

Nutzerinteraktion in Processing

Grundgerüst eines Processing-Programms mit Interaktionsmöglichkeiten

Im Folgenden lernen Sie, wie Sie Ihre Programme so erweitern können, dass auf Maus- oder Tastatureingaben des Anwenders bzw. der Anwenderin reagiert werden kann. Dazu müssen im Grundgerüst eines Processing-Programms neben den Methoden `setup` und `draw` weitere vordefinierte Methoden ergänzt werden, die bei einem Ereignis wie einem Mausklick oder einer Tastatureingabe ausgeführt werden. Die Methoden haben dabei immer die Form `void methodenname() {...}`. In die geschweiften Klammern wird der Programmcode eingefügt, der ausgeführt werden soll, wenn das entsprechende Ereignis eintritt.

Reaktion auf Mausereignisse

Abb 1. zeigt ein Programm, das neben `setup` und `draw` die Methode `mouseClicked` enthält. Diese wird immer dann ausgeführt, wenn eine Maustaste geklickt wird, während sich der Mauszeiger innerhalb des Programmfensters befindet. Im Beispiel in Abb. 1 wird bei jedem Mausklick ein Quadrat an die Position des Mauszeigers gezeichnet. Dazu werden in Zeile 8 die Systemvariablen `mouseX` und `mouseY`, in denen Processing die aktuellen Koordinaten des Mauszeigers speichert, als Parameterwerte für die Koordinaten übergeben. Weitere Beispiele für Systemvariablen im Zusammenhang mit der Maus zeigt die Tabelle in Abb. 2.

```
1 void setup(){
2   size(400, 300);
3   background(255, 255, 0);
4   fill(255, 0, 0);
5 }
6 void draw(){}
7 void mouseClicked() { //diese Methode wird bei jedem Mausklick ausgeführt
8   square(mouseX, mouseY, 30); //mouseX und mouseY liefern die aktuelle Position des
                               //Mauszeigers
9 }
```

Abbildung 1: Reaktion auf Mausklick

Hinweis: Beachten Sie, dass ein Processing-Programm, das auf Nutzereingaben reagieren soll, immer die Methode `draw` enthalten muss, auch wenn die Methode keine Anweisungen enthält.

Systemvariable	Bedeutung
<code>mouseX</code>	enthält die aktuelle x-Koordinate der Maus
<code>mouseY</code>	enthält die aktuelle y-Koordinate der Maus
<code>pmouseX</code>	enthält die vorherige x-Koordinate der Maus
<code>pmouseY</code>	enthält die vorherige y-Koordinate der Maus
<code>mouseButton</code>	enthält die Werte LEFT, RIGHT oder CENTER, je nachdem welche Maustaste gedrückt wurde

Abbildung 2: Systemvariablen zum Zustand der Maus

Aufgabe 1: Erweitern Sie das Programm *Beispiel_Mausklick* in Abb. 1 so, dass beim Klicken der linken Maustaste ein Quadrat und beim Drücken der rechten Maustaste ein Kreis an die Position der Maus gezeichnet wird.

Neben `mouseClicked` gibt es noch weitere Methoden, die auf Mausereignisse reagieren. Einen Überblick gibt die Tabelle in Abb. 3. Ausführliche Erläuterungen finden Sie in der Referenz¹.

Methode	wird ausgeführt, ...
<code>mousePressed()</code>	sobald eine Maustaste gedrückt wird, unabhängig vom Zeitpunkt des Loslassens.
<code>mouseReleased()</code>	wenn eine Maustaste wieder losgelassen wird.
<code>mouseDragged()</code>	wenn die Maus mit gedrückter Maustaste bewegt wird.
<code>mouseMoved()</code>	wenn die Maus ohne gedrückte Maustaste bewegt wird.

Abbildung 3: Methoden für die Reaktion auf Ereignisse, die von der Maus ausgelöst werden

Aufgabe 2: Öffnen Sie das Programm *Vorlage_Smileys*, das beim Starten einen Smiley an eine festgelegte Position zeichnet. Erweitern Sie das Programm so, dass bei einem Mausklick ein weiterer Smiley an die Position des Mauszeigers gezeichnet wird.

