1 Grundformen zeichnen

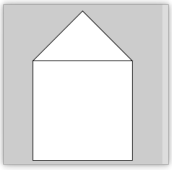
1.1 Punkte, Strecken und Polygone

Listing 1. grundformen.pde

```
// Erstes Programm
// ================================================================================
// size(x,y)
// Öffnet ein Fenster der Größe 800x600 Pixel
size(800,600);

// point(x,y)
// Zeichnet einen Punkt an die Koordinaten (42|100)
point(42,100);

// line(x1,y1,x2,y2)
// Zum Zeichnen einer Linie gibt man den Anfangs- und den Endpunkt an.
// Zeichnet eine Linie vom Punkt (100|50) zu (700|550)
line(100,50,700,550);

// triangle(x1,y1,x2,y2,x3,y3);
// Ein Dreieck braucht drei Punkte als Information.
// Zeichnet ein Dreieck mit den Punkten (50|100), (50|500) und (200|250).
triangle(50,100,50,500,200,250);

// rectMode(CORNER);
// Grundeinstellung
// rect(x, y, breite, hoehe);
// Für das Rechteck gibt man den linken, oberen Eckpunkt, sowie Höhe und Breite an.
// Zeichnet ein 200 mal 300 Pixel großes Rechteck von Punkt (300|200) aus.
rectMode(CORNER);
rect(300,200,200,300);

// rectMode(CENTER);
// rect(x, y, breite, hoehe);
// Für das Rechteck gibt man das Zentrum, sowie Höhe und Breite an.
// Zeichnet ein 200 mal 300 Pixel großes Rechteck um den Punkt (300|200).
rectMode(CENTER);
rect(300,200,200,300);

// quad(x1, y1, x2, y2, x3, y3, x4, y4);
// Viereck mit den angegebenen vier Eckpunkten.
quad(620, 550, 700, 580, 610, 590, 580, 500);

// Vieleck mit einer beliebigen Anzahl von Punkten
beginShape();
  vertex(600,100);
  vertex(600,200);
  vertex(500,250);
  vertex(550,200);
  vertex(500,150);
endShape(CLOSE);
```

A 1.1. Beschreibe, wo sich der Nullpunkt des Koordinatensystems für die Zeichenfläche befindet.

A 1.2. Zeichne das Haus aus der Grafik rechts ...



a) ... aus Grundformen.

b) ... unter Verwendung von `beginShape()` und `endShape()`.

1.2 Ellipsen zeichnen

Listing 2. ellipsen.pde

```
// Erstes Programm
// ================================================================================
// size(x,y)
// Öffnet ein Fenster der Größe 800x600 Pixel
size(800,600);

// ellipseMode(CENTER);
// Grundeinstellung
// ellipse(x, y, Breite, Höhe)
// Bei der Ellipse werden der Mittelpunkt und der Durchmesser in Breite und
// Höhe angegeben.
// Zeichnet einen Kreis mit dem Radius 100 um den Punkt (400,300)
ellipse(400,300,100,100);
// Eine Ellipse um den Punkt (200,150) mit den beiden Durchmessern 50 und 100.
ellipse(200,150, 50, 100);

// ellipseMode(CORNER);
// ellipse(x1, y1, breite, hoehe)
// Diese Ellipse wird durch das umgebende Rechteck definiert.
// Eine Ellipse vom linken oberen Punkt (200,300) in das Rechteck mit den Seiten 100 und 50.
ellipseMode(CORNER);
ellipse(200, 300, 100, 50);

// ellipseMode(CORNERS);
// ellipse(x1, y1, x2, y2)
// Diese Ellipse wird durch das umgebende Rechteck definiert.
// Zeichnet eine Ellipse in das Rechteck mit dem linken oberen Punkt (150|450)
// und dem rechten unteren Punkt (250|550).
ellipseMode(CORNERS);
ellipse(150, 450, 250, 550);


// arc(x, y, b, h, alpha, beta)
// Zeichnet eine Teilellipse mit dem Zentrum um den Punkt (x|y), den beiden
// Durchmessern h und b und den Öffnungswinkeln alpha (Anfang) und beta (Ende).
// Die Winkel werden im Bogenmaß angegeben.
ellipseMode(CENTER);
arc(600, 150, 100, 100, PI/6, 2*PI - PI/6);
// Referenzlinie für den Winkel
line(600, 150, 660, 150);

// Um den Winkel in Grad anzugeben, wird der Befehl radians(Winkel) verwendet.
arc(600, 300, 100, 100, radians(30), radians(330));

// Kleine Spielerei
arc (600, 500, 200, 100, radians(0), radians(180));
arc (550, 500, 100, 100, radians(180), radians(360));
arc (620, 450, 50, 100, radians(-90), radians(90));
```

