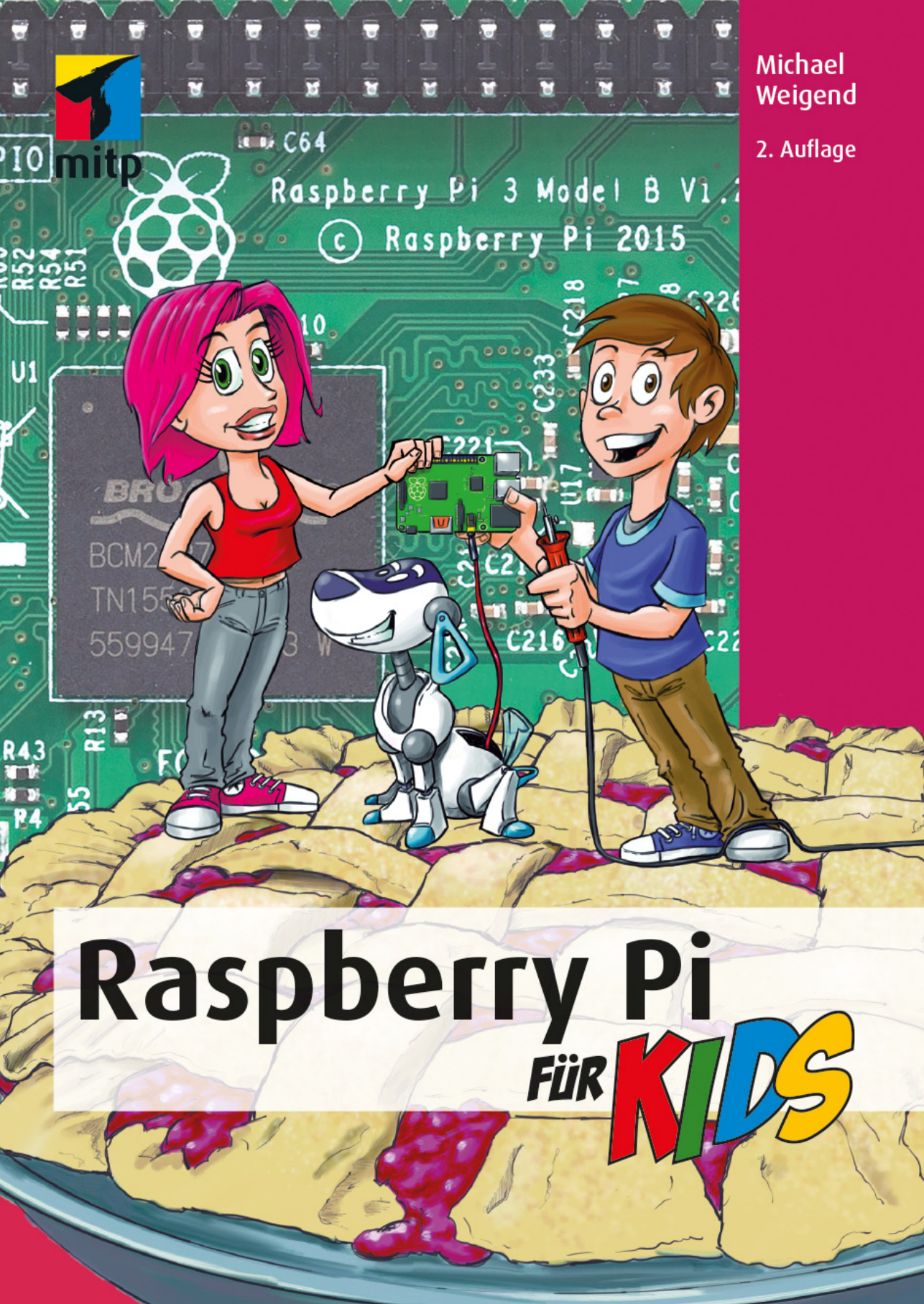
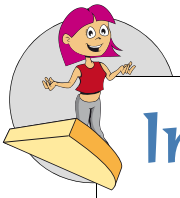

mitp
Raspberry Pi 3 Model B
© Raspberry Pi 2015
Michael Weigend
2. Auflage
Raspberry Pi
FÜR KIDS

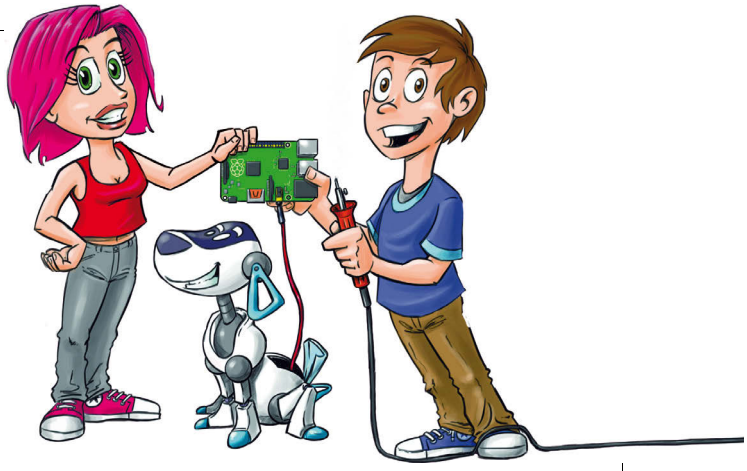


Inhalt

A

B

C



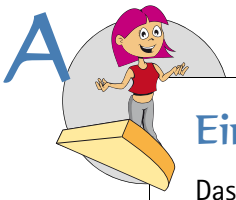
A

Scratch im Dienst der Wissenschaft

Computer spielen in den Naturwissenschaften eine wichtige Rolle. Man verwendet sie vor allem für Simulationen und zur Erfassung und Verarbeitung von Messwerten. In diesem Kapitel werden einige typische Beispiele vorgestellt. Wir simulieren eine Kellerassel, die Schutz in der Dunkelheit sucht, einen Roboter, der ein Loch in einem Kanal findet, eine Mondlandung und einen Gaschromatographen. Mit dem Picoboard und einer Partyleuchte machst du aus deinem Raspberry Pi ein Spektralphotometer zur Konzentrationsbestimmung von Farbstoffen.

Die digitale Kellerassel

Kennst du Asseln? Du findest diese kleinen krebsartigen Tiere draußen im Garten unter altem Holz oder Laub. Sie haben eine ovale Form, sind ziemlich platt, besitzen sieben Beinpaare und vorne lange Fühler. Asseln lieben die Dunkelheit. Wenn eine Assel dem Tageslicht ausgesetzt ist, krabbelt sie wild herum und sucht eine Stelle, wo es dunkel und feucht ist. Genau dieses Verhalten versuchen wir zu simulieren. Wir entwickeln mit Scratch in zwei Schritten eine digitale Assel, die laufen kann (Schritt 1) und die sich vorzugsweise unter einem Blatt aufhält (Schritt 2).



Eine krabbelnde Assel

Das Ziel des ersten Schritts ist eine Assel, die auf einer unregelmäßigen, zufälligen Zickzackbahn über den Bildschirm krabbelt. Wenn sie an den Rand stößt, prallt sie ab.

Abb. A.1: *Wenn die Assel an den Rand stößt, prallt sie ab.*

Die Kostüme

Starte Scratch und lösche die Katze. Klicke auf NEUES OBJEKT MALEN. Es öffnet sich das Fenster des Malprogramms. Male mit dem Ellipse-Werkzeug und dem Pinsel eine Assel. Auf dem Kopf sind zwei Fühler in einer besonderen Farbe (z.B. orange), die sonst nicht in deiner Assel vorkommt. Die Farbe hat eine besondere Bedeutung. Dazu später mehr. Wichtig ist, dass der Kopf mit den Fühlern nach rechts zeigt; denn das ist die Bewegungsrichtung.

In Wirklichkeit sind Asseln grau und haben sieben Beinpaare. Aber ich denke, ein bisschen künstlerische Freiheit ist erlaubt. Klicke auf OK, wenn die Assel fertig ist. Dann machst du eine Kopie des ersten Kostüms. Bearbeite die Kopie und verändere die Position der Beine ein wenig. Wenn man diese Kostüme abwechselnd zeigt, sieht es aus als würden sich die Beine bewegen und die Fühler wackeln.

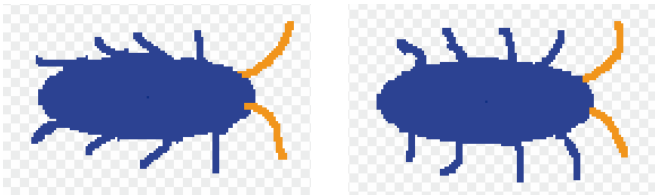


Abb. A.2: *Die beiden Kostüme der digitalen Assel.*

Gibt dem Objekt den Namen `Assel`.

Das Skript

Das Assel-Objekt hat nur ein einziges Skript, das die gesamte Bewegung bewerkstelligt.

Abb. A.3: *Das Skript der krabbelnden Assel*

So funktioniert's

1. Die Assel startet immer an derselben Stelle.
2. Die Assel dreht sich ein kleines bisschen nach rechts (wenn die Zufallszahl negativ ist) oder links. Hier wird für die unregelmäßige Zickzackbewegung gesorgt.
3. Die Assel soll vom Bildrand abprallen.
4. Hier wird für die Vorwärtsbewegung gesorgt. Durch den Kostümwechsel bewegen sich die Beine.

Suche nach einem Versteck

Im zweiten Schritt legen wir einige Blätter auf die Bühne. Die Assel soll sich nun so verhalten:

Wenn sie sich auf der freien grünen Fläche im vollen Sonnenlicht befindet, ist sie ganz aufgeregt und läuft wild herum. Sobald sie aber ein Blatt gefunden und sich darunter verstecken kann, wird sie langsam und ruhig. Sie bewegt sich die ganze Zeit, aber die meiste Zeit bleibt sie unter einem Blatt.

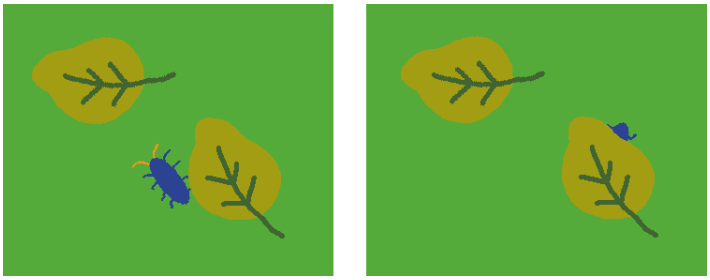


Abb. A.4: *Die Assel versteckt sich unter einem Blatt.*

Das Blatt

Zeichne ein neues Objekt, das ein Blatt darstellt. Mache von diesem Objekt eine oder mehrere Kopien und verteile sie auf der Bühne. Die Szenerie sieht interessanter aus, wenn du die Blätter etwas verdrehst. Das geht so: Über dem Reiter Skripte ist eine Miniaturansicht des Blatts. Darauf sitzt eine dünne blaue Linie, die die momentane Richtung des Objekts anzeigt (Abbildung A.5). Klicke das Ende der Linie mit der linken Maustaste an, halte die Maustaste gedrückt und drehe das Objekt.

Abb. A.5: *Ein Objekt drehen.*

Wichtig ist, dass die Blätter auf dem Bildschirm *vor* der Assel liegen, damit sich die Assel darunter verstecken kann. Das Objekt, das zuletzt auf der Bühne mit der Maus bewegt worden ist, ist immer ganz vorne. Um also ein Blatt nach vorne zu bringen, brauchst du es nur ein wenig zu verschieben. Oder aber du klickst auf den Baustein komme nach vorn in der Sammlung Aussehen.

