Project "Objectgericht Programmeren": Deel 1

Prof. Eric Steegmans
Raoul Strackx

Academiejaar 2010-2011

Deze tekst beschrijft het eerste deel van de opgave voor het project van de cursus '*Objectgericht Programmeren*'. Dit project geldt als examen voor de cursus. Het project wordt bij voorkeur uitgewerkt in groepjes van 2 studenten; indien het niet anders kan, mag het project individueel worden uitgewerkt. Indien er in de loop van het project conflicten ontstaan waardoor een verdere samenwerking onmogelijk is, moet elk van de partners het project individueel uitwerken. Na afloop van het eerste deel, mag echter een nieuwe partner gezocht worden. Dergelijke veranderingen moeten onmiddellijk gemeld worden aan *ogp-project@cs.kuleuven.be*.

Het project wordt stapgewijs opgebouwd. Het eerste deel handelt over de creatie van 1 klasse, het tweede over associaties tussen klassen. Het derde en laatste deel zal gaan over overerving en genericiteit. Na het indienen van het derde deel zal de volledige oplossing (deel 1, 2 & 3) worden verdedigd bij Prof. Steegmans.

Een team van assistenten staat in voor de opvolging van de projecten. Concreet krijgt elke student of elk team van studenten een aantal uren waarop hij/zij vragen kan stellen aan een assistent. Deze laatste fungeert min of meer als consultant, die voor een beperkte tijd kan ingehuurd worden. Vragen aan assistenten kunnen handelen over verduidelijkingen van de opgave, onduidelijkheden in de cursus, besprekingen van alternatieve oplossingen, kwaliteit en volledigheid van uitgewerkte oplossingen, ... Het is niet de bedoeling dat de assistenten zelf oplossingen of stukken ervan uitwerken.

Het project handelt over het bouwen van een deel van de kern van een *roleplaying game* (RPG) in een middeleeuwse setting. Het spel is gebaseerd op bestaande spellen.

Sommige aspecten uit de opgaven komen niet overeen met de bestaande spellen. Ze worden ingevoerd om de opgave zo gevarieerd mogelijk te maken. Je oplossing moet de regels volgen zoals ze in deze opgaven beschreven staan.

Alle grafische aspecten en alles in verband met gebruikersinteractie worden buiten beschouwing gelaten. Het project beperkt zich tot het uitwerken van die elementen, die te maken hebben met de ondergrondse doolhoven (*dungeons*) waarin het spel zich afspeelt... In de opgave worden een aantal vereisten gesteld omtrent hoe deze elementen moeten worden ondersteund in het beoogde systeem. In de eerste plaats worden een reeks functionele vereisten gesteld. Dit zijn eerder vage beschrijvingen van functionaliteiten die door het systeem moeten worden ondersteund. We verwachten dat je zelf de nodige basisfunctionaliteiten invoert zoals getters en setters voor karakteristieken. Verder zijn er een aantal niet-functionele vereisten, die zich richten op de kwaliteit van het beoogde systeem. Zo worden in dit deel op diverse punten vereisten gesteld op het vlak van aanpasbaarheid en herbruikbaarheid.

Opgave

Het eerste deel van de opgave behandelt slechts één klasse, namelijk die voor een vakje. Na dit eerste deel volgt er een *verplichte* terugkoppelingssessie bij één van de begeleidende assistenten voor deze cursus.

1 Vakje [*Square*]

In het eerste deel van de opgave wordt de klasse `Square` (vakje) uitgewerkt. Een vakje is voor het spel de kleinste eenheid van locatie, en zal later deel gaan uitmaken van een grotere kerker of doolhof, die een groot aantal met elkaar verbonden vakjes zal bevatten.

Vakjes hebben een aantal basiskenmerken die moeten worden uitgewerkt.

1.1 Temperatuur [*Temperature*]

Elk vakje in de verhaalwereld heeft altijd een welbepaalde temperatuur. Deze is beperkt van -200 tot 5000 graden Celcius voor alle vakjes. Deze grenswaarden kunnen later nog veranderen. Het is zelfs mogelijk dat na verloop van tijd verschillende grenswaarden zullen gelden voor verschillende (soorten) vakjes. Ook moet de temperatuur kunnen worden uitgedrukt in andere schalen zoals Kelvin en Farenheit.

