Objectgeoriënteerd Programmeren: WPO 5 (polymorfie)

1. Inhoud

Polymorfie

2. Oefeningen

- A: Polygon
- A: Rekenmachine
- A: Infection
- A: Waves
- E: Snake
- X: Pacman
- X: Planetendans

2.1 A: Polygon

Herneem de opgave Polygon van een voorgaande WPO. In deze opgave ga je een lijst van polygonen bijhouden. Maak in de main een lijst aan van het type Polygon. Hierbij maak je van elk type polygon (vierkant, rechthoek, enz.) een 5-tal objecten aan. Deze worden onmiddelijk in de lijst bijgehouden. Doordat de lijst aangemaakt is met het type van de abstracte klasse, kunnen nu alle objecten getekend worden door een eenvoudige for-loop en bij elk element van de lijst de methode DrawObject aan te roepen. Dit effect heet polymorfie. Doe dit! De plaatsing van objecten en de afmetingen mogen willekeurig bepaald worden.

2.2 A: Rekenmachine

Herneem de oplossing van een voorgaand WPO. Hierbij declareer je elk object volgens de superklasse. De instantiëring gebeurt echter wel door een van de kindklassen. Voordat het programma werkt, worden een aantal methoden/eigenschappen aangepast:

- maak van de superklasse een abstracte klasse,
- voorzie in deze abstracte klasse de abstracte methode double Calculate();
- hernoem de specifieke rekenmethoden in de kindklassen volgens deze abstracte methode.

Nu kan je 1 object globaal declareren volgens het type van de abstracte superklasse. In elk van de button-oproepen herinstantieer het dat object volgens de gewenste kindklasse. Nadien reken je de berekening uit door de methode Calculate aan te roepen.

2.3 A: Infection

Herneem en kopieer de oplossing Virus van een voorgaand WPO. De aanpassingen tot polymorfie zijn relatief snel toe te passen. Vervang de 2 lijsten door 1 enkele lijst van het gemeenschappelijke type (superklasse). Het tekenen van de virussen en bloedlichamen kan nu zonder verdere aanpassingen aan de klassen in 1 enkele for-loop plaatsvinden. Om het detecteren van botsingen te bepalen, zullen een aantal tussenstappen ondernomen worden:

- Alle elementen bevinden zich nu in 1 enkele lijst. We moeten nagaan of we te maken hebben met een bloedlichaam of een virus. Bouw een if structuur in die toelaat om bloedlichamen te vinden.
- In deze if-structuur zal je het bloedlichaam vergelijken met de virussen uit de lijst. Schrijf een for-loop waarin je alle elementen van de lijst overloopt. Is het een virus, dan wordt deze op positie vergeleken met het huidige bloedlichaam. Is het geen virus, dan worden de posities niet vergeleken.

Op het einde verwijder je uiteraard de gesneuvelde virussen uit de lijst. Hierbij wordt opnieuw op type vergeleken. Het toepassen van polymorfie laat toe om de code compacter en meer leesbaar te maken.

2.4 A: Waves

Deze opgave is nu een aantal keren teruggekomen. Herneem deze opdracht opnieuw. Pas deze opgave aan zodat een willekeurige wave getekend kan worden door een wave volgens de interface te declareren. Het instantiëren verloopt volgens een van de kindklassen. Roep nadien ook de gepaste methode op om te tekenen.

2.5 E: Snake

In dit spel zal je snake programmeren. In snake is het de bedoeling dat je zowel de segmenten van de slang als de objecten volgens dezelfde methodiek kan tekenen. Voorzie een gemeenschappelijke klasse/abstracte klasse/interface waarin je de basismethoden van het tekenen voorziet. De objecten die de slang moet kunnen opeten leveren elk 1 punt op. Deze worden getekend a.d.h.v. een grijs vierkantje van 10 bij 10 pixels. Op het veld kunnen zich echter ook bommen bevinden. Bommen zorgen ervoor dat de slang 3 punten verliest. Een bom wordt getekend a.d.h.v. een zwart schijfje van 10 pixels. Elk segment van de slang wordt getekend a.d.h.v. een rood vierkantje van eveneens 10 bij 10 pixels. Maak in het begin 50 objecten aan en 25 bommen. Voorzie ook een klasse snake waarin je alle segmenten gaat onderbrengen. De slang neemt met 1 segment toe indien deze een object kan opeten. De slang verliest ook 3 segmenten indien een bom geraakt wordt. De bommen en objecten worden in een gemeenschappelijke lijst (main) bijgehouden. Telkens de slang een object/bom opeet verdwijnt deze ook van de lijst en

scherm. In het spel bevindt zich 1 slang. Om het tekenen te vereenvoudigen worden de slang (segmenten) en objecten/bommen op veelvouden van 10 pixels getekend. De slang start met een score van 5 punten en sterft indien de score onder 0 gaat. Het spel is gewonnen indien alle objecten opgegeten zijn.

