Brecht Stevens
Jelle De Moerloose
Stijn De Schrijver
Wiebe Vandendriessche

Softwareproject Analyse

Concept

Wij kwamen op het idee om een schaakwebsite te maken. Op de website zal de gebruiker in staat zijn om tegen een andere gebruiker te spelen of om tegen een bot te spelen. De bot wordt gemaakt aan de hand van een algoritme.

Navigatie

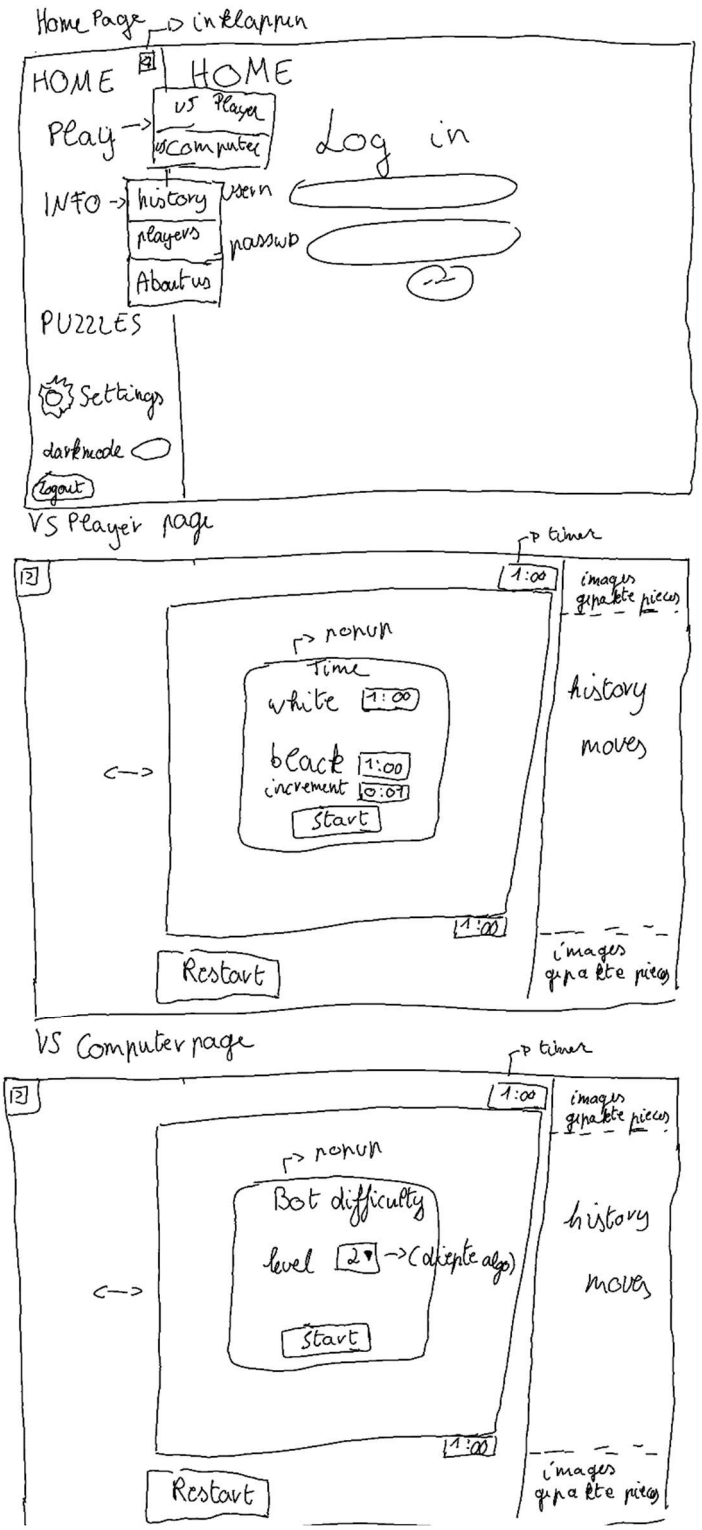
Links op de website is er een navigatiebalk waar je kan navigeren naar verschillende onderdelen van onze site. Voorlopig hebben we deze onderdelen bedacht: *Play*, *Info*, *Puzzles*, … . Er zal ook een *toggle*-knop zijn waarmee we dark-mode kunnen activeren. In het navigatie-venster zal ook een knop, *settings* aanwezig zijn. Daar kan onder andere volume en thema's voor het schaakbord aangepast worden.