Die Skripte für die Assel

Wie erreicht man, dass sich die Assel vorzugsweise unter einem Blatt aufhält? Überlegen wir zunächst einmal, wie sich die Assel in den beiden Situationen verhalten müsste, die in Abbildung A.6 dargestellt sind.

Abb. A.6: *Zwei Fälle: 1) Die Assel kommt unter einem Blatt hervor (links). 2) Die Assel hat ein Blatt gefunden (rechts).*

Erste Situation. Die Assel verlässt aus Versehen die schützende Deckung eines Blattes. Da sie sich zumindest teilweise nach dem Zufallsprinzip bewegt, kann das

schon einmal vorkommen. Jetzt müsste sie sich eigentlich umdrehen, um wieder zum Blatt zurückzukehren.

Zweite Situation. Nach langer Suche mit raschen Bewegungen und großen Richtungswechseln hat die Assel endlich ein Blatt gefunden. Sie hat es gerade mit ihrem Fühler berührt. Jetzt muss sie ihr Bewegungsmuster ändern. Bloß kein großer Richtungswechsel! Die Richtung stimmt, denn vor ihr liegt ein Blatt. Auch sollte sie jetzt langsamer werden. Denn über ihr ist ein schützendes Blatt, das sie nicht so schnell verlassen will. Hier will sie sich möglichst lange aufhalten.

Die Assel braucht also zwei Bewegungsmuster:

- Schnelle Bewegung (große Schritte) und große zufällige Richtungsänderungen.
- Langsame Bewegung (kleine Schritte) und geringe zufällige Richtungsänderung.
- Die Assel bewegt sich nach dem ersten Muster, wenn sie gerade im Freien ist. Spürt sie mit ihren Fühlern ein Blatt, bewegt sie sich nach dem zweiten Muster.
- Für jedes Bewegungsmuster schreibst du ein eigenes Skript. Je nachdem, ob die Fühler gerade das typische Grün eines Blattes erspüren oder nicht, wird das zweite bzw. das erste Muster angewendet.

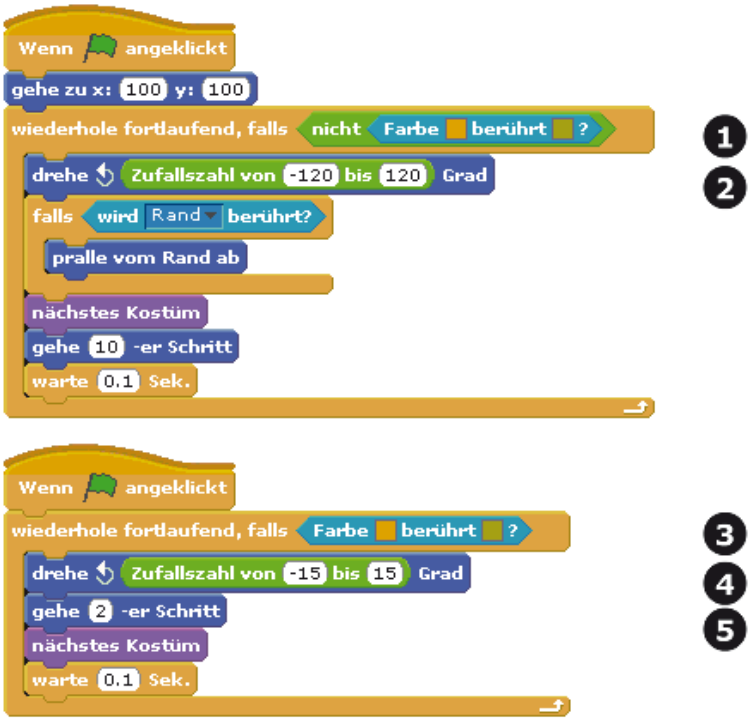
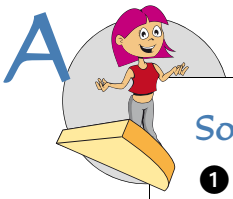


Abb. A.7: *Zwei Skripte für zwei unterschiedliche Bewegungsmuster.*



So funktioniert's

1. Mit dem Baustein 

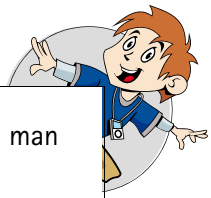
programmierst du den Farbsensor. Die erste Farbe ist die Farbe der Fühler, die zweite Farbe ist die grüne Farbe des Blattes. Um eine Farbe auszuwählen, klickst du zuerst auf die Farbe auf dem Programmbaustein. Dann sieht der Mauszeiger aus wie eine Pipette. Mit Pipetten kann man Farben aufsaugen. Wenn du nun die gewünschte Farbe auf der Bühne anklickst (das Orange der Fühler oder das Grün eines Blattes), übernimmt sie der Programmbaustein. Die restlichen Anweisungen des Skriptes werden nur ausgeführt, wenn die Assel *nicht* ein Blatt berührt.
2. Die Assel dreht sich hektisch um einen großen zufälligen Winkel nach rechts oder links (maximal 120 Grad).
3. Die folgenden Anweisungen werden ausgeführt, wenn die Assel unter einem Blatt ist.
4. Nur eine geringe zufällige Richtungsänderung (um maximal 15 Grad) findet statt.
5. Die Assel ist langsam. Sie bewegt sich nur um Zweierschritte nach vorne.

Wie funktioniert ein Gaschromatograph?

Mit einem Gaschromatographen finden Chemiker die Zusammensetzung von Stoffgemischen heraus. Der Gaschromatograph besteht aus einer langen, dünnen Röhre (auch Säule genannt), die mit einem porösen Material gefüllt ist. Durch die Röhre wird ein Gas gepumpt. Am Eingang spritzt man die Probe in den Gasstrom – z.B. Feuerzeuggas, das meist ein Gemisch aus Butan und Propan ist. Die Butanmoleküle kommen etwas langsamer durch die Säule als die kleineren Propanmoleküle. Am Ausgang der Säule ist ein Detektor. Er liefert ein Signal an einen Computer, der eine Messkurve erstellt.

Was zeigt die Animation?

Nun kann man in die Säule des Gaschromatographen nicht hineinsehen. Aber das was da drinnen passiert, kannst du in einem animierten Modell darstellen. Genau darum geht es in diesem Projekt. Abbildung A.8 zeigt Screenshots aus der Animation. Man sieht zwei unterschiedliche Arten von Molekülen: zwei langsame Butanmoleküle (modellhaft als Kreisflächen dargestellt) und ein schnelleres Propanmolekül (Quadrat). Sie wandern durch die Säule und trennen sich dabei allmählich in zwei Gruppen auf. Das Display des Detektors zeigt zunächst eine waagrechte Linie. Wenn Moleküle den Detektor treffen, entsteht ein »Hügel«



(Peak). Dieses Diagramm nennt man Gaschromatogramm. An den Peaks kann man erkennen, welche Stoffe in der untersuchten Probe enthalten sind.

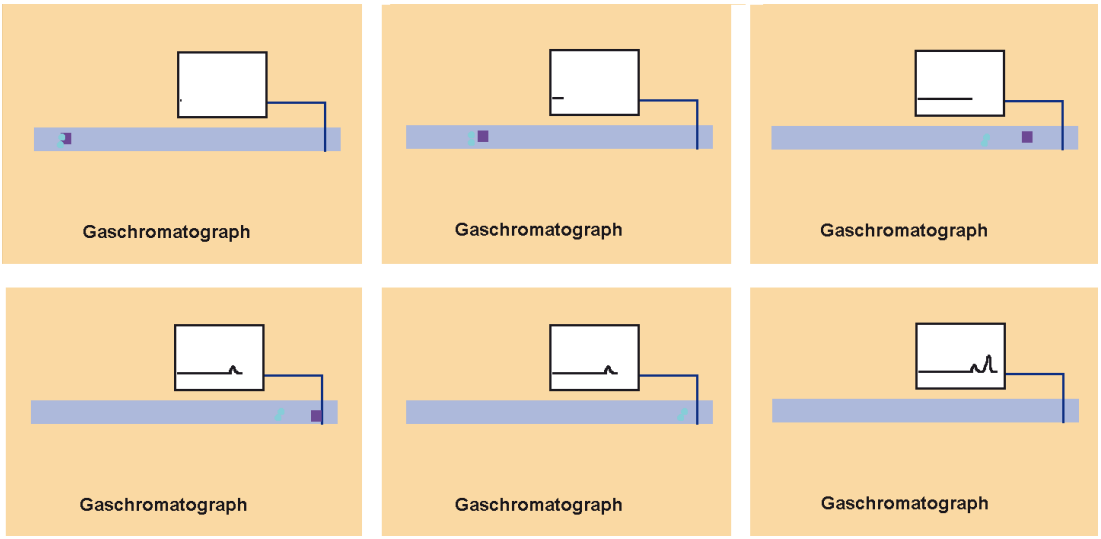


Abb. A.8: *Screenshots aus einem Scratch-Video, das erklärt, wie ein Gaschromatograph funktioniert*

Die Bühne

Das Projekt besteht aus dem Hintergrundbild der Bühne und vier Objekten. Das Objekt Kurve zeigt die Entwicklung des Chromatogramms. Die drei anderen Objekte stellen wandernde Moleküle dar (Propan, Butan1, Butan2)

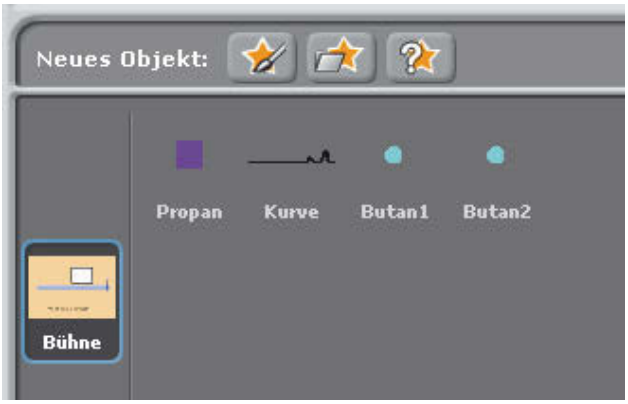


Abb. A.9: *Überblick: Die Objekte des Projekts*

Die Bühne besitzt kein Skript. Der Hintergrund der Bühne zeigt alles, was sich *nicht* verändert.

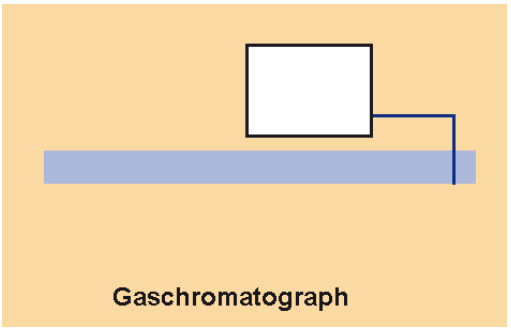


Abb. A.10: *Der Hintergrund.*

Das Diagramm

Das Diagramm wird vom Objekt Kurve mit vielen Kostümen erzeugt. Jedes Kostüm zeigt das Diagramm zu einem bestimmten Zeitpunkt. Das erste Kostüm erhält den Namen Start. Die Namen der anderen Kostüme sind unwichtig.

Abb. A.11: *Die Kostüme des Objekts* Kurve *zeigen die Entwicklung des Gaschromatogramms.*

Die gesamte Animation dauert sechs Sekunden. Nach dem Start des Programms wird das erste Kostüm gezeigt. Danach wird jede Sekunde das Bild gewechselt.

Abb. A.12: *Das Skript des Diagramm-Objekts.*

Die Moleküle

In diesem Beispiel gibt es drei Moleküle, die durch die Säule des Gaschromatographen wandern, zwei runde (etwas langsamer) und ein quadratisches (etwas schneller). Für jedes Molekül gibt es ein eigenes Objekt mit einem eigenen Kostüm und einem eigenen Skript. Die Skripte sind freilich sehr ähnlich.

