ANHANG

C

Kollisionserkennung

Für die Szenarien in diesem Buch werden – je nach Situation – unterschiedliche Methoden der Kollisionserkennung verwendet. Dieser Anhang fasst die den Greenfoot-Akteuren zur Verfügung stehenden Methoden zur Kollisionserkennung zusammen. Wir beschreiben kurz, welchen Zweck sie erfüllen und wann man sie am besten verwendet.

C.1 Übersicht über die Methoden

Die Greenfoot-Methoden zur Kollisionserkennung sind alle in der Klasse **Actor** zu finden. Es gibt insgesamt sieben Methoden, die folgendermaßen lauten:

`boolean isTouching (Class cls)`

Prüft, ob dieser Akteur irgendein Objekt der gegebenen Klasse berührt.

`List getIntersectingObjects(Class cls)`

Liefert alle Objekte zurück, die sich mit diesem Objekt überschneiden.

`Actor getOneIntersectingObject(Class cls)`

Liefert ein Objekt zurück, das sich mit diesem Objekt überschneidet.

`List getObjectsAtOffset(int dx, int dy, Class cls)`

Liefert alle Objekte zurück, die sich mit einer gegebenen Position überschneiden (relativ zur Position des aktuellen Objekts).

`Actor getOneObjectAtOffset(int dx, int dy, Class cls)`

Liefert ein Objekt zurück, das sich in der angegebenen Zelle befindet (relativ zur Position des aktuellen Objekts).

`List getNeighbours(int distance, boolean diagonal, Class cls)`

Liefert die Nachbarn des aktuellen Objekts innerhalb eines bestimmten Abstands zurück.

`List getObjectsInRange(int r, Class cls)`

Liefert alle Objekte im Umkreis **`r`** um dieses Objekt zurück.

C.2 Hilfsmethoden

Zwei dieser Methoden, **`getIntersectingObjects`** und **`getObjectsAtOffsset`**, verfügen über Hilfsmethoden, auch Convenience-Methoden genannt, die mit **`getOne`**... beginnen.

Diese Hilfsmethoden funktionieren ähnlich wie die Methode, auf der sie basieren, liefern jedoch nur einen einzigen Akteur anstelle einer Liste von Akteuren zurück. Wenn mehrere andere Akteure festzustellen sind (d.h., mehrere andere Akteure überschneiden sich gleichzeitig mit unserem), liefert die Variante, die eine Liste zurückliefert, alle betroffenen Akteure zurück. Die Variante, die nur einen Akteur zurückliefert, wählt hingegen aus allen betroffenen Akteuren willkürlich einen aus.

Diese Hilfsmethoden dienen lediglich dem Zweck, den Code zu vereinfachen, denn oft sind wir nur an einem Akteur interessiert, der sich mit unserem überschneidet. In diesen Fällen erlaubt uns die Hilfsmethode, den Akteur zu ermitteln, ohne mit Listen arbeiten zu müssen.

C.3 Niedrige kontra hohe Auflösung

Wie wir im Verlauf dieses Buches gesehen haben, kann die Auflösung (Zellengröße) der Greenfoot-Welten unterschiedlich ausfallen. Dies ist für die Kollisionserkennung von Bedeutung, da wir oft – je nach Auflösung – spezielle Methoden verwenden.

Wir unterscheiden zwischen zwei Fällen: Welten mit niedriger Auflösung, bei denen das gesamte Akteur-Bild vollständig in einer einzigen Zelle liegt (Abbildung C.1a), und Welten mit hoher Auflösung, bei denen das Bild eines Akteurs sich über mehrere Zellen erstreckt (Abbildung C.1b).

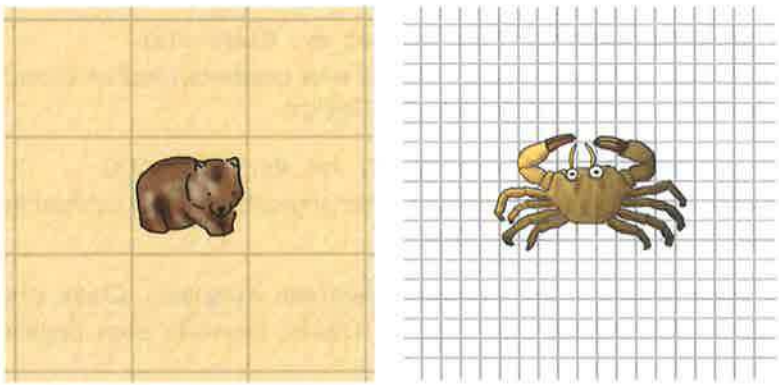

Abbildung C.1
Beispiele für Greenfoot-Welten mit a) niedriger und b) hoher Auflösung.

