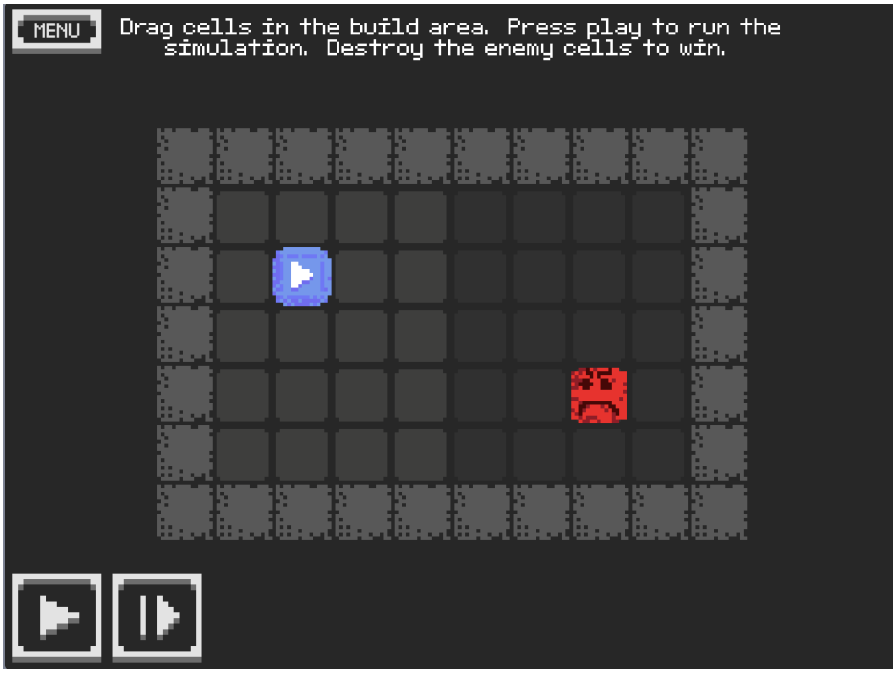
Taak Objectgericht programmeren Project
Deadline: 15 augustus 2022, 23u59

Wolfgang De Meuter
Bram Vandenbogaerde

Academiejaar 2021-2022

1 Inleiding



Figuur 1: Een voorbeeld van het spel "Cell Machine", het doel is om alle rode (vijandige) cellen te vernietigen

Dit document beschrijft de taak voor het vak „Objectgericht programmeren". Lees dit document **grondig** van het begin tot het einde, zodat het duidelijk is wat er van jou wordt verwacht.

Het doel van deze opdracht is om een minimale versie van het spel *Cell Machine*[1] te maken. *Cell Machine* is geinspireerd door *Conway's Game of Life* waar de speler een initiele configuratie

[1] `https://samhogan.itch.io/cell-machine`

maakt en daarna de logica van het spel verder de uitkomst laat bepalen. Dit resulteert in een zogenaamde *zero-player game*[2].

Figuur 1 geeft een voorbeeld van hoe het spel er uit kan zien. Het spelbord bestaat uit een aantal lege cellen en speciale cellen. De doelstelling is om de speciale cellen zo te plaatsen dat de vijandige (rode cellen) vernietigd worden wanneer het spel start. Het spel bestaat dan uit verschillende levels die verschillen in layout, het aantal vijanden, het aantal beschikbare cellen...

2 Functionele vereisten

Algemene spelstructuur:

- **Algemeen spelverloop:** Het spel bestaat uit een aantal levels die de speler moet voltooien. Elk level begint met een aantal verplaatsbare blokken (zie hieronder) die door de speler op de juiste plaats kunnen worden gezet. Er zijn ook een aantal vijanden in het spel die moeten worden vernietigd. Van zodra een blok een vijand raakt wordt zowel dat blok als de vijand vernietigd. Een level kan eindigen op twee manieren: (a) voltooid, alle vijanden zijn vernietigd (b) verloren, de blokken maken geen beweging meer, maar nog niet alle vijanden zijn vernietigd.