Play

De *player vs player* modus voorziet voor elke speler een timer die vooraf kan ingesteld worden. Ook is er een *restart*-knop voor het geval een speler opgeeft. Rechts worden de gespeelde *moves* weergegeven en onderaan de *captured pieces*.

De *player vs bot* modus heeft een heel gelijkaardig venster, alleen is er geen timer. Je zal de moeilijkheid van de bot kunnen instellen voor het starten.

Ten derde, indien er tijd over is, zullen we een *puzzle*-pagina maken. Op deze pagina krijg je Schaakpuzzels te zien die je kan oplossen voor punten. Iedere puzzel wordt moeilijker.



Figuur 1: schets 1

Brecht Stevens
Jelle De Moerloose
Stijn De Schrijver
Wiebe Vandendriessche

Info pages

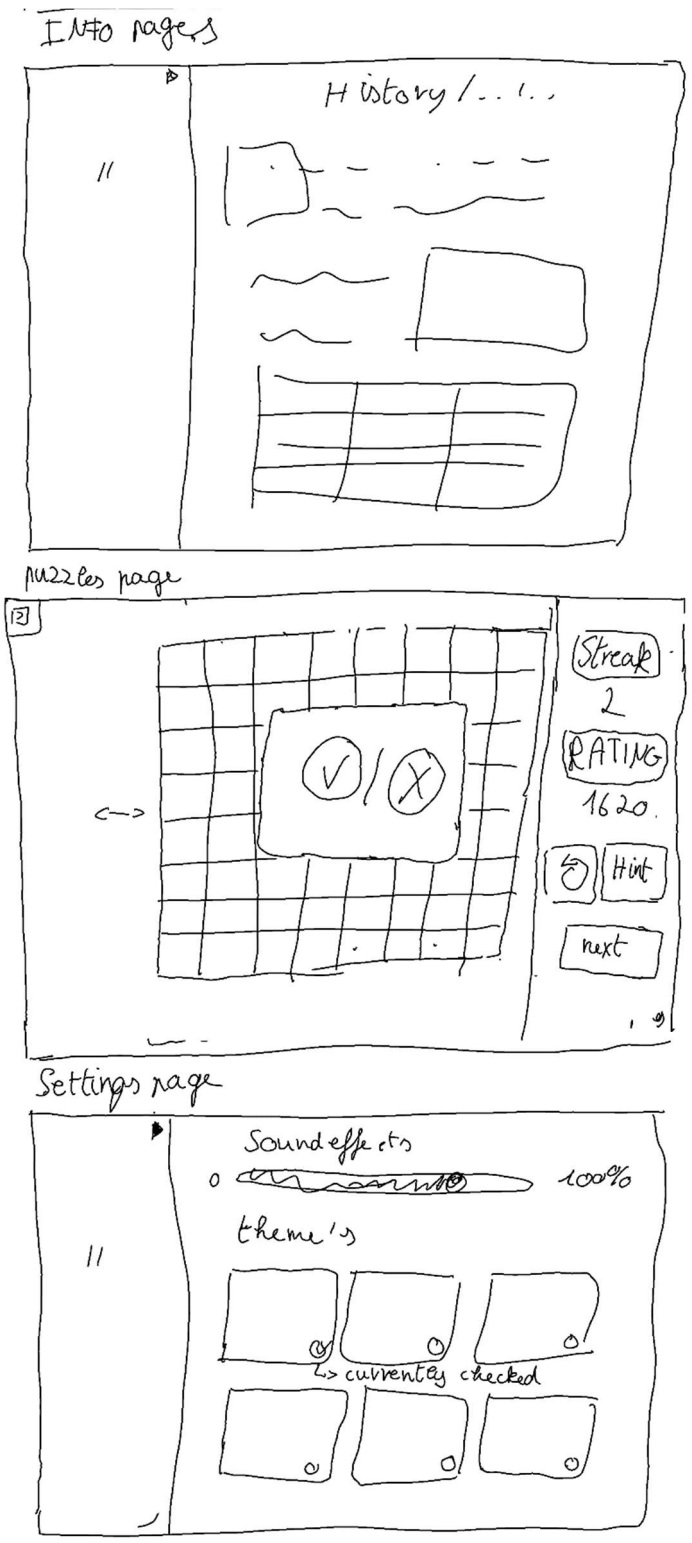
Natuurlijk is het belangrijk dat de gebruiker ook weet hoe hij moet spelen. Daarom maken we enkele infopagina's: *Rules, About us, History en Players*. Deze pagina's zullen eerder statisch zijn.