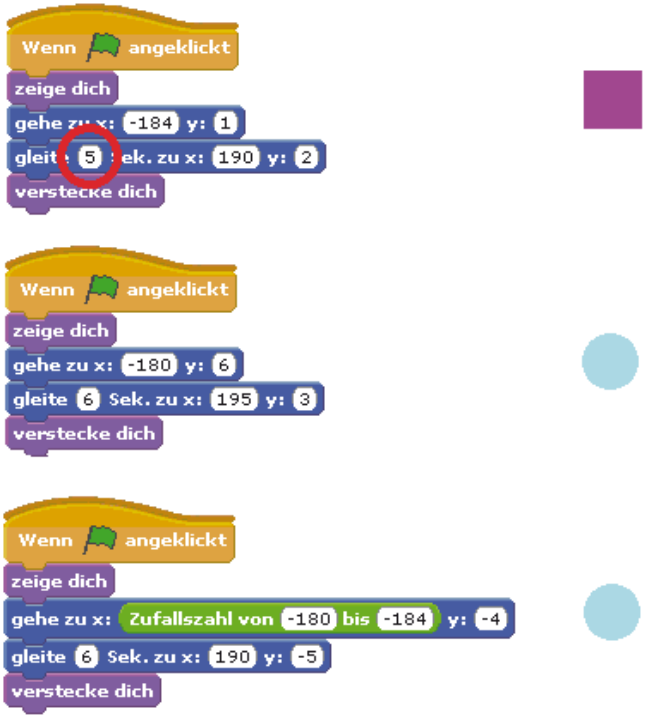


Abb. A.13: *Kostüme und Skripte der Moleküle*

Da die Moleküle am Ende versteckt werden, müssen sie sich zuerst sichtbar machen. Dann werden sie auf ihre Anfangsposition gesetzt. Die Anfangspositionen dürfen nicht genau gleich sein, damit man sieht, dass es mehrere Moleküle sind. Dann gleiten sie zu ihrer Endposition am Detektor des Gaschromatographen. Die Zeit ist beim schnelleren Propan etwas kleiner (5 Sekunden, siehe eingekreiste Stelle in Abbildung A.13) als beim langsameren Propan. Zum Schluss verschwinden die Moleküle.

Weitere Ideen

Animierte Modelle zur Erklärung von sichtbaren und unsichtbaren Vorgängen spielen in der Wissenschaft eine große Rolle. Vielleicht möchtest du dein nächstes

Referat mit einer selbstgemachten Animation aufpeppen. Frage deine Lehrerin oder deinen Lehrer. Themen gibt es genug.

- Chemische Reaktionen (z.B. Wasserstoffmoleküle stoßen mit Sauerstoffmolekülen zusammen. Dabei entstehen Wassermoleküle).
- Die Kettenreaktion im Atomreaktor.
- Wie funktioniert ein Enzym?
- Wie funktioniert ein Transistor?

Spektralanalyse

Durch Farben kann man Stoffe erkennen. Roter Traubensaft sieht anders aus als Apfelsaft, weil beide Flüssigkeiten unterschiedliche Farbstoffe enthalten. Mit einem Spektralphotometer kannst du einen unbekannten Stoff anhand seiner Farbe erkennen. Du nimmst eine Lösung des Stoffes und sendest Licht unterschiedlicher Farben hindurch. Bei unserem Eigenbau-Spektralphotometer verwenden wir die Grundfarben Rot, Grün und Blau. Du bestrahlst also die Lösung zuerst mit rotem, dann mit grünem und dann mit blauem Licht und misst jedes Mal, wie viel Licht durchkommt und wie viel von der Lösung geschluckt (absorbiert) worden ist. Das Ergebnis nennt man ein Spektrogramm (siehe Abbildung A.14). Es ist eine Art Fingerabdruck für Farben. Mit einem Spektrogramm kann man auch Farben unterscheiden, die sehr ähnlich aussehen.

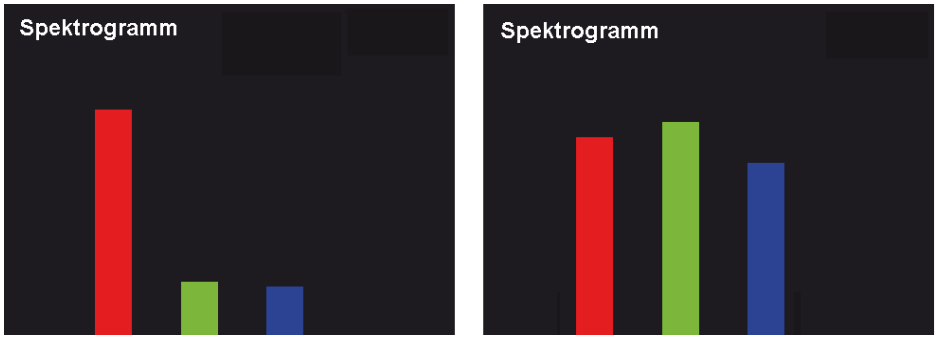


Abb. A.14: *Spektrogramme von zwei unterschiedlich gefärbten Flüssigkeiten: Roter Traubensaft und Apfelsaft.*

Die Hardware

Das Gerät, das wir bauen, nennt man Spektralphotometer. Für das Projekt brauchst du (außer dem RPi) folgende Teile:

- Eine Farbwechsellampe mit Fernbedienung, die man auch als Partybeleuchtung verwenden kann (Abbildung A.15). Du bekommst sie schon für etwa sieben Euro im Handel.

Abb. A.15: *Farbwechsellampe mit Fernbedienung*

- Eine Schreibtischleuchte oder Klemmleuchte, in die die Wechselfarbenlampe passt.
- Ein PicoBoard (siehe Kapitel 4).
- Eine Apparatur aus Pappe, in der ein Glasbehälter (Wasserglas oder Reagenzglas) für die zu untersuchende farbige Flüssigkeit lichtgeschützt untergebracht ist.

Ein Spektralphotometer aus Pappe

Im Prinzip funktioniert das Spektralphotometer so: Farbiges Licht wird durch eine Flüssigkeit geleitet. Auf der anderen Seite ist der Lichtsensor des PicoBoards. Mit hm wird gemessen, wie viel Licht durch die Flüssigkeit gelangt.

Als Apparatur kannst du die Box der Limonadenmischmaschine aus Kapitel 5 (Projekt 13) verwenden. Du musst dann nur den Grünfilter entfernen. Schraube die Birne in die Schreibtischlampe und leuchte mit der Schreibtischleuchte durch das Fenster in die Box. Der Nachteil dieser Apparatur ist, dass sie für Wassergläser konstruiert worden ist und du relativ große Mengen an Flüssigkeit brauchst.

Professioneller ist es, für die farbigen Lösungen Reagenzgläser zu verwenden. Aus Pappe und Kreppklebeband kannst du in wenigen Minuten eine geeignete Apparatur zusammenbauen. Sie besteht aus einer Grundplatte ❶, einem senkrechten Schacht für das Reagenzglas ❷ und zwei Rohren, die zur Lichtquelle und zum Lichtsensor des PicoBoards führen ❸. Schneide aus Pappkarton diese Teile aus (Abbildung A.16).

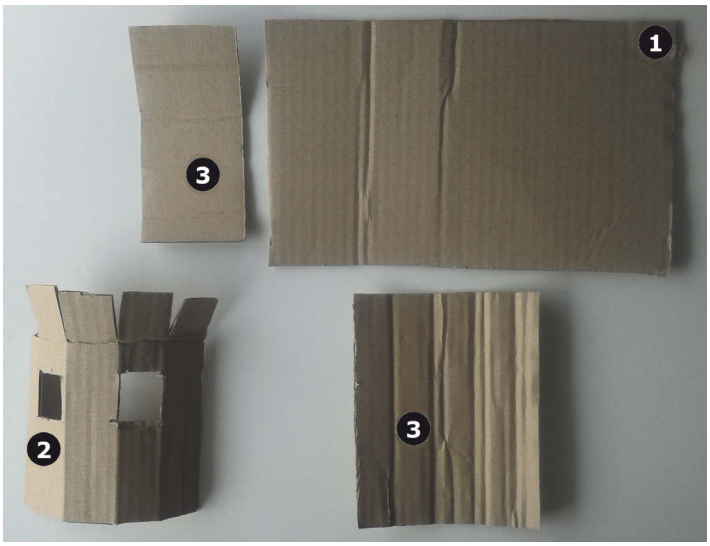


Abb. A.16: *Die Einzelteile der Apparatur*

Falte die Rohre so, dass sie einen quadratischen Querschnitt haben. Das ist leichter zusammenzubauen. Wichtig ist, dass der Schacht so eng ist, dass das Reagenzglas gerade eben hineinpasst, und später kein Licht am Reagenzglas vorbeikommt. Klebe mit Kreppklebeband alles zusammen. Das lange Seitenrohr führt in die Schreibtischleuchte zur LED-Lampe und das kürzere Seitenrohr führt zum Lichtsensor des PicoBoards (Abbildung A.17). Verwende reichlich Klebeband und achte darauf, dass möglichst wenig Tageslicht von außen in die Apparatur eindringt.

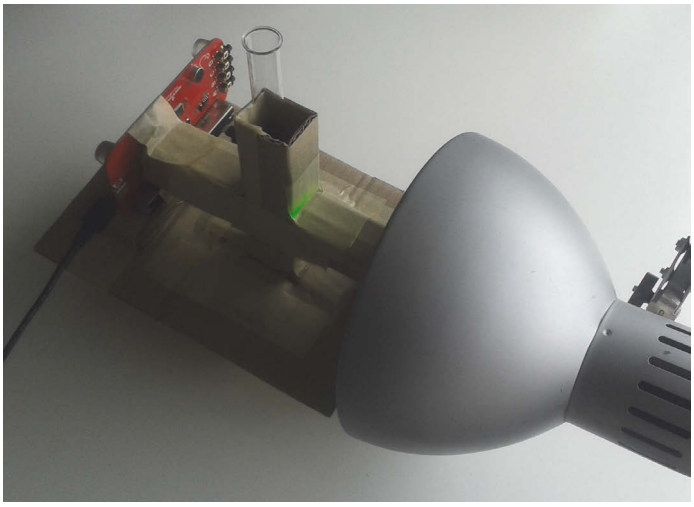


Abb. A.17: *Aufbau des Spektralphotometers*

An der Fernbedienung kannst du die Farbe der Lampe einstellen. Du verwendest nur die Grundfarben Rot, Grün und Blau. Denn alle anderen Farben werden aus den Grundfarben gemischt. Du brauchst aber monochromatisches Licht, das ist reines Licht einer bestimmten Wellenlänge (siehe Kasten).

Farbe und Wellenlänge

Licht besteht aus elektromagnetischen Wellen, so wie Radiowellen. Dabei hat jede Farbe eine bestimmte Wellenlänge. Die Wellenlänge misst man in Nanometer (nm). Ein Nanometer ist ein Milliardstel Meter, also 0,000 000 001 m. Bei guten Wechselfarbenlampen sind die Wellenlängen der Grundfarben angegeben. Typische Werte sind 625 nm (rot), 530 nm (grün) und 470 nm (blau).

So benutzt du das Spektralphotometer

Du benutzt das Spektralphotometer in zwei Schritten:

- Kalibrieren
- Messen

Mit dem *Kalibrieren* werden drei Probleme berücksichtigt:

- Die drei Grundfarben der LED-Farbwechsellampe sind nicht gleich hell.
- Das Reagenzglas und das Wasser, in dem der Farbstoff gelöst ist, können auch schon Licht absorbieren.
- Der Lichtsensor des PicoBoards hat für unterschiedliche Farben eine unterschiedliche Empfindlichkeit.

