Boter, Kaas en Eieren, waar dit programma op gebasseerd is.

De basis van dit programma is gebasseerd op “Tic-Tac-Toe programma met alpha-beta pruning en transposition table met behulp van std::map” van Harry Broeders. Graag willen wij hem bedanken voor het verlenen van dit studiemateriaal en zijn inzet tijdens de ontwikkeling van dit programma.

# Samenvatting

In dit programma is het de bedoeling dat de computer (zo efficiënt mogelijk) alle mogelijke zetten doorrekent en telkens een winnende of zo efficient mogelijke zet maakt. Het wordt al snel duidelijk dat er elke zet gekeken moet worden of er een pad is naar het desbetreffende eind van het bord. Daarom wordt er gebruik gemaakt van een datastructuur waarbij elke omliggende tegel gecontroleerd wordt. Terwijl dit gedaan wordt, wordt de volgende tegel bekeken totdat het einde van het bord bereikt is. Wanneer een pad gevonden is, wordt de zet gemaakt.

Inhoudsopgave

[1 Voorwoord 2](#_Toc417472126)

[2 Samenvatting 3](#_Toc417472127)

[3 Inleiding 5](#_Toc417472128)

[3.1 Achtergrond 5](#_Toc417472129)

[3.2 Opdrachtomschrijving 5](#_Toc417472130)

[3.3 Specificaties van de opdracht 5](#_Toc417472131)

[4 Conclusie, aanbevelingen en verbeterpunten 6](#_Toc417472132)

[4.1 Conclusie 6](#_Toc417472133)

[4.2 Aanbevelingen en verbeterpunten 6](#_Toc417472134)

# Inleiding

## Achtergrond

## Opdrachtomschrijving

## Specificaties van de opdracht

De specificaties zijn als volgt:

# De basis: Tic Tac Toe

Tic Tac Toe, of Boter, Kaas en Eieren in het Nederlands, is een bordspel van drie bij drie tegels waarbij het doel van de spelers is om drie aaneenliggende tegels te plaatsen van de betreffende speler. Dit mag horizontaal, verticaal of diagonaal: zie Figuur 1. Vaak worden er kruisjes en cirkels gebruikt om de spelers te onderscheiden.

