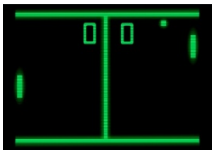


Aufgabenblatt 1: Eigenschaften von Video-/Computerspielen

Sie haben sicher schon einmal ein Computerspiel oder ein Videospiel gespielt. Heute gibt es ganz unterschiedliche Spiele. Hier sollen Sie sich überlegen, was denn ein Spiel ausmacht.

Aufgabe 1:

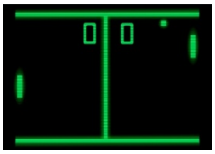
Was für Spiele kennen Sie. Geben Sie hier möglichst unterschiedliche Spiele an

Aufgabe 2:

Wie kann mit den Spielfiguren interagiert werden (in Kontakt getreten werden)?

Aufgabe 3:

Was haben unterschiedliche Spiele gemeinsam? Worin unterscheiden sie sich? Aus was für Grundelementen bestehen die Spiele?



Aufgabenblatt 2: geometrische Figuren

Ein Merkmal von Computerspielen ist, dass sie in einer bestimmten Welt integriert sind. Dabei kann es sich um ein Spielbrett, wie man es auch von „realen“ Spielen kennt oder eine virtuelle Umgebung handeln. In dieser Aufgabe sollen Sie Elemente kennen lernen, mit denen solche Umgebungen und Spielfiguren auf dem Computer gezeichnet werden können.

Aufgabe 1: Bilder aus Einzelteilen erzeugen

Überlegen Sie sich, wie Sie die folgenden Bilder erstellen können. Dabei dürfen Sie nur die folgenden geometrischen Objekte verwenden:

- Punkte
- Linien
- Rechtecke
- Ellipsen (Kreise)

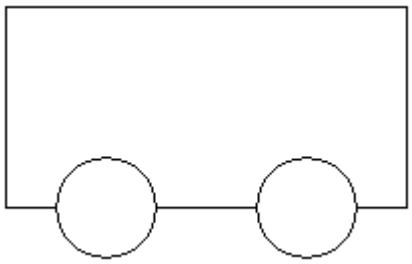


Bild 1

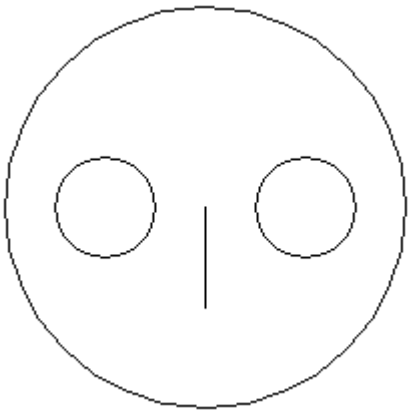


Bild 2

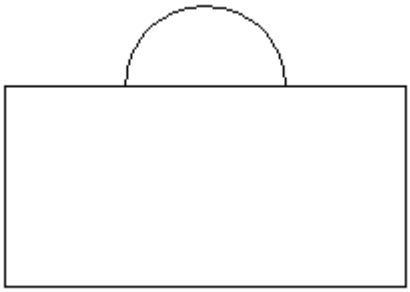


Bild 3

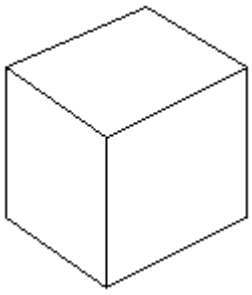
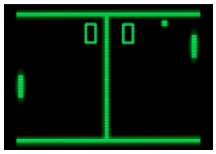
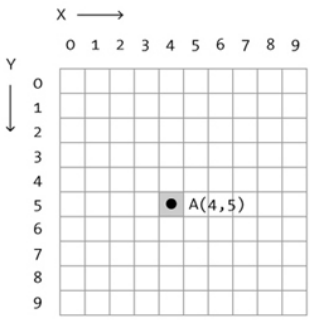
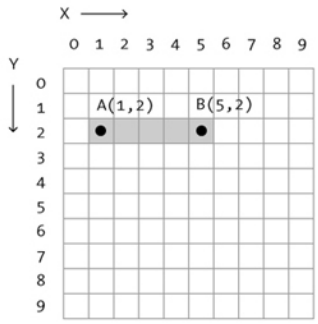
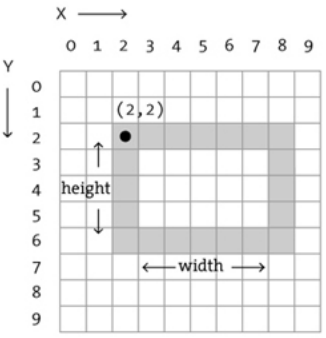
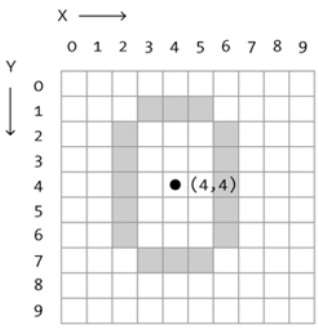
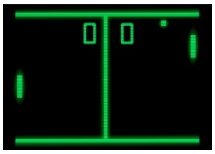