Zum Kalibrieren gibst du in den Schacht des Photometers ein Reagenzglas mit klarem Wasser (ohne Farbstoff). Das nennt man eine Blindprobe. Dann wählst du auf der Fernbedienung der LED-Lampe die Farbe Rot und drückst auf der Computertastatur die Taste R. Das Programm merkt sich den Helligkeitswert für Rot. Dann wählst du die Farbe Grün und drückst die Taste G. Wieder speichert das Scratch-Programm den Wert des Lichtsensors, diesmal für Grün. Schließlich stellst du Blau ein und drückst auf der Tastatur die Taste B.

Auf dem Bildschirm zeigen die drei Balken des Balkendiagramms den maximalen Wert. Das Spektralphotometer ist kalibriert.

Im zweiten Schritt gibst du ein Reagenzglas mit einer farbigen Flüssigkeit in den Schacht der Apparatur.

Wähle nun mit der Fernbedienung deiner Wechselfarbenlampe die Farbe Rot und klicke mit der linken Maustaste auf dem Bildschirm den roten Balken an. Dieser

rutscht nun ein Stück nach unten, wenn der Farbstoff einen Teil des roten Lichts absorbiert (schluckt). Schließlich machst du das Gleiche mit Grün und Blau und fertig ist das Spektrogramm. Wenn du deine Apparatur einmal kalibriert hast, kannst du nacheinander mehrere Flüssigkeiten untersuchen und Spektrogramme erstellen.

Das Programm

Obwohl die Bedienung des Spektralphotometers so kompliziert klingt, ist die Programmierung doch ganz einfach. Für jeden der drei farbigen Balken brauchst du nur sieben Befehle.

Die Bühne

Die Bühne hat irgendein Hintergrundbild (z.B. eine schwarze Fläche mit dem Text »Spektrogramm«) und überhaupt kein Skript.

Die Variablen

Für das Kalibrieren erzeugst du drei Variablen, die du für die Kalibrierung des Photometers benötigst. Nenne die neuen Variablen `faktor_rot`, `faktor_grün` und `faktor_blau`. Diese Variablen brauchen nicht angezeigt zu werden.

Die Objekte

Zur Vorgehensweise: Du entwickelst ein Objekt für den roten Balken. Das ist der Prototyp. Von diesem Objekt machst du später Kopien und veränderst sie ein wenig.

Erzeuge ein neues Objekt und zeichne ein möglichst hohes rotes Rechteck. Stelle die Ansicht mit der geringsten Vergrößerung ein (siehe Abbildung).

Abb. A.18: *Die kleinste Vergrößerung einstellen*

Stelle im Malprogramm den Drehpunkt für das Kostüm ungefähr so ein, wie in Abbildung A.19.

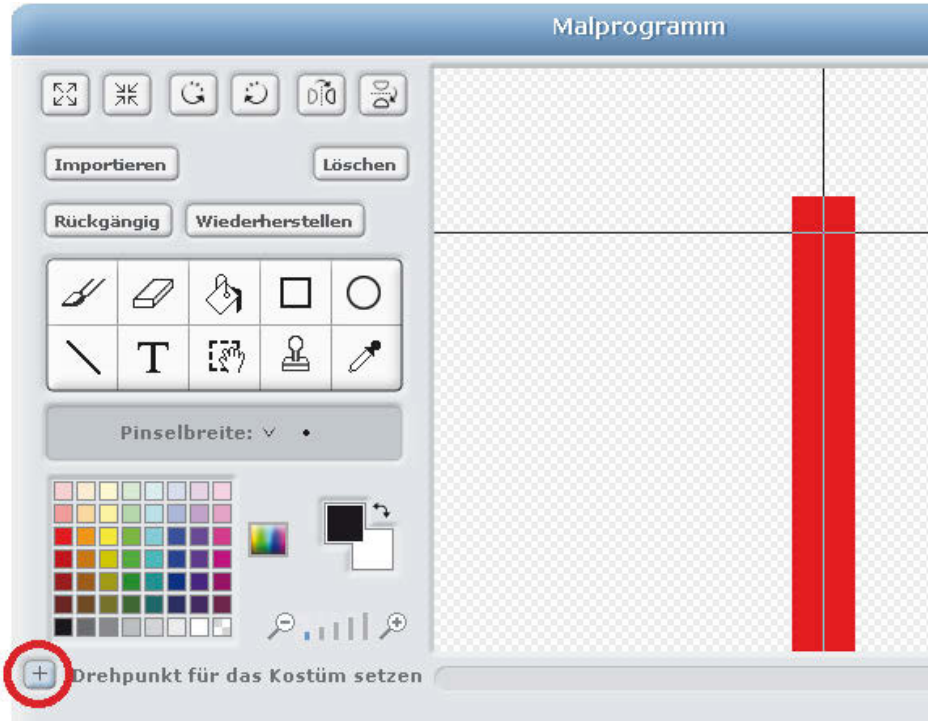


Abb. A.19: *Den Drehpunkt des Kostüms setzen.*

Gib dem Objekt den Namen `Rot`.

Schiebe auf der Bühne den Balken weit nach unten, so dass er nur ein bisschen herausragt. Lies oben die y-Position des Objektes ab. In dem Beispiel (Abbildung A.20 oben eingekreist) ist die y-Position `-180`. Das ist die unterste Position des Balkens. So steht er, wenn es dunkel ist und der Lichtsensor des PicoBoards den Wert `0` liefert.

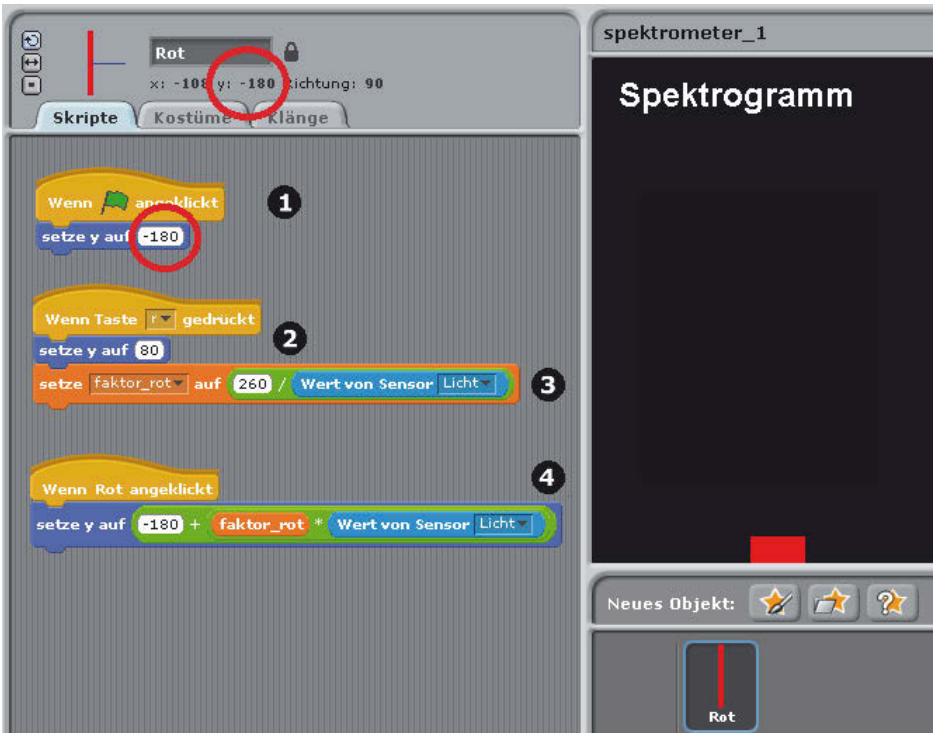


Abb. A.20: *Die Skripte des Objektes* Rot.

So funktioniert's

1. Das erste Skript wird nur einmal beim Start des Programms aufgerufen. Es setzt den roten Balken auf die unterste Position.

2. Das zweite Skript besorgt das Kalibrieren des roten Balkens. Es wird aufgerufen, wenn die Taste R gedrückt worden ist. Der Balken wird nach oben auf die Höhe y=80 verschoben. Das ist der höchste Wert. Beachte, dass der Unterschied zwischen niedrigster (-180) und höchster Position (+80) genau 260 Einheiten beträgt.

3. Hier wird nun ein Faktor berechnet, der später beim Messen benötigt wird. Der Faktor gibt an, um wie viele Pixel pro Einheit des Lichtsensors der rote Balken nach oben geschoben werden muss.

4. Wenn der rote Balken angeklickt wird, wird gemessen. Der rote Balken wird auf eine Höhe verschoben, die der gemessenen Lichtintensität entspricht. Je mehr Licht auf den Sensor fällt, desto höher ist der Balken. Für die Berechnung wird der Faktor verwendet, der beim Kalibrieren ermittelt worden ist.

Die anderen Objekte sind genauso aufgebaut. Am besten duplizierst du das Objekt Rot und wandelst die Kopien ein wenig ab.

- Klicke mit der rechten Maustaste auf das Symbol des Rot-Objekts und wähle den Befehl DUPLIZIEREN.
- Nenne das neue Objekt Grün.
- Bearbeite das Kostüm des neuen Objekts und färbe den Balken grün ein (Farbeimer).
- Ändere das Skript an drei Stellen ab. Ersetze den Buchstaben r durch g und die Variable faktor_rot durch faktor_grün (siehe Abbildung).

Abb. A.21: *Die Skripte des Objektes* Grün.

Auf die gleiche Weise erzeugst du ein Objekt für den blauen Balken.

Photometer

In der Chemie nennt man eine Maschine, die erkennt, wie viel Farbstoff in einer Lösung ist, ein Photometer. Solche Geräte sind nicht ganz billig und stehen in jedem chemischen Labor: im Krankenhaus, beim Lebensmitteluntersuchungsamt oder im Wasserwerk. Im Labor des Krankenhauses z.B. bestimmt man mit einem Photometer, wie viel Zucker (Glucose) im Blut eines Patienten ist. Dazu wird aus dem Blut das farblose Blutplasma abgetrennt und der darin enthaltene Zucker in einen Farbstoff umgewandelt. Mit dem Photometer können die Laboranten dann den Blutzuckergehalt bestimmen. Je intensiver die Farbe, desto mehr Zucker ist im Blut. In diesem Abschnitt entwickeln wir ein Scratch-Programm, das aus dem Raspberry Pi ein Photometer macht. Die Maschine wird auf eine bestimmte Fruchtsaftsorte eingestellt – sie wird kalibriert – und kann dann automatisch den Fruchtsaftgehalt einer Limonade oder Fruchtsaftschorle ermitteln.

So benutzt du das Photometer

Du baust die Apparatur auf wie in Abschnitt 6.2. Mit der Fernbedienung der LED-Farbwechsellampe wählst du die Grundfarbe (Rot, Grün oder Blau), die am meisten von der Farblösung absorbiert wird. Das ist immer die Komplementärfarbe. Wenn du roten Traubensaft untersuchst, wählst du die Farbe Grün.

Auf dem Bildschirm siehst du links drei Schaltflächen BLINDPROBE, REFERENZ und MESSEN, und rechts mehrere Anzeigen von Variablen. Mit den ersten beiden Schaltflächen kannst du das Photometer kalibrieren. Erst nach dem Kalibrieren wird gemessen. Als erstes gibst du ein Reagenzglas mit klarem Wasser in den Schacht der Apparatur und klickst auf die Schaltfläche BLINDWERT. Achte auf die angezeigten Werte (Abbildung A.22, erstes Bild)! Der aktuelle Wert des Lichtsensors wird als Blindwert I_0 gespeichert.