De temperatuur van een vakje kan veranderen doorheen de tijd. Alle methodes in verband met de temperatuur van een vakje moeten *defensief* worden uitgewerkt.

1.2 Schade door kou [*Cold Damage*]

Een extreem lage temperatuur zorgt ervoor dat wezens die zich in een vakje bevinden schade ondervinden van de kou. Deze schade moet per vakje afzonderlijk berekend worden en bedraagt 1 punt per 10 graden onder -5 graden Celcius, afgerond naar beneden. De getallen in deze berekening zullen nooit meer veranderen.

(Deze schade is een schade per tijdseenheid, dit geldt trouwens voor de meeste vormen van schade in een vakje. Met het aspect tijd houden we echter in de opgave geen rekening.)

1.3 Schade door hitte [*Heat Damage*]

Een extreem hoge temperatuur zorgt ervoor dat wezens die zich in een vakje bevinden schade ondervinden van de hitte. Deze schade moet per vakje afzonderlijk berekend worden en bedraagt 1 punt per 15 graden boven 35 graden Celcius, afgerond naar beneden. De getallen in deze berekening kunnen wel veranderen in de loop van de tijd. Ze zullen echter altijd gelijk blijven voor alle vakjes.

1.4 Vochtigheidsgraad [*Humidity*]

Vakjes hebben een bepaalde vochtigheidsgraad, uitgedrukt in een percentage. Er bestaan uiteraard geen negatieve percentages, of percentages boven 100%. Het percentage wordt geregistreerd tot op 2 cijfers na de komma. De vochtigheidsgraad van een vakje kan veranderen doorheen de tijd. Alle methodes in verband met de vochtigheidsgraad van een vakje moeten *nominaal* worden uitgewerkt.

1.5 Roesten [*Rust Damage*]

Voorwerpen gemaakt van metaal die in een vakje dat een hoge vochtigheidsgraad heeft worden gebracht, zullen langzamerhand beschadigd raken door roesten. De hoeveelheid schade wordt voor vakjes afzonderlijk berekend en bedraagt 1 punt per 7 percent vochtigheid boven 30 percent, afgerond naar beneden.

1.6 Glad [*Slippery*]

Vakjes kunnen een gladde vloer hebben, waardoor het gevaarlijk wordt om erdoor te wandelen. Men moet aan een vakje kunnen vragen of het al dan

niet glad is. Een vakje kan op drie manieren glad zijn. De eerste is wanneer de vochtigheidsgraad 100 percent is, bij temperaturen boven nul. In dat geval zet er zich water af en ontstaat er watergladheid. Een tweede is wanneer bij niet-positieve temperaturen, de vochtigheid hoger dan 10 percent is. In dat geval is er genoeg vocht aanwezig om een laagje ijs te vormen. Tenslotte kunnen vakjes glad zijn, simpelweg doordat de vloer van een glad materiaal is gemaakt, hetgeen per vakje kan worden ingesteld.

Dit is uiteraard geen juiste beschrijving van gladheid, zoals die in de echte wereld kan ontstaan. Het spel gaat echter over een fictieve wereld, waarin sommige zaken sterk vereenvoudigd worden.

Het glad zijn van vakjes moet *defensief* worden uitgewerkt.

1.7 Bewoonbaarheid [*Inhabitability*]

Hitte, koude en vochtigheid zorgen ervoor dat sommige vakjes meer schade toebrengen dan andere. Voordat een speler zijn avatar op een vakje plaatst wil hij hier graag een idee van hebben. De graad van bewoonbaarheid is hier een maatstaf voor en wordt als volgt berekend:

$$-\frac{\sqrt{\text{heat damage}^3}}{\sqrt{101 - \text{humidity}}} - \sqrt{\text{cold damage}}$$

1.8 Grenzen [*Borders*]

Elk vakje kan in 6 richtingen begrensd worden. Voorlopig nummeren we deze richtingen gewoon van 1 tot 6. Begrenzingen moeten instelbaar zijn. Een vakje kan na creatie een nieuwe grens krijgen, of een grens kan worden gewist. Voorzie ook functionaliteit om te testen of een vakje volgens een gegeven richting begrensd is. Het maximale aantal begrenzingen van een vakje zal altijd 6 blijven. Alle methodes in verband met de begrenzingen van een vakje dienen *totaal* te worden uitgewerkt.

