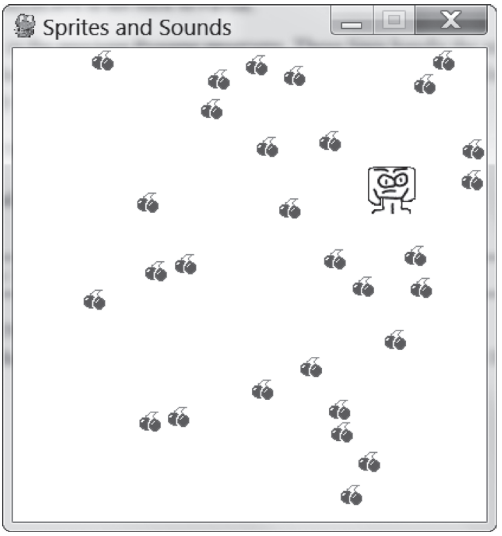


***Abb. 20–3*** *Ein Screenshot des Programms mit Sprites und Sounds*

## Der Code für das Sprites- und Sounds-Programm


Beginnen Sie eine neue Datei, geben Sie den folgenden Code ein und speichern Sie ihn als *spritesAndSound.py*. Sie können die Bild-und Tondateien, die wir in diesem Programm verwenden, von der Buch-Website unter *https://www.nostarch.com/inventwithpython/* herunterladen. Legen Sie diese Dateien im selben Ordner ab, in dem sich Ihr Programm befindet.

Wenn Sie beim Eingeben des Codes Fehlermeldungen erhalten, vergleichen Sie Ihren Code mit dem des Buches mit dem Diff Tool unter *https://www.nostarch.com/inventwithpython#diff*.

*spritesAndScounds.py*

```
 1. import pygame, sys, time, random
 2. from pygame.locals import *
 3.
 4. # Richtet pygame ein.
 5. pygame.init()
 6. mainClock = pygame.time.Clock()
 7.
 8. # Richtet das Fenster ein.
 9. WINDOWWIDTH = 400
10. WINDOWHEIGHT = 400
11. windowSurface = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT),
      0, 32)
12. pygame.display.set_caption('Sprites and Sounds')
13.
```

```
14. # Richtet die Farben ein.
15. WHITE = (255, 255, 255)
16.
17. # Richtet die Blockdatenstruktur ein.
18. player = pygame.Rect(300, 100, 40, 40)
19. playerImage = pygame.image.load('player.png')
20. playerStretchedImage = pygame.transform.scale(playerImage, (40, 40))
21. foodImage = pygame.image.load('cherry.png')
22. foods = []
23. for i in range(20):
24.     foods.append(pygame.Rect(random.randint(0, WINDOWWIDTH - 20),
          random.randint(0, WINDOWHEIGHT - 20), 20, 20))
25.
26. foodCounter = 0
27. NEWFOOD = 40
28.
29. # Richtet die Tastaturvariablen ein.
30. moveLeft = False
31. moveRight = False
32. moveUp = False
33. moveDown = False
34.
35. MOVESPEED = 6
36.
37. # Richtet die Musik ein.
38. pickUpSound = pygame.mixer.Sound('pickup.wav')
39. pygame.mixer.music.load('background.mid')
40. pygame.mixer.music.play(-1, 0.0)
41. musicPlaying = True
42.
43. # Führt die Spielschleife aus.
44. while True:
45.     # Prüft, ob ein QUIT-Ereignis eintritt.
46.     for event in pygame.event.get():
47.         if event.type == QUIT:
48.             pygame.quit()
49.             sys.exit()
50.         if event.type == KEYDOWN:
51.             # Ändert die Tastaturvariablen.
52.             if event.key == K_LEFT or event.key == K_a:
53.                 moveRight = False
54.                 moveLeft = True
55.             if event.key == K_RIGHT or event.key == K_d:
56.                 moveLeft = False
57.                 moveRight = True
58.             if event.key == K_UP or event.key == K_w:
59.                 moveDown = False
60.                 moveUp = True
61.             if event.key == K_DOWN or event.key == K_s:
62.                 moveUp = False
63.                 moveDown = True
```

```
64.         if event.type == KEYUP:
65.             if event.key == K_ESCAPE:
66.                 pygame.quit()
67.                 sys.exit()
68.             if event.key == K_LEFT or event.key == K_a:
69.                 moveLeft = False
70.             if event.key == K_RIGHT or event.key == K_d:
71.                 moveRight = False
72.             if event.key == K_UP or event.key == K_w:
73.                 moveUp = False
74.             if event.key == K_DOWN or event.key == K_s:
75.                 moveDown = False
76.             if event.key == K_x:
77.                 player.top = random.randint(0, WINDOWHEIGHT -
                      player.height)
78.                 player.left = random.randint(0, WINDOWWIDTH -
                      player.width)
79.             if event.key == K_m:
80.                 if musicPlaying:
81.                     pygame.mixer.music.stop()
82.                 else:
83.                     pygame.mixer.music.play(-1, 0.0)
84.                 musicPlaying = not musicPlaying
85.
86.         if event.type == MOUSEBUTTONUP:
87.             foods.append(pygame.Rect(event.pos[0] - 10,
                  event.pos[1] - 10, 20, 20))
88.
89.     foodCounter += 1
90.     if foodCounter >= NEWFOOD:
91.         # Fügt neue Nahrung hinzu.
92.         foodCounter = 0
93.         foods.append(pygame.Rect(random.randint(0, WINDOWWIDTH - 20),
              random.randint(0, WINDOWHEIGHT - 20), 20, 20))
94.
95.     # Zeichnet den weißen Hintergrund auf die Oberfläche.
96.     windowSurface.fill(WHITE)
97.
98.     # Bewegt die Figur.
99.     if moveDown and player.bottom < WINDOWHEIGHT:
100.        player.top += MOVESPEED
101.    if moveUp and player.top > 0:
102.        player.top -= MOVESPEED
103.    if moveLeft and player.left > 0:
104.        player.left -= MOVESPEED
105.    if moveRight and player.right < WINDOWWIDTH:
106.        player.right += MOVESPEED
107.
108.
```

```
    # Zeichnet den Block auf die Oberfläche.
    windowSurface.blit(playerStretchedImage, player)

    # Prüft, ob sich der Block mit einem Nahrungsquadrat überschneidet.
    for food in foods[:]:
        if player.colliderect(food):
            foods.remove(food)
            player = pygame.Rect(player.left, player.top,
              player.width + 2, player.height + 2)
            playerStretchedImage = pygame.transform.scale(playerImage,
              (player.width, player.height))
            if musicPlaying:
                pickUpSound.play()

    # Zeichnet die Nahrung.
    for food in foods:
        windowSurface.blit(foodImage, food)

    # Zeichnet das Fenster auf den Bildschirm.
    pygame.display.update()
    mainClock.tick(40)
```