De *Rules* pagina zal uiteraard uitleg geven over de spelregels van het schaken.

In de *About Us* zal uitleg staan over ons en over het project.

De *History* pagina zal uitleg geven over de geschiedenis van het schaken. Waar is schaken ontstaan? Wie heeft het uitgevonden? Hoe is het wereldwijd bekend geworden?

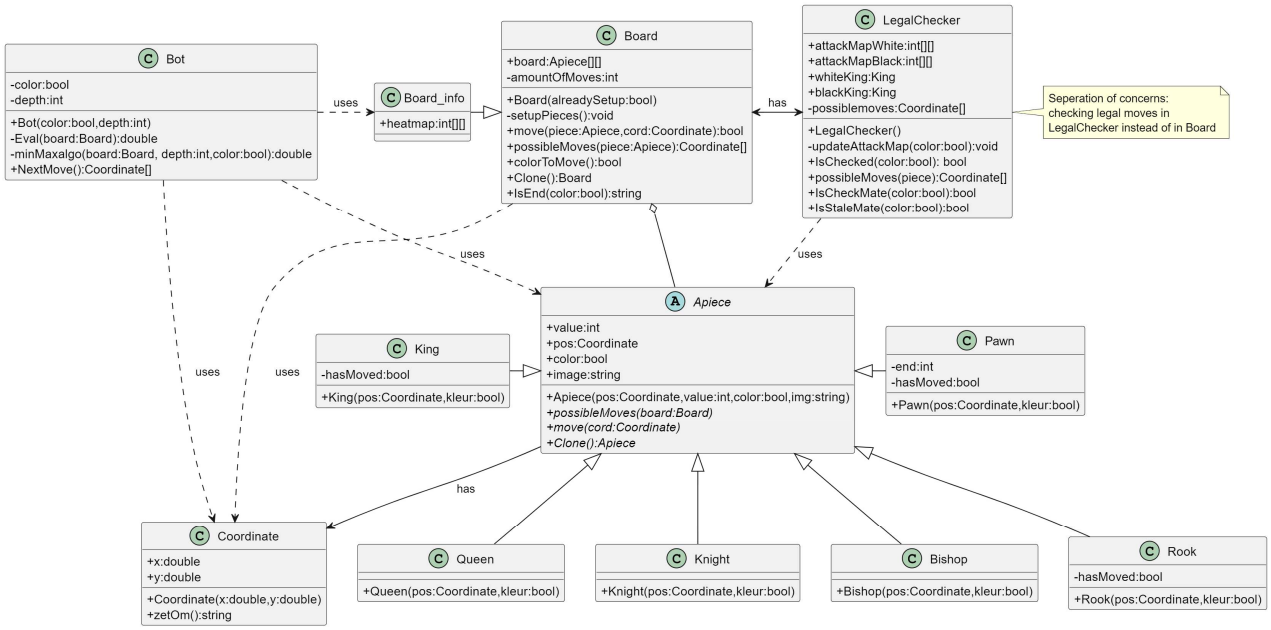
De *Players* tab zal uitleg voorzien over bekende schaakspelers.