Dann kommt der zweite Schritt des Kalibrierens. Du gibst ein Reagenzglas mit rotem Fruchtsaft in das Photometer. Das ist die Referenzlösung. Nun geht nur noch wenig grünes Licht durch das Glas. Denn Grün ist ja die Komplementärfarbe von Rot. Der Lichtsensor liefert einen sehr kleinen Wert (hier 10).

Wenn der Wert des Lichtsensors *zu* klein ist, musst du deine Referenzlösung vorher verdünnen. Sonst werden die Messwerte zu ungenau. Damit die Verdünnung bei der Rechnung berücksichtigt werden kann, ist auf dem Bildschirm ein Schieberegler eingerichtet. Damit kannst du die Konzentration der Referenzlösung in Prozent angeben. Nehmen wir an, du hast den roten Traubensaft 1 zu 5 verdünnt. Dann stellst du als Referenzkonzentration 20% ein (Abbildung A.22, zweites Bild).

Wenn du nun die Schaltfläche REFERENZ drückst, ist das Gerät kalibriert. Das Programm bestimmt den so genannten Extinktionskoeffizienten. Der wird dann für die Berechnung der Konzentration aus dem Wert des Lichtsensors verwendet.

Abb. A.22: *Das Photometer kalibrieren.*

Nun kannst du Messungen durchführen. Stelle ein Reagenzglas einer verdünnten Fruchtsaftlösung in den Schacht des Photometers und klicke die Schaltfläche Mes-

sen an. Ab jetzt wird kontinuierlich gemessen. Die Anzeige Konzentration gibt den Anteil an Fruchtsaft (oder Farbstoff) in der Probe an. Schau dir Abbildung A.23 an. Je weniger Licht den Sensor trifft, desto höher die Extinktion und desto höher die Konzentration des Farbstoffs. Ist ja logisch! Viel Farbstoff schluckt viel Licht.

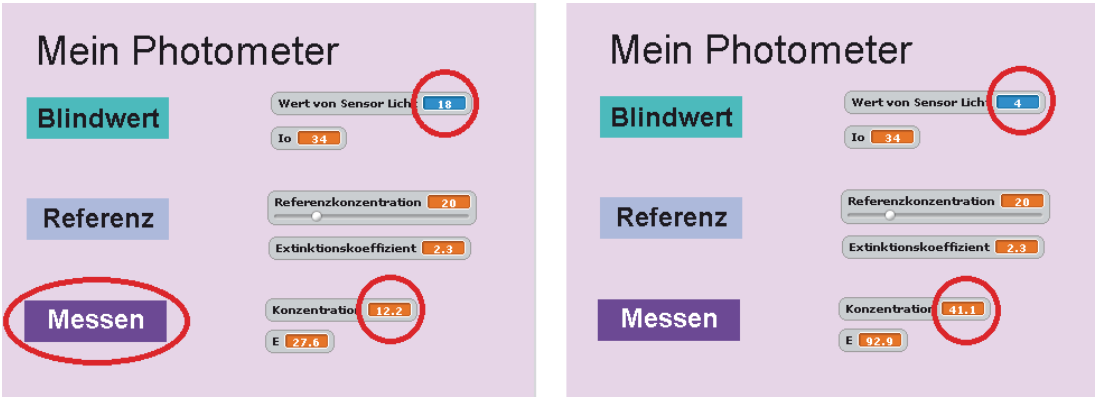


Abb. A.23: *Wenn du auf* MESSEN *geklickt hast, wird der Fruchtsaftgehalt der Probe in Prozent angegeben.*

Das Programm

Die Variablen

Welche Variablen benötigt werden, siehst du schon auf den Abbildungen des vorigen Abschnitts. Die Tabelle gibt eine Übersicht.

Name	Bedeutung
Io	Wert des Sensors *Licht* bei der Blindprobe.
Referenz-konzentration	Fruchtsaftgehalt der Referenzlösung in Prozent. Dieser kann mit einem Regler eingestellt werden (Reglerbereich von 0 bis 100).
Extinktions-koeffizient	Diese Zahl gibt an, wie stark die Farbstofflösung das Licht der LED-Lampe absorbiert.
Konzentration	Die Konzentration wird kontinuierlich berechnet, wenn MESSEN angeklickt worden ist. Die Zahl gibt den Fruchtsaftgehalt der Probe in Prozent an.
E	Die aktuelle Extinktion in Prozent. Die Zahl gibt an, wie viel des einfallenden Lichtes des LED-Lampe von der Lösung absorbiert (geschluckt) wird.

Alle Variablen werden sichtbar gemacht. Die Variable Referenzkonzentration stellst du als Regler dar.

- Klicke die Anzeige der Variablen mit der rechten Maustaste an.
- Wähle im Kontextmenü die Option REGLER.
- Klick nochmals die Anzeige mit der rechten Maustaste an.
- Wähle die Option REGLERBEREICH EINSTELLEN.
- Stelle den Bereich ein (z.B. 0 bis 100).

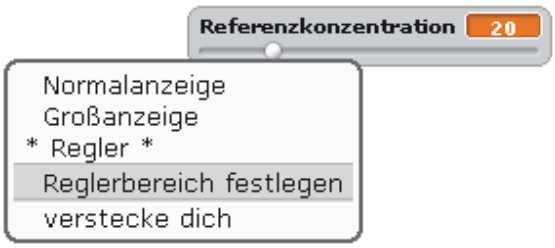


Abb. A.24: *Eine Variable als Regler darstellen und den Reglerbereich festlegen*

Anzeige des Lichtsensors

Zur Kontrolle ist die Anzeige des Lichtsensors ganz hilfreich. So bringst du sie auf den Bildschirm:

- Wähle die Befehlssammlung FÜHLEN.
- Wähle auf dem Baustein *Wert von Sensor ...* den Sensor Licht (auf das kleine Dreieck klicken).
- Setze ein Häkchen in die Checkbox neben dem Baustein (anklicken).

Abb. A.25: *Die Anzeige des Lichtsensors sichtbar machen*

Die Bühne

Die Bühne hat kein Skript. Gestalte einen Hintergrund mit einem kurzen Text.

Die Objekte

Jede der drei Schaltflächen ist ein Objekt. Das Kostüm jeder Schaltfläche ist einfach nur ein rechteckiger Kasten mit Beschriftung (BLINDWERT, REFERENZ, MESSEN). Jede Schaltfläche hat ein Skript. Dieses Skript legt fest, was passiert, wenn die Schaltfläche angeklickt worden ist.

Blindwert

Abb. A.26: *Das Skript des Objekts Blindwert*

Diese Schaltfläche Blindwert wird angeklickt, wenn sich in der Apparatur ein Reagenzglas mit Wasser befindet (Blindprobe). Wenn das Objekt angeklickt worden ist, wird der aktuelle Wert des Lichtsensors in der Variablen `Io` gespeichert. Diese Zahl wird für die Berechnung der Konzentration einer Farbstoff-Lösung benötigt.

Referenz

Die Schaltfläche REFERENZ wird angeklickt, wenn sich in der Apparatur ein Reagenzglas mit einer Referenzlösung befindet. Es ist z.B. Fruchtsaft, den du 1 zu 5 verdünnt hast. Die Konzentration (Prozentwert zwischen 0 und 100) ist per Schieberegler eingestellt worden.

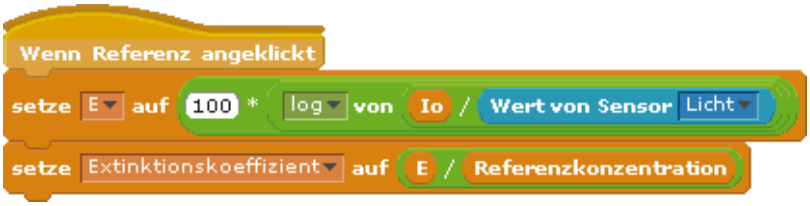


Abb. A.27: *Das Skript des Objekts Referenz*

Was passiert nach dem Anklicken? Damit es nicht zu kompliziert wird, haben wir die Rechnung auf zwei Zuweisungen verteilt. Im ersten Schritt wird die Extinktion berechnet. Das ist ein Maß dafür, wie viel Licht der Farbstoff absorbiert. Die Extinktion ist ein Begriff aus der Chemie. Wenn du dich für die Hintergründe inte-

ressierst, schau in deinem Chemiebuch (oder in der Wikipedia) unter »Lambert-beersches Gesetz« nach. Du findest vermutlich diese Formel:

$$E = \log(I_0/I)$$

Dabei ist I_0 die Lichtintensität der Blindprobe (also der Wert des Lichtsensors bei der Blindprobe) und I die Lichtintensität der gemessenen Farbstofflösung (also der aktuelle Wert des Lichtsensors). Die Funktion `log()` ist der dekadische Logarithmus (Logarithmus zur Basis 10).

In dem Skript multiplizieren wir den Wert mit 100, damit der Zahlenwert nicht zu klein wird. Scratch hat Probleme mit der Darstellung kleiner Zahlen. In der zweiten Zeile wird der Extinktionskoeffizient berechnet und gespeichert. Den brauchen wir später bei den Messungen für die Berechnung der Konzentration.

Messen

Nach dem Kalibrieren klickst du einmal die Schaltfläche MESSEN an. Danach kannst du beliebig viele Messungen mit dem Farbstoff machen, für den du das Photometer gerade kalibriert hast.

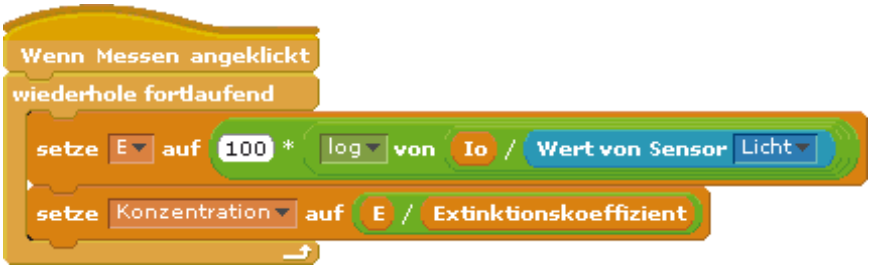
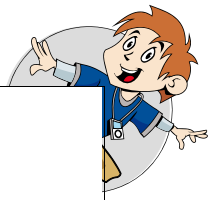


Abb. A.28: *Das Skript des Objekts Messen*

Nach dem Anklicken wird eine Endloswiederholung ausgeführt. Dabei wird zunächst die Extinktion berechnet und anschließend die Konzentration des Farbstoffs in der Probe.

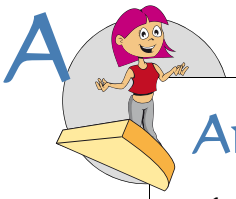


Fragen

- Auf der Bühne liegen zwei Objekte. Wie kann man dafür sorgen, dass beim Programmlauf (grüne Flagge ist angeklickt worden) Objekt A immer im Vordergrund ist?
- Ein Objekt soll sich bewegen, bis es ein zweites Objekt berührt, das sich auch bewegt. Welchen der folgenden Bausteine kann man für diese Aufgabe nicht verwenden?

- Ein Auto, das sich von links nach rechts bewegt, soll immer schneller werden. In dem Skript in der Abbildung fehlt eine Anweisung. Welche der vier Anweisungen führt nicht zu einer Erhöhung der Geschwindigkeit?

- Schau dir das rechte Bild in Abbildung A.14 an. Welche Farbe absorbiert Apfelsaft am stärksten?



