KAPITEL

10 Bilder und Töne erzeugen

Lernziele

Themen: Töne erzeugen, Bilder erzeugen, dynamische Bildänderungen, Mauseingaben bearbeiten

Konzepte: Soundformate, Soundqualitätsparameter, Bilddateiformate, RGBA-Farbmodell, Transparenz

Viele der Szenarien, die wir bisher kennengelernt haben, waren nicht nur aufgrund ihres Programmcodes interessant, der ihr Verhalten vorgab, sondern auch aufgrund ihres intensiven Einsatzes von Sound und Bildern. Nur haben wir bis dato der Erzeugung dieser Mediendateien kaum Beachtung geschenkt – und stattdessen oft auf bestehende Bilder und Sounds zurückgegriffen.

In diesem Kapitel lernen wir einige Aspekte der Erzeugung von und der Arbeit mit Mediendateien kennen. Zuerst wollen wir dir einige Hintergrundinformationen über Sounds in Computerprogrammen vermitteln, gefolgt von verschiedenen Techniken, Bilder zu erzeugen und einzusetzen.

Als positiven Nebeneffekt werden wir außerdem lernen, wie Mauseingaben zu behandeln sind.

10.1 Vorbereitende Maßnahmen

Im Gegensatz zu den vorherigen Kapiteln werden wir hier kein vollständiges Szenario erstellen, sondern anhand mehrerer kleiner Übungen bestimmte Techniken veranschaulichen, die dann in einer Vielzahl verschiedener Szenarien genutzt werden können. Der erste Übungsblock beschäftigt sich mit der Entwicklung eines Szenarios, das einen von uns selbst erzeugten Sound abspielt, wenn der Benutzer auf einen Akteur klickt.

Für diese Übungen werden wir nicht auf ein bereits teilweise implementiertes Szenario zurückgreifen, sondern ein ganz neues Szenario erstellen.

Übung 10.1 Erstelle als Vorbereitung für die Übungen in diesem Kapitel ein neues Szenario. Du kannst einen beliebigen Namen wählen.

Wie du siehst, enthält das neue Szenario automatisch die Oberklassen **`World`** und **`Actor`**, jedoch keine weiteren Klassen.

Übung 10.2 Erzeuge eine Unterklasse von **`World`** namens **`MyWorld`**. Das Hintergrundbild kannst du nach Belieben wählen. Kompiliere deinen Code.

Übung 10.3 Ändere die Größe und die Auflösung der Welt, sodass die Zellengröße ein Pixel und die Gesamtgröße 400 mal 300 Zellen (Breite mal Höhe) beträgt.

Übung 10.4 Erzeuge in deinem Szenario eine Unterklasse von **`Actor`**. Zu diesem Zeitpunkt spielt es keine große Rolle, um welchen Akteur es sich dabei handelt. Wenn du möchtest, kannst du unter den zur Verfügung stehenden Bildern der Bibliothek im Dialogfeld NEUE KLASSE eines auswählen, das dir besonders interessant erscheint. Wähle für deine Klasse einen passenden Namen. (Denke jedoch daran, dass Klassennamen immer mit einem Großbuchstaben beginnen sollten.)

Übung 10.5 Füge Code in deine Klasse **`MyWorld`** ein, der automatisch ein Akteur-Objekt in der Welt platziert.

Übung 10.6 Schreibe Code in die **`act`**-Methode deines Akteurs, der den Akteur bei jedem **`act`**-Schritt um 10 Pixel nach rechts bewegt.

Damit solltest du jetzt über ein Szenario verfügen, in dem der Akteur bei Ausführung nach rechts bewegt wird. Allerdings ist Bewegung in diesem Falle nicht unser Hauptanliegen. Wir haben die Bewegung nur hinzugefügt, um einen visuellen Anfangseffekt zu haben, mit dem wir experimentieren können.

Als nächste vorbereitende Maßnahme wollen wir uns darum kümmern, dass der Akteur auf Mausklicks reagiert.

Konzept

Wir können die Methode **`mouseClicked`** verwenden, wenn wir prüfen möchten, ob der Benutzer ein bestimmtes Objekt angeklickt hat.

Übung 10.7 In der Klasse **`Greenfoot`** gibt es mehrere Methoden zur Behandlung von Mauseingaben. Wie lauten sie? Suche danach in der Greenfoot-Klassendokumentation und schreibe sie auf.

Übung 10.8 Was ist der Unterschied zwischen **`mouseClicked`** und **`mousePressed`**?

Wenn wir auf Mausklicks reagieren wollen, können wir die Methode **mouseClicked** aus der Klasse **Greenfoot** verwenden. Diese Methode liefert einen booleschen Wert zurück und kann als Bedingung in einer **if**-Anweisung verwendet werden.

Über den Parameter, der der Methode **mouseClicked** übergeben wird, können wir das Objekt angeben das wir prüfen möchten: die Methode gibt **true** zurück, wenn es angeklickt wurde. Wenn es egal ist, wo die Maus geklickt wird, können wir als Parameter **null** angeben – die Methode wird dann bei jedem Klick **true** zurückliefern, egal wo sich die Maus befindet.

Übung 10.9 Ändere den Code in deiner Akteur-Klasse, sodass sich der Akteur nur nach einem Mausklick nach rechts bewegt. Dabei kann der Mausklick an einer beliebigen Stelle in der Welt erfolgen.

Übung 10.10 Ändere den Code jetzt so, dass sich der Akteur nur bewegt, wenn der Benutzer den Akteur anklickt. Dazu musst du der Methode **mouseClicked** anstelle von **null** den Akteur selbst als Parameter übergeben. Zur Erinnerung: Du kannst das Schlüsselwort **this** verwenden, um auf das aktuelle Objekt Bezug zu nehmen

Übung 10.11 Teste deinen Code: Platziere mehrere Akteur-Objekte in der Welt und stelle sicher, dass sich nur der Akteur bewegt, auf den du geklickt hast.

Dein Szenario sollte jetzt einen Akteur aufweisen, der auf Mausklicks reagieren kann. Dies ist eine gute Ausgangsbasis für unsere folgenden Experimente mit Sound und Bildern. (Wenn du Probleme hattest, dieses Szenario zu erzeugen, kannst du dich des Szenarios *soundtest* zu diesem Kapitel bedienen, das die gewünschte Ausgangsbasis implementiert.)

10.2 Mit Sound arbeiten

Wie wir in einem der vorherigen Kapitel gesehen haben, gibt es in der Klasse **Greenfoot** eine Methode namens **playSound**, die wir zum Abspielen einer Sounddatei verwenden können. Damit die Sounddatei abgespielt werden kann, muss sie sich im Ordner *sounds* des entsprechenden Szenario-Ordners befinden.

Als Einstieg wollen wir eine bereits existierende Sounddatei abspielen.

Übung 10.12 Wähle eine Sounddatei aus einem deiner Greenfoot-Szenarien und kopiere sie in den Ordner *sounds* deines aktuellen Szenarios. Ändere dann deinen Akteur so, dass er sich nicht bewegt, sondern den Sound abspielt, wenn du ihn anklickst.