Aufgabe 3: Implementieren Sie ein Programm, das eine Spur des Mauszeigers auf dem Bildschirm hinterlässt, wenn die Maus mit gedrückter Maustaste über den Bildschirm bewegt wird.

Aufgabe 4: Abb. 4 zeigt ein Programm, das es erlaubt, Rechtecke beliebiger Größe mit der Maus auf den Bildschirm zu zeichnen bzw. diese aufzuziehen.

a) Öffnen Sie das Programm *Beispiel_Rechtecke* und probieren Sie es aus. Erläutern Sie den Quelltext auch mithilfe von Abb. 5.

```

1  int x1;
2  int y1;
3  void setup(){
4      size(400, 300);
5      background(255, 255, 0);
6      fill(255, 0, 0);
7  }
8  void draw(){
9      if(mousePressed)
10         rect(x1, y1, mouseX - x1, mouseY - y1);
11
12 }
13 void mousePressed(){
14     x1 = mouseX;
15     y1 = mouseY;
16 }

```

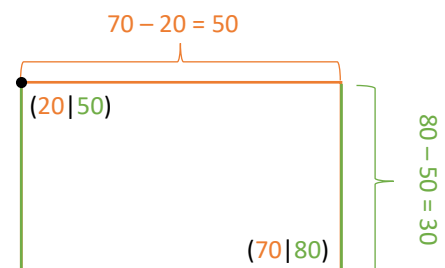


Abbildung 5: Berechnung der Höhe und Breite eines Rechtecks anhand der Koordinaten zweier gegenüberliegender Ecken

Abbildung 4: Programm für das Aufziehen von Rechtecken

- b) Testen Sie, ob und ggf. wie sich das Verhalten des Programms in Abb. 4 verändert, wenn Sie ...
- (1) das Zeichnen des Rechtecks in die Methode `mouseDragged` verlagern und die Methode `draw` leer bleibt.
 - (2) das Zeichnen des Rechtecks in die Methode `mouseReleased` verlagern und die Methode `draw` leer bleibt.
 - (3) In der `draw`-Methode als erstes die Anweisung `background(255, 255, 0);` einfügen.

¹ Ben Fry & Casey Reas. Processing – Reference. <https://processing.org/reference/> [letzter Zugriff: 10.04.2025]

- c) Ändern Sie das Programm aus Abb. 4 so, dass man einen Kreis aufziehen kann. Die Startposition legt dabei den Mittelpunkt des Kreises fest und die aktuelle Position des Mauszeigers, soll sich immer am Kreisrand befinden.

Exkurs: Lokale und globale Variablen

Vielleicht ist Ihnen aufgefallen, dass die Variablen in den Beispielprogrammen manchmal außerhalb der Methoden und manchmal innerhalb der Methoden deklariert wurden. Es stellt sich somit die Frage nach dem Unterschied und wonach entschieden wird, an welcher Stelle eine Variable deklariert werden muss.

Variablen, die außerhalb der Methoden deklariert werden, stehen während der gesamten Laufzeit des Programms zur Verfügung und können in jeder Methode verwendet werden. Man bezeichnet sie als **globale Variablen**. Ob sie oberhalb oder unterhalb der Methoden oder zwischen zwei Methoden deklariert werden, ist dabei eigentlich egal. Damit wir den Überblick leichter behalten, sollten wir uns aber angewöhnen, sie immer an den Anfang des Programms zu setzen.

Eine Variable, die innerhalb einer Methode deklariert wird, bezeichnet man als **lokale Variable**. Diese steht nur so lange zur Verfügung, bis die Ausführung der Methode oder des Blocks, in dem Sie deklariert wurde, beendet ist. Danach wird der Speicher für andere Zwecke wieder freigegeben.

Bei der Deklaration einer Variablen, sollten Sie also überlegen, ob diese global in mehreren Methoden oder nur lokal in einer benötigt wird.