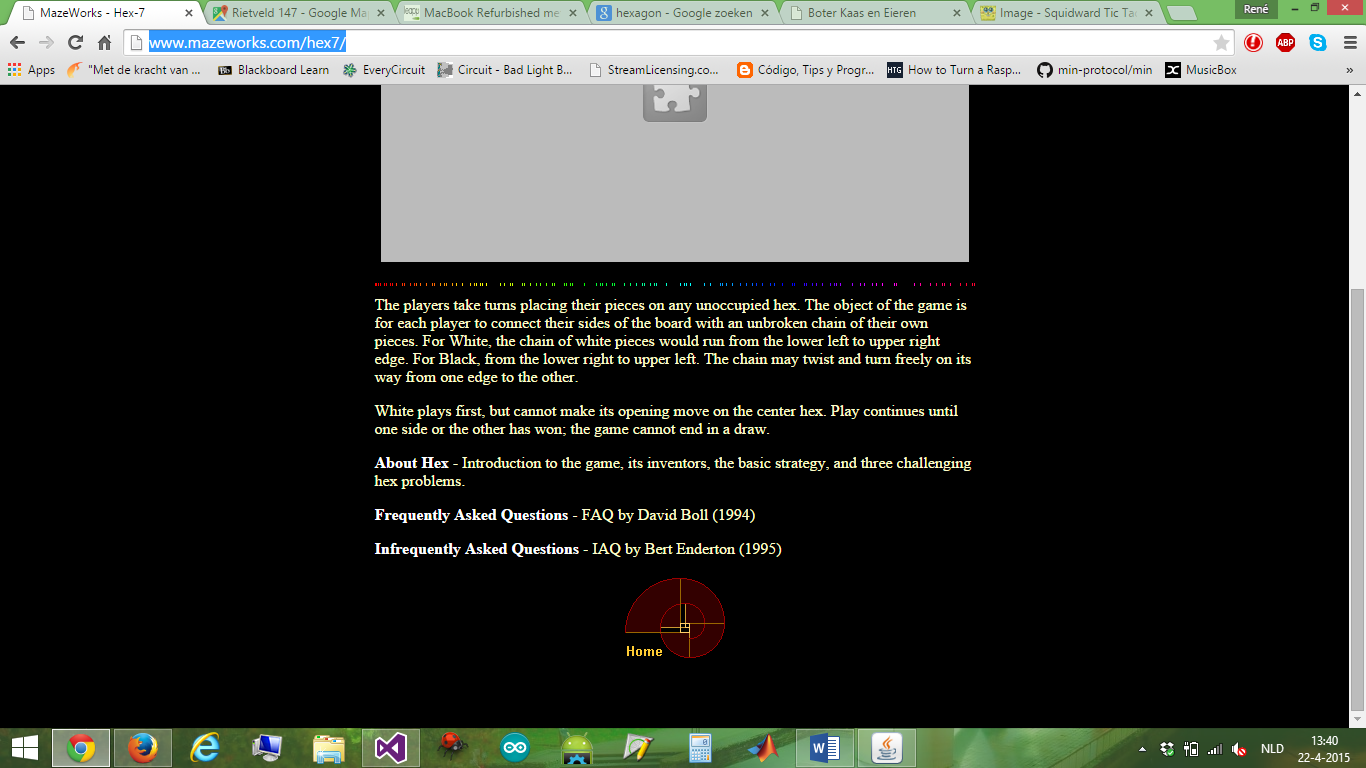


Figuur : Tic Tac Toe  
Bron: spongebob.wikia.com

Hex is gebasseerd op het “Tic Tac Toe”-programma van Harry Broeders: er is een aantal aanpassingen gemaakt zoals de winconditie, padzoek-algoritme, de manier waarop de tegels geplaatst kunnen worden en het printen van het Hex-bord in de console.

# De basis: Hex

Hex is, net als Boter, Kaas en Eieren, een bordspel met twee spelers waarbij elke speler speelt met tegels. Er mag, per beurt, één tegel gelegd worden op elke willekeurige plek. Het doel van het spel is om de overkant te bereiken met aanliggende tegels. Zie Figuur 2 voor een uitleg.



Figuur : uitleg van het bordspel Hex met een speler met zwarte steentjes en een speler met witte steentjes..  
Bron: mazeworks.com/hex7

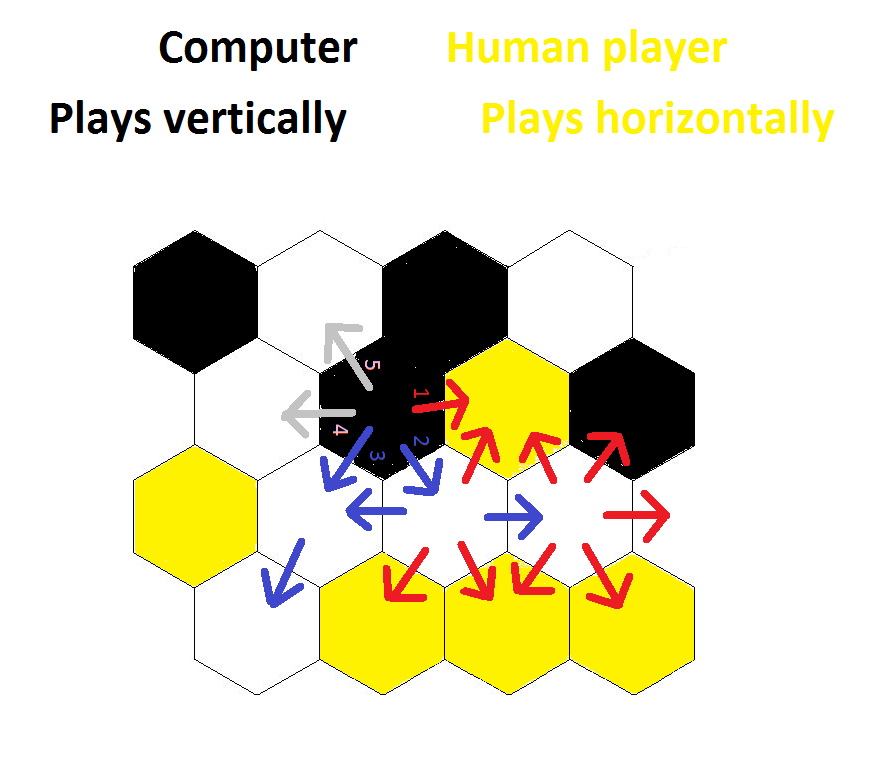
Het wordt al snel duidelijk dat bij dit spel de tegenstander geblokkeerd kan worden in zijn volgende zet. Het is daarom de truc om de overkant te bereiken, maar ook om ervoor te zorgen dat de tegenstander de overkant niet bereikt. Het is daarom aan de computer, de speler van ons programma, om het meest efficiënte pad te vinden en deze zet te plaatsen.

