UNIVERSITEIT GENT

Eindverslag Softwareproject semester 2
Informatica

Genetisch Algoritme voor Tetris

door
groep 4

Matisse Callewaert
Tomas Galle
Akash Srivastava
Nathan Salabiaku
Josse Callens

Universiteit Gent
Faculteit Ingenieurswetenschappen en architectuur
Opleiding Industriële wetenschappen: Informatica

Academiejaar 2021-2022

Inhoudsopgave

Inleiding

Figuur 1: Screenshot website

Genetische algoritmes worden al lang gebruikt en zijn toepasselijk in veel vakgebieden, waardoor het de interesse van vele informatici wekt. Het team vertrok van het idee om hier iets rond te maken. De meest voor de hand liggende toepassing was om het te gebruiken om een spel te laten spelen. Nu moest een bepaald spel gekozen worden, waarvan de moeilijkheid om het te programmeren niet in de weg mocht staan van het hoofddoel om een genetische AI op te bouwen. Een eerste optie was *snake*, maar er werd uiteindelijk gekozen voor *tetris*.

Het doel van het project was hiermee beslist: een website maken waarop een gebruiker zelf tetris kan spelen of ervoor kan kiezen een AI aan te zetten die een genetisch algoritme gebruikt om het spel zo goed mogelijk te leren spelen. Er werd beslist om de code en het genetisch algoritme uit te leggen aan de gebruiker op de website om die iets bij te leren, waardoor het verband met SDG vier (onderwijs) gelegd was. Aangezien genetische algoritmes ook gebruikt worden in de medische sector kan hier ook deels SDG drie (gezondheid) aan gelinkt worden. Het gebruik van het genetisch algoritme heeft veelbelovende implicaties in verschillende medische sectoren, waaronder radiologie, radiotherapie, oncologie, kindergeneeskunde, cardiologie, endocrinologie, chirurgie, verloskunde en gynaecologie, pulmonologie, infectieziekten, orthopedie, revalidatiegeneeskunde, neurologie, farmacotherapie, en gezondheidszorgbeheer.

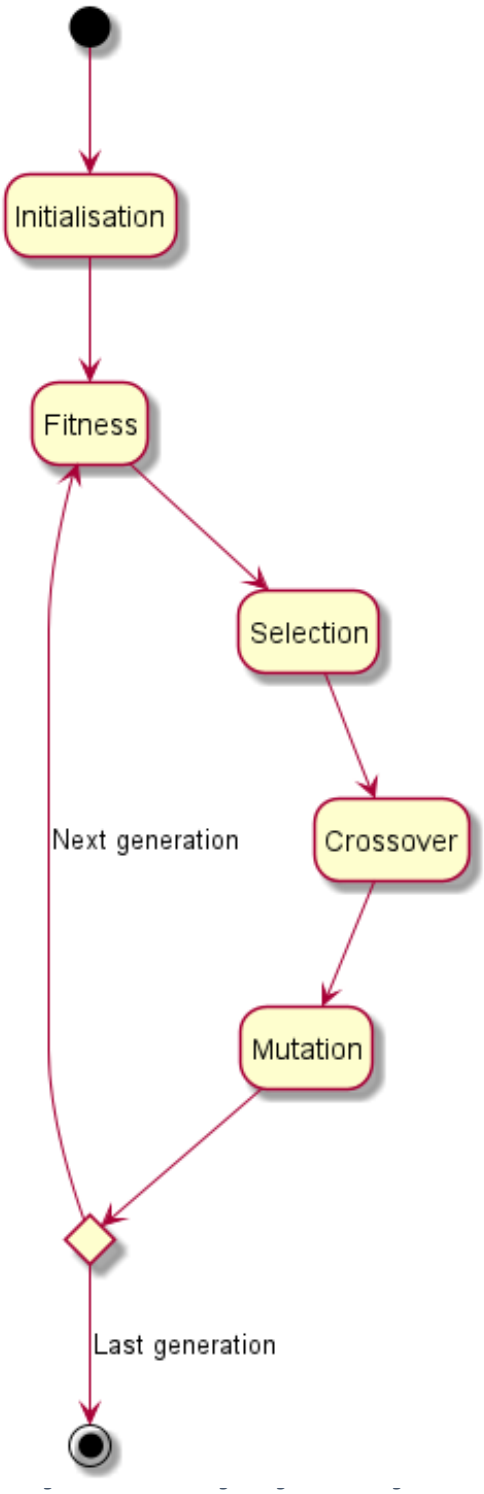
Als open data werd gebruik gemaakt van een website waarop de beste spelers van het internet werden getoond zodat de gebruiker zichzelf of de AI kon vergelijken met deze spelers.

