

# Greenfoot spiekbrief

## Wezens, spelers, acteurs, ...

Je moet de klasse *Actor* (indirect) uitbreiden en de methode *act()* implementeren. Deze methode wordt elke tijdspuls één keer uitgevoerd.

```
public class Auto extends Actor {  
  
    public void act () {  
        // wordt elke tijdspuls herhaald  
        ...  
    }  
  
}
```

## Bewegen

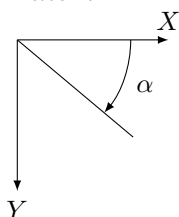
Relatieve beweging:

|                       |  |
|-----------------------|--|
| <i>move (afstand)</i> | Beweeg in de huidige richting over een <i>afstand</i> in roostereenheden |
| <i>turn (graden)</i>  | Draai (ter plaatse) over een hoek in <i>graden</i>                       |

Absolute beweging:

|                           |   |
|---------------------------|---|
| <i>setLocation (x,y)</i>  | Spring rechtstreeks naar een roosterpunt  |
| <i>setRotation (hoek)</i> | Keer u (ter plaatse) naar een richting    |
| <i>turnTowards (x,y)</i>  | Keer u (ter plaatse) naar een roosterpunt |

Coördinaten:



|             |     |
|-------------|-----|
| naar rechts | 0   |
| naar onder  | 90  |
| naar links  | 180 |
| naar boven  | 270 |

Roosterpunt (0,0) = linkerbovenhoek v/d wereld.

Opvragen:

|                       |   |
|-----------------------|---|
| <i>getX ()</i>        | Huidige X-coördinaat in het wereldrooster |
| <i>getY ()</i>        | Huidige Y-coördinaat in het wereldrooster |
| <i>getRotation ()</i> | Huidige (kijk)richting                    |

(Methoden *getX* en *getY* gooien een uitzondering op als de acteur zich niet op de wereld bevindt.)

## Imports

Om de greenfootbibliotheek te gebruiken, moet u de greenfootklassen bovenaan importeren.

```
import greenfoot.Actor;  
import greenfoot.Greenfoot;  
...
```

of korter

```
import greenfoot.*;
```

## Afbeeldingen

Afbeelding waarmee een acteur getoond wordt:

|                        |  |
|------------------------|--|
| <i>getImage ()</i>     | Huidige afbeelding (kan null zijn)                 |
| <i>setImage (afb)</i>  | Stel nieuwe afbeelding in (null = onzichtbaar)     |
| <i>setImage (naam)</i> | Idem, maar met <i>naam</i> v/h afbeeldingenbestand |

Afbeeldingen zijn van het type *GreenfootImage*.

```
GreenfootImage image =  
    new GreenfootImage ("vlag.png");  
...  
setImage (image);
```

## Type

Soms moet je niet een acteur zelf opgeven als parameter (object) maar het *type* van een object (klasse). Gebruik hiervoor de '.class'-notatie.

```
// alle beren op de wereld  
return  
    getWorld().getObjects(Beer.class);
```

Objecten van een deelklasse tellen ook mee.

## Overlappen e.d.

*intersects (andere)*

Overlapt deze acteur met de *andere* acteur?

*isTouching (type)*

Overlapt deze acteur met een acteur van dit *type*?

*removeTouching (type)*

Verwijder (hoogstens) één overlappende acteur van dit *type*.

*getIntersectingObjects (type)*

Geef alle overlappende acteurs terug van dit *type*.

*getOneIntersectingObject (type)*

Geef (hoogstens) één overlappende acteur terug van dit *type*.

## Overlappen e.d. (vervolg)

*getNeighbours*(afstand, false, type)

Geef alle acteurs terug van dit *type* binnen de gegeven *afstand*, gerekend in aantal horizontale/verticale stappen in het puntenrooster.

*getNeighbours*(afstand, true, type)

Idem, maar gerekend in aantal horizontale/verticale of *diagonale* stappen.

*getObjectsInRange*(straal, type)

Geef alle acteurs terug van dit *type* binnen de gegeven *straal*.

*getObjectsAtOffset*(dx, dy, type)

Geef alle acteurs terug van dit *type* die een gegeven punt overlappen, in relatieve coördinaten.