Antworten zu den Fragen

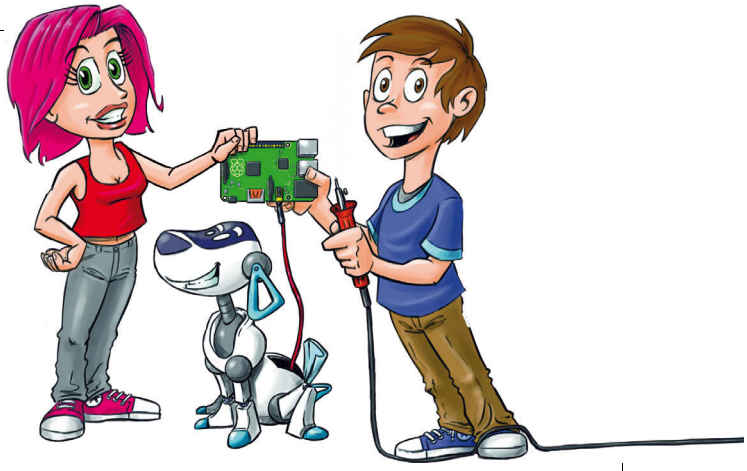
1. Die Anweisung 

setzt das Objekt in den Vordergrund.

2. Die Anweisung 

ist ungeeignet, weil sie zu einem festen Zielpunkt führt. Aber hier bewegt sich das Ziel.

3. Der Apfelsaft absorbiert am meisten blaues Licht. Denn im Spektrogramm hat der blaue Balken die geringste Höhe. Das bedeutet, dass vom blauen Licht am wenigsten am Lichtsensor ankommt.



B

Bildverarbeitung

In diesem Kapitel geht es um das automatische Erzeugen und Verarbeiten von Bildern. Du entwickelst Programme, die dekorative Muster für Geschenkpapier oder T-Shirts erzeugen. Ein »digitaler Bilderrahmen« sucht nach gespeicherten Fotos und stellt sie auf dem Bildschirm dar. Nebenbei lernst du einige Ideen der objektorientierten Programmierung kennen.

Der Rasperry Pi als Künstler

Grafische Muster für Geschenkpapier oder T-Shirts kannst du dir vom Raspberry Pi anfertigen lassen, anstatt sie von Hand – Strich für Strich – zu zeichnen.

Der Canvas

Für Grafiken aus Linien und geometrischen Formen bietet das Modul `tkinter` das Widget `Canvas`. Ein `Canvas`-Objekt (*canvas*: engl. *Leinwand*) kannst du dir als Fläche vorstellen, auf der gezeichnet wird. Das `Canvas`-Objekt beherrscht Methoden,

um grafische Objekte (Kreise, Rechtecke, Linien, Texte etc.) zu erzeugen, zu verändern oder zu löschen. Diese grafischen Objekte nennt man Items.

Jedes Item auf einem Canvas erhält bei seiner Erschaffung eine ID, eine Nummer zur Identifikation. Oft braucht man die ID nicht und speichert sie deshalb auch nicht. Wenn du aber ein Item später wiederfinden und z.B. löschen möchtest, musst du die ID speichern.

Methode	Erklärung
`create_image(x, y, image=img)`	Erzeugt ein Bild an der Stelle `(x, y)`.
`create_line(x0, y0, ...)`	Erzeugt eine zusammenhängende Linie, die aus mehreren geraden Stücken besteht.
`create_oval (x0, y0, x1, y1, ...)`	Erzeugt eine Ellipse an der Position `(x0, y0, x1, y1)`.
`Create_rectangle (x0, y0,x1, y1, ...)`	Erzeugen eines Rechtecks mit linker oberer Ecke `(x0, y0)` und rechter unterer Ecke `(x1, y1)`
`create_text(x, y,text=t, ...)`	Erzeugen eines Textobjekts an der Stelle `(x, y)`.
`delete(itemID)`	Löscht das Objekt mit der angegebenen ID.
`find_all()`	Liefert eine Liste der ID-Nummern aller Objekte auf dem Canvas.

Tabelle B.1: *Einige Methoden zum Erzeugen und Löschen von Items*

Projekt: Mikadomuster

Dieses kleine Programm erzeugt ein Muster aus Linien. Es sieht aus wie Mikadostäbchen auf dem Tisch. Man sieht das Muster im Anwendungsfenster. Es wird aber auch eine SVG-Datei erzeugt. SVG steht für *Scalable Vector Graphics*. Auf Deutsch heißt das »skalierbare Vektorgrafik«. Vektorgrafikdateien, kannst du mit einem speziellen Grafikprogramm wie *Inkscape* öffnen und weiterbearbeiten oder ausdrucken. Der Vorteil: Eine Vektorgrafik kannst du in beliebiger Größe und Qualität drucken. Sie wird beim Vergrößern nicht »pixelig«.

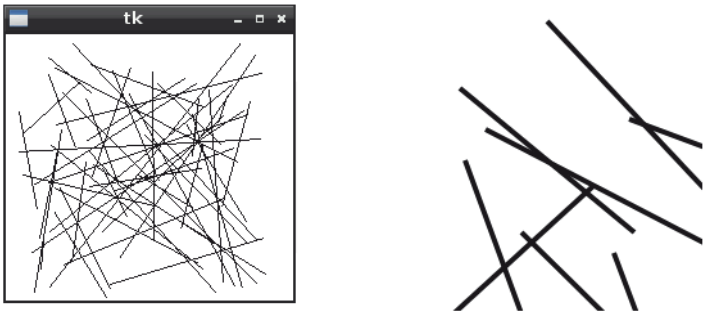


Abb. B.1: *Ein zufälliges Linienmuster. Links das Applikationsfenster, rechts ein vergrößerter Ausschnitt der SVG-Datei.*

Vorbereitung

Lege einen neuen Projektordner an. Besorge dir aus dem Internet das Modul `canvasvg.py` von Wojciech Muła und speichere es in deinem Projektorder ab. Du findest das Programm unter dem URL *http://www.math.uic.edu/t3m/hg/plink/plink/canvasvg.py*.

Wenn du deine Zeichnungen nicht als SVG-Datei abspeichern willst – vielleicht weil dir das Ganze zu kompliziert ist –, lässt du im folgenden Programm einfach die Zeilen #1 und #5 weg.

Programm

```
import canvasvg                                  #1
from tkinter import *
from random import randint
ANZAHL, SEITE = 50, 200                          #2

fenster = Tk()
bild = Canvas(master=fenster,
              width=SEITE, height=SEITE,
              bg='white')                        #3
bild.pack()
for i in range(ANZAHL):
bild.create_line(randint(0,SEITE),
randint(0, SEITE),
                     randint(0,SEITE),
                     randint(0, SEITE))          #4

canvasvg.saveall('muster_1.svg', bild)           #5
```

So funktioniert's

#1 Import des Moduls `canvasvg`. Es enthält eine Funktion, die den Inhalt eines Canvas als vektororientierte SVG-Datei abspeichert. Das Modul muss im gleichen Ordner sein wie dieses Programm.

#2 Konstanten für die Anzahl der Linien und die Größe des Canvas.

#3 Hier wird ein neues `Canvas`-Objekt erzeugt. Der Canvas ist quadratisch (Seitenlänge 200 Pixel) und hat einen weißen Hintergrund.

#4 Auf dem Canvas wird eine gerade Linie erzeugt. Die vier Argumente sind Zufallszahlen zwischen 0 und 200. Die ersten beiden Zahlen sind die Koordinaten des Anfangspunktes und die letzten beiden Zahlen sind die Koordinaten des Endpunktes der Linie.

#5 Das Bild auf dem Canvas wird als SVG-Datei mit dem Namen `muster_1.svg` gespeichert.

Objektorientierte Programmierung

Python ist eine objektorientierte Programmiersprache. Jedes Programm besteht aus Objekten, die untereinander »Botschaften austauschen«. Das klingt vielleicht ein bisschen sonderbar. In der Denkweise der objektorientierten Programmierung stellt man sich die Objekte – die Bauteile eines Computerprogramms – als lebendige Wesen vor, die Aufträge vergeben und Aufgaben ausführen.

Die Anweisung

```
>>> 'Banane'.count('a')
```

ist eine Botschaft an ein String-Objekt. Das Objekt erhält den Auftrag, die Vorkommen des Buchstaben `a` zu zählen.

Es gibt unterschiedliche Typen von Objekten: Zahlen, Zeichenketten, Listen oder die Widgets einer grafischen Benutzungsoberfläche. Die Objekte verschiedener Typen werden in so genannten Klassen definiert. Eine Klasse kannst du dir als Bauplan für Objekte vorstellen. Mit der Funktion `type()` kannst du den Namen der Klasse abfragen. Das kannst du in der Python-Shell ausprobieren.

```
>>> type('Morgen')
<class 'str'>
>>> type(1)
<class 'int'>
```

Ein Objekt besitzt Attribute und beherrscht Methoden. Attribute sind Merkmale oder Daten. Methoden sind Funktionen, die zu Objekten gehören. So haben alle Strings (neben vielen anderen Methoden) die Methode `count()`.

Wie definiert man eine neue Klasse?

Strings, Listen oder Widgets sind vorgegebene Klassen. Du kannst auch selbst eigene Klassen definieren. Eine Klassendefinition besteht aus zwei Teilen:

- Die *Kopfzeile* beginnt mit dem Schlüsselwort `class`. Danach kommt der Name der Klasse und am Ende der Zeile ein Doppelpunkt. Beispiel:

```
class App:
```

- Der *Körper* der Klassendefinition ist ein eingerückter Block.
  - Er beginnt normalerweise mit der Definition einer Methode mit dem speziellen Namen `__init__()`.Die Parameterliste beginnt mit `self`. Dieses Argument darf niemals fehlen. Es bezeichnet ein Objekt der Klasse, die wir gerade definieren. In der Methode `__init__()` werden Attribute definiert und mit Anfangswerten belegt. Die Methode `__init__()` wird immer sofort ausgeführt, sobald ein Objekt der Klasse erzeugt worden ist.
  - Im weiteren Verlauf des Körpers einer Klassendefinition werden weitere Methoden definiert. Ihre Namen sind beliebig. Sie müssen aber wieder als erstes Argument `self` enthalten.

Wichtig ist noch, dass beim Aufruf einer Methode der erste Parameter `self` weggelassen wird. Der Aufruf einer Methode hat also immer ein Argument weniger als die Definition der Methode.

Ein Programm, das Klassendefinitionen enthält, ist ein *objektorientiertes Programm*. Anfangs erscheint es abstrakter und ist schwieriger zu verstehen als ein herkömmliches (prozedurales) Programm. Das ist aber nur am Anfang so. Mit der Zeit gewöhnst du dich an die objektorientierte Ausdrucksweise. Bei umfangreicheren Projekten ist die Objektorientierung eine große Hilfe. Sie erleichtert es, das große Ganze in überschaubare Teile zu zergliedern.

Projekt: Zufallsmuster

Das Programm erzeugt eine zufällig angeordnete Gruppe weißer Rahmen auf dunklem Untergrund (siehe Abbildung B.2). In das Eingabefeld trägt man die gewünschte Anzahl von Rahmen ein. Wird dann die Schaltfläche NEUES MUSTER angeklickt, erscheint ein neues Zufallsbild und wird als SVG-Datei gespeichert.