Het project bestond uit vier delen en het team had deze taken onder zich verdeeld: het tetris spel (Matisse), de AI (Nathan), het verwerken van open data (Josse) en de UI en functionaliteit van de website (Akash en Tomas).

Op Figuur 1 ziet men de uitwerking van het project. De speler kan het spel spelen en op ‘a’ drukken om de AI te laten overnemen. Optioneel kan de speler ook op de knop ‘Best’ klikken om de AI onmiddellijk op een hoog niveau te brengen zodat het niet meer moet leren. Om de verbetering van de AI te volgen is er een grafiek toegevoegd die van elke generatie het gemiddeld aantal zetten bijhoudt, wat dus het leren van de AI visualiseert. De parameters die de AI gebruikt om het spel te spelen worden ook altijd getoond zodat de speler kan volgen hoe deze genen veranderen.

Genetische algoritmes

In Figuur 2 wordt een statendiagram van genetische algoritmes (GA afgekort) getoond.



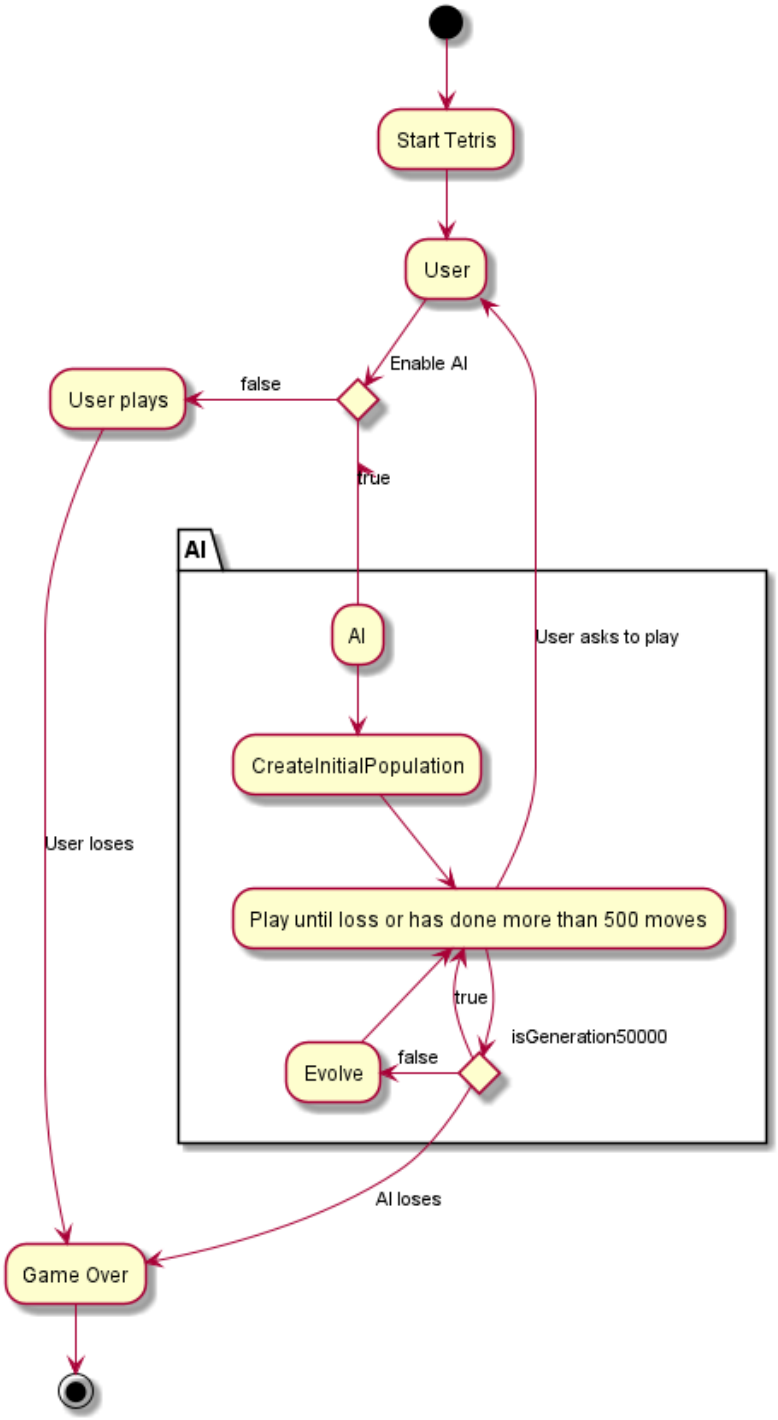
GAs zijn algoritmes gemaakt om optimalisatie-problemen op te lossen. Ze zijn gebaseerd op de evolutietheorie en natuurlijke selectie, waarin generaties van een soort zich aanpassen en evolueren om zich zo goed mogelijk aan de omstandigheden aan te passen. GAs vinden oplossingen met processen die gebaseerd zijn op natuurlijke selectie zoals selectie, cross-over en mutatie. Hoe meer generaties een GA is ondergaan, hoe meer waarschijnlijk dat die generatie de best mogelijke oplossing heeft gevonden op het gegeven probleem.