Bild 4



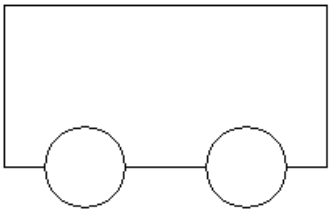
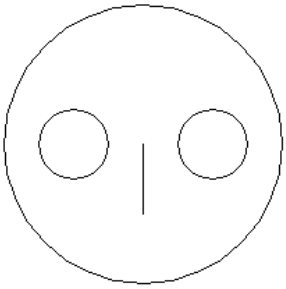
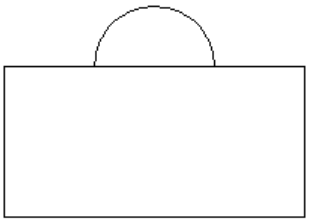
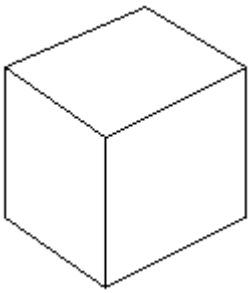
Die Anweisungen für den Computer zum Zeichnen von geometrischen Objekten sind in der folgenden Tabelle zusammengefasst.

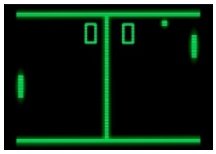
	Beispiel
Ein **Punkt** braucht zwei Angaben, eine x- und eine y-Position. Ausgehend von der oberen linken Ecke wird dann genau der Pixel ausgefüllt, der an dieser Position liegt. In Programmiersprache heisst das `point(4,5)`	
Eine **Linie** besteht aus einem Anfangs und einem Endpunkt. Diese Punkte haben jeweils eine x- und eine y-Koordinate. Im Vergleich zum Punkt braucht es also 4 Werte, damit der Computer weiss, wie die Line aussieht. Im Beispiel haben wir: `line(1,2,5,2)` Die roten Werte bezeichnen die Koordinaten des ersten, und die grünen Werte die Koordinaten des zweiten Punktes.	
Ein **Rechteck** hat eine Position (obere/linke Ecke) und eine Breite (engl. width) und eine Höhe (engl. height). Auch um ein Rechteck zu zeichnen braucht der Computer vier Werte. Zeichnen des Beispiels: `rect(2,2,7,5)` Die roten Werte bezeichnen die obere/linke Ecke. Der grüne Wert gibt an, wie viele Pixel das Rechteck breit ist, und der blaue wie viele Pixel das Rechteck hoch ist. Merke: Sind Höhe und Breite gleich, ist es ein **Quadrat**.	
Eine **Ellipse** hat einen Mittelpunkt und eine Breite (engl. width) und eine Höhe (engl. height). Zeichnen des Beispiels: `rect(4,4,5,7)` Die roten Werte bezeichnen die obere/linke Ecke. Der grüne Wert gibt an, wie viele Pixel das Rechteck breit ist, und der blaue wie viele Pixel das Rechteck hoch ist. Merke: Sind Höhe und Breite gleich, ist es ein **Kreis**.	



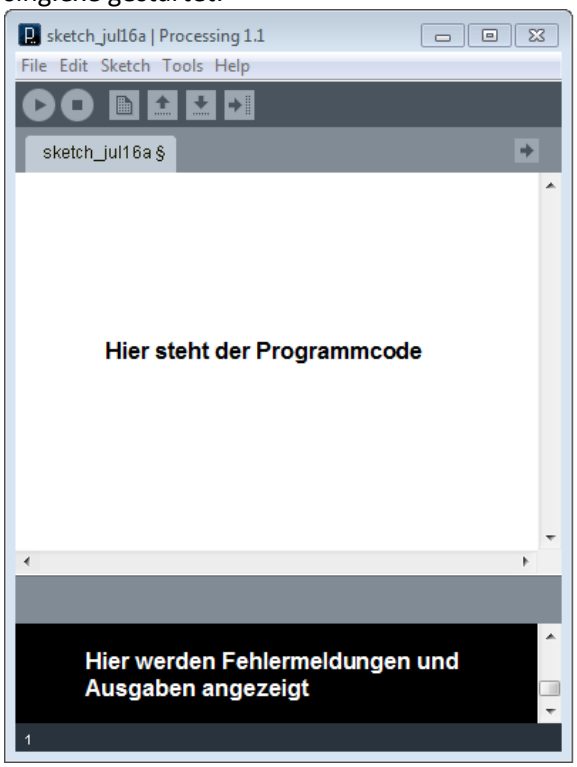
Aufgabe 2: In Computersprache übersetzen