C.4 Sich überschneidende Objekte

Methoden:

`List getIntersectingObjects(Class cls)`

Liefert alle Objekte zurück, die sich mit diesem Objekt überschneiden.

`Actor getOneIntersectingObject(Class cls)`

Liefert ein Objekt zurück, das sich mit diesem Objekt überschneidet.

Diese Methoden liefern Akteure zurück, deren Bild sich mit dem Bild des aufrufenden Akteurs überschneidet. Man spricht davon, dass Bilder sich überschneiden, wenn irgendein Teil eines Bildes einen beliebigen Teil eines anderen Bildes berührt. Diese Methoden solltest du am besten in Szenarien mit hoher Auflösung einsetzen.

Die Überschneidung wird mittels umgebender Rechtecke berechnet, sodass die Überlappung von absolut transparenten Teilen der Bilder ebenfalls als Überschneidung gewertet wird (Abbildung C.2).

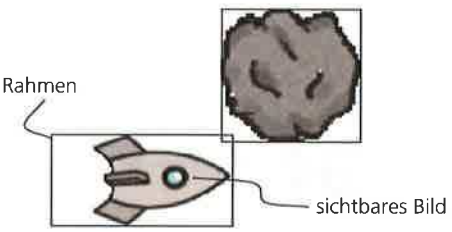

Abbildung C.2
Die Überschneidung von Akteuren mittels ihrer umgebenden Rechtecke.

Die Methode **`isTouching`** verwendet dieselbe Art von Test, gibt aber einen einfachen booleschen Wert anstelle eines Akteurs zurück:

`boolean isTouching(Class cls)`

Prüft, ob dieser Akteur irgendein Objekt der gegebenen Klasse berührt.

Mittels dieser Methoden wird oft geprüft, ob ein Akteur mit einem anderen Akteur zusammengestoßen ist. Die Ungenauigkeit, die aus der Berücksichtigung der umgebenden Rechtecke resultiert (statt des sichtbaren Teils des Bildes), kann oft vernachlässigt werden.

Der Parameter kann als Filter verwendet werden. Wenn als Parameter für diese Methode eine Klasse angegeben wird (z.B. **`Wombat.class`**), werden nur Objekte dieser Klasse berücksichtigt und alle anderen Objekte ignoriert. Lautet der Parameter **`null`**, wird jedes überschneidende Objekt zurückgeliefert.

C.5 Objekte in der Umgebung

Methoden:

`List getObjectsAtOffset(int dx, int dy, Class cls)`

Liefert alle Objekte zurück, die sich mit einer gegebenen Position überschneiden (relativ zur Position des aktuellen Objekts).

`Actor getOneObjectAtOffset(int dx, int dy, Class cls)`

Liefert ein Objekt zurück, das sich in der angegebenen Zelle befindet (relativ zur Position des aktuellen Objekts).

Diese Methoden können verwendet werden, um auf Objekte zu prüfen, die sich in einem angegebenen Abstand zur aktuellen Position des Akteurs befinden. Sie sind sowohl in Szenarien mit niedriger als auch mit hoher Auflösung nützlich.

Die Parameter *dx* und *dy* geben den Abstand als Anzahl von Zellen an. Abbildung C.3 veranschaulicht die Position am Offset (2,0) des Wombats (2 Zellen Abstand entlang der *x*-Achse und 0 Zellen Abstand entlang der *y*-Achse).

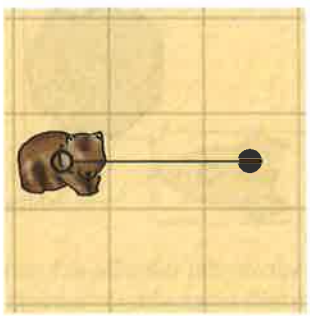

Abbildung C.3
Prüft einen gegebenen Abstand von einer Position (Beispiel hier: Abstand 2,0).