Fallstricke

Einige Betriebssysteme (vor allem Microsoft Windows) sind so konfiguriert, dass die Endungen der Dateinamen (die Erweiterungen) bei der Anzeige unterdrückt werden. Eine Datei, deren voller Name *meinsound.wav* lautet, würde dann nur als *meinsound* angezeigt. Dies stellt ein Problem dar, da wir in unserem Code den vollen Namen der Datei, einschließlich Endung, verwenden müssen. Die Anweisung

```
Greenfoot.playSound("meinsound");
```

würde fehlschlagen, da die Datei nicht gefunden würde. Wenn die Endung jedoch nicht angezeigt wird, haben wir keine Ahnung, wie sie lautet.

Die Lösung zu diesem Problem besteht darin, die Einstellungen des Betriebssystems so zu ändern, dass Endungen immer angezeigt werden. In Windows gibt es zum Beispiel ein Kontrollkästchen ERWEITERUNGEN BEI BEKANNTEN DATEITYPEN AUSBLENDEN, das auf keinen Fall markiert sein sollte. Unter Windows 10 findest du dieses Kontrollkästchen, indem du den gewünschten Ordner aufrufst und dann über den Menübefehl ANSICHT/OPTIONEN/ORDNER- UND SUCHOPTIONEN ÄNDERN zur Registerseite ANSICHT wechselst. In anderen Windows-Systemen mögen diese Menünamen anders lauten, aber das Kontrollkästchen ist auch dort immer zu finden.

Eine bereits vorhandene Sounddatei abzuspielen, ist nicht schwer. Interessanter ist es da schon, selbst Sounddateien zu erstellen.

10.3 Sound in Greenfoot aufnehmen und bearbeiten

Es gibt eine ganze Reihe von Möglichkeiten, um an Sounddateien zu gelangen. Wir können Sounds aus anderen Greenfoot-Projekten kopieren oder aus einer der kostenlosen Soundbibliotheken im Internet herunterladen. Wenn du Sounds aus dem Web herunterlädst, achte bitte auf die Urheberrechte: Nicht alles im Internet ist kostenlos – respektiere das Urheberrecht von anderen! Der einfachste Weg zu einer Sounddatei besteht in der Aufnahme eigener Sounds.

Dazu benötigst du ein Mikrofon (in vielen Laptops ist bereits ein Mikrofon vorinstalliert und oft verfügen Computer-Headsets ebenfalls über ein Mikrofon). Wenn du im Moment kein Mikrofon angeschlossen hast, kannst du diesen Abschnitt auch überspringen.

In ***Kapitel 3*** hatten wir bereits einen kurzen Blick auf den Klang-Rekorder von Greenfoot geworfen. Mit ihm können wir Sound aufnehmen, unerwünschte Teile am Anfang und am Ende abschneiden und speichern.

Übung 10.13 Verwende den Klang-Rekorder von Greenfoot (den du über das *Ausführen*-Menü öffnen kannst), um einen neuen Sound für dein Szenario aufzunehmen. Wenn der Akteur angeklickt wird, soll der Sound ertönen.

Übung 10.14 Führe eine zweite Art von Akteuren ein. Platziere mindestens eine Instanz dieser Klasse in der Welt. Dieser Akteur soll einen anderen Ton von sich geben, wenn er angeklickt wird.

Übung 10.15 Nimm ein Bild mit einer Digitalkamera oder Webcam von einem oder mehreren deiner Freunde auf. Erzeuge Akteure, die diese Freunde darstellen. Zeichne die Stimmen deiner Freunde auf und lass die Akteure diese Aufnahmen sprechen, wenn du sie anklickst.

10.4 Externe Soundaufnahme und -bearbeitung

Die Verwendung des Klang-Rekorders von Greenfoot ist schnell und leicht und ist in vielen Fällen ausreichend. Wenn wir aber ein wenig professioneller werden wollen, dann möchten wir vielleicht bestimmte Soundeffekte aufnehmen, die mit diesem einfachen Rekorder schwer zu erzeugen sind. Wenn wir an den speziellen Eigenschaften des Sounds interessiert sind, dann sollten wir eine separate Software zur Soundaufnahme einsetzen.

Es gibt eine Vielzahl von Soundaufnahmeprogrammen, von denen einige sogar kostenlos sind. Wir haben hier für unseren Beispiel-Screenshot *Audacity*[1] verwendet. Audacity ist eine gute Wahl, da es sehr leistungsstark ist, auf mehreren Betriebssystemen läuft, nichts kostet und sich relativ einfach bedienen lässt. Es gibt jedoch noch eine Menge anderer Soundaufnahmeprogramme, sodass du die Qual der Wahl hast, mit welchem Programm du arbeiten möchtest.

Abbildung 10.1 zeigt die typische Benutzeroberfläche eines Soundaufnahmeprogramms. Es gibt Steuerelemente für Aufnahme, Abspielen usw. und eine Anzeige für den wellenförmigen Verlauf des aufgenommenen Sounds (der blaue Graph).

1 Audacity kannst du dir von *http://audacity.sourceforge.net* herunterladen.

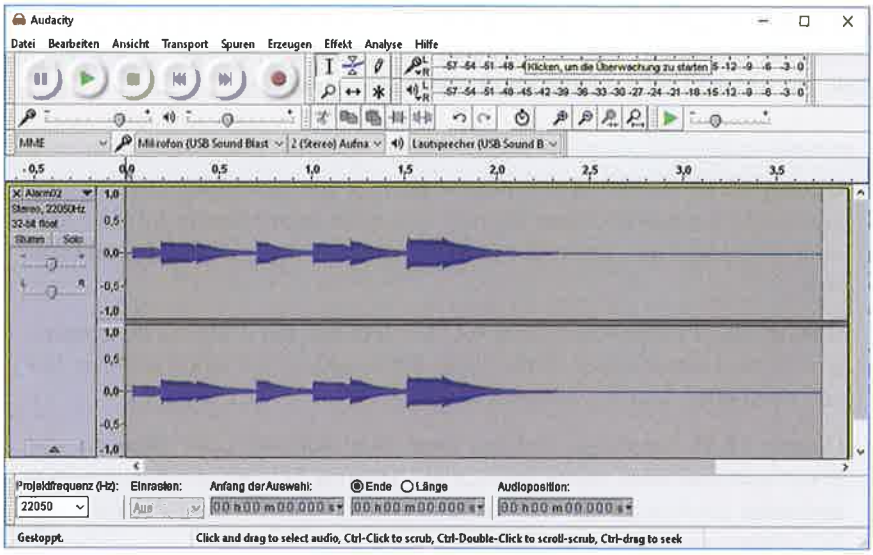

Abbildung 10.1
Ein Soundaufnahme- und -bearbeitungsprogramm.

Das Aufnehmen von Sound ist relativ einfach – in der Regel findest du das selber schnell heraus, wenn du eine Weile mit dem Programm herumspielst.