Aufgabe 5: Im Ordner zu Aufgabe 5 finden Sie fünf kurze Programmbeispiele. Beispiel 1a und 1b sowie 2a und 2b unterscheiden sich jeweils in der Definition der verwendeten Variable als globale bzw. lokale Variable.

- Erläutern Sie, wie sich die Verwendung einer lokalen bzw. globalen Variablen auf das Verhalten des Programms auswirkt.
- Begründen Sie für Beispiel 1 und 2 jeweils, ob die Verwendung einer lokalen oder einer globalen Variablen sinnvoller ist.
- Erläutern Sie, weshalb in Beispiel 3 alle gezeichneten Quadrate die gleiche Größe haben. Verändern Sie das Programm so, dass die Quadrate jeweils um 10 Einheiten größer werden.

Simulation von Buttons

In Aufgabe 3 haben Sie ein Programm zum Zeichnen mit der Maus erstellt. Die Farbe und die Dicke der gezeichneten Linien waren dabei vermutlich fest vom Programm vorgegeben. Da wir auf Mausclicks reagieren können, können wir aber auch die Funktion von Buttons simulieren, um dem Anwender oder der Anwenderin weitere Eingaben zu ermöglichen, um die Eigenschaften der gezeichneten Linien selbst festzulegen.

Als Button können wir einfach ein Rechteck zeichnen und ggf. mithilfe der Methode `text` eine Beschriftung hinzufügen. Der Methode `text` werden ein Text und eine Position übergeben. Als Schriftfarbe wird die Farbe, die mit der Methode `fill` festgelegt wurde, verwendet. Die Schriftgröße kann mit der Methode `textSize` festgelegt werden.

Damit das Rechteck die Funktion eines Buttons übernimmt, müssen die Anweisungen für die entsprechende Funktion des Buttons immer dann ausgeführt werden, wenn sich der Mauszeiger bei einem Mausclick innerhalb des Button-Rechtecks befindet. Das kann anhand der Koordinaten des Rechtecks und der aktuellen Koordinaten des Mauszeigers überprüft werden. Dazu müssen mehrere

Bedingungen mit UND verknüpft werden. Eine UND-Verknüpfung von zwei Bedingungen erfolgt in Processing mit dem Operator `&&`.

Abb. 6 zeigt ein Programm, das bei einem Mausklick auf das Rechteck mit der Beschriftung „dicker“, die Liniendicke erhöht.

```
1  int dicke = 1;
2
3  void setup() {
4      size(400, 300);
5      background(255, 255, 0);
6      fill(200);
7      rect(10, 10, 50, 30);
8      fill(0);
9      text("dicker", 15, 30);
10 }
11
12 void draw() {}
13
14 void mouseClicked() {
15     if (mouseX > 10 && mouseX < 60 && mouseY > 10 && mouseY < 40) {
16         dicke++;
17         strokeWeight(dicke);
18     }
19 }
20
21 void mouseDragged() {
22     line(pmouseX, pmouseY, mouseX, mouseY);
23 }
```

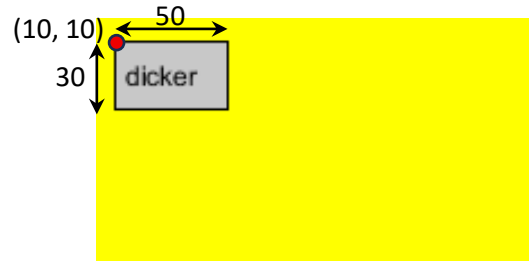


Abbildung 6: Button „dicker“

Abbildung 7: Zeichnen eines Buttons

Aufgabe 6: Öffnen Sie das Programm *Beispiel_Button* aus Abb. 6.