1.9 Vakjes Samenvoegen [*mergeWith*]

Er moet functionaliteit worden voorzien om een vakje samen te voegen met een ander vakje in alle mogelijke richtingen (grens 0, 1, 2, ...). Wanneer twee vakjes samengenomen worden volgens een welbepaalde richting, verdwijnen eventuele begrenzingen voor beide vakjes in die richting. Als je dus vakje 1 en vakje 2 samenvoegt volgens richting 2, zullen noch vakje 1 noch vakje 2 na de samenvoeging nog een grens hebben volgens richting 2. Verder krijgen beide vakjes dezelfde vochtigheidsgraad en temperatuur. De nieuwe

vochtigheidsgraad van beide vakjes wordt het gemiddelde van de vochtigheidsgraden die de vakjes voordien hadden.

De nieuwe temperatuur wordt op een iets ingewikkelder manier bepaald, rekening houdend met de hogere warmtecapaciteit van water. Van beide temperaturen wordt een *gewogen gemiddelde* genomen om de uiteindelijke temperatuur te bepalen. De gewichten zijn de som van een getal evenredig met de relatieve vochtigheidsgraad van het vakje (t.o.v. de gemiddelde vochtigheidsgraad) en een bepaalde constante. Deze constante zorgt ervoor dat volledig droge vakjes toch nog invloed hebben, en is voor beide gewichten en voor alle vakjes in de spelwereld gelijk, instelbaar en beperkt van 0,1 tot en met 0,4. De som van de gewichten moet gelijk zijn aan 2 (het aantal vakjes dat wordt verbonden).

Een voorbeeld. Stel dat de constante 0,2 bedraagt. Gezien de som van de gewichten gelijk moet zijn aan twee, en de constante in beide gewichten als term vervat zit, moet de som van de getallen die evenredig met de gemiddelde vochtigheidsgraad zullen zijn nog 1,6 zijn (2-2*0,2). Indien de vochtigheidsgraden van de vakjes 20% en 60% zijn (gemiddeld 40), moet de berekening van deze termen dus resulteren in 0,4 en 1,2 (gezien 0,4 = a*0,2/0,4 en 1,2 = a*0,6/0,4 met a = 0,8). De uiteindelijke gewichten zijn dan 0,6 en 1,4.

De methode om de nieuwe temperatuur te berekenen is systeemafhankelijk, wat betekent dat de berekening zou kunnen veranderen indien we de methode op een ander platform implementeren. De nieuwe temperatuur zal echter steeds tussen de temperaturen van de samengevoegde vakjes liggen.

Vakjes dienen geen referentie te hebben naar vakjes waarmee ze zijn verbonden. Het samenvoegen van vakjes wordt *defensief* uitgewerkt.

1.10 Constructoren

Er moeten constructoren voorzien worden die toelaten om een vakje aan te maken met een bepaalde temperatuur en vochtigheidsgraad, en om een standaardvakje te maken zonder argumenten.

2 Hoofdprogramma

De werking van de ontwikkelde klasse moet je kunnen demonstreren. Schrijf hiervoor een hoofdprogramma dat een aantal vakjes aanmaakt. Hierbij moeten zeker volgende vakjes worden gecreëerd:

1. Vakje 1:
    (a) Temperatuur: 150 graden Celcius

(b) Vochtigheidsgraad: 59.01%

(c) Glad

(d) Grenzen: 1, 2, 3

2. Vakje 2:

(a) Temperatuur: 40 graden Celcius

(b) Vochtigheidsgraad: 89.90%

(c) Niet glad

(d) Grenzen: 2, 3, 4

Voeg na de creatie vakje 1 samen met vakje 2 langs grens 2. Print hierbij zowel de staat van vakje 1 en 2 voor en na het samenvoegen naar de standaarduitvoer.

3 Testen

Voor de klasse `Square` moet(en) (een) JUnit 4 testklasse(n) worden gemaakt die de nodige testen bevatten. Uiteraard is het de bedoeling dat je eigen testen foutloos kunnen worden uitgevoerd.