Übung 10.16 Öffne ein Soundaufnahmeprogramm und nimm einen Sound auf. Spiele diesen Sound ab. Klingt er wie erwartet? Wenn nicht, lösche ihn und versuche es erneut.

Wie auch beim Klang-Rekorder von Greenfoot gibt es bei der Aufnahme eines Sounds am Anfang und am Ende oft eine Verzögerung oder ein Rauschen, das wir nicht benötigen. Wie du beispielsweise in Abbildung 10.1 siehst, herrscht am Anfang und am Ende der Sounddatei eine Zeitlang Stille (die geraden horizontalen Linien links und rechts des Graphen). Wenn wir die Sounddatei so, wie sie hier zu sehen ist, speichern, würde beim Abspielen des Sounds zunächst eine halbe Sekunde ein Verzögerungseffekt eintreten (da nichts zu hören wäre).

Du solltest auch hier, genau wie beim Klang-Rekorder von Greenfoot, die Sounddatei kürzen: die unerwünschten Bits am Anfang und am Ende abschneiden. Gute Soundbearbeitungsprogramme bieten noch viel mehr: Du kannst Segmente aus der Mitte herausschneiden, kopieren und wieder einfügen, um so einen Soundeffekt mehrmals zu wiederholen, Teile des Sounds aufnehmen und vieles mehr.

Übung 10.17 Bearbeite deine Sounddatei, sodass sie nur den von dir gewünschten Sound enthält. Entferne vom Anfang, Ende oder aus der Mitte alle Bereiche, die nicht zu hören sind, unerwünschte Geräusche enthalten oder sonst wie nicht benötigt werden.

Eine der interessantesten Fähigkeiten eines Soundbearbeitungsprogramms besteht allerdings im Einsatz von *Filtern* oder *Effekten*. Filter, um Sound zu verändern oder gänzlich neue Soundelemente zu erzeugen. So lassen sich viele verschiedene Soundeffekte erzeugen.

Durch die Anwendung von Filtern, wie Verstärken, Echo, Rückwärts, Geschwindigkeitsänderungen, auf einfache aufgenommene Geräusche (wie Sprechen, Klatschen, Pfeifen, Rufen) lässt sich eine Vielzahl von weiteren Effekten erzielen.

Übung 10.18 Wenn dein Soundprogramm Filter (manchmal auch *Effekte* genannt) unterstützt, wende einige dieser Filter auf deinen aufgenommenen Sound an. Wähle drei deiner Lieblingsfilter und beschreibe in Schriftform, was sie machen. Gib ein Beispiel dafür an, wo sich dieser Effekt sinnvoll einsetzen ließe.

Übung 10.19 Produziere die folgenden Sounds: einen Hasen, der eine Karotte frisst; eine Explosion; den Klang von zwei harten Objekten, die zusammenstoßen; einen „Spiel beendet"-Sound für Spieler, die das Spiel verloren haben; einen Spielende-Sound, wenn der Spieler gewonnen hat; eine Roboterstimme; das Geräusch eines Sprungs (wenn ein Akteur im Spiel springt).

Wenn die Bearbeitung des Sounds abgeschlossen ist, kannst du die Datei speichern.

10.5 Sounddateiformate und Dateigrößen

Sounddateien können in vielen verschiedenen Formaten und Codierungen abgespeichert werden, was leider sehr schnell zu Verwirrungen führt.

Konzept

Sounds können in einer Vielzahl von verschiedenen **Formaten** und **Codierungen** abgespeichert werden. Nicht alle Programme können alle Soundformate spielen. Für Greenfoot verwenden wir in der Regel das Format **WAV**.

Greenfoot kann Dateien im WAV-, AIFF-, AU- und MP3-Format abspielen. Einige dieser Formate sind jedoch sogenannte „Containerformate"; das heißt, sie können verschiedene Codierungen enthalten, von denen Greenfoot leider nur einige lesen kann. Folglich kann Greenfoot zum Beispiel nicht alle WAV-Dateien abspielen.

Wenn du deine eigenen Sounds speicherst, solltest du als Format „signed 16-bit PCM" wählen. Bei diesem Format kannst du am sichersten sein, dass das Abspielen klappt. In vielen Soundaufnahmeprogrammen musst du hierzu allerdings den Menübefehl EXPORTIEREN und nicht den Standardbefehl SPEICHERN wählen. Achte darauf, dass du deine Datei in diesem Format abspeicherst.

Wenn du eine Sounddatei verwenden möchtest, die Greenfoot nicht abspielen kann (die du dir vielleicht aus dem Internet heruntergeladen hast), kannst du diese in der Regel in deinem Soundbearbeitungsprogramm öffnen und in das gewünschte Format konvertieren.

WAV ist ein gutes Format für kurze Soundeffekte. MP3, ein anderes beliebtes Format, wird hauptsächlich für Musik und andere längere Aufnahmen eingesetzt. Es ist ein proprietäres Format (das heißt, wenn du ein Programm schreibst, das MP3 erzeugt, musst du Lizenzgebühren bezahlen), deshalb unterstützen es viele der

kostenlosen Soundbearbeitungsprogramme nicht. In Greenfoot kannst du zwar MP3-Dateien abspielen (du könntest deinen eigenen MP3-Player schreiben!), aber zum Speichern unserer selbsthergestellten Soundeffekte eignet es sich nicht.

Beim Speichern deiner Sounddatei musst du außerdem sicherstellen, dass dein Greenfoot-Szenario diese Datei auch finden kann. Die Sounddatei muss in dem Ordner *sounds* innerhalb deines Szenario-Ordners liegen, damit dein Code darauf zugreifen kann. Der Klang-Rekorder von Greenfoot weiß das und speichert die Dateien automatisch dort. Wenn du ein externes Soundbearbeitungsprogramm verwendest, musst du selbst dafür sorgen, dass die Datei am richtigen Ort abgelegt wird.

Übung 10.20 Speichere deinen aufgenommenen Sound in einem Greenfoot-tauglichen Format. Verschiebe die Datei in den *sounds*-Ordner deines Szenarios. Ändere den Code deiner Akteur-Klasse, sodass dein Sound abgespielt wird, wenn der Akteur angeklickt wird.

Übung 10.21 Ändere deinen Code so, dass ein Sound abgespielt wird, wenn du mit der linken Maustaste klickst, und ein anderer, wenn du mit der rechten Maustaste klickst. Dazu musst du erst Informationen einholen, welche Taste beim Klicken mit der Maus gedrückt wurde. Greenfoot stellt dir hierfür Methoden zur Verfügung – konsultiere die Greenfoot-Klassendokumentation, um Näheres darüber herauszufinden.

Übung 10.22 Ändere deinen Code so, dass der Akteur beim Anklicken einen Sound abspielt und sich zu einer neuen Zufallsposition bewegt.