- Erläutern Sie den Quelltext des Programms in Abb. 6. Verwenden Sie Abb. 7, um die Bedingungen, die in der Methode `mouseClicked` geprüft werden, zu erklären.
- Erweitern Sie das Programm in Abb. 6 um einen „Button“, mit dem man die Dicke der gezeichneten Linie verringern kann.
Vorsicht: Achten Sie darauf, dass die Dicke nicht negativ wird.
- Erweitern Sie das Programm, um zwei verschieden farbige Buttons, mit denen man zwischen zwei Linienfarben wechseln kann.

Reaktion auf Tastaturereignisse

Um auf das Ereignis, dass eine Taste auf der Tastatur gedrückt wurde, zu reagieren, kann einem Processing-Programm die Methode `keyPressed` hinzugefügt werden. Der Inhalt der Methode wird bei jedem Tastendruck einmal ausgeführt. Alternativ können wir die Methode `keyTyped` verwenden. Anders als `keyPressed` wird die Methode `keyTyped` nicht ausgeführt, wenn Tasten gedrückt werden, die Steuerungsfunktionen übernehmen wie z. B. die *Shift*-Taste. Während `keyPressed` bei Eingabe von 'A' zweimal ausgeführt wird (einmal für das Drücken der Taste *Shift* und einmal für das Drücken der Taste 'a'), wird `keyTyped` nur einmal ausgeführt.

Welche Taste gedrückt wurde, ist in der Systemvariablen `key` gespeichert.

Beispiel für eine Programmsteuerung mithilfe von Tastatureingaben

In Aufgabe 1 wurde über die linke bzw. rechte Maustaste ausgewählt, ob ein Quadrat oder ein Kreis gezeichnet wird. Alternativ können wir unser Programm nun so erstellen, dass vor dem Zeichnen durch Drücken von Tasten ausgewählt werden kann, welche Figur gezeichnet werden soll. Abb. 8 zeigt ein entsprechendes Programm. Die Auswahl der Figur erfolgt über die Tasten 'q' für Quadrat bzw. 'k' für Kreis. Wird eine der beiden Tasten gedrückt, übernimmt das Programm den Wert der Systemvariablen `key` in die Variable `figur` vom Typ Zeichen (`char`), so dass dort das Zeichen 'q' oder 'k' gespeichert wird. Die Methode `mouseClicked` zeichnet nun ein Quadrat, wenn die Bedingung in Zeile 10 erfüllt ist, also der Anwender vorher Taste 'q' für Quadrat gedrückt hat, oder einen Kreis, wenn die Bedingung in Zeile 13 erfüllt ist, weil der Anwender Taste 'k' gedrückt hat.

```
1 char figur = 'x';
2 void setup(){
3     size(400, 300);
4     background(255, 255, 0);
5     fill(255, 0, 0);
6 }
7 void draw(){}
8
9 void mouseClicked(){
10    if(figur == 'q'){
11        square(mouseX, mouseY, 30);
12    }
13    if(figur == 'k'){
14        circle(mouseX, mouseY, 30);
15    }
16 }
17 void keyPressed(){
18    if(key == 'q' || key == 'k'){ //Verknüpfung von zwei Bedingungen mit ODER
19        figur = key;
20    }
21 }
```

Abbildung 8: Reaktion auf Tastatureingaben

Aufgabe 7: Erweitern Sie das Programm *Beispiel_FigurenZeichnen* in Abb. 8 um weitere Figuren, die mithilfe von Tasten ausgewählt und gezeichnet werden können.

Das Programm in Abb. 8 würde auch funktionieren, wenn wir in den Bedingungen innerhalb der Methode `mouseClicked` direkt die Systemvariable `key` abfragen würden, vorausgesetzt es werden nach 'q' oder 'k' keine weiteren Tasten gedrückt. Das Vorgehen, in der Methode `keyPressed` den Wert zunächst in einer Variablen zu speichern, ist jedoch hilfreich, wenn wir noch andere Eingaben über die Tastatur erlauben wollen. Deshalb wird der Wert von `key` nur in der Variablen `figur` gespeichert, wenn die Taste 'q' oder die Taste 'r' gedrückt wurde.