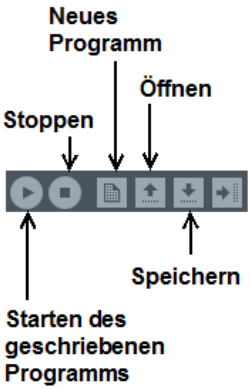
Überlegen Sie Sich, wie sie die Objekte aus Aufgabe 2.1 so formulieren können, dass es der Computer auch versteht.

Bild	Code
	ellipse(100,100,150,50);
	
	
	
	



Nun wollen wir unser erstes Programm schreiben. Das Programm mit dem wir dies tun, heisst Processing. Wenn das Programm nicht bereits geöffnet ist, wird es einfach durch Doppelklick auf processing.exe gestartet.





Aufgabe 3: Erstes Programm schreiben

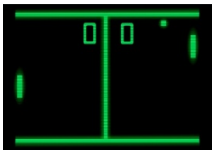
Geben Sie im Feld, in dem der Programmcode stehen soll folgendes ein und starten Sie das Programm mit dem Pfeilsymbol:

```
void setup(){
  size(400,400);
  rect(20,40,100,50);
}
```

Zwischen den geschweiften Klammern { } stehen die Anweisungen an den Computer welche unsere „Szene“ zeichnen sollen. Als erstes wird hier jeweils noch angegeben, wie gross die Spielfläche (size) sein soll. In diesem Fall ist es eine Fläche der Grösse 400 Pixel auf 400 Pixel.

Aufgabe 4: Eigene Objekte zeichnen

Ersetzen Sie nun den Befehl rect(20,40,100,50) durch die Anweisungen, welche Sie in Aufgabe 2 aufgeschrieben haben.



Aufgabenblatt 3: Farben

Bis hier waren unsere Bilder nur aus den Farben Grau, Weiss und Schwarz. In dieser Aufgabe soll nun Farbe in unsere Szene kommen.

Aufgabe 1: Die Hintergrundfarbe

Die Hintergrundfarbe kann mit dem Befehl background() gesetzt werden. Diese Anweisung benötigt drei Werte. Der erste Wert gibt an, wie gross der Rot-Anteil ist, der zweite Wert gibt an wie gross der Grün-Anteil ist und der dritte Wert gibt an, wie gross der Blau-Anteil ist. Diese Farben werden auch **RGB**-Farben genannt (Rot-Grün-Blau-Farben). Die Werte liegen bei allen drei Farben zwischen 0 (nicht vorhanden) und 255 (so viel wie möglich).
Nehmen Sie das folgende Programm als Ausgangslage:

```
void setup(){
  size(400,400);
  background(0,0,0);
}
```

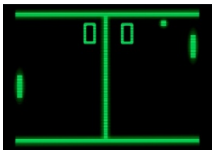
Füllen Sie nun die folgende Tabelle aus. Das heisst, geben Sie an, welche Hintergrundfarbe bei den gegebenen Farben erscheint.

Rot	Grün	Blau	Farbe ?
0	0	0	
255	255	255	
255	0	0	
0	255	0	
0	0	255	
255	255	0	
0	255	255	
255	0	255	
120	0	0	
100	255	255	
255	150	150	

Natürlich können auch geometrische Formen farbig gezeichnet werden:
Mit der Anweisung fill() kann die Farbe geändert werden, mit der die gezeichneten Objekte ausgefüllt werden. Auch hier werden wieder die drei Werte für Rot, Grün und Blau angegeben.
Beispiel:

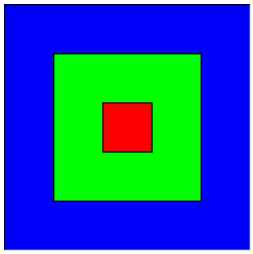
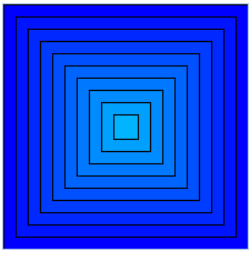
```
void setup(){
  size(400,400);
  fill(0,0,0);
  rect(100,100,200,200);
}
```

Welche Farbe hat das gezeichnete Quadrat?