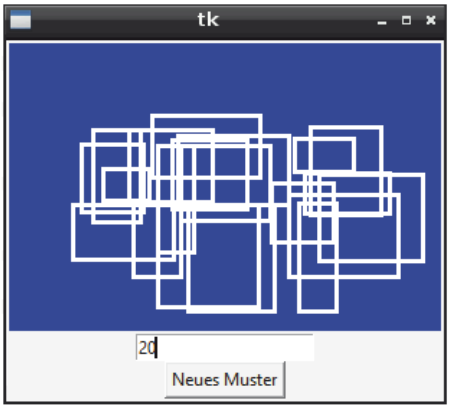
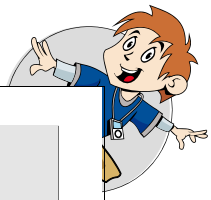


Abb. B.2: *Interaktiver Generator für Zufallsmuster*

Programm

Das Programm verwendet wieder das Modul canvasvg, das zuerst herunter geladen und im Projektordner gespeichert werden muss.

```
# rahmen.pyw
from tkinter import *
from random import *
import canvasvg

class App:                                          #1
  def __init__(self, breite, höhe):                #2
      self.höhe, self.breite = höhe, breite         #3
      #Widgets
      self.fenster = Tk()
      self.bild = Canvas(master=self.fenster,
                  width=breite, height=höhe,
                  bg='blue')
      self.eingabe = Entry(master=self.fenster)
      self.button = Button(master=self.fenster,
                        text='Neues Muster',
                        command=self.zeichnen)
      #Layout
      self.bild.pack()
      self.eingabe.pack()
      self.button.pack()
      self.fenster.mainloop()

  def löschen(self):                                #4
      '''Alle Rechtecke löschen'''
      items = self.bild.find_all()                  #5
```



```
        for item in items:
            self.bild.delete(item)                  #6

    def zeichnen(self):
        '''Zufällige Rechtecke zeichnen'''
        self.löschen()                              #7
        anzahl = int(self.eingabe.get())            #8
        for i in range(anzahl):
            x = randint(20, self.breite-80)         #9
            y = randint(20, self.höhe-80)           #10
            höhe = randint(20, 80)                  #11
            breite = randint(20, 80)
            self.bild.create_rectangle(x, y,
                       x+breite, y+höhe,
                       outline='white', width=3)    #12
        canvasvg.saveall('muster_1.svg', self.bild)

a = App(300, 200)                                   #13
```

So funktioniert's

#1 Kopfzeile der Klassendefinition. Hier beginnt die Definition der Klasse `App`.

#2 Die Methode `__init__()`initialisiert ein Objekt. Das erste Argument ist `self`. Damit ist ein Objekt der Klasse bezeichnet. Die Methode wird automatisch aufgerufen, wenn ein Objekt dieser Klasse erzeugt wird (Zeile #13).

#3 Hier werden zwei Attribute definiert. Sie stellen die Breite und Höhe des Canvas mit dem Muster dar. Die Namen der Attribute beginnen immer mit `self.`. Anschließend werden Widgets als Attribute definiert. Auch ihre Namen fangen mit `self.` an.

#4 Eine Methode, die alle Items auf dem Canvas löscht. Auch sie hat das Argument `self`.

#5 Die Canvas-Methode `find_all()` liefert eine Folge aller IDs von Items, die auf den Canvas gezeichnet worden sind.

#6 Hier wird das Item mit der ID-Nummer `item` gelöscht.

#7 Hier wird die Methode `löschen()` aufgerufen. Das Objekt schickt also eine Botschaft an sich selbst. Beachte, dass in der Parameterliste (Klammer nach dem Funktionsnamen) das Argument `self` *weggelassen* wird.

#8 Hier wird der Inhalt aus dem Eingabefeld gelesen (ein String) und in eine ganze Zahl (Typ `int`) umgewandelt. Hoffentlich steht in dem Eingabefeld auch eine Zahl! Sonst gibt es eine Fehlermeldung.

#9 Hier wird eine Zufallszahl für eine x-Koordinate erzeugt. Der Zahlenbereich ist so gewählt, dass später das Rechteck vollständig auf dem Canvas zu sehen ist und nicht über den Rand hinausgeht.

#10 Nach dem gleichen Prinzip wird eine zufällige y-Koordinate gewählt.

#11 Höhe und Breite des Rechtecks sind Zufallszahlen zwischen 20 und 80.

#12 Hier wird ein neues Rechteck auf den Canvas gezeichnet. Die ersten vier Zahlen sind die Koordinaten des linken oberen und des rechten unteren Eckpunktes.

#13 Hier wird ein Objekt der Klasse App erzeugt. Die Parameter sind die Breite und Höhe des Canvas mit dem Muster.

Projekt: Ein digitaler Bilderrahmen

Der digitale Bilderrahmen zeigt Bilder, die als JPEG-Dateien im Projektordner gespeichert sind. Immer, wenn man auf die Schaltfläche WEITER klickt, wird ein neues Bild aus dem Ordner zufällig ausgewählt und angezeigt.

Zur Vorbereitung legst du einen Projektordner an. In diesen Ordner speicherst du das Bilderrahmenprogramm und einige JPEG-Dateien mit Digitalfotos. Ihre Dateinamen enden auf .jpg.

Abb. B.3: *Ein Foto im digitalen Bilderrahmen (Dünengebiet in der Nähe von Bergen, Nordholland)*

Das Modul PIL

Digitalfotos sind meist im JPEG-Format. Um aus einer JPEG-Datei ein PhotoImage-Objekt zu gewinnen, brauchst du die *Python Imaging Library* (PIL). PIL gibt es für die Python-Versionen 2.2 bis 2.7, aber (noch) nicht für Python 3. Deshalb verwenden wir in diesem Abschnitt ausnahmsweise Python 2.7, das auf deinem Raspberry Pi vorinstalliert ist.

PIL ist nicht vorinstalliert. Du musst es zuerst herunterladen und installieren. Sorge dafür, dass dein RPi mit dem Internet verbunden ist und gib im LXTerminal-Fenster folgendes Kommando ein:

```
$ sudo apt-get install python-imaging python-imaging-tk
```

Mit PIL kannst du Bilder verarbeiten, die als JPEG- oder PNG-Datei gespeichert sind. Dabei gehst du so vor:

Du erzeugst ein neues Objekt der Klasse Image:

```
bild = Image.open(dateiname)
```

Dann bearbeitest du das Bild mit den Methoden der Image-Objekte (siehe Tabelle B.2). Schließlich muss aus dem Image-Objekt ein PhotoImage-Objekt gewonnen werden, damit man es auf einem Tkinter-Widget darstellen kann. Alle Einzelheiten sind im Programmbeispiel erklärt.

Attribut/Methode	Erklärung
copy()	Liefert ein neues Image-Objekt als Kopie des Bildes.
crop((*x0*,*y0*,*x1*,*y1*))	Aus einem Bild wird ein kleineres Stück ausgeschnitten. Das Argument ist ein 4-Tupel, das eine Bounding-Box mit den Eckpunkten (*x0*, *y0*) und (*x1*, y1) beschreibt. Zurückgegeben wird ein neues Image-Objekt, das nur aus den Pixeln in der Bounding-Box besteht.
getPixel(*x*, *y*)	Liefert den Wert des Pixels an Position (x, y).
load()	Liefert ein Objekt, das schnelleren Zugriff auf Pixel erlaubt. Siehe Beispiel in Kapitel 18.
resize(size=(*width*, *height*))	Liefert ein neues Image-Objekt mit einer (vergrößerten oder verkleinerten) Kopie mit Breite *width* und Höhe *height*.

Tabelle B.2: *Einige Methoden und ein Attribut von Image-Objekten aus dem Modul PIL*

Attribut/Methode	Erklärung
save(*dateiname*)	Das Bild wird gespeichert. Das Argument ist ein Dateiname (String).
size	Das Attribut beschreibt die Größe des Bildes als Tupel der Form (*breite*, *höhe*).

Tabelle B.2: *Einige Methoden und ein Attribut von Image-Objekten aus dem Modul PIL (Forts.)*

Programmierung

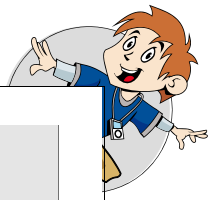
In diesem Projekt verwendest du Python 2, da es PIL noch nicht für Python 3 gibt. Du startest also nicht IDLE3, sondern IDLE.

Python 2 unterscheidet sich ein klein wenig von Python 3. In diesem Projekt musst du Folgendes beachten:

- Das `tkinter`-Modul heißt bei Python 2 `Tkinter` (großer Anfangsbuchstabe).
- Wenn du bei Python 2 Nicht-ASCII-Zeichen wie die deutschen Umlaute verwendest, musst du die Kodierung angeben. Das ist im Prinzip kein Problem. Wenn du versuchst das Programm zu starten, sagt dir IDLE, was zu tun ist. Am einfachsten ist es aber, wenn du in deinem Programmtext auf Umlaute und `ß` verzichtest.

Programm

```
#! /usr/bin/python                              #1
from Tkinter import *                           #2
from PIL import Image, ImageTk
from random import choice
import os

GROESSE = (1200, 800)                           #3

def zeigeBild():
    global bildTk
    datei = choice(bilder)                      #4
    bild = Image.open(datei)                    #5
    bild = bild.resize(size=GROESSE)            #6
    bildTk = ImageTk.PhotoImage(bild)           #7
    label.config(image=bildTk)
```



```
dateien = os.listdir('.')                          #8
bilder = []                                        #9
for d in dateien:                                  #10
    if d.endswith('.jpg'):
        bilder.append(d)

#widgets
fenster = Tk()
label = Label(master=fenster)
label.pack()
button=Button(master=fenster,
              command=zeigeBild,
              text='Weiter')
button.pack(side=LEFT)
zeigeBild()
fenster.mainloop()
```

So funktioniert's

#1 `python` ist der Name des Python-2.7-Interpreters.

#2 Bei Python 2.7 beginnt `Tkinter` mit großem `T`.

#3 Dieses Tupel enthält die Breite und Höhe des Bildes, das auf dem Label gezeigt wird. Das Verhältnis von Breite zu Höhe sollte genauso sein wie bei den gespeicherten Bilddateien. Anderenfalls werden die Bilder verzerrt angezeigt.

#4 Aus der Liste der Bilddateien wird nach dem Zufallsprinzip ein Dateiname ausgewählt.

#5 Hier wird ein `Image`-Objekt (Modul PIL) erzeugt.

#6 Das Bild wird auf die Größe 1200 mal 800 gebracht (oder was auch immer in der Konstanten `GROESSE` festgelegt ist).

#7 Aus dem `Image`-Objekt wird ein `PhotoImage`-Objekt erzeugt. Nur `PhotoImage`-Objekte können auf einem Label angezeigt werden.

#8 Die Funktion `listdir()` aus dem Modul `os` liefert eine Liste aller Dateinamen in dem angegebenen Verzeichnis. Das Argument `'.'` bezeichnet das Projektverzeichnis, in dem auch das Python-Programm gespeichert ist.

#9 Wir bauen eine Liste aller Bilddateinamen des Verzeichnisses auf. Anfangs ist diese Liste leer.

#10 Wenn ein Dateiname aus der Liste der Dateinamen mit `'.jpg'` endet, wird er an die Liste `bilder` angehängt.

Fragen

1. Zu welchem Zweck braucht man die ID-Nummer eines Items auf dem Canvas?
2. Mit welcher Standardfunktion findet man heraus, zu welcher Klasse ein Objekt gehört?
3. Welche Besonderheit hat der erste Parameter einer Methodendefinition (`self`)?
4. Eine Klasse kann man sich als Bauplan für Objekte vorstellen. Von einer Klasse können mehrere Objekte existieren. Worin können sich die Objekte einer Klasse unterscheiden, obwohl sie nach dem gleichen Bauplan geschaffen worden sind?

Aufgabe: Bunte Texte

Entwickle ein Programm, das ein zufälliges Muster aus bunten Texten erzeugt. In das Eingabefeld gibt man ein Wort ein. Nach dem Knopfdruck erscheint das Wort 20 Mal an zufälligen Stellen und in einer zufällig ausgewählten Farbe.

Abb. B.4: *Ein zufälliges Muster aus Wörtern*

Tipps

Wandle das Programm aus Abschnitt »Projekt: Zufallsmuster« auf Seite 31 ein wenig ab.