Extra: Zorg ervoor dat de slang zichzelf niet opeet. Indien dit gebeurt, is de slang op slag dood.
Hint: Gebruik 4 knoppen (up, down, richt, left) om de slang in het veld te loodsen. Eventueel kan je hiervoor de pijltjestoesten gebruiken.

2.6 X: Pacman

Hieronder kan men de opgave van Pacman (of een variant erop) terugvinden. In deze opgave ga je zelf achterhalen op welke manier je klassen, overerving en polymorfie gaat toepassen. Hierbij mag je de opgave aanpassen zodat er objectgeoriënteerd programmeren aan te pas komt.

In bijlage kan het project voor deze opgave gevonden worden. Je kan van hieruit vertrekken om deze opgave tot een goed einde te brengen. In deze opgave zal je een alternatieve versie van pacman programmeren. In dit spel ben je Scooby die door een beige schijf voorgesteld wordt. Scooby vertrekt van rechtsonder in het doolhof en moet naar de uitgang (blauw vierkantje) bewegen. Door de gepaste knoppen te gebruiken kan je Scooby doorheen het doolhof verplaatsen. Er is echter één probleem: de val (groen-blauw vierkantje) vlak voor de uitgang. Om die hindernis te kunnen voorbijgaan moet je 150 punten kunnen afgeven. Die punten kan je in het doolhof winnen door de gekleurde voorwerpen te verzamelen. Om deze opgave te onwikkelen zal je in een aantal tussenstappen te werk gaan.

1. Zorg ervoor dat je het doolhof en de voorwerpen kan tekenen. Figuur 1 geeft exact weer wat het resultaat is als je het doolhof tekent vanuit de 2D array.

2. Eens je het doolhof kan tekenen, teken je Scooby in het doolhof (beige schijf).

3. Nadien zorg je ervoor dat Scooby doorheen het doolhof kan bewegen. Dit kan je doen door de 4 buttons op het formulier te voorzien. Merk op dat Scooby nooit op een muur terecht kan komen!

4. Vervolgens kan Scooby de nodige voorwerpen oprapen door ernaar te bewegen. Eens Scooby op de positie van een voorwerp staat, worden de punten van dit voorwerp toegevoegd aan de huidige score en verdwijnt dit voorwerp ook van het bord. Pas hiervoor de 2D array aan!

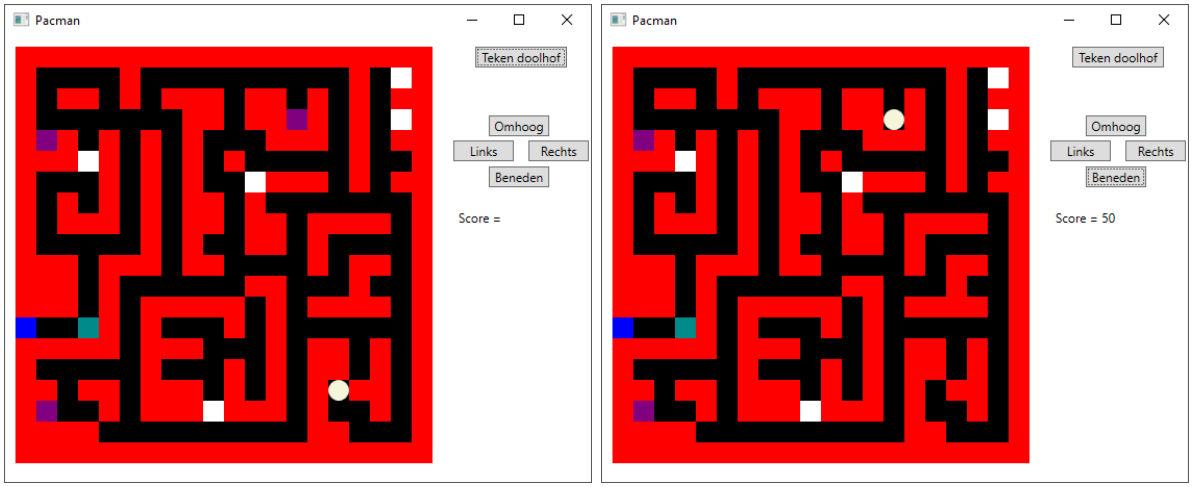
5. Indien Scooby verliest (score kleiner dan 0), verschijnt er “Game over”. In het geval dat Scooby de uitgang bereikt heeft, verschijnt er “Gewonnen”.

De 2D-array bevat volgende karakters (char) waarop je het spel kan baseren:

- ‘*’: is een muur en wordt door een rood vierkant getekend,

- spatie (‘ ’): is een gewoon wandelpad waar Scooby op kan lopen,
- ‘Y’: 50 extra bonuspunten (paars vierkantje),
- ‘X’: 15 extra bonuspunten (wit vierkantje) ,
- ‘D’: de val vlak voor de uitgang (donker cyaan vierkantje),
- ‘U’: de uitgang (blauw vierkantje)

De canvas neem je 400 bij 400 pixels groot. Elk te tekenen figuur heeft een grootte van 20 bij 20 pixels. Zorg ervoor dat Scooby telkens van cel naar cel kan bewegen (+1, -1). Schrijf de nodige functies/procedures om de code overzichtelijk te houden!