Sounddateien können schnell sehr groß werden. Dies stellt kein besonderes Problem dar, solange das Szenario nur lokal verwendet wird, Aber wenn das Szenario exportiert wird, zum Beispiel auf die Greenfoot-Website, dann kann die Größe eine wichtige Rolle spielen. Sound- und Bilddateien machen in der Regel den größten Teil eines Greenfoot-Szenarios aus und die Größe der Sounddateien wird die Zeit zum Herunterladen eines Szenarios beeinflussen.

Konzept

Das **Sampleformat**, die **Samplefrequenz** und die **Stereo/Mono**-Einstellung einer Soundaufnahme bestimmen die Dateigröße und die Tonqualität.

Um zu vermeiden, dass unsere Sounddateien zu groß werden, sollten wir auf unsere Codiereinstellungen achten. Beim Aufnehmen und Speichern von Sounds müssen wir einen Kompromiss zwischen Dateigröße und Tonqualität finden. Wir können den Sound entweder in einer sehr hohen Qualität speichern, mit der Folge, dass die Dateien sehr groß werden, oder in einer schlechteren Qualität, was zu kleineren Dateien führt. Die wichtigsten Einstellungen hierbei sind

- das Sampleformat (normalerweise 16 Bit, 24 Bit oder 32 Bit)
- die Samplefrequenz gemessen in Hertz (Hz), die normalerweise von 8.000 bis 96.000 Hz reicht
- Stereo- kontra Mono-Aufnahme (Stereo-Aufnahmen umfassen zwei getrennte Spuren und produzieren deshalb doppelt so viel Daten).

Wenn du jetzt Abbildung 10.1 erneut betrachtest, wirst du feststellen, dass der Sound in diesem Screenshot mit 16 Bit, 22.050 Hz und in Stereo aufgenommen wurde.

Manchmal sind die Standardeinstellungen für Soundaufnahmeprogramme sogar noch höher (zum Beispiel 32 Bit und eine höhere Samplefrequenz). Dies ist eine viel höhere Aufnahmequalität als für einfache Soundeffekte benötigt wird. (Eine solche Qualität benötigen wir allenfalls für die Aufnahme klassischer Musik oder Ähnlichem, aber nicht für kurze Soundeffekte wie *Peng*!) Im Allgemeinen empfiehlt es sich deshalb, die Sounds in niedrigerer Qualität abzuspeichern, es sei denn, du benötigst wirklich eine höhere Qualität.

Übung 10.23 Finde heraus, wie du in deinem Soundaufnahmeprogramm Sampleformat, Samplefrequenz und Stereo/Mono-Aufnahmen einstellst. In einigen Programmen kannst du bestehende Sounds umwandeln. In anderen Programmen kannst du diese Einstellungen nur bei einer Neuaufnahme vornehmen. Versuche einmal, deinen Sound mit verschiedenen Sampleformaten, Samplefrequenzen und in Stereo und Mono aufzunehmen. Speichere deinen Sound in jeweils eigenen Dateien und vergleiche die Dateigrößen. Lege eine Tabelle mit Dateigrößen für unterschiedliche Einstellungen an. Welche Änderung hat die größte Auswirkung auf die Dateigröße?

Übung 10.24 Höre dir die in der vorstehenden Übung erzeugten Sounds an. Kannst du einen Unterschied feststellen? Wie weit kannst du die Qualität (und damit die Dateigröße) reduzieren, ohne dass die Aufnahme zu sehr darunter leidet.

10.6 Erweiterte Steuerung: die Klasse GreenfootSound

Bis hierhin waren unser Umgang mit Sound sehr einfach: wir haben eine Sounddatei abgespielt. Dies ist für kurze Soundeffekte ausreichend und die **`playSound`**-Methode von Greenfoot ist eine gute Möglichkeit dazu.

Manchmal benötigen wir jedoch mehr Kontrolle über den Sound. Wenn wir Hintergrundmusik einsetzen oder wenn wir einen MP3-Player schreiben, müssen wir Sounds starten und anhalten können, die Lautstärke verändern können und so weiter. Greenfoot hat eine eigene Klasse dafür: **`GreenfootSound`**.

Übung 10.25 Suche die Dokumentation für die Klasse **`GreenfootSound`** in der Klassendokumentation von Greenfoot. Untersuche sie. Welcher Wertebereich für die Lautstärke wird akzeptiert?

Übung 10.26 Verändere deinen Code, sodass er nun ein **`GreenfootSound`**-Objekt für den Sound verwendet. Dieses Sound-Objekt sollte im Konstruktor deines Akteurs erzeugt und in einer Instanzvariablen gespeichert werden. Wenn du dann auf deinen Akteur klickst, sollte dieser Sound abgespielt werden.

Übung 10.27 Ändere deinen Code noch einmal, sodass er nun mit dem Abspielen deines Soundeffekts in einer Schleife beginnt, wenn du auf deinen Akteur klickst, und mit dem Abspielen pausiert, wenn du noch einmal klickst. Du kannst dann den Sound wiederholt abspielen und pausieren lassen, indem du mehrere Male auf deinen Akteur klickst.

Mehr Übungen zur Sound-Klasse findest du am Ende dieses Kapitels im Abschnitt *Vertiefenden Aufgaben*.

10.7 Mit Bildern arbeiten

Wie bereits in früheren Kapiteln kurz angesprochen (z.B. als wir den Asteroiden-Hintergrund in ***Kapitel 9*** erstellt haben), gibt es zwei Möglichkeiten, um sich Bilder für Akteure und Hintergründe zu besorgen: Wir können bereits fertige Bilder aus Dateien verwenden oder selbst Bilder in einem eigenen Programm zeichnen.

Beide Methoden wollen wir hier ein wenig näher beleuchten.

10.8 Bilddateien und Bildformate

Es gibt mehrere Möglichkeiten, an Bilddateien für unsere Szenarien zu gelangen. Am leichtesten ist es natürlich, die Bilder der Greenfoot-Bilderbibliothek zu verwenden. Diese werden automatisch angezeigt, wenn wir neue Klassen erzeugen. Darüber hinaus gibt es einige gute Bibliotheken im Internet, in denen du kostenlose Icons und Bilder finden kannst. (Achte aber darauf, dass die Bilder, die du verwenden willst, auch tatsächlich zur freien Verfügung stehen – nicht alles, was es im Internet gibt, ist kostenlos oder lizenzfrei. Respektiere die Urheberrechte und Lizenzbedingungen anderer.)

Die interessanteste Alternative besteht jedoch darin, unsere Bilder selbst zu erstellen und so Szenarien zu entwickeln, die einzigartig und unverwechselbar sind.

Es gibt eine ganze Reihe von Grafikprogrammen, mit denen wir Bilder erzeugen können. *Photoshop* ist vielleicht das bekannteste kommerzielle Programm und mit Sicherheit ein sehr gutes. Doch nicht jeder hat das Glück, im Besitz dieses Programms zu sein. Es gibt jedoch auch kostenlose Open-Source-Software, die eine ähnliche Funktionalität bereitstellt. *Gimp*[2] ist ein exzellentes kostenloses Programm, das über viele Features verfügt und die Mühe der Installation lohnt. Du findest aber auch viele noch einfachere Programme, die du verwenden kannst.