# Het programma

Er zijn drie grote aanpassingen gemaakt aan het “Tic Tac Toe”-programma van Harry Broeders: het padzoek-algoritme, de manier waarop de tegels geplaatst worden en de winconditie. Er is besloten om deze onderdelen individueel toe te passen en op werking te controleren.

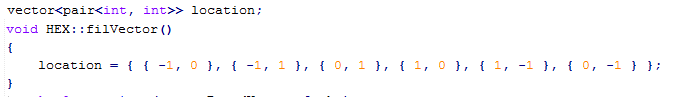
## Padzoek-algoritme

Het programma maakt voor elke tegel gebruik van een functie om een pad naar het eind van het bord te vinden. Deze functie wordt voor elke volgende tegel recursief aangeroepen en geeft het resultaat terug aan de vorige tegel. In Figuur 3 is te zien hoe er een pad gevonden kan worden door de speler met de zwarte tegels.



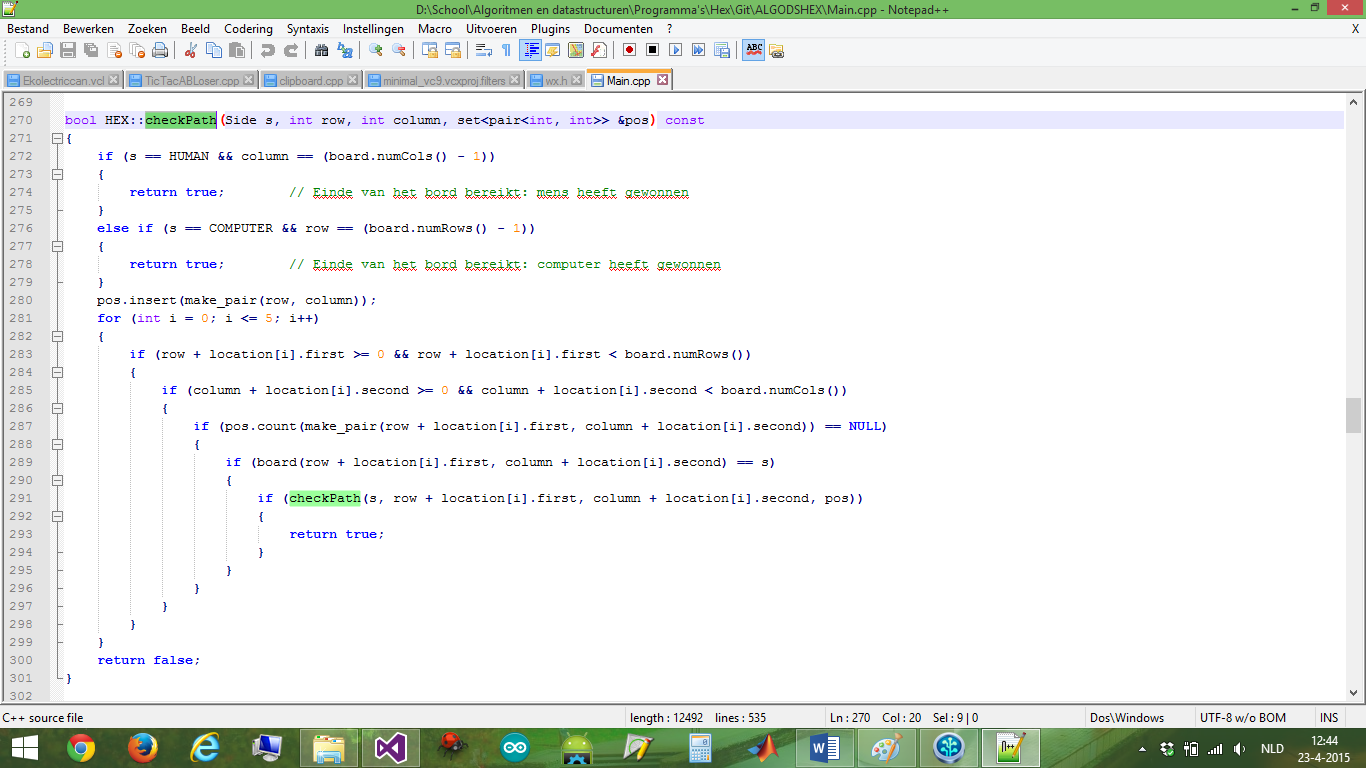
Figuur : padzoek-algoritme, schematisch.  
Merk op dat er vijf mogelijkheden zijn en dat de functie  
voor elke tegel (recursief) wordt gebruikt