Wir könnten z. B. zusätzlich die Steuerung der Dicke der Umrandung über die Tastatur anbieten. Hierfür gibt es verschiedene Möglichkeiten. Wir können z. B. fünf Werte für die Dicke zur Verfügung stellen, die mithilfe der Tasten 1 bis 5 direkt ausgewählt werden. Wir können die Dicke aber auch ähnlich wie bei den beiden Buttons bei Drücken der Plus-Taste um 1 erhöhen und bei Drücken der Minus-Taste um 1 verringern, sofern sie aktuell größer als 1 ist. Abb. 9 zeigt die Umsetzung der zweiten Variante.

Wir benötigen eine globale Variable `dicke`, in der wir uns die aktuelle Dicke merken. Diese ist zu Beginn 1, deshalb weisen wir der Variablen `dicke` zu Beginn den Wert 1 zu. Die Methode `keyPressed` in Abb. 9 enthält nun im Vergleich zu Abb. 8 zwei weitere Bedingungen in Zeile 7 und Zeile 11. Wenn die Plus-Taste gedrückt wurde, wird die Variable `dicke` um 1 erhöht und die Methode `strokeWeight` zum Setzen der Dicke der Umrandung mit dem neuen Wert als Parameter ausgeführt. Wenn die Minus-Taste gedrückt wurde und die Variable `dicke` nicht schon beim Wert 1 angekommen ist, wird der Wert um 1 verringert und ebenfalls die Methode `strokeWeight` mit dem aktuellen Wert ausgeführt.

```
1  int dicke = 1;
2  void keyPressed() {
3
4      if(key == 'q' || key == 'k'){
5          figur = key;
6      }
7      if(key == '+'){
8          dicke++;
9          strokeWeight(dicke);
10     }
11     if(key == '-' && dicke > 1){
12         dicke--;
13         strokeWeight(dicke);
14     }
15 }
```

Abbildung 9: Unterscheiden der Tastatureingaben mithilfe mehrerer if-Konstruktionen

```
1  int dicke = 1;
2  void keyPressed() {
3      switch(key) {
4          case 'k':
5          case 'q': figur = key;
6                      break;
7          case '+': dicke++;
8                      strokeWeight(dicke);
9                      break;
10
11         case '-': if(dicke > 1){
12                     dicke--;
13                     strokeWeight(dicke);
14                 }
15                 break;
16     }
17 }
```

Abbildung 10: Unterscheiden der Tastatureingaben mithilfe einer Mehrfachverzweigung mit switch-case

Mehrfachverzweigungen

Je mehr Tasten in der Methode `keyPressed` zu berücksichtigen sind, desto mehr Fälle müssen unterschieden und entsprechende Bedingungen überprüft werden. Deshalb lohnt es sich, dafür eine etwas einfachere Schreibweise in Form einer Mehrfachverzweigung einzuführen, die in der Variante rechts in Abb. 10 zu sehen ist. Hinter dem Schlüsselwort `switch` in Zeile 3 steht in Klammern die Variable, die in jeder Bedingung überprüft werden soll (hier `key`). Jeder Wert, mit dem die Variable verglichen werden soll, steht hinter dem Schlüsselwort `case`. Optional kann abschließend hinter dem Schlüsselwort `default` noch angegeben werden, welche Anweisungen ausgeführt werden sollen, wenn keiner der Fälle zutrifft.

Das Beispiel in Abb. 10 unterscheidet die vier Fälle, dass die Systemvariable `key` die Werte 'k', 'q', '+' oder '-' annimmt. Hinter dem Doppelpunkt stehen dann die Anweisungen, die in dem jeweiligen Fall ausgeführt werden sollen. Es fällt auf, dass in Zeile 4 für den Fall 'k' kein Befehl angegeben ist. Das liegt daran, dass ab dem Fall, der zutrifft, auch die Anweisungen, die bei allen nachfolgenden Fällen stehen, ausgeführt werden, bis die Anweisung `break` gefunden wird. Da für den Fall 'k' und 'q' die gleiche Aktion ausgeführt werden soll, muss diese nur einmal hinter dem zweiten Fall angegeben werden. Da bei '+' und '-' unterschiedliche Anweisungen ausgeführt werden sollen, muss in Zeile 9 und Zeile 15 die Anweisung `break` stehen.