Een GA start met een populatie van een gekozen grootte van chromosomen met elk een eigen set van willekeurige gen-waarden. Een chromosoom is te vergelijken met een persoon die een aantal parameters bijhoudt, dit zijn de genen. Nadat elk chromosoom geëvalueerd is op basis van de respectievelijke gen-waarden en een fitness-score heeft gekregen, begint de reproductie. Dit proces kan variëren van toepassing tot toepassing, maar over het algemeen worden de hoogst scorende chromosomen gekozen om kinderen met elkaar te maken. Tijdens het maken van een kind wordt elke parameter willekeurig van een van de ouders gekozen, wat dus cross-over is. Na dit proces heeft elke gen een kleine kans om te muteren, zodat willekeurige variatie verandering kan brengen over de generaties heen. De reproductie wordt herhaald tot de nieuwe generatie even groot is als de vorige, en dit proces blijft zichzelf herhalen.

Het grootste probleem met genetische algoritmes is om goede parameters voor de mutatie te zoeken; een te grote kans of te grote stap zorgt voor te veel variatie, waardoor het kan zijn dat er nooit een goede oplossing gevonden wordt, maar wanneer deze parameters te klein zijn is er te weinig variatie en kan het zijn dat er een “oplossing” gevonden wordt omdat deze niet meer veranderd, maar dat dit niet de beste is. Er is veel discussie en documenten die onderzoek bevatten over wat de beste parameters zijn, maar uit eigen ondervinding is trial and error een degelijke manier om deze te bekomen.