Om ervoor te zorgen dat het programma elk vlak van de tegel benadert, is gebruik gemaakt van een (statische) vector, gevuld met pairs. Deze pairs zijn vervolgens gevuld met twee integers: de waarden van de x- en y-as waar naar gekeken moet worden ten opzichte van de huidige tegel. In Figuur 4 is te zien hoe deze vector, genaamd “location”, geïnitialiseerd wordt.



Figuur : vector met de locaties van elke aanliggende tegel.  
Merk op dat er zes mogelijke locaties zijn voor een hexagon

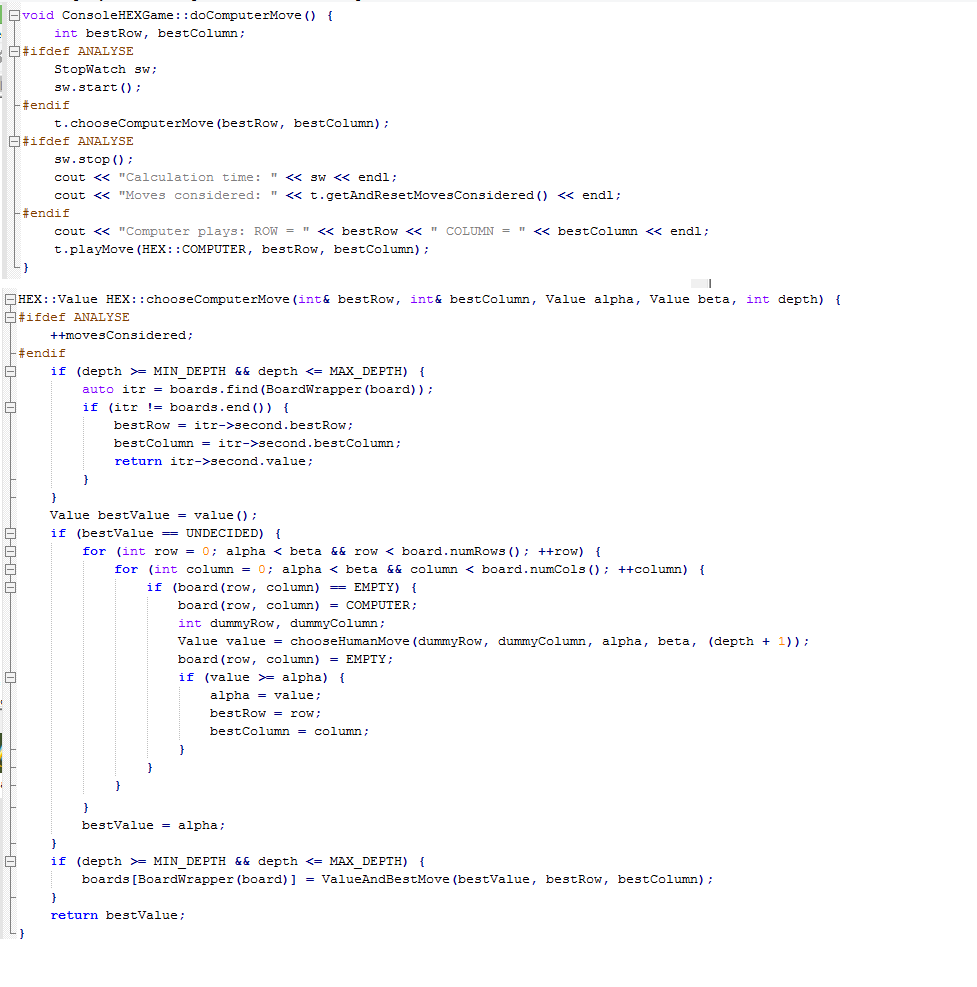
In Figuur 5 is te zien hoe het padzoek-algoritme geïmplementeerd is. Hier wordt de vector uit Figuur 4 gebruikt om bij elke omliggende tegel naar een pad te zoeken. Te zien is dat telkens wanneer de functie (recursief) aangeroepen wordt, er een nieuw pair met de huidige locatie aangemaakt wordt.   
Deze pair word opgeslagen in een set. Om te voorkomen dat het padzoek- algoritme niet twee keer op dezelfde plek komt word er in de set gekeken of de volgende tegel al in de set zit. Er is voor een set gekozen omdat hierin gezocht kan worden met de snelheid O(log n). Het recursief aanroepen van deze functie stopt wanneer er een pad gevonden is.