Figuur 2: schets 2

Brecht Stevens
Jelle De Moerloose
Stijn De Schrijver
Wiebe Vandendriessche

Klassendiagram schaakmodel



Uitleg:

Apiece (+ afgeleiden):

Deze abstracte klasse houdt alles bij dat een piece zou moeten hebben: waarde, positie, kleur, link naar juiste afbeelding. In deze abstracte klasse is de constructer veel uitgebreider dan in de pieces apart omdat elke soort (pawn,rook,…) piece bepaalde zelfde eigenschappen hebben . De abstracte klasse Apiece legt ook aan elke subklasse op om zijn possibleMoves() en move() en Clone() functie te implementeren. Enkele afgeleide klasse hiervan hebben extra attributen die te maken hebben met hun bewegingen op het bord (Pawn en King).

Board:

Het schaakbord (Board) bestaat uit een 2D array die alle pieces bijhoudt, een LegalChecker en het aantal moves dat gespeeld is (deze wordt gebruikt om te weten aan welke kleur de beurt is). Een bord kan worden aangemaakt met een boolean als parameter, deze bepaalt of alle pieces worden klaargezet (setupPieces()) in de standaardstartopstelling of leeg gelaten wordt. Dit is niet altijd wenselijk want in de functie Clone() wordt een nieuw bord gemaakt in een toestand die vaak anders is. Voor de rest bevat het bord de functies possibleMoves() en Move() voor de mogelijke moves van een piece te vragen en deze echt te moven. Ook heeft deze klasse een functie die weergeeft in welke staat het spel zich begeeft (IsEnd()), deze geeft een specifieke string terug voor elk van deze staten: checkmate, stalemate en OK.

LegalChecker:

Dit is een klasse die het board bijhoudt en het board helpt om te checken welke moves helemaal legaal zijn, hierbij wordt er rekening met checks en discovered checks. Het zal ook aan het bord vertellen of het checkmate of stalemate is. De checks worden gecontrolleerd aan de hand van 2

attackmaps voor de twee kleuren, deze bevatten elk een 2D array van integers die bijhouden hoeveel aanvallers elk vakje heeft. Omdat de positie van de koningen vaak nodig is om de checks te controleren, houdt LegalChecker een referentie naar deze bij.

Coordinate:

Simpele klasse die x en y coördinaat bijhoudt en deze kan omzetten naar een string in schaakvorm, zo wordt (3,6)-> “C7” .

Board_info:

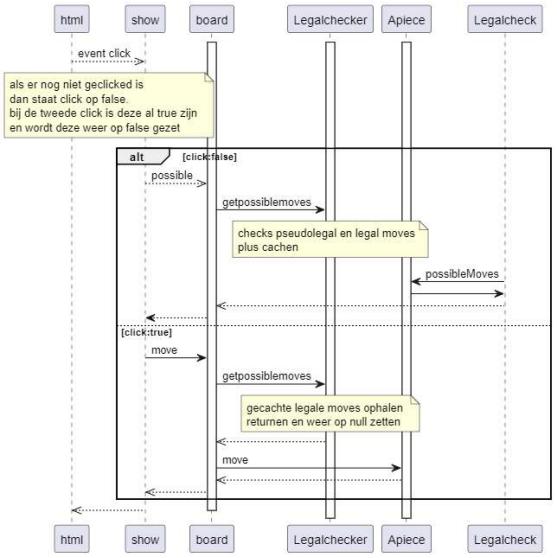
Dit is een afgeleide klasse van Board die zal worden gebruikt als er tegen de bot wordt gespeeld, hierin zal extra informatie zoals heatmaps kunnen worden gemaakt en bijgehouden die de bot helpen om zijn positie te evalueren.