Das Erstellen gut aussehender, attraktiver Grafiken erlernt man nicht von heute auf morgen. Im Rahmen dieses Buches können wir deshalb nicht ausführlich darauf eingehen. Durch ein wenig Herumspielen und viel Üben wirst du viele Tech-

2 *http://www.gimp.org*

niken und Tricks selbst herausfinden. Hier wollen wir uns auf die technischen Aspekte der Verwendung dieser Bilder konzentrieren.

Wichtig ist die Frage zu klären, welche Dateiformate beim Speichern der Bilder zu verwenden sind. Wie schon bei den Sounds gilt es zwischen Qualität und Dateigröße abzuwägen. Bilddateien sind unter Umständen sehr groß (viel größer als die Codedateien in unseren Szenarien), sodass sie schnell die gesamte Download-Größe unseres Projekts dominieren. Dies ist, wie zuvor, vor allem zu beachten, wenn wir unser Szenario auf einen Webserver wie die Greenfoot-Website exportieren wollen. Unterschiedliche Bildformate können Dateigrößen aufweisen, die sich um den Faktor 10 oder mehr unterscheiden, was zur Folge hat, dass sich ein Szenario 10-mal schneller herunterladen lässt (weil es nur ein Zehntel so groß ist), wenn wir die Formate gut gewählt haben.

Greenfoot kann Bilder lesen, die im JPEG-, PNG-, GIF-, BMP- und TIFF-Format vorliegen. Für die meisten Anwendungen dürften JPEG und PNG die am besten geeigneten Formate sein.

JPEG-Bilder haben den Vorteil, dass sie sich sehr gut komprimieren lassen. Das bedeutet, dass sie in einer sehr kleinen Dateigröße abgespeichert werden können. Dies gilt besonders für Farbbilder wie Fotos und Hintergründe (deshalb verwenden Digitalkameras dieses Format). Beim Speichern von JPEG-Bildern können wir bei vielen Grafikprogrammen wählen, wie stark wir die Datei komprimieren wollen. Je stärker wir komprimieren, desto kleiner wird die Datei – und desto schlechter die Qualität. Gimp zum Beispiel stellt beim Speichern von Bildern im JPEG-Format einen Schieberegler zur Verfügung, mit dem die Qualität eingestellt werden kann. Je mehr du die Qualität reduzierst, umso kleiner wird deine Datei. Der Qualitätsverlust ist besonders offensichtlich, wenn wir scharfe, kontrastreiche Kanten haben, wie es zum Beispiel bei geschriebenem Text oder bei Strichzeichnungen der Fall ist.

Konzept

Das Bildformat **JPEG** komprimiert große Bilder sehr gut. Es eignet sich deshalb besonders für Hintergrundbilder.

Übung 10.28 Erzeuge ein Bild in deinem Grafikprogramm und speichere es als JPEG-Datei. Speichere es in mindestens vier verschiedenen Qualitätseinstellungen. Öffne dann die verschiedenen Dateien nebeneinander und vergleiche sie. Vergleiche auch die Dateigrößen. Welche Qualitätseinstellung ist für dein Bild ein guter Kompromiss zwischen Bildqualität und Dateigröße?

Übung 10.29 Erstelle in deinem Grafikprogramm einen neuen Hintergrund für das Szenario, das du für die vorherigen Abschnitte dieses Kapitels erzeugt hast. Speichere ihn als JPEG-Datei und verwende ihn in deinem Szenario. Wie groß (Breite und Höhe) sollte das Bild sein? Was passiert, wenn es zu groß ist? Was, wenn es zu klein ist?

Übung 10.30 Wie gelingt es dem JPEG-Algorithmus deiner Meinung nach, Dateien kleiner zu machen? Wie funktioniert das? Versuche einmal, ein paar Theorien und Ideen hierfür zu entwickeln.

Übung 10.31 Wie komprimiert JPEG eigentlich Dateien? Recherchiere ein wenig im Internet und halte deine Antwort schriftlich fest.

Konzept

Pixel in einem Bild haben einen **Transparenzwert**, der festlegt, ob wir durch sie hindurchsehen können. Pixel können auch teilweise transparent sein. Wenn sie vollständig transparent sind, sind sie unsichtbar.

Das zweite Bildformat, das für uns sehr nützlich ist, lautet PNG.

PNG-Bilder haben den Vorteil, dass sie Transparenz sehr gut unterstützen. Jedes einzelne Pixel kann teilweise oder vollständig transparent sein. Dadurch ist es uns möglich, nicht rechteckige Bilder zu erzeugen. (Wie wir in ***Kapitel 9*** besprochen haben, sind Bilder rechteckig, doch durch die transparenten Anteile wird der Eindruck eines beliebigen Umrisses erweckt.)

Eine gute Komprimierung zusammen mit der Fähigkeit, Transparenzinformationen zu verarbeiten, macht PNG zu einem idealen Format für Akteur-Bilder. (In JPEG-Bildern können keine transparenten Pixel dargestellt werden, sodass dieses Format hier nicht verwendet werden kann, es sei denn, der Akteur ist zufällig rechteckig. Für Hintergründe stellt dies im Allgemeinen kein Problem dar, da wir hier normalerweise keine Transparenz einsetzen.)

Konzept

Das Bildformat **PNG** ist oft die beste Wahl für Akteur-Bilder, da es Transparenzinformationen speichert und sich relativ gut komprimieren lässt.

Es besteht selten die Notwendigkeit, BMP-, TIFF- oder GIF-Bilder zu verwenden. BMP lässt sich nicht so gut komprimieren wie die anderen Formate und unterstützt auch keine transparenten Pixel. TIFF-Bilder wahren zwar recht gut die Qualität, gehen aber einher mit großen Dateigrößen. GIF ist ein geschütztes Format, das erfolgreich durch das bessere – und kostenlose – PNG-Format ersetzt wurde.

Übung 10.32 Erstelle zwei neue Bilder für deinen Akteur (der Akteur wird zwischen diesen beiden Bildern hin- und herwechseln). Speichere sie im PNG-Format. Mache eines dieser Bilder zum Standardbild für deine Akteur-Klasse.

Übung 10.33 Ändere deinen Akteur-Code, sodass er zwischen den beiden Bildern hin- und herschaltet, wenn der Akteur mit der Maus angeklickt wird.

Übung 10.34 Ändere deinen Akteur-Code erneut, sodass er das zweite Akteur-Bild nur anzeigt, wenn die Maus gedrückt gehalten wird. Mit anderen Worten, der Akteur soll zunächst sein Standardbild zeigen; wenn dann die Maus auf dem Akteur gedrückt gehalten wird, zeigt er das zweite Bild, und sobald die Maustaste losgelassen wird, erscheint das Ursprungsbild wieder.

