Project "Objectgericht Programmeren": Deel 3

Prof. Eric Steegmans
Raoul Strackx

Academiejaar 2010-2011

Deze tekst beschrijft het derde deel van de opgave voor het project van de cursus '*Objectgericht Programmeren*'. Het dient te worden uitgewerkt in dezelfde groepjes van twee studenten als deel 2, tenzij dit door bijzondere omstandigheden onmogelijk geworden is. In dat geval moet er individueel worden verdergewerkt en moet dit gemeld worden aan prof. Eric Steegmans.

Indien het project individueel wordt gemaakt, moeten sommige delen van de opgave niet worden uitgewerkt, dit is aangegeven op die plaatsen.

Opgave Deel 3

In het derde deel van de opgave worden nog een aantal zaken omtrent het doolhof gewijzigd en toegevoegd, zoals een aantal verschillende soorten vakjes en kerkers, en het verbinden van kerkers.

1 Vakjes

Naast de 'gewone' vakje zoals beschreven in het 2e deel van de opgave, zijn er speciale soorten vakjes die hieronder beschreven worden. Vakjes kunnen evenwel nooit van soort veranderen.

1.1 Opgevulde vakjes [*Rock*]

Sommige vakjes bestaan volledig uit rots. Er kan zich niets in bevinden en men kan er zich ook niet doorheen bewegen. Deze vakjes zijn volledig omringd door muren zonder deuren en hebben steeds een vochtigheidsgraad van nul en een temperatuur die het gemiddelde is van alle errond liggende vakjes (nul indien er geen naastliggende vakjes zijn).

1.2 Transparante vakjes [*Transparent*]

Sommige vakjes zijn transparant. Een transparant vakje is in geen enkele richting afgebakend door een volle muur. Verder heeft een dergelijk vakje minstens 1 deur en hoogstens 2 deuren. In het laatste geval moeten de 2 deuren zich bevinden in tegenovergestelde richtingen.

1.3 Teleportatievakjes [*Teleportation Squares*]

Teleportatievakjes zijn vakjes die de mogelijkheid verschaffen om naar een ander vakje te bewegen, dat zich niet noodzakelijk vlak naast het huidige vakje bevindt. Dit gebeurt op een magische manier, en het is zelfs mogelijk om naar een vakje geteleporteerd te worden dat niet behoort tot de dezelfde kerker.

Teleportatie is niet noodzakelijk tweerichtingsverkeer. Als je vanuit een vakje geteleporteerd wordt naar een ander vakje, betekent dat niet noodzakelijk dat je vanuit dit laatste vakje geteleporteerd wordt naar het eerste vakje. Teleportatie wordt nominaal uitgewerkt.

Zowel de 'gewone' vakjes uit deel 1 & 2 als transparante vakjes kunnen teleportatievakjes zijn. Opgevulde vakjes zullen avatars nooit kunnen teleporteren.

Enkel voor groepjes van twee. Een teleportatievakje moet de mogelijkheid bieden om naar meerdere vakjes te teleporteren. Bij iedere teleportatie wordt willekeurig één van de betrokken vakjes als bestemming gekozen.

1.4 Extra functionaliteit

Opgelet! In dit deel van de opgave is er een verschil tussen de mogelijkheid van een avatar te bewegen tussen vakjes en de vakjes die samen een ruimte vormen. In het eerste geval zal er ook rekening gehouden moeten worden met teleportatievakjes. Deze vormen immers artificiële nieuwe buren buiten de bestaande fysieke buren. Bij het berekenen van een ruimte is er deze moeilijkheid niet, aangezien fysische eigenschappen zoals temperatuur niet geteleporteerd worden.

Er moet een methode gedefinieerd worden waarmee kan nagegaan worden of je vanuit een vakje kan navigeren naar een ander vakje. Daarbij moet rekening gehouden worden met teleportatie. Als je beweegt vanuit een dergelijk vakje naar een buurvakje, is er een kans dat je niet in die buur terecht komt, maar wel in één van de vakjes die als bestemming van de teleportatie geregistreerd staat.

Van deze methode wordt een formele specificatie verwacht als je gaat voor een score van 16 of meer.

2 Kerkers [*Dungeons*]

2.1 Vakjes in Kerkers

In dit deel van de opgave moeten kerkers geparametriseerd kunnen worden in het soort vakjes dat ze bevatten. Het moet dus mogelijk zijn om kerkers aan te maken waarin bijvoorbeeld enkel rotsvakjes kunnen zijn, of enkel teleportatievakjes, ... Controles omtrent het type van vakjes in een kerker moeten bij compilatie gebeuren (en mogen dus niet uitgesteld worden tot uitvoeringstijd).

2.2 Soorten Kerkers

Er zijn vanaf nu meerdere soorten kerkers in het spel. Kerkers kunnen echter nooit van type veranderen. Bovendien zijn enkel de volgende soorten kerkers mogelijk:

1. *Plateaus [Levels].* Plateaus zijn tweedimensionale dungeons waarin vakjes slechts één mogelijke z-coördinaat kunnen hebben. Ze bevinden zich dus in een vlak.

2. *Schachten [Shafts].* Schachten zijn groepjes vakjes die slechts in één coördinaat kunnen variëren. Er kunnen geen opgevulde vakjes in aanwezig zijn, en er mogen geen muren met deuren staan tussen aangrenzende vakjes binnen dezelfde schacht.

3. *Samengestelde Kerkers [Composite Dungeons].* Samengestelde kerkers bestaan uit een aantal plateaus, schachten of andere samengestelde kerkers. Kerkers die deel uitmaken van een samengestelde kerker noemen we subkerkers. Het toevoegen en verwijderen van subkerkers verloopt gelijkaardig aan het toevoegen van vakjes aan plateaus en schachten.