A 1.3. Beschreibe wie man das Aussehen eines Ellipsenabschnitts (`arc()`) über einen siebten Parameter beeinflussen kann. Nutze dazu die Referenzfunktion (Hilfe → Referenz) von Processing und zeichne jeweils ein Beispiel.

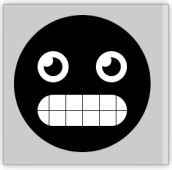
A 1.4. Zeichne die beiden Grafiken rechts jeweils mit Processing.

1.3 Hintergrund und Konturen

Listing 3. konturundco.pde

```
// Kontur und Füllung
// ================================================================================
// size(x,y)
// Öffnet ein Fenster der Größe 800x600 Pixel
size(800,600);
// Hintergrund hellgrau
background(220);

// Schwarzes Dreieck
fill(0);
triangle(50,150,250,150,150,50);
// Graues Dreieck
fill(127);
triangle(300,150,500,150,400,50);
// Weißes Dreieck
fill(255);
triangle(550,150,750,150,650,50);


fill(255);
// Weißer Kreis mit dünnem Rand
strokeWeight(1);
ellipse(100, 250, 80, 80);
// Weißer Kreis mit dickem Rand
strokeWeight(5);
ellipse(200, 250, 80, 80);

// Weißer Kreis mit grauem Rand
stroke(127);
ellipse(300, 250, 80, 80);
// Schwarzer Kreis mit weißem Rand
fill(0);
stroke(255);
ellipse(400, 250, 80, 80);

// Schwarzer Kreis ohne Rand
noStroke();
ellipse(500, 250, 80, 80);

// Leerer Kreis mit Rand
noFill();
stroke(255);
ellipse(550, 250, 80, 80);
```

A 1.5. Zeichne das Gesicht aus der Grafik rechts.



1.4 Farben und Transparenz

Listing 4. farben.pde

```
// Farben und Transparenz
// ================================================================================
// size(x,y)
// Öffnet ein Fenster der Größe 800x600 Pixel
size(800,600);
// Hintergrund in Antikweiß
background(250,235,215);

// Kein Rand
noStroke();

// Roter Kreis
fill(255,0,0);
ellipse(100, 150, 80, 80);
// Grüner Kreis
fill(0,255,0);
ellipse(200, 150, 80, 80);
// Blauer Kreis
fill(0,0,255);
ellipse(300, 150, 80, 80);

// Cyan Kreis
fill(0,255,255);
ellipse(500, 150, 80, 80);
// Magenta Kreis
fill(255,0,255);
ellipse(600, 150, 80, 80);
// Gelber Kreis
fill(255,255,0);
ellipse(700, 150, 80, 80);

// heller
fill(255,127,127); ellipse(100, 250, 80, 80);
fill(127,255,127); ellipse(200, 250, 80, 80);
fill(127,127,255); ellipse(300, 250, 80, 80);
fill(127,255,255); ellipse(500, 250, 80, 80);
fill(255,127,255); ellipse(600, 250, 80, 80);
fill(255,255,127); ellipse(700, 250, 80, 80);

//dunkler
fill(127,0,0); ellipse(100, 350, 80, 80);
fill(0,127,0); ellipse(200, 350, 80, 80);
fill(0,0,127); ellipse(300, 350, 80, 80);
fill(0,127,127); ellipse(500, 350, 80, 80);
fill(127,0,127); ellipse(600, 350, 80, 80);
fill(127,127,0); ellipse(700, 350, 80, 80);

// Transparenz
fill(255,255,255, 40); ellipse(100, 400, 80, 80);
fill(255,255,255, 80); ellipse(200, 400, 80, 80);
fill(255,255,255,120); ellipse(300, 400, 80, 80);
fill(255,255,255,160); ellipse(500, 400, 80, 80);
fill(255,255,255,200); ellipse(600, 400, 80, 80);
fill(255,255,255,240); ellipse(700, 400, 80, 80);
```

A 1.6. Zeichne die Olympischen Ringe mit Processing.