Analyse Diagrammen

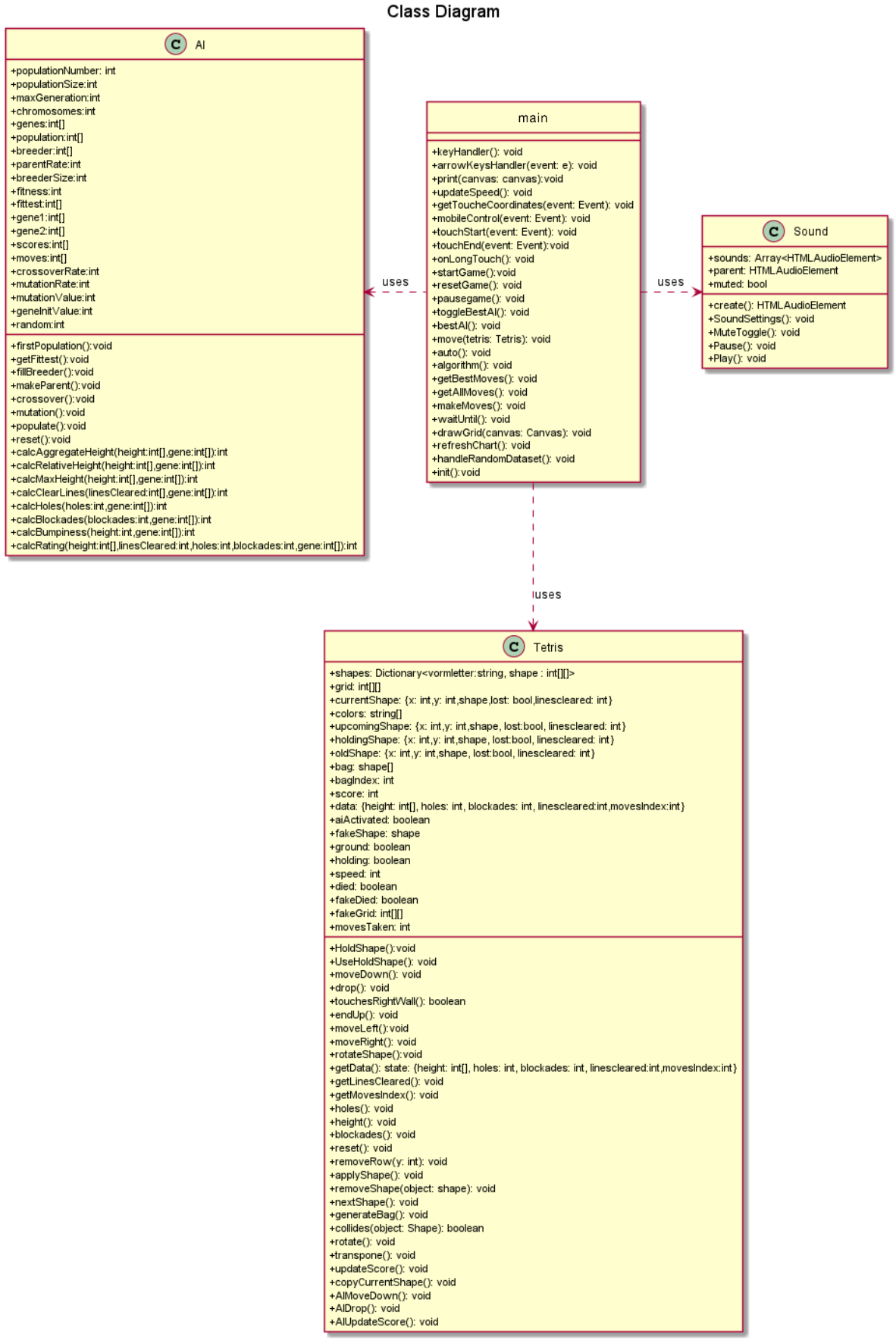
Algemeen statendiagram



Figuur 3: Statendiagram

In Figuur 3 wordt het algemene verloop van een spel voorgesteld. De speler kan kiezen om zelf te spelen of om de AI te laten overnemen, en kan later nog kiezen om dit te veranderen. De AI begint met een willekeurige populatie en zal een spel spelen tot het verloren heeft, of tot het over 500 zetten gaat. Daarna maakt hij een nieuwe generatie gebaseerd op de beste chromosomen van de vorige als dit nog niet de laatste generatie was.