10.9 Bilder zeichnen

Die zweite Möglichkeit, Bilder für unsere Akteure und Hintergründe zu erzeugen, ist sie direkt im Quelltext zu zeichnen. Beispiele hierfür haben wir bereits in einigen der früheren Szenarien gesehen, z.B. als wir die Sterne in dem Asteroiden-Programm gemalt haben. Jedes Pixel in einem Bild wird durch zwei Werte definiert: seinen Farbwert und seinen Transparenzwert (auch *Alpha-Wert* genannt).

Der Farbwert besteht wiederum aus drei Komponenten: dem Rot-, Grün- und Blauanteil.[3] Farben, die auf diese Weise dargestellt werden, heißen normalerweise auch RGB-Farben.

3 Dieses Modell zur Repräsentation von Farben ist nur eines von mehreren. Es gibt andere, die in der Computergrafik oder im Druckwesen Verwendung finden. Da dieses Modell jedoch in der Java-Programmierung am weitesten verbreitet ist, werden wir uns hier allein auf dieses Modell konzentrieren.

Das bedeutet, dass wir ein Pixel (mitsamt Farben und Transparenz) durch vier Zahlen darstellen können. Die Reihenfolge lautet normalerweise:

(R, G, B, A)

Das bedeutet, die ersten drei Werte definieren die rote, grüne und blaue Komponente (in dieser Reihenfolge) und der letzte Wert ist der Alpha-Wert.

In Java liegen all diese Werte im Bereich [0..255] (null bis 255 einschließlich). Ein Farbwert von 0 zeigt an, dass in dieser Komponente keine Farbe vorhanden ist, während 255 die Farbe in voller Stärke angibt. Ein Alpha-Wert von 0 ist voll transparent (unsichtbar), während 255 opak (100% deckend) ist.

Abbildung 10.2 zeigt tabellarisch einige mögliche Farben (ohne Transparenz, d.h. alpha = 255). Die Tabelle in Abbildung 10.2 (siehe auch Anhang E) wurde mit dem Greenfoot-Szenario *color-chart* erzeugt, das du in dem Ordner *Kapitel10* der Buchszenarien findest.

0,0,0	0,0,51	0,0,102	0,0,153	0,0,204	0,0,255
0,51,0	0,51,51	0,51,102	0,51,153	0,51,204	0,51,255
0,102,0	0,102,51	0,102,102	0,102,153	0,102,204	0,102,255
0,153,0	0,153,51	0,153,102	0,153,153	0,153,204	0,153,255
0,204,0	0,204,51	0,204,102	0,204,153	0,204,204	0,204,255
0,255,0	0,255,51	0,255,102	0,255,153	0,255,204	0,255,255
51,0,0	51,0,51	51,0,102	51,0,153	51,0,204	51,0,255
51,51,0	51,51,51	51,51,102	51,51,153	51,51,204	51,51,255
51,102,0	51,102,51	51,102,102	51,102,153	51,102,204	51,102,255
51,153,0	51,153,51	51,153,102	51,153,153	51,153,204	51,153,255
51,204,0	51,204,51	51,204,102	51,204,153	51,204,204	51,204,255
51,255,0	51,255,51	51,255,102	51,255,153	51,255,204	51,255,255
102,0,0	102,0,51	102,0,102	102,0,153	102,0,204	102,0,255
102,51,0	102,51,51	102,51,102	102,51,153	102,51,204	102,51,255
102,102,0	102,102,51	102,102,102	102,102,153	102,102,204	102,102,255
102,153,0	102,153,51	102,153,102	102,153,153	102,153,204	102,153,255
102,204,0	102,204,51	102,204,102	102,204,153	102,204,204	102,204,255
102,255,0	102,255,51	102,255,102	102,255,153	102,255,204	102,255,255
153,0,0	153,0,51	153,0,102	153,0,153	153,0,204	153,0,255
153,51,0	153,51,51	153,51,102	153,51,153	153,51,204	153,51,255
153,102,0	153,102,51	153,102,102	153,102,153	153,102,204	153,102,255
153,153,0	153,153,51	153,153,102	153,153,153	153,153,204	153,153,255
153,204,0	153,204,51	153,204,102	153,204,153	153,204,204	153,204,255
153,255,0	153,255,51	153,255,102	153,255,153	153,255,204	153,255,255
204,0,0	204,0,51	204,0,102	204,0,153	204,0,204	204,0,255
204,51,0	204,51,51	204,51,102	204,51,153	204,51,204	204,51,255
204,102,0	204,102,51	204,102,102	204,102,153	204,102,204	204,102,255
204,153,0	204,153,51	204,153,102	204,153,153	204,153,204	204,153,255
204,204,0	204,204,51	204,204,102	204,204,153	204,204,204	204,204,255
204,255,0	204,255,51	204,255,102	204,255,153	204,255,204	204,255,255
255,0,0	255,0,51	255,0,102	255,0,153	255,0,204	255,0,255
255,51,0	255,51,51	255,51,102	255,51,153	255,51,204	255,51,255
255,102,0	255,102,51	255,102,102	255,102,153	255,102,204	255,102,255
255,153,0	255,153,51	255,153,102	255,153,153	255,153,204	255,153,255
255,204,0	255,204,51	255,204,102	255,204,153	255,204,204	255,204,255
255,255,0	255,255,51	255,255,102	255,255,153	255,255,204	255,255,255

Abbildung 10.2
Die RGB-Farbtabelle.

Übung 10.35 Wie sehen die Pixel aus, die einen Farb-/Alpha-Wert von (255, 0, 0, 255) haben? Oder von (0, 0, 255, 128)? Von (255, 0 255, 230)?

In Greenfoot werden Farben durch Objekte der Klasse **`Color`** aus dem Paket **`java.awt`** repräsentiert. Nach dem Importieren dieser Klasse können wir Farbobjekte entweder mit RGB-Werten erzeugen:

```
Color mycol = new Color (255, 12, 34);
```

oder mit RGB-Werten und einem Alpha-Wert:

```
Color mycol = new Color (255, 12, 34, 128);
```

Wenn wir keinen Alpha-Wert angeben, ist die Farbe vollständig deckend.

Übung 10.36 Erstelle ein neues Greenfoot-Szenario namens *color-test*. Erzeuge darin eine Welt mit einem Hintergrund, der ein Muster anzeigt. Erzeuge eine Unterklasse von **`Actor`** namens **`ColorPatch`**. Programmiere die Klasse **`ColorPatch`** so, dass sie ein neues **`GreenfootImage`**-Objekt fester Größe gefüllt mit einer festen Farbe erzeugt. Verwende dieses Bild als das Bild des Akteurs. Experimentiere mit verschiedenen Farb- und Alpha-Werten.

Übung 10.37 Ändere deinen Code, sodass der Farbfleck bei seiner Erzeugung ein Bild von beliebiger Größe, gefüllt mit einer beliebigen Farbe von beliebiger Transparenz erhält.

Übung 10.38 Ändere deinen Code erneut, sodass das Bild nicht gefüllt ist, sondern darin 100 kleine farbige Punkte an zufälligen Positionen gezeichnet werden.