Bot:

Deze bot wordt gemaakt voor een bepaald kleur en met een bepaalde moeilijkheidsniveau (depth) en heeft een publieke functie die de volgende move teruggeeft van de bot. Deze functie bepaald de beste volgende zet aan de hand van de minimaxfunctie. De minimaxfunctie maakt op zijn beurt ook gebruik van de evaluatiefunctie die een hypothetische positie een waarde toekend. De evaluatiefunctie kan ver worden uitgebreid moet verschillende algoritmes om zo een zo goed mogelijke zet te berekenen.

Brecht Stevens
Jelle De Moerloose
Stijn De Schrijver
Wiebe Vandendriessche

Sequentiediagrammen

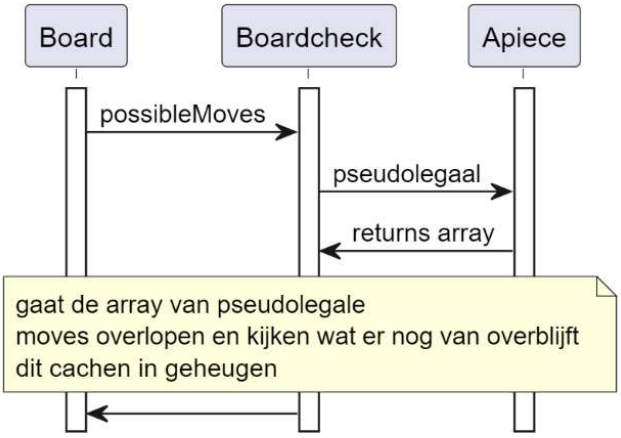


De moves werken aan de hand van pseudolegale en legale moves.

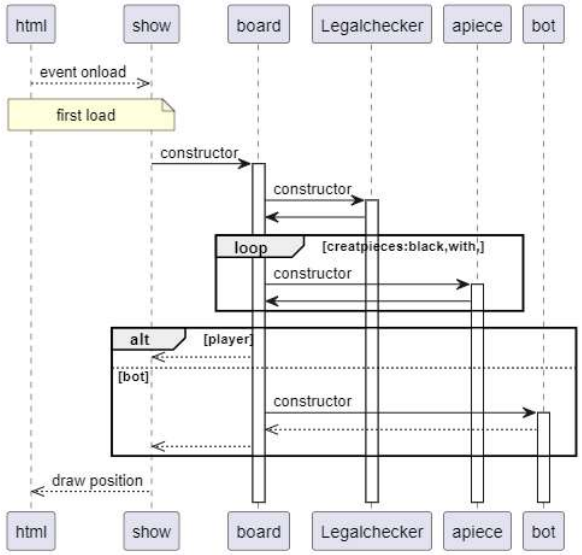
Pseudolegale moves zijn de zetten die een stuk mogen doen in het algemeen. Voor een paard is dit bijvoorbeeld een L-vorm

Legale-moves zijn de pseudolegale moves maar ook rekening houdend met checks, discovered checks en castelen.

De pseudolegale moves worden gecontroleerd in de piece zelf. De legale zetten daarintegen worden in een klasse legalchecker gedaan. Deze legale moves worden gecached in de Legalchecker zodat er niet dubbel wordt gerekend.



Dit is een uitvergroot deel van het checken van de legale zetten hier word het cachen nog eens duidelijk gemaakt. De reden dat het gecached moet worden is omdat we dan eerst de functie oproepen voor de legale zetten aan te duiden op het bord. De tweede keer als een zet wordt gedaan om te controleren of het een zet is die gedaan mag worden.