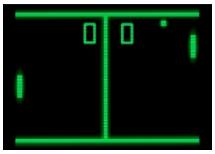


Aufgabe2: Farbige Figuren Zeichnen

Zeichnen Sie die folgenden Zwei Figuren und schreiben Sie den Code in die Tabelle:

HINWEIS (freiwillig): mit dem Befehl noStroke() werden die schwarzen Ränder der Rechtecke nicht mehr gezeichnet.



Aufgabenblatt 4: Variablen

In allen Computerspielen können sich die Zustände (Werte) des Hintergrunds, der Spielfiguren oder des Spielers verändern. Diese Zustände müssen irgendwo gespeichert werden. Z.B. möchte man sich merken, wo sich der Spielball, der Gegner oder das Ziel gerade befindet. Dazu werden Variablen benötigt.

Werte in Processing

In Processing gibt es einige Schlüsselwörter, welche für Werte stehen. Einige Beispiele dafür sind:

- height: Höhe des Fensters
- width: Breite des Fensters
- mouseX: x-Koordinate der Maus
- mouseY: y-Koordinate der Maus

Soll nun eine Linie von Links-Oben nach rechts-unten gezeichnet werden, kann das auch geschrieben werden als:

```
line(0,0,width,height);
```

Mit diesen Werten kann auch gerechnet werden

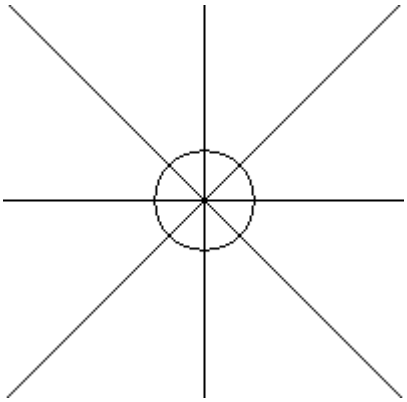
Code	Bedeutung
+	Addition
-	Subtraktion
/	Division (ganzzahlig)
*	Multiplikation

Soll also ein Kreis in der Mitte des Fensters gezeichnet werden, sieht das so aus:

```
ellipse(width/2,height/2,50,50);
```

Aufgabe 1:

Zeichnen Sie mit Hilfe von Rechenoperationen und den genannten Schlüsselwörtern das folgende Bild.



Das Bild soll immer gleich gezeichnet werden, auch wenn die Werte in size() verändert werden.

Aufgabe 2:

Zeichnen Sie mindestens eines der Bilder aus Aufgabenblatt 2 so, dass sie immer in der Mitte des Fensters sind, egal was für eine Grösse für das Fenster am Anfang gewählt wird.

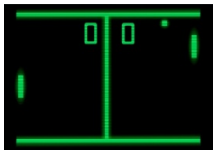
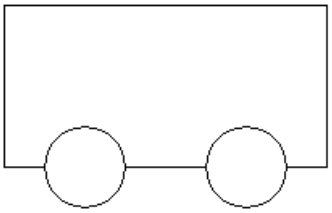
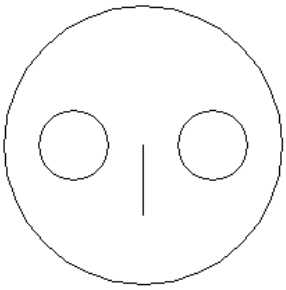
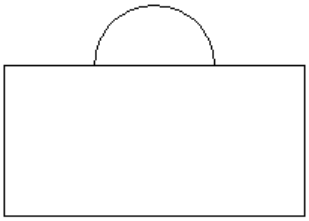


Bild	Code
	
	
	

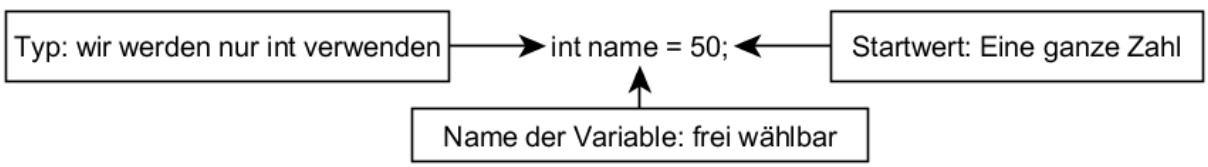
Testen der Zentrierung

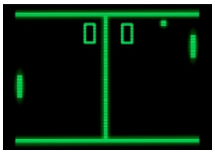
Nun können Sie Ihren Code testen, indem Sie den Code für das Zeichnen im folgenden Programm an der markierten Stelle einfügen.

```
void setup(){
  size(400,400);
// bedeutet, dass das Fenster eine veränderbare Grösse hat.
  frame.setResizable(true);
}
void draw(){
  // Zeichencode
}
```