    - Vakjes behoren in principe slechts tot één Level of Shaft tegelijk, maar ze behoren indirect ook tot eventuele samengestelde kerkers die de Level of Shaft als subkerker bevatten.
    - Op coördinaten in een samengestelde kerker die niet behoren tot een subkerker, kunnen geen vakjes worden toegevoegd.
    - Een subkerker moet volledig in zijn bevattende kerker passen.

- Meerdere subkerkers van een bepaalde kerker mogen niet onderling overlappen.
- Er moeten in alle soorten kerkers inspectoren worden voorzien om de bevattende kerker van een kerker op te vragen, en ook één die de wortelkerker teruggeeft. Naast een methode om de rechtstreekse subkerkers op te vragen, biedt een samengestelde kerker ook een methode aan die een verzameling van alle bevatte Levels en Shafts teruggeeft (inclusief die van de eventuele samengestelde subkerkers).
- Bij het toevoegen van een subkerker aan een samengestelde kerker worden coördinaten meegegeven die de relatieve afstand tussen de oorsprongen van de subkerker en de kerker aangeven.

2.3 Iterator

Er moet een iterator worden aangeboden die alle vakjes in een kerker één voor één teruggeeft. De volgorde waarin vakjes worden afgeleverd mag je zelf bepalen. Wel is het zo dat de iterator direct moet inwerken op de interne gegevensstructuur (en dus bijvoorbeeld niet op een gegevensstructuur waarin je alle vakjes vooraf verzameld hebt).

Verder moet er een methode worden aangeboden die alle rotsvakjes teruggeeft binnen een kerker met een temperatuur van minstens 200 graden Celcius. Als je gaat voor een score van 18 of meer moet deze methode veralgemeend worden tot een methode waarmee je eender welke conditie kan meegeven waaraan de geselecteerde vakjes moeten voldoen (vb. alle vakjes met een vochtigheid van minstens 50% en een temperatuur beneden de 0 graden Celcius, alle teleportatievakjes die een gegeven vakje als bestemming hebben, alle vakjes die behoren tot een schacht, ...). Je moet al deze condities niet expliciet uitwerken. Het moet enkel duidelijk zijn dat je dergelijke condities kan meegeven als argument.

3 Testen

Je moet een consistente verzameling testen uitwerken voor de methode om na te gaan of je vanuit een vakje kan bewegen naar een ander vakje. Verder beslis je zelf voor welke methodes je testen uitwerkt. We verwachten niet dat je dat consistent doet voor alle methodes. Je moet ons enkel kunnen overtuigen dat je dergelijke testen kan schrijven.

4 Hoofdprogramma

Illustreer de correcte werking van je programma aan de hand van een samengestelde kerker, waarin minstens één schacht en één plateau aan mekaar grenzen via n vakje met een volle muur op de scheiding, en waarin een teleportatievakje uit die schacht een vakje uit dat plateau als bestemming heeft. Dat vakje bevindt zich in een ruimte van minstens 5 andere vakjes. Toon aan dat je vanuit het teleportatievakje in de schacht kan bewegen naar een willekeurig vakje binnen die ruimte.

Praktische Richtlijnen

We verwachten dat dit deel van het project volledig kan worden afgewerkt in 50 uur.

1 Ontwerptekening

Naast de specificatie en implementatie van de gevraagde klassen, moet ook een **schets van het ontwerp** worden gemaakt. Deze bevat alle ontwikkelde klassen en interfaces, behalve zelfgedefinieerde uitzonderingsklassen en testklassen. Het ontwerp moet uitgetekend worden in UML. Voor interfaces moet enkel de naam worden opgenomen in de tekening; voor klassen moet de naam worden opgenomen, samen met alle attributen die in die klasse aanwezig zijn. De ontwerptekening mag geen opsomming bevatten van methodes.

2 Inleveren

1. Het project moet ingediend worden vóór woensdag 25 mei om 17u00. 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 selecteert (Select the resources to export), en neem zowel de .java source files als de gegenereerde .class files op! Noem je JAR naar je eigen naam en voornaam en naar die van je eventuele partner (vb. EricSteegmansRaoulStrackx.JAR). Bij het indienen moet je bevestigen tot je oplossing effectief werd ingeleverd!

3. De ontwerptekening moet niet elektronisch worden ingeleverd. Ze moet enkel op papier worden meegenomen naar de verdeging. De tekening mag geproduceerd zijn met een werktuig zoals MagicDraw. Ze mag ook met de hand getekend worden.

3 Verdediging

1. Het project moet mondeling verdediging worden bij prof. Eric Steegmans. De verdedigingen starten op 30 mei. Op Toledo zal rond 20 mei een lijst verschijnen met mogelijke data waarop het project kan worden verdedigd. Een concrete afspraak voor je verdediging zal je vanaf dan kunnen aanvragen via een e-mail naar prof. Steegmans. Meer informatie hierover volgt later op Toledo.

2. Je moet je project verdedigd hebben ten laatste 2 werkdagen voor je deliberatie. Als je deliberatie bijvoorbeeld plaats heeft op vrijdag 1 juli, dan kan je ten laatste verdedigen op dinsdag 28 juni. Vind je deliberatie bijvoorbeeld plaats op dinsdag 5 juli, dan kan je ten laatste verdedigen op donderdag 30 juni.

3. Je moet de ontwerptekening in UML op papier meebrengen naar de verdediging. Zonder deze ontwerptekening kan de verdediging niet doorgaan! Vermeld duidelijk je naam en studierichting op deze schets.

4. Groepjes van twee dienen samen naar de verdediging te komen.

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

Veel succes!