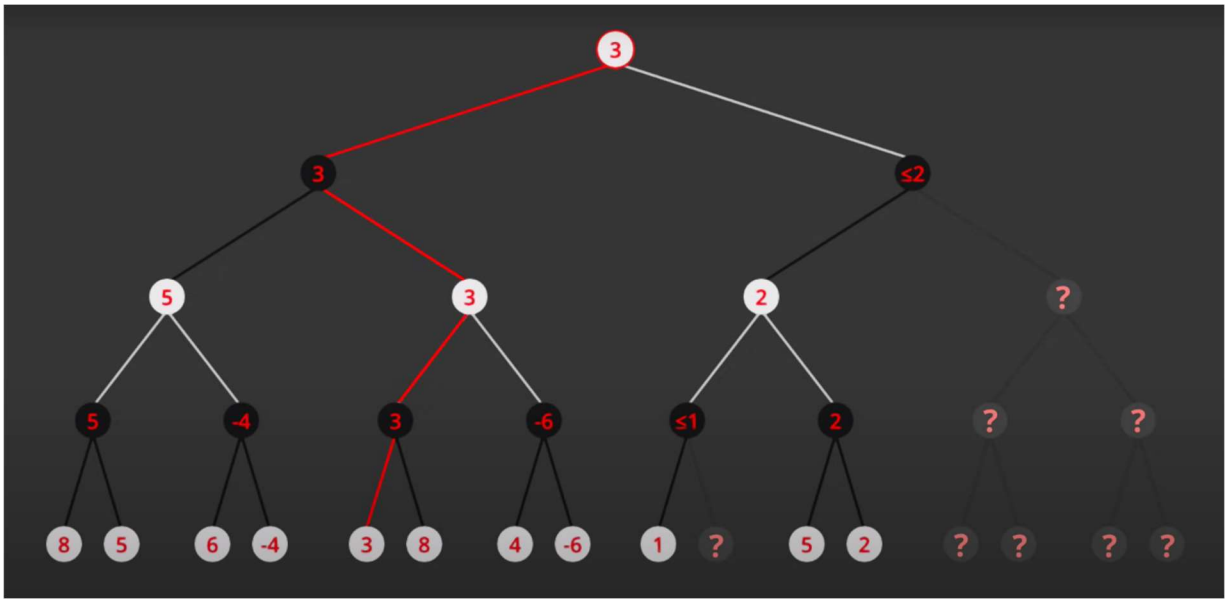
Dit is het diagram dat toont hoe alle objecten worden aangemaakt.

Brecht Stevens
Jelle De Moerloose
Stijn De Schrijver
Wiebe Vandendriessche

Minimax algoritme met pruning

Basiswerking van het algoritme

Hieronder is een voorbeeld gegeven van hoe het algoritme werkt. Het algoritme werkt in een boomstructuur met allemaal verschillende knooppunten. Deze knooppunten ontstaan omdat het algoritme zichzelf een bepaald aantal keer oproept. De witte lijnen zijn de mogelijke moves van een wit schaakstuk en de zwarte lijnen zijn de mogelijke zetten voor de zwarte stukken.



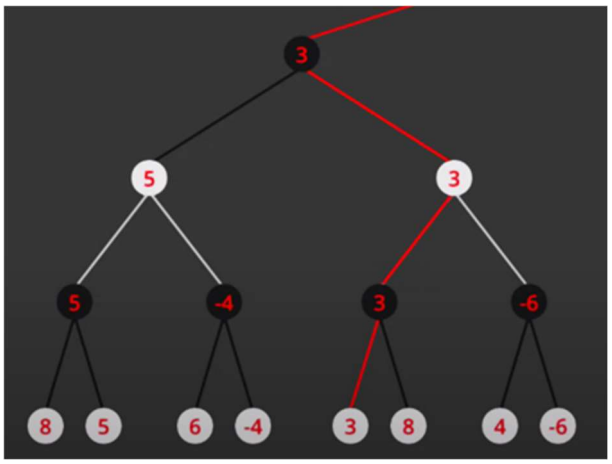
Figuur 3: Minimax boomstructuur

Het bovenste knooppunt stelt een schaakstuk voor en de 2 aftakkingen stellen elk een mogelijke beweging voor. Daarna gaat het algoritme een aantal stappen vooruit denken (hier zijn dat er 4). Bij de volgende knooppunten gebeurt hetzelfde als bij het eerste maar dit voor de andere kant. Dit gaat zo verder tot de ingestelde diepte is bereikt.