Eigene Variablen (für die etwas Schnelleren; wird für "Pong" gebraucht)

Deklaration von Variablen:





Aufgabe 3: Erste Variable

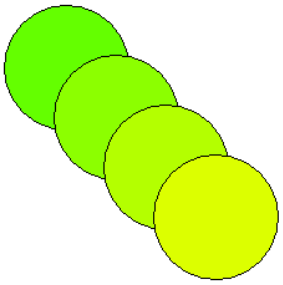
In dieser Aufgabe werden wir eine Variable definieren, und deren Wert anschliessend zum Zeichnen verwenden. Die Definition der Variable muss ganz am Anfang des Programms stehen. Die Variable mit dem Namen pos wird auf den Startwert 100 gesetzt. Anschliessend wir dieser Wert (50) an vier Stellen im Programm verwendet. Bei der Verwendung kann dem Wert auch etwas hinzugezählt, oder abgezogen werden.

```
int pos=100;
void setup(){
  size(400,400);
  background(255,255,255);
  ellipse(pos, 100, 100, 100);
  ellipse(pos+10, 100, 100, 100);
  ellipse(pos+20, 100, 100, 100);
  ellipse(pos+30, 100, 100, 100);
}
```

Kopieren Sie den obigen Code, testen Sie ihn und machen Sie anschliessend folgende Änderungen:

a. Ändern Sie die Variable pos so, dass der Mittelpunkt des ersten Kreises nicht an der Position 100/100 ist, sondern an der Position 50/100.
b. Ändern Sie das Programm so ab, dass sich die Kreise weniger überlappen.
c. Verwenden Sie die Variable pos auch noch für das Setzen bzw. Veränderung der Farbe.
d. Der Kreis soll diagonal wandern. Ändern Sie den Code entsprechend.

Das Resultat könnte am Ende z.B. so aussehen:



Aufgabe 4: Variable verändern

In der Aufgabe 3 wurde der am Anfang in der Variablen gespeicherte Wert immer wieder verwendet, aber nicht verändert. Der folgende Code macht genau das gleiche wie der gegebene Code aus Aufgabe 3. Nun wird der Wert von pos aber jeweils verändert.

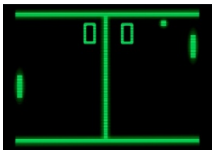
```
int pos= 100;
void setup(){
  size(400,400);
  background(255,255,255);
  ellipse(pos, 100, 100, 100);
  pos=pos+10;
  ellipse(pos, 100, 100, 100);
  pos=pos+10;
  ellipse(pos, 100, 100, 100);
  pos=pos+10;
```

Wert von pos = 100

Wert von pos = 110

Wert von pos = 120

Wert von pos = 130

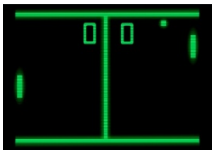


```
  ellipse(pos, 100, 100, 100);
}
```

Kopieren Sie den obigen Code, testen Sie ihn und machen Sie anschliessend wieder folgende Änderungen:

a. Ändern Sie die Variable pos so, dass der Mittelpunkt des ersten Kreises nicht an der Position 100/100 ist, sondern an der Position 50/100.
b. Ändern Sie das Programm so ab, dass sich die Kreise weniger überlappen.
c. Verwenden Sie die Variable pos auch noch für das Setzen bzw. Veränderung der Farbe.
d. Der Kreis soll diagonal wandern. Ändern Sie den Code entsprechend.

Welche Variante bevorzugen Sie? Die Version von Aufgabe 3 oder von Aufgabe 4?



Aufgabenblatt 5: Interaktion

Bis jetzt haben wir einfach statische Objekte gezeichnet. In einem Computerspiel gibt es aber immer auch Elemente, welche sich bewegen, seien dies Spielfiguren oder Gegner. In diesem Aufgabenblatt geht es nun um die Interaktion mit dem Programm. Das Programm soll einerseits auf die Position der Maus, und andererseits auf Tasten reagieren.

