Schritt 5: Blöcke und Paddel

Der Spielgegenstand Block

Die Blöcke tun nichts. Lösche allen überflüssigen Code aus der Klasse Block.

Der Spielgegenstand Paddel

Das Paddel soll mit den Pfeiltasten ← und → bis zum linken und rechten Spielfeldrand bewegt werden können.

Finde in der Dokumentation zu den Klassenbibliotheken von Greenfoot heraus, welche Methoden es in der Klasse Greenfoot zur Tastensteuerung gibt.

Ergänze nun den Quellcode in der Methode `act()` in deiner Paddel-Klasse. Der nachfolgende Text kann dir dabei vielleicht helfen (vgl. Roboter-AB4)

```
falls Links-Taste gedrückt
    setzeAnPosition(x-Position - 1, y-Position);
falls Rechts-Taste gedrückt
    setzeAnPosition(x-Position + 1, y-Position);
```

Teste, ob dein Paddel das Gewünschte leistet (vorher ist die Run-Schaltfläche zu drücken). Der Geschwindigkeitsregler sollte dabei auf etwa 50% stehen. Wenn sich dein Paddel zu langsam bewegt, kannst du die x-Position anstatt um 1 Pixel z.B. um 2, 3, oder vielleicht 10 Pixel ändern. Finde einen geeigneten Wert.

Für Experten: Du wirst bemerkt haben, dass sich dein Paddel etwas über die Spielfeldgrenzen links und rechts hinaus bewegen lässt. Sorge dafür, dass sich das Paddel nur bis an die Spielfeldränder bewegen lässt.

Startzustand

Es ist umständlich, jedes Mal die Welt neu mit den Objekten zu füllen. In Greenfoot kann man eine einmal erstellt Welt abspeichern, so dass die Objekte nach jeder neuen Compilierung oder Drücken des "Reset"-Buttons wieder eingefügt werden. Füge in deine Welt das Paddle, einige Blöcke und einen Ball ein. Klicke dann mit der rechten Maustaste auf die Welt und wähle "Save the world".

Reflexion

Der Ball soll nun am Paddel reflektiert werden, falls er es trifft. Wenn ein Actor-Objekt auf ein zweites Actor-Objekt trifft, erhält man das zweite Actor-Objekt mit der Actor-Methode `getOneIntersectingObject(java.lang.Class cls)`.

Falls diese beiden Objekte sich nicht überlappen, liefert diese Methode den Wert `null` zurück.

Der zugehörige Quelltextschnipsel, um zu erkennen, ob es eine Überlappung gibt, lautet:

```
if (getOneIntersectingObject(Paddel.class) != null) {
    ...
}
```

Dann soll sich die Richtung des Balls so ändern, als wenn er am unteren Spielfeldrand reflektiert worden wäre.

Der Parameter `Paddel.class` in dieser Methode ist etwas ungewöhnlich. Gemeint ist damit, dass auf Kontakt mit irgendeinem Objekt der Klasse Paddel geprüft wird und nicht auf Kontakt mit dem konkreten Paddel auf dem Spielfeld.

Erstelle eine neue Methode `reflektierePaddel()` und füge den zugehörigen Methodenaufruf **an geeigneter Stelle** in `act()` ein. Teste dein Programm.

Nun fehlt nur noch die Reflexion des Balls an den Blöcken und das Entfernen des getroffenen Blocks. Erstelle dazu eine Methode `reflektiereBlock()` und füge den zugehörigen Methodenaufruf an geeigneter Stelle in `act()` ein.

```
private void reflektiereBlock() {
    Actor block = getOneIntersectingObject(Block.class);
    if (block != null) {
        dy = -dy;
        getWorld().removeObject(block);
    }
}
```

Im Unterschied zu `reflektierePaddel()` wird ein Verweis auf den getroffenen Block einige Zeilen später benötigt, weil man genau diesen Block entfernen will. Aus diesem Grund muss man ihn in einer lokalen Variablen speichern.

Mit dem Methodenaufruf `getOneIntersectingObject(Block.class)` prüft man zunächst auf Kontakt mit irgendeinem Objekt der Klasse Block (`Block.class`). Erst dann, wenn diese Methode einen `Actor` zurück liefert, hat man den konkreten Block gefunden, den man anschließend entfernen kann.

<<< Zurück zu Schritt 4 | **Breakout: Schritt 5** | Kapitelübersicht | Weiter zu Schritt 6 >>>