De witte knooppunten prefereren de hoogste waarde van de onderliggende waarden en de zwarte prefereren de laagste waarde.

Het berekenen van de beste move begint links onderaan bij de zwarte vertakking (figuur 4). Eerst worden de waardes van de onderste witte knooppunten berekend en dan gaat de bovenliggende knoop de laagste waarde hiervan nemen. Dit gebeurt ook bij de volgende zwart vertakking, deze gaat de waarde -4 nemen. Dan gaat het bovenliggende knooppunt de hoogste waarde nemen.

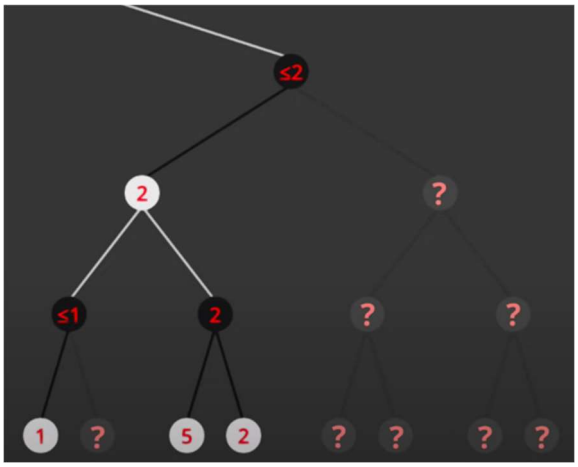
Dit proces wordt herhaald voor alle andere takken en knopen. Uiteindelijk gaat de bovenste knoop de hoogste waarde nemen van de rechtstreeks onderliggende knopen.



Figuur 4: linkse vertakking

Brecht Stevens
Jelle De Moerloose
Stijn De Schrijver
Wiebe Vandendriessche

Bij het rechterdeel (figuur 5) zijn er vraagtekens te zien, dit is omdat deze waarden niet moeten berekend worden. Dit kan bij grote vertakkingen veel rekenwerk en tijd besparen.



Figuur 5: rechtse vertakking

In figuur 5 beginnen de berekeningen ook linksonderaan. Hier wordt de waarde 1 berekend, hieruit kan afgeleid worden dat de waarde van de bovenliggende knoop 1 of lager is want die gaat de laagste waarde kiezen. Stel dat de waarde 1 zou doorsijpelen naar de bovenste zwarte knoop zou de eerste knoop de waarde 3 nemen en niet 1.

Hierna wordt de volgende vertakking berekend. Hiervan gaat de waarde 2 doorsijpelen naar de bovenste zwarte tak. Deze tak gaat de laagste waarde nemen, dus de waarde zal 2 of lager zijn als de andere vertakkingen een lagere waarde zouden genereren. Dit is onnodige omdat de eerste tak de hoogste waarde gaat nemen (in dit geval is dit 3), omdat 3 hoger is dan 2 is de rest onnodig rekenwerk.

Brecht Stevens
Jelle De Moerloose
Stijn De Schrijver
Wiebe Vandendriessche

Activiteitendiagram

Hiernaast staat het activiteitendiagram voor het algoritme. Het diagram is opgedeeld in 3 stukken waaronder het hoofdgedeelte en 2 whileloops die worden opgeroepen in het hoofdgedeelte. Dit is gedaan om praktische redenen zodat het overzicht bewaard wordt.