Klasse diagram

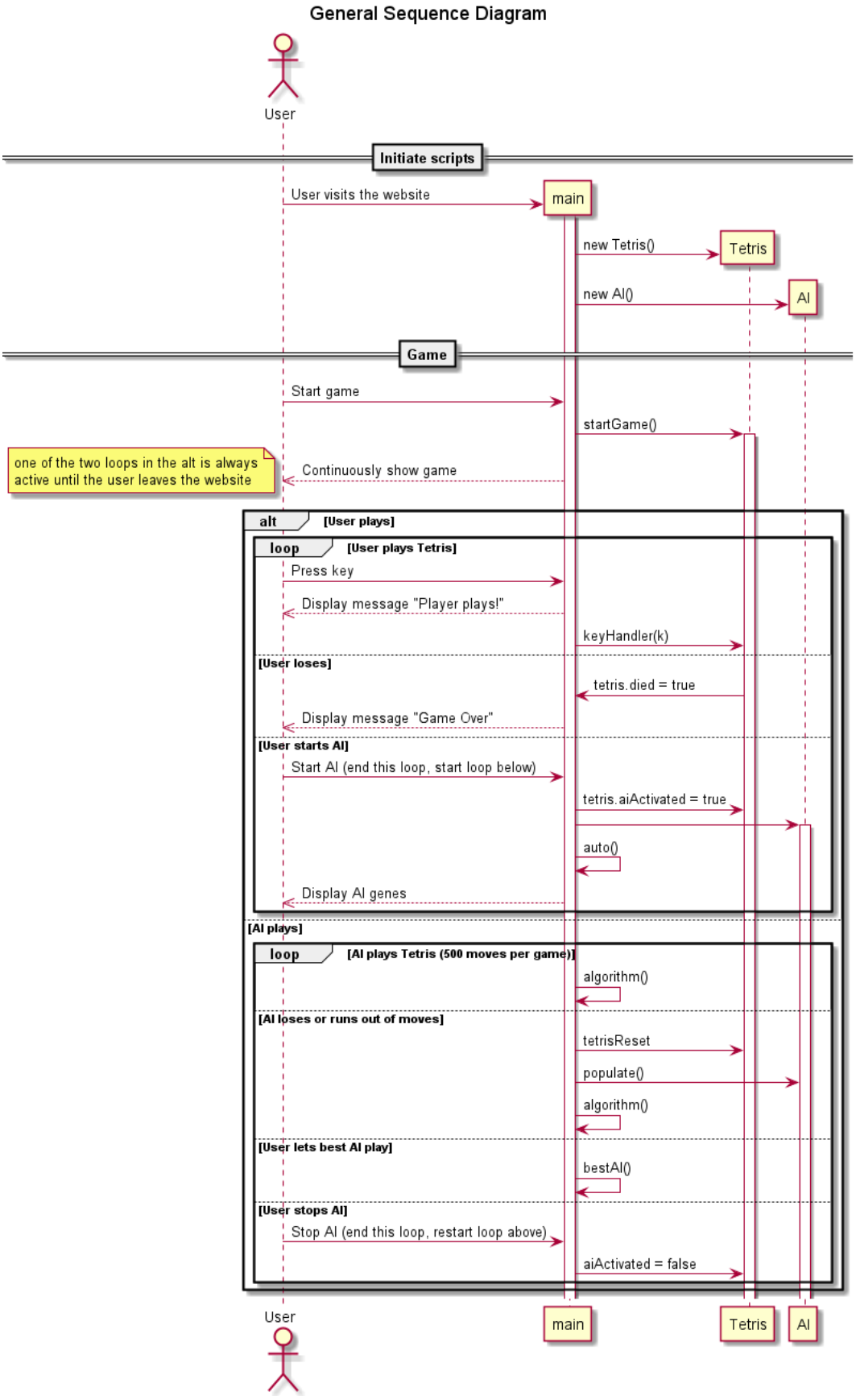


Figuur 4: Klassendiagram

In Figuur 4 worden enkel de klassen getoond die te maken hebben met het spel, niet met de UI (User Interface) of de UX (User Experience). Het team besloot om 3 klassen te gebruiken: de AI-klasse die alle AI logica bezit, de tetris klasse die de logica voor het spel bezit en de voorgebouwde Sound klasse om geluidseffecten toe te voegen. Main verbindt deze klassen met elkaar en beheert de data en communicatie tussen deze klassen. Het verzorgt ook het verloop van het spel en het wisselen tussen manueel spelen en de AI laten spelen.