## Das Fenster und die Datenstruktur einrichten

Der meiste Code dieses Programms entspricht dem aus dem Programm zur Kollisionserkennung aus Kapitel 19. Wir betrachten daher nur die Abschnitte, die Sprites und Toneffekte hinzufügen. Zuerst legen wir in Zeile 12 die Beschriftung der Titelzeile des Fensters fest, die unser Programm beschreibt:

```
12. pygame.display.set_caption('Sprites and Sounds')
```

Um die Beschriftung festzulegen, müssen Sie den String `('Sprites and Sounds')` an die Funktion `pygame.display.set_caption()` übergeben.

### Ein Sprite hinzufügen

Nachdem wir jetzt den Fenstertitel eingerichtet haben, benötigen wir unsere Sprites. Wir verwenden für den Spieler drei Variablen und nicht nur eine, wie im vorherigen Programm.

```
17. # Richtet die Blockdatenstruktur ein.
18. player = pygame.Rect(300, 100, 40, 40)
19. playerImage = pygame.image.load('player.png')
20. playerStretchedImage = pygame.transform.scale(playerImage, (40, 40))
21. foodImage = pygame.image.load('cherry.png')
```

Die Variable player in Zeile 18 speichert ein Rect-Objekt, das Position und Größe des Spielers verfolgt. Die Variable player enthält kein Bild des Spielers. Am Programmanfang wird die obere linke Ecke des Spielers auf (300, 100) festgelegt, und der Spieler hat eine Anfangsgröße und -breite von 40 Pixeln.

Die zweite Variable, die den Spieler darstellt, heißt playerImage (Zeile 19). Der Funktion pygame.image.load() wird ein String mit dem Dateinamen des zu ladenden Bilds übergeben. Der Rückgabewert ist ein Surface-Objekt, auf dem die Grafik der Bilddatei gezeichnet ist. Wir speichern dieses Surface-Objekt in playerImage.