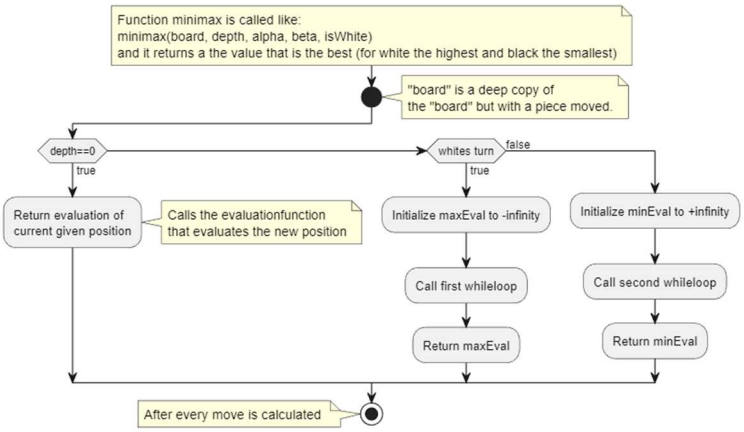
Het algoritme zit in een functie die bij het oproepen de volgende variabelen nodig heeft: board, depth, alpha, beta en isWhite.

In de variabele “board” wordt een clone gemaakt van de toestand van het huidige schaakbord met een nieuwe mogelijk zet in verwerkt.

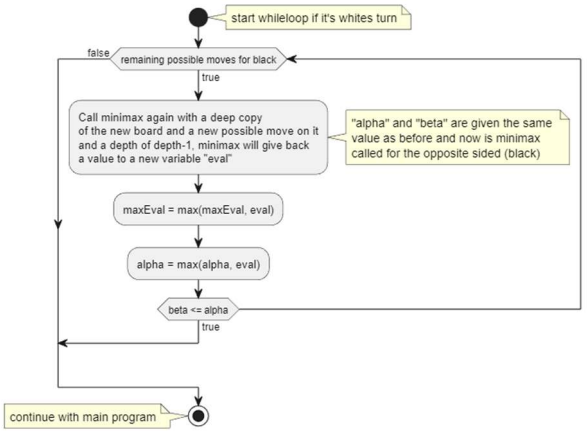
Depth is een integer en die bepaald hoeveel stappen het algoritme moet vooruitdenken.

Alpha en beta zijn nodig voor het kunnen schrappen van bepaalde takken als die berekeningen overbodig zijn.

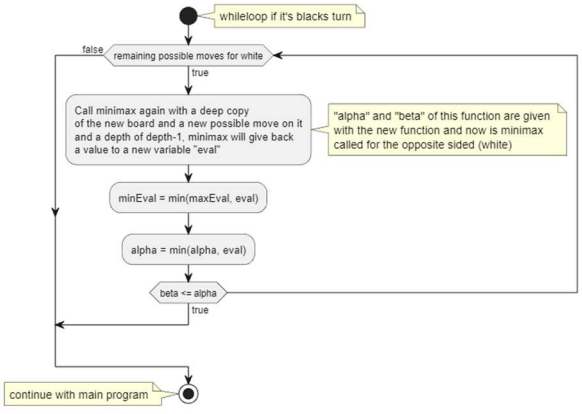
De variabele isWhite is een referentie voor welke kant de waarden moeten berekend worden.



Figuur 6: hoofdgedeelte



Figuur 7: whileloop 1



Figuur 8: whileloop 2

Brecht Stevens
Jelle De Moerloose
Stijn De Schrijver
Wiebe Vandendriessche

Bronnen

Chess Wiki
https://www.chessprogramming.org/Main_Page?fbclid=IwAR2qOof_TcYPxKgVHaoEP_kO4XFNen4EVrOtp3lYzZJD3KtzDLMNL03nQuY

I created a chess AI – Code Bullet
https://www.youtube.com/watch?v=DZfv0YgLJ2Q&t=55s

Algoritms explained: minimax and alpha-beta pruning
https://www.youtube.com/watch?v=l-hh51ncgDI

Coding adventure: chess
https://www.youtube.com/watch?v=U4ogK0MIzqk&t=614s

Minimax:
https://en.wikipedia.org/wiki/Minimax