Aufgabe 1: Verändern der Hintergrundfarbe abhängig von der Mausposition

Wir wollen mit einem neuen Basisprogramm starten:

```
void setup(){
  size(400,400);
}
void draw(){
  background(255,255,255);
}
```

a) Übernehmen Sie dieses Programm und starten Sie es.
b) Ändern Sie das Programm nun wie folgt ab:

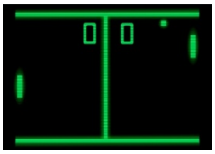
```
void setup(){
  size(400,400);
}
void draw(){
  background(mouseX,mouseY,255);
}
```

Starten Sie das Programm erneut, und beobachten Sie was passiert. Wie können Sie sich das erklären?

Aufgabe 2: Zeichnen einer Figur an der Mausposition

Übernehmen Sie nun das folgende Programm:

```
void setup(){
  size(400,400);
  background(255,255,255);
}
void draw(){
  fill(0,0,0);
  ellipse(mouseX,mouseY,10,10);
}
```

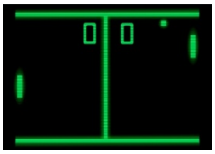


Alles was fett markiert ist, wird nun von Processing immer wieder aufgerufen. Die Variablen mouseX und mouseY sind wie height und width Variablen von Processing. Diese beiden bezeichnen die Position der Maus auf dem Bildschirm. Notieren Sie Sich die Lösungen der folgenden Aufgaben oder speichern Sie den Programmcode ab.

a) Ändern Sie das Programm so ab, dass anstelle der Ellipse ein Rechteck an der Position der Maus gezeichnet wird.
b) Ändern Sie das Programm so ab, dass anstelle der Ellipse ein Punkt gezeichnet wird.

Zusatzaufgaben (freiwillig)

c) Ändern Sie das Programm so ab, dass zwischen der Mausposition und der oberen linken Ecke eine Linie gezeichnet wird.
d) Überlegen Sie, wie Sie die Grösse des zu zeichnenden Objekts von der Position der Maus abhängig machen können. Erstellen Sie dazu ein Programm.



Aufgabenblatt 6: Verzweigungen (Bedingte Programmausführung)

Aufgabe 1: Nur zeichnen, wenn die Maus gedrückt ist

Wir starten mit dem letzten Programm aus Aufgabenblatt 5.

```
void setup(){
  size(400,400);
  background(255,255,255);
}
void draw(){
  fill(0,0,0);
  ellipse(mouseX,mouseY,10,10);
}
```

Der Kreis soll hier nur gezeichnet werden, wenn die Maus gedrückt ist. Zum Testen, ob die Maus gedrückt ist, wird die Befehlsfolge `if (test){…}` verwendet.
Ersetzen Sie nun die Zeile `ellipse(mouseX, mouseY, 10,10)` durch die Zeilen:

```
if (mousePressed){
  ellipse(mouseX, mouseY, 10,10);
}
```

Starten Sie Ihr Programm.
Das Programm zeichnet jetzt nur, wenn die Maus gedrückt wird (engl. pressed = gedrückt).

Aufgabe 2: Nur im oberen Teil des Feldes zeichnen

Man kann natürlich auch mehr testen, als nur, ob die Maus gedrückt ist oder nicht. Hier sollen Sie nur noch einen Teil der Fläche für das Zeichnen zulassen.

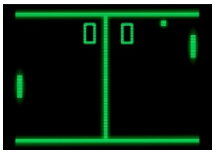
```
void setup(){
  size(400,400);
  background(255,255,255);
}
void draw(){
  if (mouseX > 200){
     ellipse(mouseX, mouseY, 10,10);
  }
}
```

Ersetzen Sie den mousPressed durch mouseX>200. Was macht das Programm?

- Ändern Sie das Programm so ab, dass nur im unteren Bereich des Feldes gezeichnet werden kann
- Ändern Sie das Programm so ab, dass nur links oder rechts gezeichnet werden kann.

Verwenden Sie dazu die folgenden Operationen:

Symbol	Beschreibung	Beispiel
<	Testet, ob das links kleiner ist als das rechts	mouseX<200
>	Testet, ob das links grösser ist als das rechts	mouseX>200
<=	Testet, ob das links kleiner oder gleich ist wie das rechts	mouseX<=200
>=	Testet, ob das links grösser oder gleich ist wie das rechts	mouseX<=200



Aufgabe 3 (für die etwas schnellen): Zwei Begrenzungen einführen

Wir wollen nun nicht nur eine Bedingung, sondern mehrere auf einmal testen. Dafür können wir die einzelnen Test zusammenhängen.