10.10 Bilddateien und dynamisches Zeichnen kombinieren

Die wahrscheinlich interessantesten visuellen Effekte erzielst du, wenn du statische Bilder aus Dateien mit dynamischen Änderungen kombinierst, die über das Programm vorgenommen werden. Wir können zum Beispiel eine statische Bilddatei als Ausgangsbasis nehmen und dann darauf mit verschiedenen Farben malen, sie vergrößern oder verkleinern oder sie durch Verändern des Transparenzwertes verblassen lassen.

Dies wollen wir anhand eines Raucheffekts veranschaulichen. In unserem nächsten Szenario bewegen wir einen Ball über den Bildschirm, der eine Rauchfahne hinter sich herzieht (siehe Abbildung 10.3).

Übung 10.39 Erzeuge ein neues Szenario namens *smoke* und erstelle dafür einen relativ neutralen Hintergrund.

Übung 10.40 Erzeuge einen Ball, der sich mit konstanter Geschwindigkeit über den Bildschirm bewegt. Wenn er den Rand berührt, prallt er davon ab. (Der Bildschirm ist mehr oder weniger ein Kasten, in dem der Ball hin- und herspringt.)

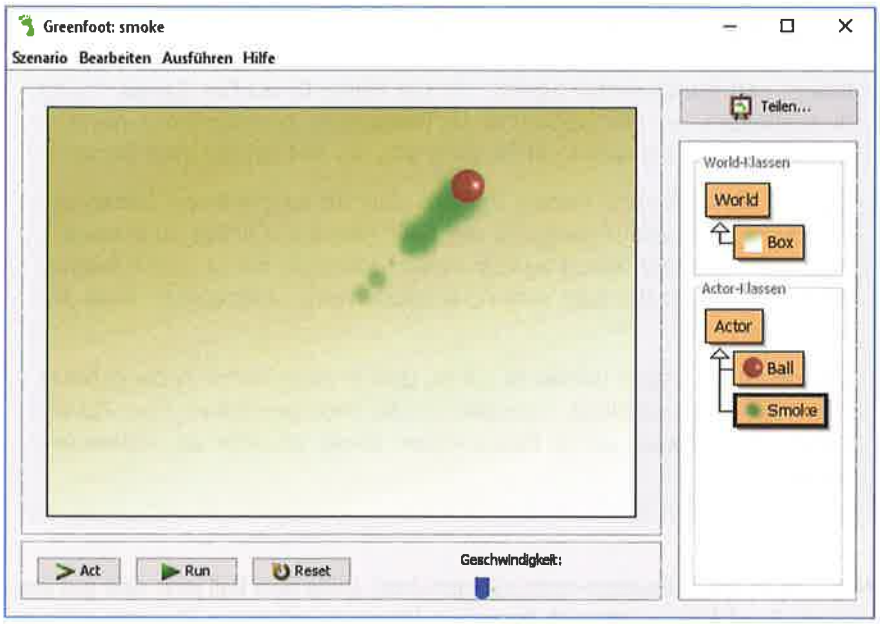

Abbildung 10.3
Der Rauchfahnen-Effekt.

Jetzt wollen wir uns dem Rauchfahneneffekt zuwenden. Dazu erzeugen wir zuerst eine kleine Rauchwolke (Abbildung 10.4).

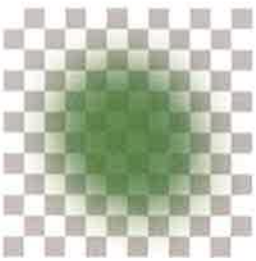

Abbildung 10.4
Das Bild einer kleinen Rauchwolke.

Beachte, dass der schachbrettartige Hintergrund nicht Teil des Bildes ist – damit soll nur veranschaulicht werden, dass die Rauchwolke halbtransparent ist (wir können den Hintergrund dahinter erkennen).

Dein Rauch muss nicht grün sein – du kannst jede andere beliebige Farbe wählen, sofern sie transparent ist. In einem guten Grafikprogramm musst du dazu lediglich einen Punkt mit einem großen, weichen, halbtransparenten Farbpinsel setzen. Wenn du Schwierigkeiten hast, dieses Bild zu erzeugen, kannst du die bereits vorhandene Bilddatei *smoke.png* aus dem Ordner *smoke* zu diesen Buchszenarien verwenden.

Übung 10.41 Erzeuge das oben beschriebene Bild der Rauchwolke.

Übung 10.42 Erzeuge in deinem Szenario eine Akteur-Klasse **`Smoke`**. Nach dem Einfügen in die Welt sollte der **`Smoke`**-Akteur relativ schnell ausgeblendet werden. Das bedeutet, in jedem **`act`**-Schritt sollte das Bild der Akteurs kleiner und transparenter werden. (In der Klasse **`GreenfootImage`** findest du Methoden, um die Größe und die Transparenz zu verändern.) Wenn es volltransparent oder sehr klein ist, sollte sich das Bild aus der Welt löschen.

Übung 10.43 Ändere deinen Ball so, dass er Rauchfahnen hinter sich zurücklässt. Bei jeder Bewegung des Balls eine Rauchwolke zu erzeugen, wäre wahrscheinlich etwas zu viel: Versuche einmal, nur in jedem zweiten Bewegungsschritt des Balls eine neue Rauchwolke zu erzeugen. Teste deinen Code.

Übung 10.44 Ändere deinen Rauch so, dass er nicht immer in der gleichen Geschwindigkeit verblasst. Integriere in die Geschwindigkeit eine Zufallskomponente, sodass einige Rauchwolken etwas schneller als andere verblassen.

Wenn du all diese Übungen nachvollzogen hast, sollte dein Ball jetzt eine gut aussehende Rauchfahne hinter sich herziehen. Wenn du mit diesen Übungen Schwierigkeiten hattest oder wenn du deine Lösung mit unserer vergleichen möchtest, kannst du einen Blick in das *smoke*-Szenario zu diesem Kapitel werfen.

Zusammenfassung der Programmiertechniken

Das Erzeugen eigener Sounds und Bilder ist unerlässlich für alle, die attraktive Spiele, Simulationen und andere grafische Anwendungen entwickeln wollen. Darüber hinaus sollte man über Kenntnisse der Sound- und Bilddateiformate verfügen, um genau zwischen Dateigröße und -qualität abwägen zu können.

Sounds können direkt in Greenfoot aufgenommen oder mit einer Vielzahl von Soundaufnahmeprogrammen aufgenommen und bearbeitet werden, wobei. Verschiedene Parameter bestimmen die Tonqualität und die Dateigröße. Für Greenfoot-Szenarien verwenden wir in der Regel das Format WAV mit relativ geringer Qualität.

Auch Bilder können in vielen verschiedenen Formaten abgespeichert werden. Die einzelnen Formate unterscheiden sich darin, wie gut sie Dateien komprimieren, die Bildqualität wahren und Transparenzinformationen speichern. JPEG und PNG sind die am häufigsten verwendeten Formate in Greenfoot-Szenarien.