*getOneObjectAtOffset*(dx, dy, type)

Idem, maar geeft hoogstens één dergelijke acteur terug.

## De wereld

De wereld is een object van een klasse die *World* uitbreidt. Bij constructie geef je de dimensies op van het puntenrooster en de grootte van de cellen.

Bijv. schaakbord met cellen van 50×50 pixels:

```
public class Schaakbord extends World {
    public Schaakbord () {
        super (8, 8, 50);
    }
    ...
}
```

Bijv. 80000 cellen van 2×2 pixels, 200 rijen, 400 kolommen (= 400 cellen breed, 200 cellen hoog):

```
super (400, 200, 2, false);
```

Vierde argument *false* laat toe dat acteurs zich *buiten* de wereld begeven.

Ook *World* heeft een methode *act()* die elke tijdspuls één keer uitgevoerd worden.

## Acteurs op de wereld

|                                  |  |
|----------------------------------|--|
| <i>addObject</i> (acteur, x, y)  | Voeg de <i>acteur</i> toe op een roosterpunt                                 |
| <i>removeObject</i> (acteur)     | Verwijder de <i>acteur</i>   |
| <i>removeObjects</i> (lijst)     | Verwijder alle acteurs in de <i>lijst</i>                                    |
| <i>getObjects</i> (type)         | Geef alle acteurs uit de wereld terug van een bepaald <i>type</i>            |
| <i>getObjectsAt</i> (x, y, type) | Idem, maar moeten ook het roosterpunt overlappen. (In absolute coördinaten.) |

Een acteur kan zijn wereld opvragen met de methode *getWorld()*.

Voorbeeld: verwijder jezelf van de wereld

```
getWorld().removeObject(this);
```

## Toetsenbord

(Klassenmethoden uit de klasse *Greenfoot*.)

|                                    |  |
|------------------------------------|--|
| <i>Greenfoot.isKeyDown</i> (toets) | Is de <i>toets</i> op dit moment ingedrukt?                  |
| <i>Greenfoot.getKey</i> ()         | De laatste ingedrukte toets (sinds de vorige <i>getKey</i> ) |

Toetsen worden voorgesteld als strings:

|                |                                |
|----------------|--------------------------------|
| Lettertoetsen  | a b c ...                      |
| Cijfertoetsen  | 0 1 2 ...                      |
| Leestekens     | . ? ! , ; : ...                |
| Pijltjes       | right down left up             |
| Functietoetsen | F1 F2 ... F12                  |
| Andere         | enter space tab                |
| 'Dode' toetsen | escape backspace shift control |

(De methode *getKey()* geeft ook hoofdletters terug.)

```
// gebruik de pijltjestoetsen
// om van richting te veranderen
if (Greenfoot.isKeyDown("left")) {
    turn (270);
} else if (Greenfoot.isKeyDown("right")) {
    turn (90);
} else if (Greenfoot.isKeyDown("down")) {
    turn (180);
}
move (1);
```

## Diversen

(Klassenmethoden uit de klasse *Greenfoot*.)

*Greenfoot.getRandomNumber*(n)

'Willekeurig' geheel getal in het bereik [0, n-1]

*Greenfoot.playSound*(bestandsnaam)

Speel een geluid (*naam* = bestand)

## Volgorde

(Methoden van *World*)

*setPaintOrder*(Klasse1.class, Klasse2.class, ...)

Volgorde waarin objecten worden getekend. Objecten van *Klasse1* worden *boven* die van *Klasse2* getekend, ...

*setActOrder*(Klasse1.class, Klasse2.class, ...)

Volgorde waarin de 'act()'s worden opgeroepen. Eerst bij objecten van *Klasse1*, ...

## Simulatie

(Klassenmethoden uit de klasse *Greenfoot*.)

|                               |   |
|-------------------------------|---|
| <i>Greenfoot.stop</i> ()      | Leg de simulatie stil                                 |
| <i>Greenfoot.setSpeed</i> (s) | Stel de snelheid v/d simulatie in (s tussen 1 en 100) |

## Belangrijk.

Er staat nog meer informatie in de elektronische documentatie (API).