Praktische Richtlijnen

We verwachten dat het eerste deel van het project volledig kan afgewerkt worden in 20 uur. Hierbij veronderstellen we dat de materie die aan bod komt in de cursus, vooraf volledig verwerkt is. Daarenboven kan het project enkel binnen die tijdsspanne worden uitgewerkt, indien je op een beredeneerde en gedisciplineerde manier te werk gaat. We raden sterk aan om de principes van *Extreme Programming* zoveel mogelijk toe te passen bij de uitwerking van het project.

1 Verdere richtlijnen rond de oplossing

- Enkel de expliciet opgelegde vereisten moeten voorzien worden. Je zal geen indruk maken door bijkomende spelelementen te voorzien in je oplossing. Integendeel, Extreme Programming schrijft voor alleen die dingen uit te werken die momenteel gevraagd worden.

- Hoewel de opgave vrij uitgebreid is, hebben we niet gepoogd alle mogelijke details in te vullen. Alles wat niet uitdrukkelijk vermeld is in de opgave mag je dan ook naar eigen inspiratie invullen. Kies daarbij steeds voor de gemakkelijkste oplossing.
- Dit project moet uitgewerkt worden in Java 5 of hoger.
- Tenzij uitdrukkelijk anders vermeld wordt in de opgave, moeten alle aspecten van de ontwikkelde klassen zowel formeel als informeel worden gespecificeerd.
- Als opgelegd wordt dat methodes rond een bepaalde karakteristiek nominaal, totaal of defensief moeten uitgewerkt worden, geldt dit ook voor het manipuleren van die karakteristieken in meer complexe methodes (vb. constructoren). Als er niets wordt opgelegd, mag er vrij gekozen worden.
- Methodes die schade-waarden teruggeven, dienen als return-type `int` te gebruiken.
- Er moeten geen definities voorzien worden van methodes die geërfd worden uit Object (zoals equals, clone, toString, ...).
- Indien je ergens een uitzondering dient te gooien, en er kan geen zinnig gebruik gemaakt worden van `IllegalArgumentException`, dan dien je zelf een uitzonderingsklasse te definiëren.

2 Inschrijven

Je dient ons mee te delen met wie je het project zal maken, **ook indien je het 1ste deel individueel maakt**. Enkel de groepjes die zich inschrijven krijgen een begeleider. Inschrijven gebeurt door **voor 2 maart om 12u** een mailtje te sturen naar *ogp-project@cs.kuleuven.be*. Er moet ook instaan (voor beide studenten) welke studierichting je volgt. Als je samenwerkt, stuurt één van beide studenten een mail met in cc de andere student.

3 Inleveren

1. Het project moet **door beide studenten** ingediend worden vóór **vrijdag 25 maart om 9u**. Afwijkingen op dit tijdstip zullen slechts in zeer uitzonderlijke gevallen worden toegestaan.

2. Het indienen gebeurt via Toledo. Dien je oplossing als **JAR-bestand** in. Je kan met de Eclipse omgeving eenvoudig het JAR-bestand genereren met de export functie die je o.a. in het file menu vindt. Let erop dat je al je ontwikkelde klassen (ook de testklassen) selecteert (Select the resources to export), en neem zowel de **.java source files** als de gegenereerde .class files op! Vermeld je naam en studierichting in commentaar bij het indienen. Bij het indienen moet je bevestigen tot je oplossing effectief werd ingeleverd!

4 Feedback

Dit eerste deel van het project zal besproken worden met een assistent. Het is de bedoeling dat je zo te weten komt of je al dan niet goed bezig bent. Deze sessies zullen plaatsvinden tussen 28/03 en 08/04. Concrete informatie volgt op Toledo.

In de loop van het project zal er begeleiding door een assistent mogelijk zijn, ook hierover zal meer informatie op toledo te vinden zijn.

Mocht deze opgave op bepaalde punten onduidelijk zijn, dan zal verdere toelichting gegeven worden via Toledo. Om tegenstrijdigheden in antwoorden te vermijden, kunnen vragen **(enkel i.v.m. de opgave)** gestuurd worden naar: *ogp-project@cs.kuleuven.be*

Veel succes!