Durch die Kombination von Bildern aus einer Datei mit dynamischen Bildoperationen (wie Verkleinern/Vergrößern und Transparenzänderungen) können wir in unseren Szenarien attraktive visuelle Effekte erzielen.

Zusammenfassung der Konzepte

- Wir können die Methode **`mouseClicked`** verwenden, um zu prüfen, ob der Benutzer ein bestimmtes Objekt angeklickt hat.
- Sounds können in einer Vielzahl von verschiedenen **Formaten** und **Codierungen** abgespeichert werden. Nicht alle Programme können alle Soundformate spielen. Für Greenfoot verwenden wir in der Regel das Format **WAV**.
- Das **Sampleformat**, die **Samplefrequenz** und die **Stereo/Mono**-Einstellung einer Soundaufnahme bestimmen die Dateigröße und die Tonqualität.
- Das Bildformat **JPEG** komprimiert große Bilder sehr gut. Es eignet sich deshalb besonders für Hintergrundbilder.
- Pixel in einem Bild haben einen **Transparenzwert**, der festlegt, ob wir durch sie hindurchsehen können. Pixel können auch teilweise transparent sein. Wenn sie vollständig transparent sind, sind sie unsichtbar.
- Das Bildformat **PNG** ist oft die beste Wahl für Akteur-Bilder, da es Transparenzinformationen speichert und sich relativ gut komprimieren lässt.

Vertiefende Aufgaben

Hier findest du einige Aufgaben, um deine Fertigkeiten im Bereich Sound- und Bildbearbeitung einzuüben.

Musik abspielen

Übung 10.45 Erstelle ein neues Szenario für einen (sehr einfachen) MP3-Player. Erzeuge einen Akteur für eine Starttaste. Speichere eine MP3-Datei in dem Sound-Ordner des Szenarios. Programmiere den Player so, dass beim Anklicken der Starttaste die MP3-Datei abgespielt wird. Ein weiterer Tastenklick stoppt das Abspielen der Datei.

Übung 10.46 Stelle sicher, dass sich das Bild der Taste verändert: Es sollte das typische „Abspielen"-Dreieck angezeigt werden, wenn kein Sound abgespielt wird, und dann zum typischen „Pause"-Zeichen (zwei vertikale Balken) wechseln, während der Sound läuft.

Übung 10.47 Füge eine Stopptaste hinzu. Diese Taste sollte das Abspielen des Sounds beenden. Wenn du die Taste dann erneut drückst, beginnt der Sound wieder am Anfang (nicht von der Stelle, an der sie gestoppt wurde). Achte darauf, dass sich das Bild der Taste entsprechend ändert, wenn das Abspielen beendet ist.

Übung 10.48 Füge zwei Tasten hinzu, mit denen man die Lautstärke lauter und leiser stellen kann.

Übung 10.49 Füge eine Anzeige hinzu, die die aktuell eingestellte Lautstärke anzeigt.

Übung 10.50 Füge drei Tasten hinzu, mit denen man drei unterschiedliche Songs auswählen kann (die du in deinem Sound-Ordner ablegen musst).

Bilder bearbeiten

Übung 10.51 Erstelle eine Kopie deines *fatcat*-Szenarios aus ***Kapitel 2*** und öffne es. Ändere den Konstruktor von `MyCat`, sodass die Größe der Katze nun zufällig gewählt wird (wobei die aktuelle Gesamtgröße das Maximum darstellt). Verkleinere dazu im Konstruktor das Bild der Katze um einen zufälligen Wert. Teste den Code, indem du ein paar Katzen in der Welt platzierst.

Übung 10.52 Schreibe die `act`-Methode um, sodass die Katze langsam verblasst, wenn du das Szenario ausführst (so wie die Grinsekatze in *Alice im Wunderland*).

Übung 10.53 Verändere deine Katze so, dass sie nur verblasst, nachdem du sie angeklickt hast.

Für die nächsten Übungen benötigen wir das Szenario *path-follower* aus dem *Kapitel10*-Ordner. Öffne das Szenario. Es zeigt ein Lebewesen (ein *Greep*), der einem Pfad folgen soll. Dazu untersucht es die Farbe des Untergrunds: Wenn der Boden bräunlich ist, dann ist es ein Pfad, ansonsten ist es kein Pfad.

Im Moment ist das Szenario noch unvollständig. (Probiere es aus!) Ein Großteil des Codes ist implementiert, aber die letzten Feinheiten, bei denen es um die Farben geht, fehlen noch. Das wird nun deine Aufgabe sein.

Wir werden das Verfolgen eines Pfads implementieren, indem wir uns die Farbe des Welthintergrunds an der Position der beiden Augen vom Greep geben lassen. Wenn wir mit dem linken Auge etwas Grünes sehen, dann drehen wir uns ein wenig nach rechts, und wenn wir mit dem rechten Auge etwas Grünes sehen, dann drehen wir uns nach links. Auf diese Weise bleiben wir auf dem Pfad.

In drei Methoden der Klasse **Greep** fehlt noch Code: in **`leftEyeColor`**, in **`rightEyeColor`** und in **`isPath`**. Alles andere ist fertig.

Wir beginnen mit der Methode **`leftEyeColor`**. Diese Methode soll die Farbe des Welthintergrunds an der Position des linken Auges (als ein Objekt vom Typ **`Color`**) zurückgeben.

Suche die Methode. Du wirst feststellen, dass die Methode bereits die Koordinaten des linken Auges erhält. Diese werden in einem Objekt der Klasse **`Point`** gespeichert und wir bekommen die *x*- und *y*-Koordinaten des Auges, indem wir Methoden auf dem **`Point`**-Objekt aufrufen.

Du musst noch Folgendes tun:

- Zugriff auf die Welt erlangen
- das Bild des Welthintergrunds von der Welt holen
- die *x*- und *y*-Koordinaten des Auges vom **`Point`**-Objekt abfragen
- die Farbe des Hintergrundbilds an der Position des Auges bekommen
- diese Farbe zurückgeben

Eventuell musst du die Dokumentation von **`World`** und **`GreenfootImage`** hinzuziehen und die Namen der Methoden herausfinden, um diese Aufgaben auszuführen.

Farben lesen

Übung 10.54 Vervollständige den Code der Methode **`leftEyeColor`**, sodass sie die Farbe unter dem linken Auge zurückgibt.

Übung 10.55 Vervollständige den Code der Methode **`rightEyeColor`**, sodass sie die Farbe unter dem rechten Auge zurückgibt.

Übung 10.56 Vervollständige den Code der Methode **`isPath`**, sodass sie genau dann **`true`** zurückgibt, wenn die angegebene Farbe zum Pfad gehört und nicht zum Gras. Dazu kannst du den Rotanteil der Farbe untersuchen: Ist der Wert für Rot größer als 160 ist, dann kannst du annehmen, dass wir einen Pfad angucken und kein Gras. Sieh in der Dokumentation der Klasse **`Color`** nach, wie man den Rotanteil der Farbe herausfindet.