A 1.7. Zeichne einen halbkreisförmigen Regenbogen mit Processing unter Verwendung von `fill()` und `noStroke()`.

1.5 Text

Listing 5. hallowelt.pde

```
// Text schreiben
//============================================================================
size(800, 600);                                                  // Fenstergröße
background(255,140,0);                                            // Hintergrund

// Orientierungskreuz
stroke(255);                                                 // Strichfarbe weiß
line(400,0,400,600);
line(0,100,800,100);
line(0,200,800,200);
line(0,300,800,300);
line(0,400,800,400);
line(0,500,800,500);

// Text in Grafik schreiben
fill(255);                                                     // Textfarbe weiß
text("Hallo Welt!", 400, 100);                          // Text normal linksbündig

// zentriert
textAlign(CENTER);
text("Hello World!", 400, 200);

// rechtsbündig
textAlign(RIGHT);
text("Moin erstmal!", 400, 300);

// Schriftgröße (in Punkten)
textSize(32);
textAlign(CENTER);
text("Große Schrift!", 400, 400);

textSize(48);
fill(0);
text("Sehr große schwarze Schrift!", 400, 500);
```

2 Statischer und Dynamischer Modus

Listing 6. statischerModus.pde

```
1: // Statischer Modus
2:
3: // In Konsole schreiben
4: println("Jetzt geht's ...");
5: println("los!");
```

A 2.1. Ersetze in Zeile 5 `println()` durch `print()`. Beschreibe, was die beiden Befehle unterscheidet.

Listing 7. dynamischerModus.pde

```
 1: // Dynamischer Modus
 2: //============================================================================
 3:
 4: // Wird am Anfang ausgeführt
 5: void setup()
 6: {
 7:   // Anzahl der Wiederholungen von draw() pro Sekunde
 8:   frameRate(1);
 9:   // In Konsole schreiben
10:   println("Jetzt geht's ...");
11: }
12:
13: // Wird in einer Endlosschleife wiederholt
14: void draw()
15: {
16:   // In Konsole schreiben
17:   println("los!");
18: }
```

A 2.2. Beschreibe, in welcher Zeile der Befehl wie verändert werden muss, damit der Text "los!" fünfmal in der Sekunde erscheint.

3 Position des Mauszeigers

Listing 8. drawing1.pde

```
1: // Steuern eines Kreises mit dem Mauszeiger
2: //============================================================================
3:
4: void setup()
5: {
6:   //Fenster öffnen
7:   size(800,800);
8:   // Hintergrundfarbe
9:   background(255,140,0);
10: }
11:
12: void draw()
13: {
14:   // Bild löschen
15:   background(255,140,0);
16:   // Kreis zeichnen an Mauszeigerposition
17:   ellipse(mouseX,mouseY,21,21);
18: }
```

A 3.1. Nenne die Variablen, in denen die aktuellen Koordinaten des Mauszeigers stehen.

A 3.2. Schreibe ein Programm, in dem eine Linie vom Punkt (0|0) zur aktuellen Position des Mauszeigers führt.

A 3.3. Schreibe ein Programm, dass ein Fadenkreuz aus einer horizontalen und einer waagerechten Linie erzeugt. Der Schnittpunkt der Linien liegt genau auf dem Mauszeiger.

A 3.4. Kommentiere in Listing 8 den Befehl in Zeile 15 aus, um ein kleines Zeichenprogramm zu bekommen. Beschreibe die Schwächen des Programms.