Ein anderer Akteur befindet sich an diesem Abstand, wenn irgendein Teil des Bildes dieses Akteurs sich mit dem Mittelpunkt der angegebenen Zelle überschneidet. Der Parameter **`cls`** bietet auch hier die Möglichkeit, die zu berücksichtigenden Objekte zu filtern (siehe oben).

Diese Methoden werden oft verwendet, um den Bereich vor oder unter einem Akteur zu prüfen (um zu testen, ob sich der Akteur vorwärtsbewegen kann bzw. ob er auf etwas steht).

C.6 Nachbarn

Methode:

`List getNeighbours(int distance, boolean diagonal, Class cls)`

Liefert die Nachbarn des aktuellen Objekts innerhalb eines bestimmten Abstands zurück.

Diese Methode wird verwendet, um Objekte zu ermitteln, die sich in den Zellen um den aktuellen Akteur befinden. Sie eignet sich deshalb besonders für Szenarien mit niedriger Auflösung.

Achte auf die Schreibweise des Methodennamens. Er lautet tatsächlich **`getNeighbours`** (mit englischer Schreibweise) – denn Greenfoot ist kein amerikanisches System.[1]

Die Parameter geben den zu berücksichtigenden Abstand von dem aufrufenden Akteur an und ob diagonal liegende Zellen ebenfalls zu prüfen sind. Abbildung C.4 zeigt die benachbarten Zellen bei einem Abstand 1 mit und ohne die diagonalen Zellen.

Ein Abstand von *N* wird definiert als alle Zellen, die von der Position des Akteurs aus in *N* Schritten erreicht werden können. Der Parameter **`diagonal`** gibt an, ob diagonale Schritte in diesem Algorithmus erlaubt sind.

Wie schon bei den vorherigen Methoden bietet der Parameter **`cls`** die Möglichkeit, nur Objekte einer angegebenen Klasse zu berücksichtigen.

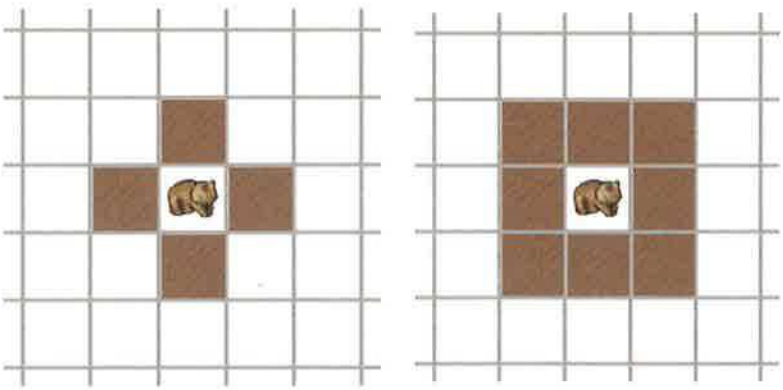

Abbildung C.4
Beispiel der Methode **`getNeighbours`**:
a) Nachbarn ohne Diagonalen – **`false`**.
b) Nachbarn mit Diagonalen – **`true`**.

1 In England heißen Nachbarn „neighbours", in den USA „neighbors" ohne u. Daher kann man an der Schreibweise erkennen, dass Greenfoot englische Wurzeln hat.

C.7 Objekte im Umkreis

Methode:

`List getObjectsInRange(int r, Class cls)`

Liefert alle Objekte im Umkreis **r** um dieses Objekt zurück.

Diese Methode liefert alle Objekte zurück, die in einem gegebenen Umkreis um den aufrufenden Akteur liegen. Ein Objekt befindet sich in diesem Umkreis, wenn seine Position eine Zelle ist, deren Mittelpunkt sich in der Entfernung **r** oder weniger von dem aufrufenden Akteur befindet (Abbildung C.5). Der Umkreis **r** wird in Zellen gemessen.

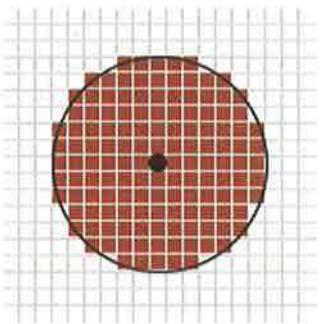

Abbildung C.5
Die Zellen in einem gegebenen Umkreis um eine Position.

Diese Methode ist am nützlichsten in Szenarien mit hoher Auflösung. Wie schon bei den vorherigen Methoden, kann auch hier ein Klassenfilter angewendet werden.