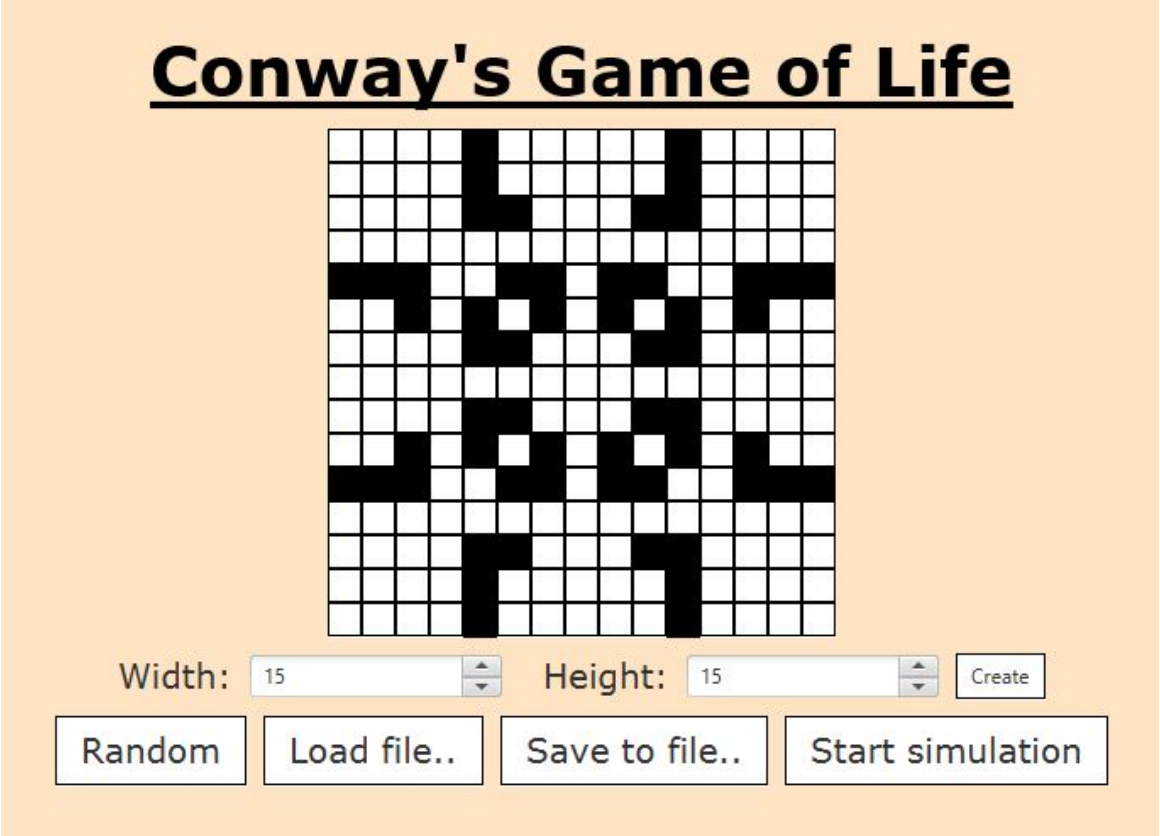
FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

Project Object Georiënteerd Programmeren
Conway's Game of Life

9 november 2020



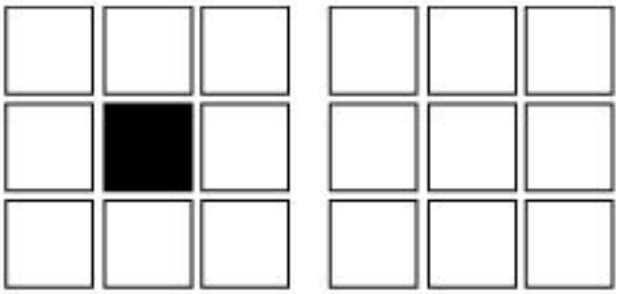
1 Inleiding

Game of life, een cellulaire automaat, is bedacht door een Britse wiskunde John Conway en is ook bekend als Conway's game of life. Het 'spel' bestaat in zijn origineel ontwerp uit een tweedimensionaal raster waarbij elk vierkante **cel** *levend of dood* kan zijn. Op basis van enkele regels kan een cel leven, dood gaan of zich verdubbelen. Na een initiële configuratie van het raster start de simulatie die oneindig lang kan doorgaan. Bepaalde combinaties van cellen resulteren in stabiele, oscilerende of bewegende patronen. In volgende video vertelt John Conway hemzelf over het ontstaan van het 'spel'.

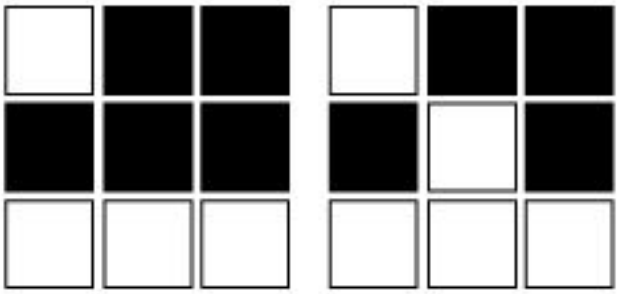
2 Regels

Elke cel heeft 8 buren in de tweedimensionale ruimte. Afhankelijk van onderstaande regels is een cel dood of levend in de volgende generatie.

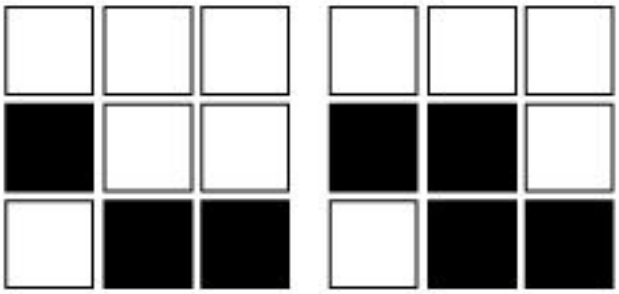
- **Onderpopulatie:** levende cellen sterven als ze minder dan 2 levende buren hebben.
- **Overpopulatie:** levende cellen sterven als ze meer dan 3 levende buren hebben.
- **Reproductie:** dode cellen worden levend als ze exact 3 levende buren hebben.
- **Geen verandering:** in de andere gevallen blijven levende cellen levend en dode cellen dood.



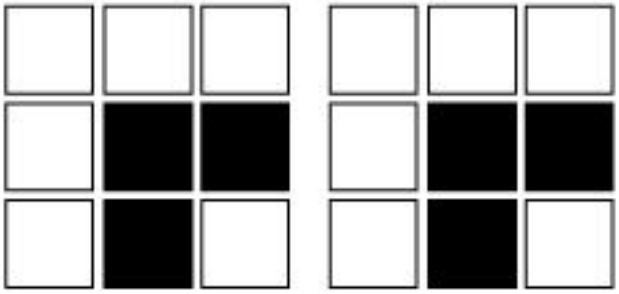
Loneliness
A cell with less than 2 adjoining cells dies.



Overcrowding
A cell with more than 3 adjoining cells dies.



Reproduction
An empty cell with more than 3 adjoining cells comes alive.



Stasis
A cell with exactly 2 adjoining cells remains the same.

3 Opgave

De opgave voor dit project bestaat uit het implementeren van de logica die nodig is om de gegeven GUI te laten werken. In principe kan het tweedimensionele raster oneindig groot zijn, maar deze implementatie beperkt zich tot een *(n x m)* matrix.

4 Kennismaking - zoekoefening

Voorlopig krijg je alleen de code van de GUI. Deze is grotendeels onverstaanbaar, en dat is helemaal OK. We vragen nog geen codeerwerk, enkel voorbereidend werk:

1. Leid uit de code af welke constructoren en methodes de klasse `World` nodig heeft; noteer de hoofdingen (inclusief returntype en parameterlijst).
2. Beschrijf wat deze methodes van de klasse `World` volgens jou moeten doen.
3. Denk na over bijkomende klassen. We vragen je om in de uiteindelijke implementatie minstens één extra klasse te gebruiken. Welke zou dat kunnen zijn? Is er meer dan één kandidaat?
4. Stel een werkplan op. Welke methodes zou je eerst schrijven? Welke tussentijdse testen kan je zelf doen, voor je de code samen laat werken met de GUI?
5. Schets eventueel al een paar methodes: bespreek binnen je team welke structuren nodig zijn (if / while / for), hoe informatie bewaard wordt,...
6. Stel dat je de initiële wereld vanuit een bestand wil inlezen, wat moet er dan in het bestand staan? (Concreet: geef de bestandsinhoud voor de wereld uit de eerste afbeelding.)

```
World

Meerdimensionele Map? x co -> Map met y co -> bool alive
     -> maakt zoeken makkelijker?


constructor(int width, int height)
constructor(String bestandsnaam)

void .nextGeneration() //gebruikt toggleCell, isAlive om nieuwe
staat te bepalen.
Alle punten overlopen en checken hoeveel neighbours .isAlive

int height .getHeight() // getter
int width .getWidth() // getter

bool .isAlive(int x, int y) // check bool van co
Bool checken uit Map

void .toggleCell(int x, int y) // bool 0 -> 1
Bool aanpassen in Map

void .randomCells() // random startcellen kiezen
Alle cellen overlopen en random true of false genereren:
Random.nextBoolean()

void .saveToFile(String bestandsnaam) // exporteren naar file met
vaste opmaak die ook door loadFile gelezen wordt. Co die 0 of 1 zijn
opslaan per lijn + width en heigth

// void .loadFile(String bestandsnaam) toevoegen
```

```
15,15
0,0,0,0,1,0,0,0,0,0,1,0,0,0,0
0,0,0,0,1,0,0,0,0,0,1,0,0,0,0
0,0,0,0,1,1,0,0,0,1,1,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1,1,1,0,0,1,1,0,1,1,0,0,1,1,1
...
```

0) Extra klone

1) constructors → ((file))
→ (1)(int lengte, int breedte)

2) getters

3) isAlive , Togglecell , Randomcell

4) Nextgeneration

5) save & load file