Listing 9. drawing2.pde

```
1: // Kleines Zeichenprogramm
2: //============================================================================
3: void setup()
4: {
5:   size(800,800);                                          // Fenster öffnen
6:   background(255,140,0);                                 // Hintergrundfarbe
7:   stroke(255);                                               // Linienfarbe
8:   strokeWeight(5);                                          // Linienbreite
9: }
10:
11: void draw()
12: {
13:   // Zeichnet eine Linie von vorheriger Mausposition zur aktuellen Mausposition
14:   line(pmouseX, pmouseY, mouseX, mouseY);
15: }
```

A 3.5. Nenne die Variablen, in denen die Koordinaten des Mauszeigers von der vorherigen Ausführung der Funktion `draw()` stehen.

4 Eventfunktionen für Mausklick und Tastendruck

Listing 10. mousepressed.pde

```
1: // Auf Mausklick reagieren
2: //============================================================================
3:
4: void setup()
5: {
6:   size(800,600);                                              // Fenstergröße
7:   background(255,140,0);                                 // Hintergrundfarbe setzen
8: }
9:
10: // Tut nichts, muss aber da sein.
11: void draw() {}
12:
13: // Reagiert auf Mausklick
14: void mousePressed()
15: {
16:   // Malt das Fenster in einer zufälligen Farbe an.
17:   background(random(256), random(256), random(256));
18: }
```

A 4.1. Beschreiben Sie den Befehl `random()`.

A 4.2. Schreibe ein Programm, dass auf Mausklick an die aktuelle Position des Mauszeigers einen weißen Punkt setzt.

Listing 11. keypressed.pde

```
1: // Auf Tastendruck reagieren
2: //============================================================================
3:
4: void setup()
5: {
6:   size(800,600);                                              // Fenstergröße
7:   background(255,140,0);                                 // Hintergrundfarbe setzen
8: }
9:
10: // Tut nichts, muss aber da sein.
11: void draw() {}
12:
13: // Reagiert auf Mausklick
14: void keyPressed()
15: {
16:   // Malt das Fenster in einer zufälligen Farbe an.
17:   background(random(256), random(256), random(256));
18: }
```

A 4.3. Schreibe ein Programm, dass zuerst ein graues Fenster zeigt. Auf Tastendruck soll es rot werden und auf Mausklick grün.

5 Variablen

5.1 Grundlagen

Listing 12. variablen1.pde

```
 1: // Variablen deklarieren, typisieren und initalisieren
 2: //============================================================================
 3:
 4: // Variablen werden am Anfang des Programms deklariert und typisiert.
 5: int pille;                                        // ist eine Ganzzahl (integer)
 6: float palle;                                   // ist eine Fließkommazahl (float)
 7:
 8: // Die erste Zuweisung eines Wertes bezeichnet man als Initalisierung.
 9: pille = 7;
10: palle = 8.85;                  // Gleitkommazahlen werden mit Punkt geschrieben.
11:
12: // Deklarieren, Typisieren und Initialisieren geht auch in einer Zeile.
13: float pillepalle = 9.81;
14:
15: // = ist der Zuweisungsoperator. Links von ihm steht immer eine Variable.
16: // Nicht mit dem Gleichheitszeichen verwechseln!!!
17:
18: // Addition
19: pillepalle = pille + palle;
20: println(pillepalle);
21: // Subtraktion
22: pillepalle = pille - palle;
23: println(pillepalle);
24: // Multiplikation
25: pillepalle = pille * palle;
26: println(pillepalle);
27: // Division
28: pillepalle = pille / palle;
29: println(pillepalle);
```

Harte Regeln für Variablennamen

Variablenname dürfen ...

- nicht identisch zu einem Schlüsselwort in Java bzw. Processing sein (z.B. `int`, `float`, `line`, `text`, ...)
- aus Buchstaben (a-z, A-Z), Ziffern (0-9), dem Unterstrich und $ bestehen.
- nicht mit einer Ziffer beginnen.
- kein Leerzeichen und keinen Bindestrich enthalten.

Weiche Regeln (Konventionen)

Variablennamen ...

- beginnen immer mit einem kleinen Buchstaben.
- folgen der CamelBack-Regel, wenn sie aus verbundenen Einzelwörtern besteht (z.B. `altePositionX`, `personVorname`, `hintergrundFarbeRot`)
- sind möglichst aussagekräftig (z.B. `zaehler` anstatt `z`, `nenner` anstatt `n`)
- verzichten auf Unterstrich und $.