Figuur 5: het padzoek-algoritme.  
Merk op dat deze functie recursief wordt aangeroepen   
voor elke aanliggende tegel

## Plaatsen van de tegels

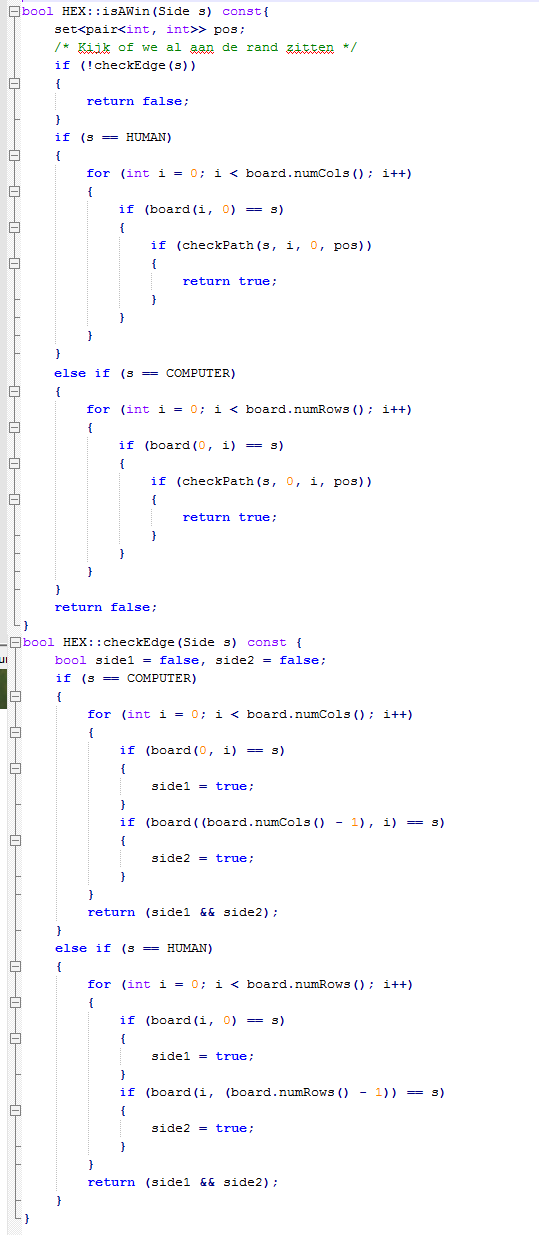
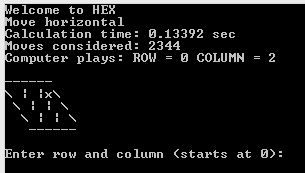
Net als bij het “Tic Tac Toe”-programma wordt chooseComputerMove() en chooseHumanMove() gebruikt om de volgende zet te bepalen. De waarden voor “beste kolom”en “beste rij” worden vervolgens opgeslagen en gebruikt om de tegels te plaatsen. Wanneer een “bestValue” gevonden is, wordt het (recursief) aanroepen gestopt. In figuur 6 is te zien hoe een waarde gekozen en gespeeld wordt.