Verwende eine Liste aus Farbnummern, aus denen eine Farbe ausgewählt werden kann. Die drei Ziffern (Hexadezimalziffern) der Farbnummer geben an, wie groß der Anteil der Farben Rot, Grün und Blau ist. Hier ein Beispiel:

```
FARBEN = ['#10F', '#E12', '#3C1', '#AD0', '#232']
```

In der Methode `zeichnen()` verwendest du folgende Anweisungen:

```
self.bild.create_text(x, y,
                      font=('Arial', 30),
                      text=self.eingabe.get(),
                      fill=choice(FARBEN))
```

Die Anweisung setzt den Text, der im Eingabefeld `self.eingabe` eingetragen ist, an die Position `x`, `y`. Die Textfarbe ist eine zufällige Farbe, die mit `choice()` aus der Liste von Farbnummern ausgewählt wird. Diese Zufallsfarbe wird dem Schlüsselwortargument `fill` zugewiesen.

Antworten zu den Fragen

1. Die ID-Nummer braucht man, wenn man das Item wieder vom Canvas entfernen will.
2. Die Standardfunktion `type()` liefert die Bezeichnung der Klasse. Beispiel:

```
>>> type(1.0)
<class 'float'>
```

3. Der erste Parameter `self` bezeichnet das Objekt. Er wird beim Aufruf weggelassen.
4. Die Attribute der Objekte einer Klasse können unterschiedliche Werte besitzen.

Lösung der Aufgabe

Programm

```
# text_bild.py
from tkinter import *
from random import *

FARBEN = ['#10F', '#E12', '#3C1', '#AD0', '#232']
ANZAHL = 20                                        #1

class App:
    def __init__(self, breite, höhe):
        self.höhe, self.breite = höhe, breite
```

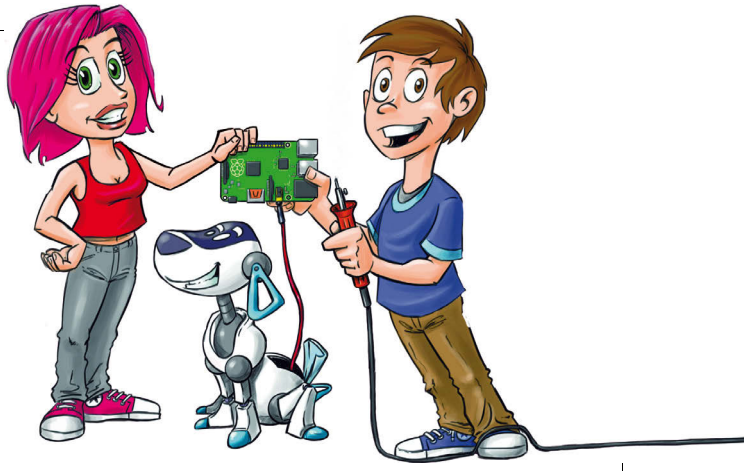
```
        self.fenster = Tk()
        self.bild = Canvas(master=self.fenster,
                           width=breite, height=höhe,
                           bg='white')
        self.bild.pack()
        self.eingabe = Entry(master=self.fenster)
        self.eingabe.pack()
        self.button = Button(master=self.fenster,
                             text='Neues Muster',
                             command=self.zeichnen)
        self.button.pack()
        self.fenster.mainloop()

    def löschen(self):
        items = self.bild.find_all()
        for item in items:
            self.bild.delete(item)

    def zeichnen(self):
        self.löschen()
        for i in range(ANZAHL):                    #2
            x = randint(0, self.breite)
            y = randint(0, self.höhe)
            self.bild.create_text(x, y,            #3
                          font=('Arial', 30),
                          text=self.eingabe.get(),
                          fill=choice(FARBEN))

a = App(300, 200)
```

So funktioniert's

#1 Hier werden zwei Konstanten definiert. Man kann sie an dieser Stelle leicht finden und abwandeln, wenn man möchte. Zum Beispiel kann man der Liste neue Farben hinzufügen oder die gewünschte Anzahl an Wörtern auf dem Canvas verändern.

#2 Mit `range(ANZAHL)` wird festgelegt, wie oft der Anweisungsblock der Iteration ausgeführt wird. Hier hat `ANZAHL` den Wert 20. Also wird 20 Mal das Wort auf den Canvas geschrieben.

#3 Der Text aus dem Eingabefeld wird in einer zufälligen Farbe an die zufällige Position `x`, `y` geschrieben.



C

Experimente mit der Infrarotkamera

Hast du das Infrarot-Kameramodul NoIR? Dann könnte dieser Abschnitt für dich interessant sein. Infrarotstrahlen sind elektromagnetische Wellen mit einer Wellenlänge zwischen 780 nm (Nanometer = milliardstel Meter) und 1 mm. Das sichtbare Licht liegt im Bereich 380 nm (violett) bis 780 nm (rot). Strahlung mit längerer Wellenlänge ist Infrarot und ist für den Menschen völlig unsichtbar.

Mit dem Kommando

```
$ raspistill -t 0 -p '200, 200, 800, 800'
```

schaltest du die Kamera ein. Du siehst ein dauerhaftes Preview-Bild der Größe 800 Pixel mal 800 Pixel. Nichts wird gespeichert. Nimm die Fernbedienung eines Fernsehers, halte sie vor die Kamera und drücke einige Tasten. Auf dem Bildschirm siehst du zwei ungewöhnliche Dinge:

- Der schwarze Kunststoff vorne an der Fernbedienung erscheint völlig durchsichtig. Das Material schluckt keine IR-Strahlung.
- Vorne an der Fernbedienung ist ein pulsierendes Leuchten erkennbar. Über Infrarot sendet die Fernbedienung Steuersignale an den Fernseher.

Halte in der Nähe eines Fensters eine Pflanze vor die Infrarotkamera. Die grünen Blätter sehen ungewöhnlich hell aus. Sie erscheinen beinahe weiß (Wood-Effekt). Denn das Blattgrün lässt die IR-Strahlung des Sonnenlichtes durch und schluckt aber einen großen Teil des sichtbaren Lichts (alles außer Grün). Die Infrarotstrahlung wird von dem Wasser in den Zellen reflektiert und dann von der NoIR-Kamera aufgenommen.

Mit Strg + C kannst du die Ausführung von `raspistill` abbrechen.

Abb. C.1: *Eine Fernbedienung und eine Pflanze mit dunkelgrünen Blättern, aufgenommen mit dem NoIR-Modul*

Infrarot-Beleuchtung

Um die Kamera im Dunklen nutzen zu können, brauchst du eine Infrarot-Lichtquelle. Wenn du sehr an Nachtaufnahmen interessiert bist, lohnt sich für dich vielleicht die Anschaffung eines Infrarotscheinwerfers (ab etwa 5 € als Bausatz und ab 20 € als fertiges Gerät). Den kannst du auch für Nachtfotografie mit einer guten Digitalkamera verwenden.

Für Experimente mit dem Raspberry Pi reicht eine einzelne LED, die du über den RPi steuern kannst. Für etwa 2 € bekommst du ein Sortiment von zehn LEDs. Sicherlich sind IR-Leuchtdioden auch einzeln erhältlich. Infrarot-LEDs brauchen mehr Strom als normale LEDs (Tabelle C.1).

Färbung	Wellenlänge	Stromstärke	Vorwärtsspannung
klar	870 nm	100 mA	1,35 V
blau	925 nm	100 mA	1,35 V

Tabelle C.1: *Typische Werte für Infrarot-LEDs (Hersteller Kemo)*

Das bedeutet, dass du IR-LEDs nicht über die üblichen Ausgänge des GPIO steuern kannst. Du musst die LEDs über einen Widerstand mit Pin 2 (5 Volt) verbinden. Wenn du einen Lötkolben hast, löte den Widerstand an die LED und verbinde beides mit einem 1-m-langen Kabel. Du kannst die Kabelenden aber auch einfach verdrillen (Abbildung C.2).

Schneide ein ein female-female-Jumper-Kabel in der Mitte durch und löte die beiden Enden an das Kabel. Markiere z.B. das negative Ende mit einem Stück Klebeband. Nun kannst du deine LED leicht an Pin 2 (5 Volt) und Pin 6 (Masse) des GPIO anschließen. Mit Alufolie kannst du dir einen Reflektor basteln.

Abb. C.2: *Die IR-Leuchtdiode wird durch Verdrillen mit einem Kabel und einem 50-Ohm-Widerstand verbunden.*

Verdunkle das Zimmer und probiere deine IR-Leuchte aus!

Experiment: Wie verhalten sich Kellerasseln im Dunklen?

Kennst du Asseln? Diese kleinen krebsartigen Tiere findest du draußen an feuchten Orten, unter Laub oder totem Holz. Asseln sind extrem lichtscheu. Man sieht sie eigentlich niemals im Hellen. Wenn du auf einer Wiese ein Stück Holz umdrehst und Asseln darunter findest, krabbeln sie aufgeregt herum und verschwinden in Windeseile unter irgendeinem anderen Gegenstand. Aber wie verhalten sie sich im Dunkeln? Sitzen sie still an einer Stelle oder bewegen sie sich? Sind sie in Gruppen unterwegs oder verteilen sie sich so, dass jeder alleine ist?

Mit einem Umzugskarton kannst du ein Labor zur Verhaltensforschung in der Dunkelheit aufbauen. Auf die Oberseite des Kartons schneidest du kleine Lö-

cher und montierst den RPi mit Kamera und einer IR-Leuchtdiode zur Beleuchtung. Der Karton muss groß sein, damit die Kamera genügend Abstand zum Boden hat.

Bitte denke daran, die Asseln nach deinem Versuch wieder in ihre natürliche Umgebung zurückzubringen.

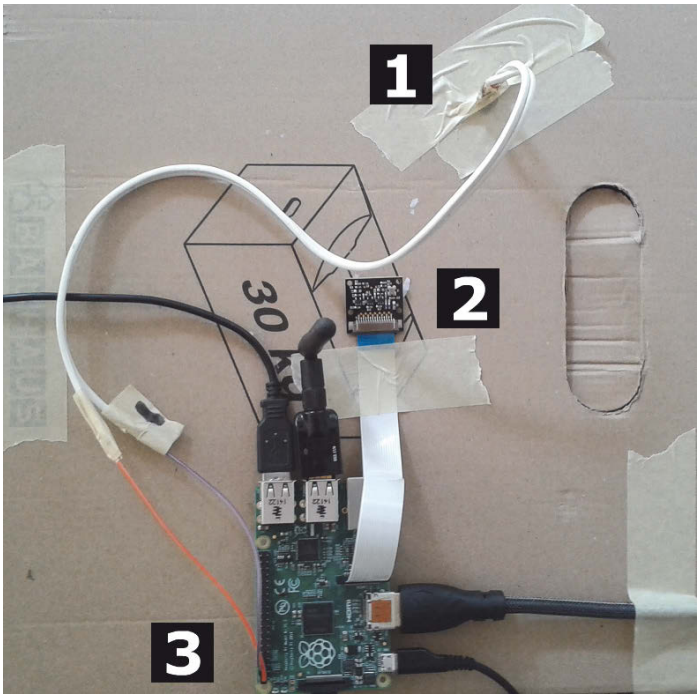


Abb. C.3: *Blick auf die Oberseite eines großen Umzugskartons. Du siehst einen Raspberry Pi mit Kamera (2). Die LED (1) ist an Pin 2 (+5 V) und an Pin 6 (Masse, 0 V) des GPIO angeschlossen (3).*

Übrigens, wenn du die Ergebnisse des Versuchs deutest, solltest du Folgendes bedenken: Eine Szenerie, die nur mit IR bestrahlt wird, ist für *uns Menschen* völlig dunkel. Aber vielleicht haben Asseln eine andere Wahrnehmung.