  Eens de configuratie van de speler klaar is, wordt het level gestart door op de afspeelknop te drukken, daarna verloopt het spel in een aantal iteraties (ook wel generaties genoemd) volgens de regels van de geplaatste blokken (zie hieronder).

- **Grid:** Het spel bestaat uit een grid van cellen. Het grid is verantwoordelijk om de code aan te roepen die zorgt voor de beweging van een cel. Het moet ook ondersteuning bieden om cellen toe te voegen en te verwijderen, zodat indien nodig cellen zichzelf kunnen verwijderen of nieuwe cellen op bepaalde plaatsen kunnen maken.

  Zorg er ook voor dat jouw implementatie herbruikt kan worden voor andere *cell-based* spelletjes (bijvoorbeeld mijnveger). Het gebruik van een type parameter is hier dus aangewezen.

- **Grafische interface** Alle spelelementen hierboven en onder beschreven dienen visueel weergegeven te kunnen worden. Je kan inspiratie opdoen van de speelweergave in figuur 1. De grafische interface dient geimplementeerd te worden in **Java Swing**.

Speciale cellen:

- **Duw cel:** Een duw cel is een speciaal type cel dat zelf beweegt in een bepaalde richting en duwbare cellen in dezelfde richting duwt als het aan het bewegen is. Zorg ervoor dat het gemakkelijk is om verschillende variaties te maken van een dergelijke cel. Maak dan een duw cel dat na vijf iteraties stopt met bewegen, om te tonen dat deze cellen gemakkelijk uitbreidbaar zijn.

- **Duwbare cellen:** Een duwbare cel is een cel dat geduwd kan worden door een duw cel of andere duwbare cellen. Alle cellen zijn duwbare cellen, behalve de onbeweegbare cellen. Zorg ervoor dat de logica om een cel te laten bewegen herbruikbaar is en dat het gemakkelijk is om het gedrag van cellen aan te passen.

  Er zijn ook speciale cellen die slechts in een bepaalde richting kunnen bewegen, maak gebruik van de gepaste klassen.

[2] `https://en.wikipedia.org/wiki/Zero-player_game`

- **Draaicellen:** Een draaicel laat toe om de richting van bewegende cellen te veranderen. Deze cel kan zelf ook verplaatst worden door een duw cel, maar beweegt normaal niet.
- **Generator cellen:** Een generator cel dupliceert de cel die zich voor zichzelf bevindt.
- **Onbeweegbare cellen:** Een onbeweegbaare cel is een cel die niet kan bewegen, maar wel kan worden gedupliceerd met een generator cel.
- **Ondraaibare cel:** Een ondraaibare cel verschilt van de onbeweegbare cel in het feit dat het wel kan worden voortgeduwd maar stopt wanneer het bij een draaicel komt.

3 Niet-Functionele Vereisten

Het project heeft de volgende niet-functionele vereisten:

- **Code structuur:** We verwachten dat je code goed gestructureerd is en verspreid is over meerdere bestanden. Het is **niet** toegestaan om het hele project in een *Scala Worksheet* te programmeren. Maak hiervoor gebruik van een *sbt* project zoals gezien in de WPOs.
- **Code ontwerp:** Een belangrijk onderdeel van de taak is dat je een goed ontwerp maakt voor je objectgericht systeem. Maak dus gebruik van de concepten die je hebt geleerd tijdens de hoorcollege's en WPOs (i.e., traits, overerving, abstracte klassen, ...).

  Zorg er voor dat er een duidelijke scheiding is tussen de code die de grafische interface verzorgt, en de eigenlijke spellogica.
- **Code kwaliteit:** De kwaliteit van je code in het algemeen zal ik rekening gebracht worden. Maak voldoende gebruik van abstractie, en vermijd code duplicatie, magische constanten, ... Zorg voor voldoende documentatie in je code zelf, zodanig dat het duidelijk is hoe je code werkt en hoe deze gelezen moet worden.
- **Tests:** Zorg ervoor dat jouw project *unit tests* bevat waar bepaalde onderdelen van je code afzonderlijk worden getest. Maak hiervoor gebruik van *ScalaTest* zoals gezien in de WPOs. De tests moeten duidelijk maken dat jouw implementatie werkt naar behoren. Er zijn geen automatische UI tests vereist.

