VRIJE
UNIVERSITEIT
BRUSSEL

Conway's Game of Life

computersystemen project januari 2017

Asma Oualmakran
0507834

8 januari 2017

1 Conway's Game of Life

Het spel bestaat uit een raster, hierin bevinden zich vierkantjes die cellen moeten voorstellen. Elke cel heeft een van volgende twee statussen: levend of dood. De levende cellen worden ook wel de populatie van die generatie genoemd.

Via een algoritme zal het speelveld zicht steeds updaten. Zo zal:

1. Een **dode** cel met drie *levende* buren, tot leven komen.
2. Een **levende** cel met twee of drie *levende* buren zal levend blijven.
3. Een **levende** cel met vier of meer *levende* buren zal sterven.

Elke update van het speelveld, gemaakt volgens deze regels, wordt gezien als een generatie.

Het doel van Conway's game of life is om een start veld te maken dat een zo groot mogelijk aantal generaties overleefd. Het is niet onmogelijk dat uw populatie nooit sterft.

2 Functionaliteiten

2.1 Behaalde Doelstellingen

- **Berekenen van volgende generatie**

  Deze berekening wordt momenteel foutief uitgevoerd waardoor de volgende generatie niet de verwachte waarde heeft. De fout hiervoor ligt bij het tellen van de buren zoals beschreven in sectie 3.3 of bij het gebruik van meerdere array's, sectie 3.5, waar de resultaat array niet juist wordt geïnnitialiseerd alvorens aan de berekening te beginnen.

- **Tekenen van de populatie**

- **Tellen van de generaties**

  Hoewel we het aantal generaties tellen, worden deze momenteel nog niet visueel weergegeven in het scherm.

2.2 Niet Behaalde Doelstellingen

Door tijdsgebrek en het wegvallen van mijn teamgenoot, besproken in sectie 3.4, zijn niet alle, door ons zelf opgelegde, doelstellingen behaald.

- **Opstellen van een start populatie door middel van de muis**

  Aangezien een start populatie noodzakelijk is voor het berekenen van de daaropvolgende generaties, kan de start populatie aangemaakt worden door in de code de array *_start* aan te passen met de indexen van de cellen die tot de start populatie behoren. Deze cellen worden dan door de procedure *SetStartState* in *_gridArray* levend gemaakt.

- **Opslaan en inladen van de populatie**

  Deze is niet behaald vanwege de verdeling van de taken. Deze verdeling wordt verder besproken in sectie 3.4.

- **Het pauzeren en herstarten van een spel**

  De gameloop, waarin herhaaldelijk de volgende generatie wordt berekend en getekend, stopt na elke iteratie. Wanneer de gebruiker een toets heeft ingeduwd zal hij de volgende generatie bereken. Omdat er momenteel met het principe wordt gewerkt waarin de gebruiker bevestiging moet geven voor we verder gaan, was er geen nood aan het implementeren van een pauze systeem. Momenteel wordt er van een Ëen stap per keer p̈rincipe gebruikt gemaakt om het testen van het programma te vergemakkelijken.

3 Problemen

3.1 Tekenen van meerdere cellen

Het was enkel mogelijk om een afzonderlijke cel te tekenen ongeacht de staat ervan. Dit probleem kwam voor in de procedure *DrawGrid*, die meerdere cellen moet tekenen om het volledige speelveld grafisch weer te geven. Deze procedure maakt gebruik van de procedure *DrawSquare* waar het probleem schuilt. Deze procedure kon niet meerdere keren achter elkaar worden opgeroepen. Dit probleem is opgelost door manueel alle registers, die werden gepusht alvorens de procedure op te roepen, achteraf weer te poppen in plaats van de esp waarde, na de oproep, aan te passen.

3.2 Het tekenen van _gridArray2

De adressen van de array's worden bijgehouden in macro's, en hier lag ook het probleem. De macro's werden gebruikt om de adressen van *_gridArray* en *_gridArray2* bij te houden, zijn vervangen door hun eigenlijke adres hard-coded mee in de code te steken. Het nadeel hiervan is dat de code dan minder modulair wordt.

3.3 Het tellen van de buren

Het aantal levende buren van een cell word op een foutieve manier berekend. Hierdoor is het resultaat voor de volgende generatie niet correct. Voor dit probleem is er echter nog geen oplossing gevonden. Het vermoeden is dat het tellen van de cellen wel correct gebeurd maar dat de resultaat array, zoals beschreven in sectie 3.5, niet correct wordt leeggemaakt waardoor het eindresultaat foutief is.

3.4 Groepswerk

Het groepswerk is slecht verlopen, hierbij moest ik lang wachten op mijn teamgenoot die steeds onze afspraken om samen te zitten en het project te bespreken uitstelde. Uiteindelijk heb ik besloten om alvast alleen aan het project te beginnen. Echter was er toen reeds veel tijd verloren door het wachten op mijn team genoot.
Aangezien hij wel altijd van plan was mee te werken aan het project, tot op het laatste moment, hebben we de taken via een facebook conversatie verdeeld nadat ik hem had gezegd niet langer op hem te wachten en dat ik reeds was begonnen. Een week voor de deadline heeft te kennen gegeven niet langer te willen deelnemen aan het project.

3.5 Meerdere Arrays

In dit ontwerp wordt er gebruik gemaakt van meerdere array's. Twee array's om de volgende populatie te berekenen. Een array bevat de huidige populatie die noodzakelijk is voor het berekenen van de volgende generatie. De andere array is leeg en zal de resulterende populatie bevatten. Om te voorkomen dat de huidige populatie verloren gaat, wordt na de berekening de resultaat array gewisseld met de populatie array. De populatie array wordt de nieuwe lege

resultaat array. Om dit te bereiken, wordt de generatie teller gebruikt, bij een even generatie wordt *_gridArray* gebruikt en bij een oneven *_gridArray2*. Bij iedere iteratie wordt de laatst verwerkte array weggeschreven naar *_bufferArray* die op zijn beurt naar de videobuffer wordt weggeschreven. Door deze manier van werken te kiezen, is het enkel mogelijk om met byte array's te werken. Dit komt doordat de videobuffer een byte array is en afhankelijk is van informatie uit de andere array.
Dit zorgt voor geen problemen aangezien de inhoud van de array's enkel kleuren zijn en deze hebben een waarde tussen 0 en 15.