Figuur 6: functies om de zet van de computer te bepalen   
zoals in het originele “Tic Tac Toe”-programma

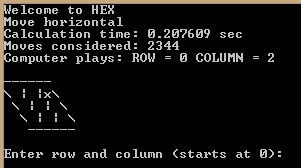
## Winconditie

Om de winconditie te bepalen is de functie “isAWin()” gebruikt. In deze functie word eerst gecontroleer of de desbetrevende speler een tegel aan bijde randen heeft met behulp an de functie “checkEdges()”. Als dit zo is word er gezocht naar een pad gezocht. Dit word gedaan door één van de rande af te lopen op zoek naar een tegel van de desbetrevende speler, als deze is gevonde word er hiervanuit gezocht naar een pad met de checkPath()-functie. In Figuur 7 is te zien hoe deze functie is geschreven.  
Door gebruik te maken van de functie “checkEdges()” word de tijd die nodig is voor het uitrekenen van de zetten verminderd.



Figuur : Tijd met de functie “checkEdges()”.

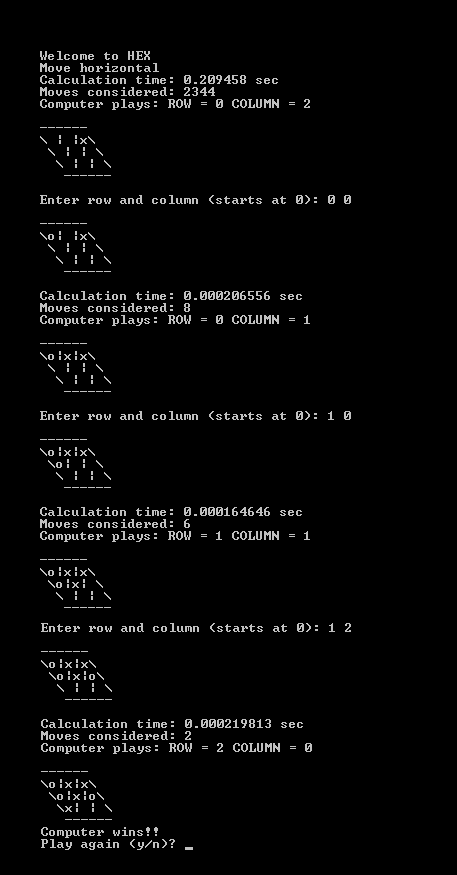
Figuur 9: bepalen van de winconditie



Figuur 8: Tijd zonder de functie “checkEdges()”.

# Conclusie, aanbevelingen en verbeterpunten

Hex is een spel waarbij de speler van de ene naar de andere zijde moet spelen, de één horizontaal en de ander verticaal. In dit programma speelt een persoon tegen de computer, waarbij de computer verticaal speelt. De basis van dit programma is afgeleid van het programma “Tic Tac Toe” van Harry Broeders. Er wordt gebruik gemaakt van een padzoek-algoritme: een functie waarin recursief gekeken wordt of er een pad naar de andere kant van het bord is. Ook is de winconditie aangepast, namelijk door te kijken of er aan beide kanten van het bord een tegel van de desbetreffende speler ligt en of er een pad hiertussen ligt.   
In dit spel is Alpha-Beta-Pruning toegepast om het berekenen van het mogelijke aantal zetten te verminderen. Hoewel dit programma gebasseerd is op een bord van drie bij drie tegels, werkt het ook met meer tegels. Het aantal mogelijke zetten en de tijd om dit te berekenen neemt dan wel drastich toe en het is dan ook niet meer reeël om hier op te wachten.  
Om dit te verbeteren zoe er een maximale zoekdiepte moeten worden toegepast. Hiervoor moet een algoritme worden gemaakt die een schatting kan maken van de winkans van de computer.  
Het idee hiervoor is is om te kijken hoe dichtbij de computer is bij het maken van een verbinding tussen de randen, ook kan er een hogere waarden worden gegeven aan zetten die de tegenspeler hinderd.



Figuur 10: het resultaat