De website werd ook op gsm met touch controls ondersteund, maar schijnbaar zonder reden stopte deze code tijdens het ontwikkelingsproces met werken. Aangezien de focus van het project hier niet op lag werd besloten om hier niet verder op in te gaan.

Algemeen sequentiediagram



Figuur 5: Algemeen sequentiediagram

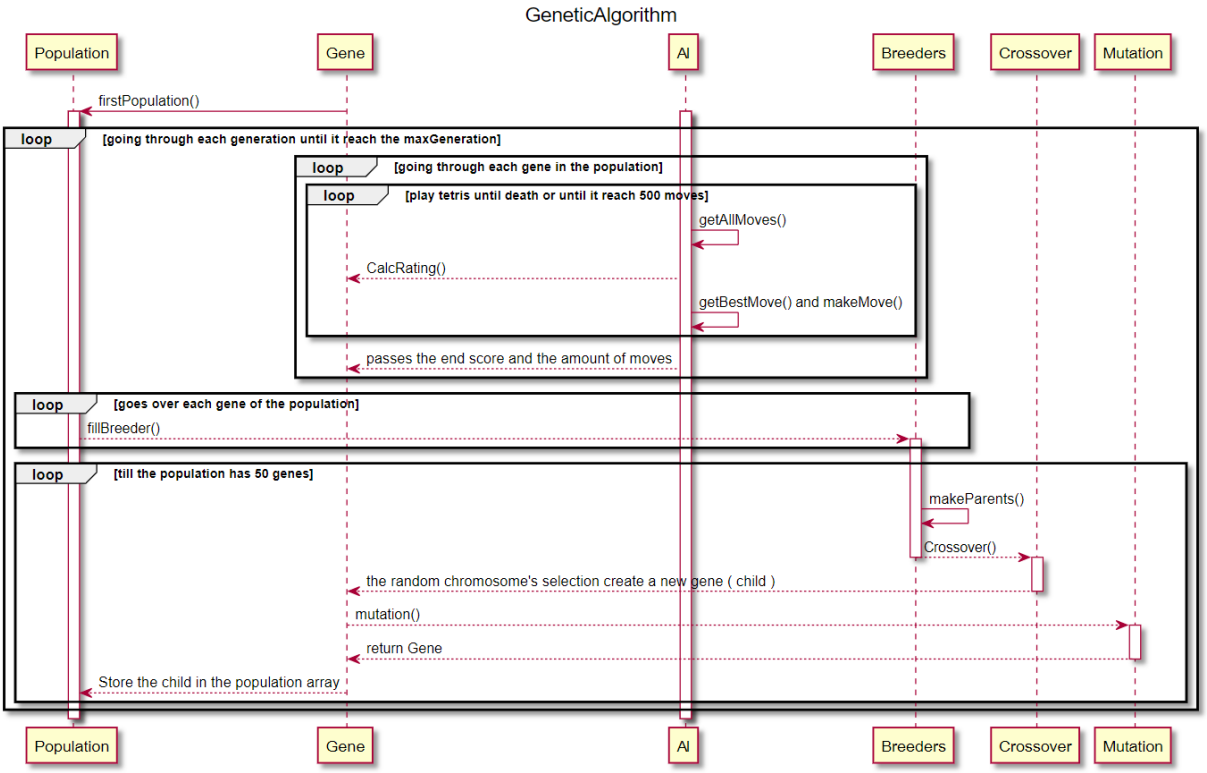
In Figuur 5 wordt de volgorde van de belangrijkste functies getoond. Hier wordt getoond hoe de AI het spelen overneemt, en het leren wordt later in meer detail beschreven. Dit diagram maakt ook duidelijk dat het spel altijd in een van twee staten is, die eerder in het statendiagram werden beschreven. Het maakt ook duidelijk dat tetris en de AI nooit direct met elkaar communiceren, maar dit via main doen, wat ook duidelijk in het klassendiagram gevisualiseerd werd.

De auto() en algorithm() functies worden op main zelf opgeroepen, wat er op het eerste gezicht raar uit ziet. Dit is echter logisch als men naar de code kijkt, omdat dan duidelijk wordt dat deze functies allerlei andere functies van zowel tetris en AI oproepen en de data hiertussen beheert.