Symbol	Beschreibung	Beispiel
&	Testet, ob das links und das rechts zutrifft	(mousePressed) & (mouseX<10)
|	Testet, ob das links oder das rechts zutrifft	(mousePressed) | (mouseX<10)

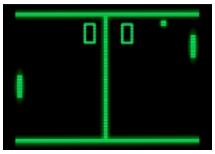
Ändern Sie Ihr Programm so ab, dass nur gezeichnet wird, wenn:

- die Maus gedrückt wird, und man sich im oberen Bereich befindet.
- die Maus gedrückt wird oder man sich im oberen Bereich befindet.
- man sich im oberen rechten Bereich befindet
- man sich im unteren linken Bereich befindet

Sie können natürlich auch wieder mit Farben arbeiten.
Ändern Sie Ihr Programm so ab, dass

- Im oberen Bereich rot gezeichnet wird und im unteren blau

Speichern oder notieren Sie sich jeweils den Programmcode



Aufgabenblatt 7a: Geheimbotschaft entschlüsseln

In dieser Aufgabe sollen Sie eine geheime Botschaft in einem Programm verstecken. Der Benutzer des Programms findet die Botschaft indem er sie durch „Zeichnen“ aufdeckt. Kopieren Sie als Beispiel den folgenden Code.

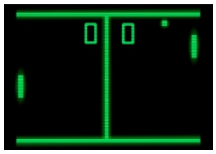
```
void setup(){
  size(400,400);
  background(255);
  noStroke();
}

void draw(){
  if (mousePressed){
    rect(mouseX,mouseY,10,10);
    if (mouseX<200){
      if (mouseX>50){
        if (mouseY<120){
          if (mouseY>100){
            fill(255,0,0);
            rect(mouseX,mouseY,10,10);
            fill(255,255,255);
          }
        }
      }
    }
    if (mouseX<50+85){
      if (mouseX>50+65){
        if (mouseY<250){
          if (mouseY>120){
            fill(255,0,0);
            rect(mouseX,mouseY,10,10);
            fill(255,255,255);
          }
        }
      }
    }
  }
}
```

Was ist die geheime Botschaft des Programms?

Verschlüsseln Sie nun Ihre eigene Geheimbotschaft, nach dem gegebenen Beispiel. Folgende Varianten können Sie dabei ausprobieren:

- Geheimbotschaft erscheint nur, wenn der geheime Schlüssel gedrückt wird (zum Beispiel die Taste g)
- Die verschiedenen Teile erscheinen mit unterschiedlichen Vorbedingungen. Ein Teil erscheint, wenn die Maus gedrückt ist, ein anderer, wenn eine bestimmte Taste gedrückt wird.



Aufgabenblatt 7b: Einfache Pong Variante

Nun haben wir fast alles beisammen, um unser kleines Pong-Spiel zu programmieren. Bisher haben wir folgendes gesehen:

- Zeichnen von Figuren
- Setzen von Farben
- Erstellen und Verwenden von eigenen Variablen
- Verwenden von Processing-Variablen
- Bedingte Programmausführung

Wir werden unser Pong-Spiel in zwei Teilen programmieren:

1. Der Balken, der rauf und runter fährt.
2. Der Ball, der an den Seiten, bzw. dem Balken abprallt.

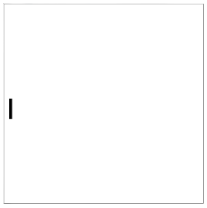
Der Schläger

Als erstes soll der Schläger simuliert werden. Dafür werden wir ein Rechteck verwenden, welches am linken Rand von oben nach unten wandern kann. Für die Steuerung werden wir die Tasten ‚w' (für aufwärts) ‚y' (für abwärts) verwenden.

Aufgabe 1: setup-Methode

In der Methode setup soll folgendes gemacht werden:

- Die Grösse des Fensters wird auf 400x400 Pixel gesetzt
- Der Hintergrund wird auf Weiss gesetzt
- Die Füllfarbe wird auf Schwarz gesetzt.
- Der Schläger wird mit folgenden Eigenschaften gezeichnet: Abstand vom linken Rand sind 10 Pixel, ungefähr in der Mitte des Feldes mit der Breite 5 Pixel und der Höhe 40 Pixel.