Figuur 1: Pacman. Links bij het opstarten van het programma en rechts wanneer Scooby een voorwerp heeft gevonden.

Extra: Indien je Scooby met de pijltjestoetsen kan besturen krijg je een bonuspunt.

2.7 X: Komeet

Hieronder kan men de opgave van de Komeet (of een variant erop) terugvinden. In deze opgave ga je zelf achterhalen op welke manier je klassen, overerving en polymorfie gaat toepassen. Hierbij mag je de opgave aanpassen zodat er objectgeoriënteerd programmeren aan te pas komt. Pas de opgave aan zodat je verschillende hemellichamen kan tekenen en dat ze elk afzonderlijk kunnen bewegen.

In ons zonnestelsel bevinden zich een zeer groot aantal hemellichamen, waaronder onze vertrouwde zon en enkele kometen. Afgezien van een aantal niet gemeenschappelijke eigenschappen worden hemellichamen gekenmerkt door de massa, de grootte (diameter), de bewegingssnelheid, de kleur en de positie (2D). Maak een klasse aan dat deze eigenschappen van de hemellichaam bevat. In ons zonnestelsel worden alleen de zon en één komeet getekend. Teken

beide hemellichamen door deze beide de gepaste eigenschappen te geven.
Natuurlijk blijven hemellichamen niet stationair binnen het zonnestelsel evolueren. Deze bewegen. De beweging van de hemellichamen wordt voornamelijk bepaald door de huidige snelheid van het hemellichaam en de zwaartekracht dat deze ondervindt t.g.v. andere hemellichamen. De grootte van de zwaartekracht t.g.v. een ander hemellichaam wordt beschreven als:

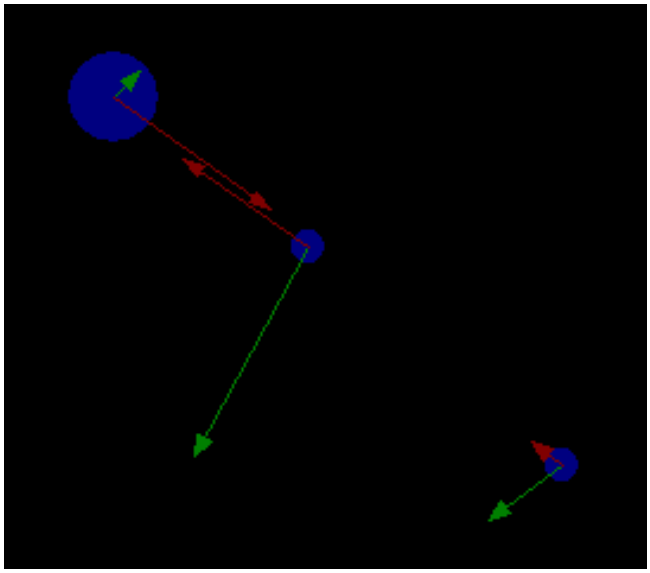
$$F = G \cdot \frac{m_1 \cdot m_2}{r^2} \tag{1}$$

waarbij F de grootte van de zwaartekracht is, m_1 de massa van hemellichaam 1 is, m_2 de massa van hemellichaam 2 en r de afstand tussen beide hemellichamen. Ga dus na hoe de afstand tussen beide hemellichamen berekend kan worden (maak hiervoor een functie aan).
In deze opgave mag de zon statisch blijven, dus enkel de positie van de komeet verandert. De nieuwe positie en snelheid van de komeet worden als volgt bepaald:

$$p_{komeet} = p_{komeet} + v_{komeet} \cdot t + \frac{at^2}{2} \tag{2}$$

$$v_{komeet} = v_{komeet} + at \tag{3}$$

De zwaartekracht, snelheid en positie worden uitgedrukt in vectoren (dus X en Y). Het verband tussen het vectorieel verband en de numerieke waarden in formules 1, 2 en 3 wordt gegeven door de hoek tussen de posities van de zon en de komeet.
Gebruik een timer om het effect van de gravitatie op de komeet weer te geven. Teken ook de situatie.
Hint: $F = m \cdot a$



Figuur 2: Voorbeeld van 3 hemellichamen. Hier worden ook de snelheidsvector (groen), samen met de zwaartekracht (rood) dat ondervonden wordt door elk lichaam getekend. De pijlen hoeven niet getekend te worden.

Extra: Kan je deze opgave uitbreiden naar meerdere hemellichamen waarbij deze allemaal kunnen bewegen (geen enkel lichaam is stationair)?