Alle info die de speler moet zien (bv. speelveld, huidige score, volgende tetromino, ...) wordt in main verwerkt en getoond op de webpagina door periodisch een zelfgemaakte print() functie uit te voeren.

Het wisselen tussen zelf spelen en de AI gebeurt op basis van een boolean, waardoor de gameloop in main kan bepalen welke functies moeten uitgevoerd worden.

Sequentie diagram van het genetisch algoritme



Figuur 6: Sequentiediagram algoritme

In Figuur 6 wordt de werking van het genetisch algoritme en zijn constant gebruik van loops getoond. Wanneer de website start, wordt een initiële populatie gemaakt van 50 chromosomen met willekeurige genen die de parameters voorstellen. De chromosomen in de populatie worden 1 per 1 doorlopen door de AI, die ze zal gebruiken om zijn moves te raten. Op het moment dat hij zal sterven of zijn 500 moves zal halen, gaat hij dat chromosoom een rating toekennen, door de eindscore en de het aantal gemaakte zetten. Eenmaal heel de populatie overlopen is, worden de beste chromosomen (eerst worden de scores vergelijken en als ze gelijk zijn wordt er dan gekeken naar het aantal zetten) doorgegeven aan een nieuwe array “Breeders” en de populatie wordt dan leeg gemaakt. Twee willekeurige chromosomen worden uit de Breeders gehaald om een nieuw chromosoom te maken door cross-over en dat nieuwe chromosoom zal potentieel mutatie ondergaan. Het nieuwe chromosoom zal een oude vervangen en dit wordt herhaald tot alle de volledige vorige populatie vervangen is. Dit proces wordt herhaald tot het aantal generaties (die de n-de populatie voorstellen) gelijk staat met de “Maxgeneration”.

Extra info over het project

Struikelblokken

Hoewel het project relatief vlot tot stand kwam, verliep het coderen van een project zoals dit vanaf nul natuurlijk niet zonder problemen. Bijvoorbeeld stopte zoals eerder vermeld de gsmsupport plots met werken. Tijdens het maken van het leaderboard waren er wat problemen met toestemming om de data te halen van de verschillende websites die het team geprobeerd heeft. De AI werd op een andere tak van de git gemaakt dan de rest van de websites. Om deze twee takken samen te voegen werd er wat gesleuteld. Daarnaast paste het team i18n toe, dit verliep niet overal even vlot wanneer er variabelen in de tekst geïntegreerd moesten worden. Naast het oplossen van bugs kropen ook meerdere uren in het in orde maken van de asynchrone functies, maar uiteindelijk kwam de website in zijn geheel tot zijn recht.

Leaderboardpagina

De leaderboardpagina gebruikt een fetch call om data over de top Tetris spelers op tetr.io op te halen van hun API. Hier kon het team de opgedane kennis uit het vak Gebruikersinterfaces toepassen op het project. De data van de API werd omgezet in een JSON-bestand en zo kon ze getransformeerd worden naar een leesbare tabel via JavaScript.

Bronvermelding

Bullet, C. (2020, september 13). *Using A.I. to DOMINATE NERDS in TETRIS*. Opgehaald van YouTube: https://youtu.be/os4DcbpL0Nc

Mohammad. (2022, mei 16). *The Ultimate Guide to JavaScript Localization*. Opgehaald van Phrase: https://phrase.com/blog/posts/step-step-guide-javascript-localization/#Loading_translations_asynchronously

mzmousa. (2017, oktober 19). *Evolving Tetris AI based on genetic algorithms*. Opgehaald van GitHub: https://github.com/mzmousa/tetris-ai

Osk. (sd). *ch.tetr.io API*. Opgehaald van Tetr.io: tetr.io

The Applications of Genetic Algorithms in Medicine. (2015, november 30). Opgehaald van PubMed Central: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4678452/

yukikwi. (sd). Opgehaald van codesandbox: https://codesandbox.io/s/admiring-almeida-xrmv5d?file=/src/App.js