### Die Größe eines Sprites verändern

In Zeile 20 verwenden wir im Modul pygame.tranform eine neue Funktion. Die Funktion pygame.transform.scale() kann ein Sprite verkleinern oder vergrößern. Das zweite Argument ist ein Tupel mit der neuen Breite und Höhe des Bilds im ersten Argument. Die Funktion scale() gibt ein Surface-Objekt mit einem in der neuen Größe gezeichneten Bild zurück. Im Programm dieses Kapitels vergrößern wir das Sprite des Spielers, je mehr Kirschen er ist. Wir speichern das Originalbild in der Variablen playerImage und das vergrößerte Bild in der Variablen playerStretchedImage.

In Zeile 21 rufen wir erneut load() auf, um ein Surface-Objekt mit der darauf gezeichneten Kirsche zu erzeugen. Achten Sie darauf, dass sich die Dateien player.png und cherry.jpg im selben Ordner wie die Programmdatei befinden, denn sonst kann pygame sie nicht finden und gibt eine Fehlermeldung aus.

## Die Musik und die Toneffekte einrichten

Als Nächstes müssen wir die Sound-Dateien laden. In pygame gibt es zwei Module für Sound. Das Modul pygame.mixer kann während des Spiels kurze Soundeffekte wiedergeben. Das Modul pygame.mixer.music spielt die Hintergrundmusik ab.

### Sound-Dateien hinzufügen

Rufen Sie die Konstruktorfunktion pygame.mixer.Sound() auf, um ein pygame.mixer.Sound-Objekt (oder kurz ein Sound-Objekt) zu erzeugen. Dieses Objekt verfügt über die Methode play(), die bei Aufruf den Toneffekt wiedergibt.

```
37. # Richtet die Musik ein.
38. pickUpSound = pygame.mixer.Sound('pickup.wav')
39. pygame.mixer.music.load('background.mid')
40. pygame.mixer.music.play(-1, 0.0)
41. musicPlaying = True
```

Zeile 39 ruft `pygame.mixer.music.load()` auf, um die Hintergrundmusik zu laden, und Zeile 40 `pygame.mixer.music.play()`, um sie abzuspielen. Der erste Parameter teilt `pygame` mit, wie oft die Hintergrundmusik nach dem ersten Start abgespielt werden soll. Wenn wir den Wert 5 übergeben, würde `pygame` die Hintergrundmusik insgesamt sechsmal abspielen. Wie übergeben hier den Parameter –1, ein Spezialwert, der die Musik unendlich oft wiederholt.

Der zweite an `play()` übergebene Parameter ist die Position in der Sounddatei, an der das Abspielen beginnen soll. Ein Wert von `0.0` spielt die Hintergrundmusik von Anfang an ab. Ein Wert von `2.5` lässt die Hintergrundmusik an der Position 2,5 Sekunden nach dem Anfang beginnen.

### Den Ton ein- und ausschalten

Die Taste M schaltet die Hintergrundmusik ein oder aus. Wenn `musicPlaying` auf `True` gesetzt ist, wird die Hintergrundmusik gerade abgespielt, und wir sollten sie anhalten, indem wir `pygame.mixer.music.stop()`aufrufen. Ist `musicPlaying` auf `False` gesetzt, wird die Hintergrundmusik im Moment nicht abgespielt, und wir sollten sie starten, in dem wir `play()` aufrufen. Die Zeilen 79–84 enthalten die Anweisungen, um das zu tun:

```
79.             if event.key == K_m:
80.                 if musicPlaying:
81.                     pygame.mixer.music.stop()
82.                 else:
83.                     pygame.mixer.music.play(-1, 0.0)
84.                 musicPlaying = not musicPlaying
```

Ob nun die Musik gespielt wird oder nicht, wir schalten den Wert in `musicPlaying` einfach um. Einen booleschen Wert umzuschalten, bedeutet, seinen Wert auf den genau entgegengesetzten Wert einzustellen. Die Zeile `musicPlaying = not musicPlaying` setzt die Variable auf `False`, wenn sie gerade `True` ist, und auf `True`, wenn sie gerade `False` ist. Stellen Sie sich das Umschalten wie das Betätigen eines Lichtschalters vor: Das Betätigen des Schalters bringt ihn in die entgegengesetzte Position.