Aufgabe 8: Erweitern Sie das Programm *Abb10_SwitchCase* aus Abb. 10 um weitere Optionen, die mithilfe der Tasten ausgewählt werden können, z. B:

- ob die Form gefüllt sein soll oder nicht
- die Farbe der Füllung (z. B. können fünf Farben zur Auswahl mit Tasten belegt werden)
- die Größe der Figur

Nutzereingaben über ein Eingabefeld

Spätestens bei der Wahl der Farbe oder Größe in Aufgabe 8 fällt auf, dass es relativ umständlich ist, komplexere Tastatureingaben, die eigentlich aus mehr als einem Zeichen bestehen, mithilfe der Methoden `keyPressed` oder `keyTyped` zu erfassen.

Um Texteingaben zu erfassen, bietet es sich daher an, eine Java-Bibliothek in das Processing-Programm einzubinden, die es ermöglicht, ein Eingabefeld anzuzeigen, in das der Anwender oder die Anwenderin einen Text eingeben kann.

Abb. 11 zeigt ein Processing-Programm, bei dem über ein Eingabefeld die Größe des zu zeichnenden Quadrats eingegeben werden kann. In der ersten Zeile des Programms wird mit dem folgenden Befehl die benötigte Java-Bibliothek eingebunden:

```
import static javax.swing.JOptionPane.*;
```

Durch das Einbinden der Bibliothek steht die Methode `showInputDialog` zur Verfügung, die in Zeile 9 als Reaktion auf einen Mausklick aufgerufen wird, um ein Dialogfenster zu öffnen. Als Parameter wird der Methode `showInputDialog` ein Text übergeben, der dem Anwender als Erklärung oder Aufforderung zu dem Eingabefeld im Dialogfenster angezeigt werden soll. Der Rückgabewert der Methode ist die Eingabe des Anwenders oder der Anwenderin als Zeichenkette (`String`)². Mit der folgenden Codezeile würde beispielsweise die Eingabe in der Variablen `antwort` vom Typ Zeichenkette gespeichert:

```
String antwort = showInputDialog("Wie heißt du?");
```

Wenn eine Zahl als Eingabe benötigt wird, kann die eingegebene Zeichenkette in eine Ganzzahl oder Gleitkommazahl umgewandelt werden. Voraussetzung ist, dass die Eingabe nur aus Ziffern besteht. Die Umwandlung geschieht in Abb. 11 in Zeile 9 mithilfe der Methode `Integer.parseInt`. Die Breite, die der Anwender oder die Anwenderin eingegeben hat, wird als Ganzzahl in der Variablen `breite` gespeichert und dann in Zeile 10 in der Methode zum Zeichnen des Quadrates verwendet.

```
1 import static javax.swing.JOptionPane.*;
2 void setup() {
3     size(400, 300);
4     background(255, 255, 0);
5     fill(255, 0, 0);
6 }
7 void draw() {}
8 void mouseClicked() {
9     int breite = Integer.parseInt(showInputDialog("Bitte gib die Breite
10                                     des Quadrates ein!"));
11     square(mouseX, mouseY, breite);
12 }
```

Abbildung 11: Nutzereingabe über ein Textfeld

² Der Umgang mit Zeichenketten wird in einem separaten Leitfaden ausführlich thematisiert

Aufgabe 9: Erstellen Sie ein Programm, das bei einem Mausklick ineinander geschachtelte Kreise oder Quadrate wie in Abb. 12 an der Position des Mauszeigers zeichnet. Der Anwender oder die Anwenderin soll bei jedem Mausklick eingeben können, wie viele Figuren ineinander geschachtelt werden.

Sie können für das Programm die Vorlage *Vorlage_FigurenSchachteln* verwenden oder ein eigenes Programm erstellen.