Aufgabe 2: Position des Schlägers speichern

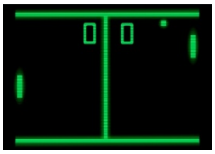
Was wir uns nun noch merken wollen, ist die Position des Schlägers. Dies machen wir mit einer Variablen pos. Definieren Sie diese Variable ganz am Anfang des Programms, und setzten Sie den Wert auf die y-Koordinate des Schlägers.

Aufgabe 3: Bewegung des Schlägers

Nun soll sich der Schläger natürlich noch bewegen. Schreiben Sie die folgende Methode:

```
void draw(){
  if(keyPressed){
    if (key=='w'){
      pos=pos-3;
    }
    if (key=='y'){
      pos=pos+3;
    }
  }
  rect(10,pos,5,40);
}
```

Testen Sie nun dieses Programm. Macht es das, was Sie von ihm erwarten würden?



Aufgabe 4: Löschen des Hintergrunds

Anscheinend muss das vorher gezeichnete wieder gelöscht werden. Dies wird so gelöst, dass zu Beginn der Methode draw der Hintergrund wieder auf Weiss gesetzt wird. Fügen Sie diese Anweisung hinzu.

Aufgabe 5: Schläger im Spielfeld behalten

Nun haben wir noch das Problem, dass der Schläger aus dem Spielfeld gesteuert werden kann. Das wollen wir natürlich verhindern.
Bevor der Wert von pos um 3 reduziert wird, soll vorher geprüft werden, ob der Wert denn noch über 0 liegt. Das wird wie folgt gemacht:

```
if (pos>0) {
      pos=pos-3;
}
```

Schreiben Sie nun einen analogen Test, der prüft, ob der Schläger schon unten am Bildschirm angekommen ist. Denken Sie dabei daran, dass die Position die obere linke Ecke des Schlägers bezeichnet. Der Schläger hat aber noch eine bestimmte Länge.

Der Ball

Und nun wollen wir noch den Ball simulieren, zuerst einmal in einem eigenen kleinen Programm.
Wir brauchen:

- Zwei Variablen für die Position des Balles(x=200 und y=200 Position)
- Zwei Variablen für die Geschwindigkeit des Balles. (speedx=2 und speedy=1)

Aufgabe 6: Die Variablen

Für den Ball brauchen wir vier Variablen. Zwei Variablen, um uns die Position (x- und y-Koordinate) zu merken. Deklarieren Sie diese zwei Variablen.
Zwei weitere Variablen speichern die Geschwindigkeit des Balles in der x- bzw. y-Achse. Deklarieren Sie diese zwei Variablen (speedx und speedy) und setzen Sie den Wert von speedx auf 2, sowie den von speedy auf 1.

Aufgabe 7: setup-Methode

In der Methode setup soll das analoge gemacht werden wie beim Schläger:

- Die Grösse des Fensters wird auf 400x400 gesetzt.
- Der Hintergrund wird auf Weiss gesetzt.
- Die Füllfarbe wird auf Schwarz gesetzt.
- Der Ball wird in der Mitte des Fensters gezeichnet. (an der Position x/y) Der Durchmesser des Balles beträgt 10 Pixel.

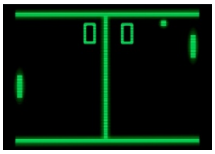
Aufgabe 8: Bewegung des Balls

In der Methode draw wir als erstes die Position des Balles verändert. Dazu wird zur x Koordinate der Wert von speedx hinzugezählt. Zur y Koordinate wird der Wert von speedy hinzugezählt.
Anschliessend soll der Ball an der Position wieder gezeichnet werden.
Testen Sie Ihr Programm.

Aufgabe 9: Abprallen des Balles

Mit dieser Lösung wird der Ball einfach aus dem Spielfeld rausfliegen. Wir möchten aber, dass er abprallt. Fügen Sie also die folgenden Zeilen zwischen dem Verändern der Position und dem Zeichnen des Balles ein:

```
  if(x>390){
    speedx=speedx*-1;
```



```
}
if(x<25){
  speedx=speedx*-1;
}
if(y<5){
  speedy=speedy*-1;
}
if(y>395){
  speedy=speedy*-1;
}
```

Aufgabe 10: Zusammenfügen

Fügen Sie nun die beiden Programme zusammen.

Aufgabe 11: Testen, ob der Ball den Schläger trifft

Diese Aufgabe ist sehr komplex, und ist die Abschlussaufgabe des Moduls. Die folgenden Zeilen sollen so geändert werden, dass nur dann die Anweisung speedx=speedx*-1 ausgeführt wird, wenn sich die Position des Schlägers auf der Höhe des Balles befindet:

```
if(x<25){
  speedx=speedx*-1;
}
```