## Den Spieler im Fenster zeichnen

Der in `playerStretchedImage` abgelegte Wert ist ein `Surface`-Objekt. Zeile 110 zeichnet das Sprite des Spielers mit `blit()` in das `Surface`-Objekt des Fensters (das in `windowSurface` gespeichert ist).

```
109.     # Zeichnet den Block auf die Oberfläche.
110.     windowSurface.blit(playerStretchedImage, player)
```

Der zweite Parameter der Methode `blit()` ist das `Rect`-Objekt, das angibt, wo auf dem `Surface`-Objekt das Sprite gezeichnet werden soll. Das Programm verwendet das in `player` abgelegte `Rect`-Objekt, das die Position des Spielers im Fenster festhält.

## Auf Kollisionen prüfen

Dieser Code entspricht dem des vorhergehenden Programms, enthält jedoch einige neue Zeilen:

```
114.         if player.colliderect(food):
115.             foods.remove(food)
116.             player = pygame.Rect(player.left, player.top,
                   player.width + 2, player.height + 2)
117.             playerStretchedImage = pygame.transform.scale(playerImage,
                   (player.width, player.height))
118.             if musicPlaying:
119.                 pickUpSound.play()
```

Wenn das Spieler-Sprite eine der Kirschen isst, vergrößert es sich in Höhe und Breite um zwei Pixel. In Zeile 116 wird dem neuen Wert von `player` ein neues `Rect`-Objekt zugewiesen, das zwei Pixel größer ist, als das alte `Rect`-Objekt.

Während das `Rect`-Objekt Position und Größe des Spielers festhält, wird sein Bild in `playerStretchedImage` als `Surface`-Objekt gespeichert. In Zeile 117 erzeugt das Programm ein neues installiertes Bild durch Aufrufen von `scale()`.

Ein Bild zu skalieren, verzerrt es häufig ein wenig. Wenn Sie ein bereits skaliertes Bild weiter skalieren, vergrößern sich auch die Verzerrungen. Wenn Sie jedoch jedes Mal das Ausgangsbild – indem wir `playerImage` und nicht `playerStretchedImage` übergeben – auf die aktuell geforderte Größe skalieren, verzerren Sie es nur einmal.

Zeile 119 ruft schließlich die Methode `play()` für das `Sound`-Objekt in der Variablen `pickUpSound` auf. Dies geschieht aber nur, wenn `music` auf `True` gesetzt ist (der Ton also eingeschaltet ist).

## Die Kirschen im Fenster zeichnen

In den vorherigen Programmen haben Sie die Funktion `pygame.draw.rect()` aufgerufen, um ein grünes Quadrat für jedes in der `food`-Liste abgelegte `Rect`-Objekt zu zeichnen. In diesem Programm wollen wir jedoch stattdessen die Kirsch-Sprites zeichnen. Rufen Sie die `blit()`-Methode auf und übergeben Sie das in `foodImage` abgelegte `Surface`-Objekt, auf dem sich die Kirschen-Bilder befinden:

```
121.     # Zeichnet die Nahrung.
122.     for food in foods:
123.         windowSurface.blit(foodImage, food)
```

Die Variable `food`, die die einzelnen `Rect`-Objekte in `foods` für jeden Durchlauf der `for`-Schleife enthält, teilt der `blit()`-Methode mit, wo das `foodImage` gezeichnet werden soll.

## Zusammenfassung

Sie haben Ihr Spiel um Bilder und Töne erweitert. Die Bilder, auch Sprites genannt, sehen viel besser aus als die einfachen Formen im vorherigen Programm. Sprites können kleiner oder größer skaliert werden, sodass wir sie in jeder gewünschten Größe ausgeben können. Das in diesem Kapitel vorgestellte Spiel hat außerdem einen Hintergrund und spielt Toneffekte ab.

Nachdem wir jetzt wissen, wie wir ein Fenster erstellen, Sprites anzeigen, grafische Grundformen zeichnen, Eingaben über Tastatur und Maus abfragen, Töne wiedergeben und Kollisionserkennung einsetzen, sind wir soweit, ein Grafikspiel in `pygame` zu schreiben. In Kapitel 21 werden all diese Elemente für unser bislang am weitesten fortgeschrittene Spiel kombiniert.