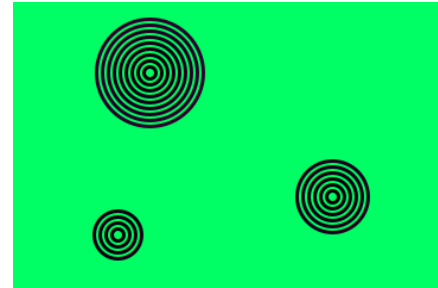


Abbildung 12: Mögliche Ausgabe des Programms zu Aufgabe 9

Projekte

Sie haben nun verschiedene Konzepte der Nutzerinteraktion kennengelernt. Verwenden Sie diese, um ein etwas umfangreicheres Programm selbständig zu erstellen.

Projektidee 1: Kombinieren Sie das Gelernte aufbauend auf Aufgabe 6 zu einem etwas komplexeren Zeichenprogramm, das weitere Funktionen enthält. Folgende Funktionen könnten beispielsweise noch ergänzt werden:

- Ein Radiergummi, mit dem Teile der Zeichenfläche wieder gelöscht werden können.
- Vollständiges Löschen der bisherigen Zeichnung
- Weitere Buttons zur Auswahl der Farben, Formen oder anderer Optionen
- Mit der Maus angeklickte Positionen werden mit Linien verbunden, so dass ein Polygon entsteht
- Speichern eines gezeichneten Bildes mithilfe der Methode `save` (s. Referenz).

Projektidee 2: Verwenden Sie die Zeichenfunktionen von Processing, um ein Pong-Spiel zu erstellen.

Zwei schwarze Rechtecke dienen als Schläger. Der linke Schläger kann z. B. mithilfe zweier Tasten auf der Tastatur hoch und runter bewegt werden, der linke Schläger mithilfe der linken und der rechten Maustaste. Die Spielenden müssen die Schläger so positionieren, dass der Ball vom Schläger abprallen kann. Fliegt der Ball aus dem linken oder rechten Spielfeldrand, hat der jeweilige Spielende, der den Ball nicht aufhalten konnte, verloren. Am oberen und unteren Rand prallt der Ball ab.

Ergänzen Sie Ihr Spiel um weitere Funktionalitäten, beispielsweise um eine Anzeige des Punktestands.

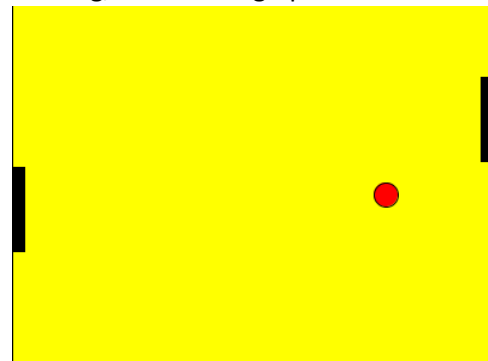


Abbildung 13: Exemplarisches Programmfenster zur Projektidee Pong-Spiel

Hinweis: Damit der Ball nicht flackert und sich nicht zu schnell bewegt, kann es hilfreich sein mithilfe der Methode `frameRate` die Anzahl der pro Sekunde gezeichneten Bilder herunterzusetzen, z. B. `frameRate(4)`.

Projektidee 3: Gestalten Sie ein Quiz mit drei Fragen nach dem Prinzip von „Wer wird Millionär?“. Dem Anwender wird jeweils eine Frage mit 4 Antwortmöglichkeiten angezeigt. Die Auswahl der Antwort kann über Buttons oder über die Tastatur erfolgen.

Dieses Werk ist lizenziert unter einer [Creative Commons Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz](#).

Für die korrekte Ausführbarkeit der Quelltexte in diesem Leitfaden und der beiliegenden Quelltexte zu den Beispielen und Aufgaben wird keine Garantie übernommen. Auch für Folgeschäden, die sich aus der Anwendung der Quelltexte oder durch eventuelle fehlerhafte Angaben ergeben, wird keine Haftung oder juristische Verantwortung übernommen.