4 Startproject

Voor deze taak is er een startproject voorzien op *Canvas*. Het is verplicht om jouw implementatie te bouwen op dit startproject, het is niet toegelaten om bestaande code in dit project aan te passen. Het project bevat een aantal hulpklassen in Java, alsook een aantal *sprites* die nuttig kunnen zijn voor de grafische weergave van je spel. **Maak gebruik** van deze hulpklassen om je UI te bouwen.

Buiten de reeds meegeleverde *Java* bestanden, dient het project in **Scala** te worden geschreven.

5 Indienen

Je dient je gehele project in te dienen als een **ZIP-archief** met naam *voornaam-achternaam-rolnummer.zip*, andere bestandsformaten zullen niet worden aanvaard. Er zijn bepaalde bestanden en mappen in je project die je niet dient mee sturen in je zip-archief aangezien deze door Scala zelf worden gegenereerd. Ervanuitgaande dat je je project hebt aangemaakt met Intellij IDEA

en dat de standaard mapstructuur gerespecteerd werd[3] dien je enkel onderstaande bestanden en mappen toe te voegen aan je zip-archief:

- SRC/: dit bevat de broncode van je project.
- BUILD.SBT: dit bevat de project-configuratie (o.a., de naam van je project, welke versie van Scala er gebruikt moet worden, maar ook welke libraries, zoals ScalaTest, nodig zijn voor het project uit te voeren).
- PROJECT/BUILD.PROPERTIES: dit bestand bevat welke versie van SBT gebruikt moet worden om je project te compileren (andere mappen in PROJECT/ moeten niet toegevoegd worden!).
- Indien je bepaalde SBT-plugins gebruikt hebt, voeg je ook de benodigde bestanden van deze plugins toe.
- Als je project ook data-bestanden (zoals tekstbestanden of een database) nodig heeft, voeg je ook deze ook toe aan je zip-archief! Kortweg: elk bestand dat nodig is voor je project te compileren en uit te voeren.

Overige mappen (zoals TARGET/, PROJECT/TARGET/ en NULL/COURSIER[4]) bevatten gecompileerde bestanden of bibliotheken die door SBT gedownload zijn. Deze moeten dus niet toegevoegd worden aan het zip-archief.

Controleer steeds of je zip-archief volledig is door het opnieuw uit te pakken op een andere locatie, en het vervolgens te laten importeren door Intellij IDEA (sleep de map naar de rechterkant van het opstart-scherm, om het project te importeren[5]). Controleer of je project gecompileerd kan worden, en dat er geen tests falen.

Tot slot, willen we jullie er aan herinneren dat jullie de taak samen mogen bespreken maar dat samen aan code werken **niet** is toegestaan. Elke vorm van code-uitwisseling tussen studenten zal als **plagiaat** beschouwd worden; hierbij zullen zowel gebruiker als verlener van de code volgens de desbetreffende regels van het examenreglement gesanctioneerd worden.

De deadline voor deze taak is **15 augustus, 23u59**, late indiening van je project is niet toegestaan en resulteert in een afwezig voor het hele vak! Voor vragen mag je altijd contact opnemen met `bram.vandenbogaerde@vub.be`.

[3]Indien je je project op een andere manier hebt gemaakt, of de standaardpaden werden niet gerespecteerd, dan zijn deze instructies mogelijks niet 100% correct.

[4]Niet al deze mappen zijn altijd aanwezig.

[5]Wanneer je reeds een project geopend hebt, kan je dit scherm terug openen door je huidige project te sluiten (FILE – CLOSE PROJECT).