A 5.1. Die Variablen `height` und `width` enthalten die Höhe und die Breite des aktuellen Programmfensters. Nutze für die folgenden Aufgaben diese Variablen.

a) Schreibe ein Programm, das durch eine senkrechte und eine waagerechte Linie das Fenster mittig teilt.
b) Schreibe ein Programm, das ein weißes Rechteck in ein Programmfenster zeichnet, dessen Ränder den Abstand 20 Pixel vom Fensterrand haben. Benutze für die linke obere Ecke die Variablen `x1` und `y1` an und für die untere rechte Ecke die Variablen `x2` und `y2`.

5.2 Hoch- und Runterzählen

Listing 13. variablen2.pde

```
 1: // Hochzählen
 2: //============================================================================
 3: // Globale Variablen
 4: int zaehler = 0;
 5: // Hintergrundfarbe
 6: int bgred = 255;
 7: int bggreen = 140;
 8: int bgblue = 0;
 9:
10: //============================================================================
11: void setup()
12: {
13:   size(200,200);                                          // Fenster öffnen
14:   background(bgred,bggreen,bgblue);                      // Hintergrundfarbe
15:   fill(255);                                              // Textfarbe weiß
16:   frameRate(1);                                  // Aktualisierungen pro Sekunde
17:   textAlign(CENTER);                                      // Text zentriert
18:   textSize(32);                                                // Textgröße
19: }
20:
21: //============================================================================
22: void draw()
23: {
24:   // Variablen hochzählen
25:   zaehler = zaehler + 1;
26:   // Fenster löschen
27:   background(bgred,bggreen,bgblue);
28:   // Zähler ausgeben
29:   text(zaehler,100,100);
30: }
```

A 5.2. Ersetze Zeile 25 in Listing 13 durch `zaehler++;` und starte das Programm. Beschreibe, was der Befehl macht.

A 5.3. Schreibe das Programm 13 so um, dass es von 100 runterzählt.

A 5.4. Ergänze das Programm 13 so, dass es bei Tastendruck den Hintergrund auf rot ändert und bei Mausklick auf grün.

A 5.5. Schreibe ein Programm unter Berücksichtigung folgender Anweisungen.

a) Erstelle ein Fenster der Größe 640 x 640.
b) Erstelle zwei Variablen `x` und `y` vom Typ `float` und dem Wert `320`.
c) Erstelle einen Smiley mit dem Durchmesser 100 und dem Mittelpunkt (`x`|`y`). Alle Elemente des Smileys (Kreis, Augen, Mund) sollen keine absoluten Koordinaten enthalten, sondern durch Addition und Subtraktion mit den Variablen `x` und `y` erstellt werden. `x` und `y` dürfen dabei ihren Wert nicht ändern.
d) Überführe das Programm in den aktiven Modus (mit `setup()` und `draw()`). Ersetze dabei `x` und `y` mit `mouseX` und `mouseY`, so dass der Smiley dem Mauszeiger folgt.

5.3 Modulo

Listing 14. modulo1.pde

```
 1: // Beispiel Modulo
 2: //============================================================================
 3: // Globale Variablen
 4: int zaehler = 0;
 5:
 6: void setup()
 7: {
 8:   size(200,200);                                          // Fenstergröße
 9:   frameRate(1);                                 // Aktualisierungen pro Sekunde
10:   fill(255);                                               // Textfarbe
11:   textAlign(CENTER);                                  // Textausrichtung
12:   textSize(32);                                           // Textgröße
13: }
14:
15: void draw()
16: {
17:   zaehler++;                                          // Um eins hochzählen
18:
19:   // Lokale Variable
20:   int out = zaehler % 3;                        // Rest der Division mit 3
21:
22:   // Ausgabe
23:   background(255,140,0);                                // Bild löschen
24:   text(out,100,100);                                   // Text ausgeben
25: }
```

A 5.6. Ersetze Zeile 20 durch `zaehler = zaehler % 3;`.

a) Gebe an, wie Zeile 24 dann aussehen muss.
b) Teste, ob das Programm so funktioniert.
c) Beschreibe den Unterschied zwischen den beiden Programmversionen.

A 5.7. Schreibe ein Programm unter Berücksichtigung folgender Anweisungen.

a) Erstelle ein Fenster der Größe 640 x 640.
b) Erstelle zwei Variablen `x` und `y` vom Typ `int`, die den Mittelpunkt eines Kreises darstellen.
c) Lasse den Kreis nun von links nach rechts in mittlerer Höhe über das Fenster wandern.
d) Lasse den Kreis, wenn er den rechten Rand erreicht, wieder nach links springen. Verwende dazu den Operator Modulo (%) und die Systemvariable `width`.
e) Durch jeden Mausklick soll der Kreis 20 Einheiten nach unten springen.
f) Wird der untere Rand erreicht, soll der Kreis nach oben springen.

5.4 Variablentypen

Ganze Zahlen

Typ	Größe	Wertebereich
byte	8 Bit	-128 bis 127
short	16 Bit	-32 768 bis 32 767
int	32 Bit	-2 147 483 648 bis 2 147 483 647
long	64 Bit	ca. -9×10^{18} bis 9×10^{18}

Gleitkommazahlen

Typ	Größe	Stellen
float	32 Bit	7 bis 8
double	64 Bit	15 bis 16

Wahrheitswerte

Typ	Beschreibung	Wertebereich
boolean	Wahrheitswert	`true` und `false`

Zeichen

Typ	Beschreibung	Werte
char	einzelnes Zeichen	z.B. `'a'`, `'7'` oder `'%'`

Zeichenketten

Typ	Beschreibung	Werte
String	Reihung von Zeichen	z.B. `"Hallo Welt"`

5.5 Zeichenketten

Listing 15. mousekoordinaten.pde

```
 1: // Mauskoordinaten anzeigen
 2: //============================================================================
 3: void setup()
 4: {
 5:   size(640,640);                                              // Fenstergröße
 6:   fill(255);                                                    // Textfarbe
 7: }
 8:
 9: void draw()
10: {
11:   // Lokale Variable vom Typ String nur gültig in dieser Funktion
12:   String txt = "";
13:   // Konkatenation mehrerer Zeichenketten zu einer
14:   txt = "x = " + mouseX + " y = " + mouseY;
15:   // Ausgabe
16:   background(255,140,0);                                       // Bild löschen
17:   text(txt,10,20);                                           // Text ausgeben
18: }
```

A 5.8. Verändern Sie das Programm 15, damit es noch den Abstand vom Nullpunkt (0|0) ausgibt. Verwenden Sie dazu den Satz des Pythagoras und die Wurzelfunktion `sqrt()`.

6 Entscheidungen

6.1 if

Listing 16. steine1.pde

```
// IF-Entscheidung: Steine
//============================================================================
// Globale Variablen
float x = 0;                                          // Position des Steins X
float y = 0;                                          // Position des Steins Y

void setup()
{
  size(640,640);                                          // Fenstergröße
  background(255,140,0);                                // Hintergrundfarbe
  // Startpunkt in der Mitte des Fensters
  // width und height stehen erst nach size() bzw. setup() zur Verfügung
  x = width/2;
  y = height/2;
}

void draw()
{
  // Stein zeichnen
  ellipse(x,y,20,20);                                   // Ellipse zeichnen
  // Neue Position
  x = x + random(-20,20);
  // Randkontrolle
  if (x < 0) {                  // Links über den Rand: Springe zum rechten Rand
    x = width + x;
  }
  if (x > width) {              // Rechts über den Rand: Springe zum linken Rand
    x = x - width;
  }
}
```

A 6.1. Zeichne einen Programmablaufplan und ein Struktogramm für die Funktion `draw()` aus Listing 16.

A 6.2. Verändere das Programm 16 so, dass ...

a) die Steine zufällig auch höher und tiefer gesetzt werden.
b) jeder Stein ein zufälliges Grau bekommt.
c) jeder Stein eine zufällige Breite und Höhe bekommt.

6.2 Der bewegte Ball

Listing 17. ball1.pde

```
// Ball von links nach rechts
//============================================================================
// Globale Variablen
float x = 0;                                          // Position des Steins X
float y = 0;                                          // Position des Steins Y
float speed = 2;                                 // Geschwindigkeit in X-Richtung

void setup()
{
  size(640,640);                                              // Fenstergröße
  background(255,140,0);                                    // Hintergrundfarbe
  // Startpunkt in der Mitte des Fensters
  // width und height stehen erst nach size() bzw. setup() zur Verfügung
  x = width/2;
  y = height/2;
}

void draw()
{
  // Ball zeichnen
  background(255,140,0);
  ellipse(x,y,20,20);
  // Neue Position
  x = x + speed;
  if (x >= width) {         // Wenn über rechten Rand, dann gehe zum linken Rand
    x = 0;
  }
}
```

A 6.3. Zeichne einen Programmablaufplan und ein Struktogramm für die Funktion `draw()` aus Listing 17.

A 6.4. Verändere das Programm 17 so, dass der Ball sich von rechts nach links bewegt.

A 6.5. Verändere das Programm 17 so, dass der Ball von oben nach unten fliegt.

A 6.6. Nimm das Programm 17 als Ausgangspunkt und schreibe ein Programm mit den folgenden Kriterien.

- Benutze zwei Variablen `xspeed` und `yspeed` für die Geschwindigkeit des Balls in alle Richtungen.
- Der Ball soll sich nach dem Start in eine zufällige Richtung bewegen.
- An den Rändern soll der Ball zum gegenüberliegenden Punkt springen.

A 6.7. Wenn der Ball an einer senkrechten Wand abprallen soll, dann ändert man das Vorzeichen der Geschwindigkeit. Für eine senkrechte Wand die Geschwindigkeit in x-Richtung und bei einer waagerechten Wand die Geschwindigkeit in y-Richtung. Programmiere das Programm aus der vorherigen Aufgabe so um, so dass der Ball an der Wand abprallt.

6.3 if-else

Listing 18. ball2.pde

```
 1: // Ball von links nach rechts mit Farbwechsel
 2: //============================================================================
 3: // Globale Variablen
 4: float x = 0;                                          // Position des Steins X
 5: float y = 0;                                          // Position des Steins Y
 6: float speed = 3;                                 // Geschwindigkeit in X-Richtung
 7:
 8: void setup()
 9: {
10:   size(640,640);                                              // Fenstergröße
11:   background(255,140,0);                                   // Hintergrundfarbe
12:   // Startpunkt in der Mitte des Fensters
13:   // width und height stehen erst nach size() bzw. setup() zur Verfügung
14:   x = width/2;
15:   y = height/2;
16: }
17:
18: void draw()
19: {
20:   // Ballfarbe festlegen
21:   if (x >= width/2) {
22:     fill(255,0,0);
23:   } else {
24:     fill(0,255,0);
25:   }
26:   // Ball zeichnen
27:   background(255,140,0);                                  // Fenster löschen
28:   ellipse(x,y,20,20);
29:   // Neue Position
30:   x = x + speed;
31:   if (x >= width) {         // Wenn über rechten Rand, dann gehe zum linken Rand
32:     x = 0;
33:   }
34: }
```

A 6.8. Zeichne einen Programmablaufplan und ein Struktogramm für die Funktion `draw()` aus Listing 18.

6.4 Tasten abfragen

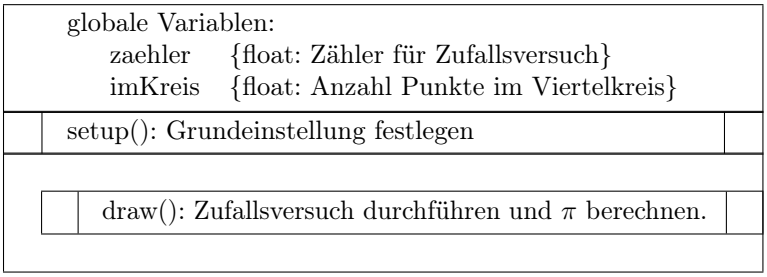
Listing 19. steuern1.pde

```
// if-Entscheidung
// Steuern mit Tasten
//============================================================================

int x = 0;
int y = 0;

// Wird am Anfang ausgeführt
void setup()
{
  size(640,640);                                          // Fenstergröße
  background(255,140,0);                                // Hintergrundfarbe
  fill(255);                                                 // Füllfarbe
  // In der Mitte positionieren
  x = width/2;
  y = height/2;
}

// Wird in einer Endlosschleife wiederholt
void draw()
{
  // Neue Position durch Steuerung
  // Variable key enthält die zuletzt gedrückte Taste
  // w drücken für oben
  // Einzelner Buchstabe in einfachen Anführungzeichen ist ein Char!
  if (key == 'w') {
    y = y-1;
  }
  // s drücken für unten
  if (key == 's') {
    y = y+1;
  }
  // Bildschirm löschen
  background(255,140,0);
  // Ball zeichnen
  ellipse(x,y,30,30);
}
```

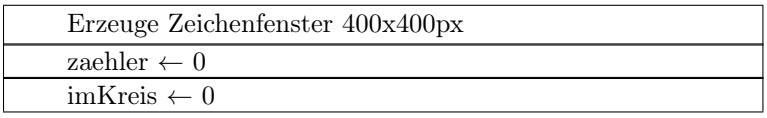
6.5 Anwendung π-Bestimmung

Zur Bestimmung der Zahl *pi* kann man sich einer Monte-Carlo-Simulation bedienen. Dabei wird ein Viertelkreis mit dem Radius 1 betrachtet. In dem dazugehörigen Quadrat mit der Kantenlänge 1 wird zufällig ein Punkt mit den Koordinaten $(x|y)$ gesetzt. Ist der Abstand vom Nullpunkt kleiner oder gleich 1, dann gehört er zum Kreis. Teilt man nun die Anzahl der Punkte im Kreis durch alle Punkte, dann erhält man die Fläche des Viertelkreises. Multipliziert man mit 4, dann erhält man die Fläche des Vollkreises und, da der Radius 1 ist, die Zahl π.

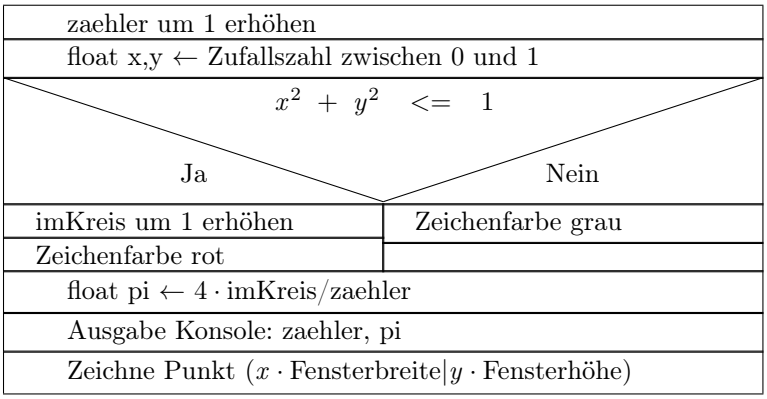
Das folgende Struktogramm zeigt den Algorithmus zur Bestimmung von *pi* nach dieser Methode.



setup()



draw()



A 6.9. Schreibe nach dem Struktogramm das Programm zur Bestimmung von π in Processing.

A 6.10. Untersuche, welche Framerate sinnvoll für das Programm ist.

A 6.11. Die Ausgabe auf der Konsole bremmst das Programm aus. Die Ausgabe soll daher nur ungefähr jede Sekunde erfolgen. Pflege diese Änderung in Dein Programm ein.

6.6 Verschachtelte Entscheidungen

Listing 20. steuern2.pde

```
 1: // if-Entscheidung verschachtelt
 2: // Steuern mit Tasten
 3: //============================================================================
 4:
 5: int x = 0;
 6: int y = 0;
 7:
 8: // Wird am Anfang ausgeführt
 9: void setup()
10: {
11:   size(640, 640);                                          // Fenstergröße
12:   background(255, 140, 0);                                // Hintergrundfarbe
13:   fill(255);                                                 // Füllfarbe
14:   // In der Mitte positionieren
15:   x = width/2;
16:   y = height/2;
17: }
18:
19: // Wird in einer Endlosschleife wiederholt
20: void draw()
21: {
22:   // