

1 Mandelbrotfiguren

In deze opdracht willen we je laten oefenen met de grafische bibliotheek van java: *Swing*. Dit doe je aan de hand van een klassieker op het gebied van grafische applicaties: het tekenen van *Mandelbrotfiguren*. Ook voor deze opdracht zijn de gebruikelijke ontwerpscriteriën voor OO-applicaties van toepassing, d.w.z. je programma dient modulair te zijn opgezet waarbij de afhankelijkheden tussen modules zo gering mogelijk moeten zijn. Maak voor het minimaliseren van afhankelijkheden gebruik van het MVC-principe.

2 Leerdoelen

Na afloop van deze opdracht ben je in staat om:

- een model voor een gegeven toepassing te maken;
- verschillende GUI-componenten te implementeren;
- het MVC-principe toe te passen voor een GUI-applicatie.

3 Probleemschets: Mandelgetallen

Voor elk punt (a, b) van het platte vlak, waarbij a en b reële getallen zijn, kan een bijbehorend getal worden bepaald – laten we dit het *mandelgetal* noemen. Om het mandelgetal te kunnen uitrekenen, bekijken we eerst de volgende reeks van punten, startende bij (a, b) :

$$\begin{aligned}(x_0, y_0) &= (a, b) \\ (x_{n+1}, y_{n+1}) &= (x_n^2 - y_n^2 + a, 2 \times x_n \times y_n + b)\end{aligned}$$

Het idee is om te stoppen met de reeks zodra het laatste punt een afstand van meer dan 2 tot $(0, 0)$ heeft. Het mandelgetal is nu gelijk aan de waarde van n in deze reeks. Voor sommige punten (a, b) geldt meteen al dat hun afstand tot de oorsprong groter dan 2 is. Van deze punten is het mandelgetal dus gelijk aan 0. Voor andere punten moet je enkele waarden uit de reeks uitrekenen om dit te bereiken: die hebben een groter mandelgetal. Er zijn ook punten waarbij je deze situatie nooit krijgt, oftewel, $\forall n : \sqrt{x_n^2 + y_n^2} < 2$. Deze punten hebben mandelgetal *oneindig*. Het is eenvoudig in te zien dat het punt $(0, 0)$ zo'n oneindig mandelgetal heeft, maar er zijn er veel meer.

In de praktijk kunnen we met het uitrekenen van de reeks voor een bepaald beginpunt niet oneindig lang doorgaan; daarom stoppen we bij een bepaald maximum aantal keren, bijvoorbeeld 100 keer. Is dan de afstand nog steeds niet groter dan 2, dan nemen we maar aan dat het mandelgetal oneindig is.

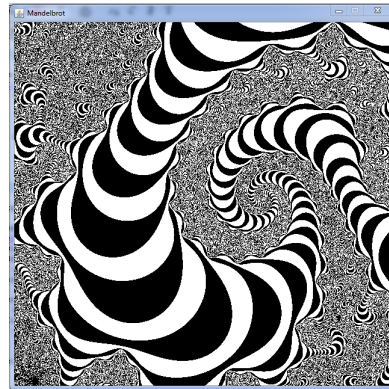
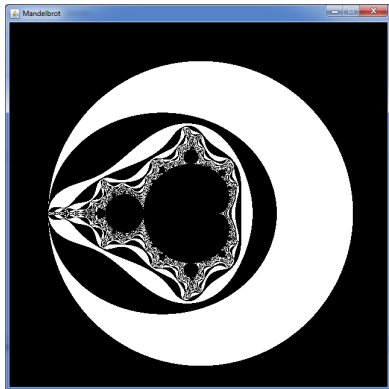
Een voorbeeld

Als voorbeeld berekenen we het mandelgetal van het punt $(0.5, 0.8)$. We beginnen dus met $(x_0, y_0) = (0.5, 0.8)$; de teller staat op 0. De afstand van $(0.5, 0.8)$ tot $(0, 0)$ is volgens Pythagoras wortel $\sqrt{0.25 + 0.64} = \sqrt{0.89}$ wat ongeveer 0.94 is, en dat is niet groter dan 2. We mogen dus doorgaan. We berekenen met de formule $x_1 = 0.5^2 - 0.8^2 + 0.5 = 0.11$ en $y_1 = 2 * 0.5 * 0.8 + 0.8 = 1.6$. De afstand van $(0.11, 1.6)$ tot $(0, 0)$ is volgens Pythagoras ongeveer 1.6038, dus we mogen nog steeds doorgaan. We berekenen met de formule weer een volgende element uit de reeks, met

$x_2 = 0.11^2 - 1.6^2 + 0.5 = -2.0479$ en een $y_2 = 2 * 0.11 * 1.6 + 0.8 = 1.152$. De afstand van $(-2.0479, 1.152)$ tot de oorsprong is royaal meer dan 2, dus de teller blijft staan op 2. Het mandelgetal van $(0.5, 0.8)$ is dus 2.

De Mandelbrot-figuur

Aan de hand van mandelgetallen kunnen we een kleur aan elk punt toekennen. Een mogelijkheid is deze: punten met een oneven mandelgetal worden wit, punten met een even mandelgetal worden zwart. Ook punten met mandelgetal oneiding worden zwart. Geef je elk punt van het scherm de aldus bepaalde kleur, dan zie je een afbeelding van de Mandelbrot-figuur. Het interessante gedeelte van de figuur zit rond de oorsprong. Voor x en y tussen -2.5 en 2.5 is het resultaat in de linkerfiguur hieronder weergegeven.



De figuur is opmerkelijk grillig, zeker als je bedenkt hoe relatief eenvoudig het algoritme eigenlijk is. Het wordt nog duidelijker als je verder inzoomt op een onderdeel van de figuur, zoals in het rechterplaatje is gebeurd.

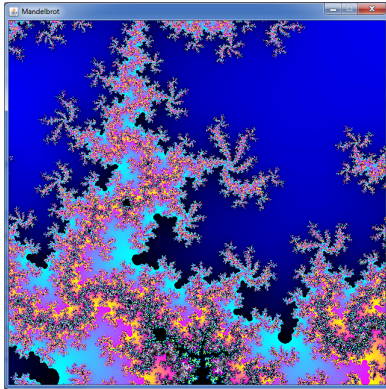
De userinterface

We gaan een programma maken waarmee de gebruiker kan inzoomen op de Mandelbrotfiguur. Dit kan op verschillende manieren welke je *allemaal* dient te realiseren.

- De gebruiker kan het middelpunt van het weer te geven vlak invoeren tezamen met de *schaalfactor*. Deze laatste bepaalt de eenheid langs beide assen, oftewel de daadwerkelijke grootte van ieder beeldpunt. Als je bijvoorbeeld een venster hebt van 500 bij 500 beeldpunten en een schaalfactor van 100 dan is de concrete breedte van het scherm 5 bij 5. Geef je dan als centrum $(0, 0)$ op dan wordt het gebied voor x en y tussen -2.5 en 2.5 weergegeven. Met een schaalfactor van 10000 en $(1, -0, 5)$ als centrum ligt x tussen 0.975 en 1.025 en en y tussen -0.5025 en $-0, 4975$. Gebruik in je implementatie `JTextFields` om deze gebruikersinformatie in te kunnen lezen. Voeg bovendien een extra `JTextField` waarmee het maximum aantal herhalingen kan worden aangegeven alsook een `JButton` die de gebruiker kan aanklikken zodat het aangegeven deel van de Mandelbrotfiguur worden getoond.
- Ook kan de gebruiker met de muis op de figuur klikken. We onderscheiden nu twee situaties.
 1. De gebruiker laat de muisknop direct weer los. In dat geval wordt het aangeklikte punt het nieuwe middelpunt van de figuur, en de schaalfactor wordt verdubbeld; de gebruiker zoomt daarmee in op het aangeklikte punt. De nieuwe waarden van midden en schaal worden ook in de tekstvelden weergegeven. Als de gebruiker de shift-toets tijdens het klikken ingedrukt houdt, dan dient de schaalfactor te worden gehalveerd i.p.v verdubbeld. Het effect is dat er wordt uitgezoomd.

2. De gebruiker selecteert een bepaald gebied door de muis naar een andere positie te slepen. Er wordt nu ingezoomd op het geselecteerde gebied. Zorg ervoor dat gedurende het slepen duidelijk is welk gebied op dat moment geselecteerd is (Hint: geef dit gebied aan door een rechthoek in *XOR-mode* te tekenen). Pas ook de tekstvelden weer aan. Het vergt enig puzzelwerk om te bepalen wat je nieuwe tekengebied wordt bij zo'n selectie. Werk dit eerst op papier uit, omdat het erg lastig kan zijn dit achter het scherm te verzinnen.

De kleuren



De hierboven gegeven kleuring is wel heel erg eenvoudig. Daar kun je het programma in eerste instantie mee testen. Maar in het definitieve programma moeten de punten gekleurd worden, waarbij de rood-, groen- en blauwcomponenten alledrie van het mandelgetal afhangen. Hier is alvast een voorbeeld. We gaan niet vertellen hoe we dit plaatje precies hebben gekregen; dat mag je zelf uitzoeken. Er zijn nog vele andere mogelijkheden om dat te doen: Wees creatief!

4 Hulpbestanden

Voor deze opdracht staan op Blackboard enkele bestanden die je mag gebruiken als inspiratiebron voor je eigen programma. Hiervan zijn mijn name het interface `Grid` en de klasse `GridView` die dit interface implementeert, interessant. De definitie van `Grid` luidt:

```
public interface Grid {  
    int getWidth ();  
    int getHeight ();  
  
    void setPixel (int x, int y, int [] rgb);  
}
```

Dit interface is een abstractie van het (concrete) tekenvlak. Met `setPixel` kunnen individuele pixels (tussen $(0,0)$ en $(W-1, H-1)$) gekleurd worden. De waarden W en H worden door `getWidth` en `getHeight` opgeleverd. De gewenste kleur wordt als een rijtje van 3 integers gerepresenteerd die overeenkomen met de rood-, groen- en blauwcomponent van deze kleur. Van `GridView` zelf hoef je de precieze werking niet te begrijpen. Deze klasse is enkel bedoeld om je een boel zoekwerk te besparen. Het is niet de bedoeling dat je `Grid` of `GridView` aanpast. De overige klassen illustreren het gebruik van `Grid` en kunnen naar believen worden aangepast.

5 Producten

Als producten moet je een JAVA applicatie, zoals gebruikelijk bestaande uit een verzameling klasdefinities, inleveren, die voldoen aan de kwaliteitscriteria zoals die gesteld zijn voor deze cursus.

6 Inleveren van je producten

Vóór zondag 3 mei, 11:00 uur, via Blackboard.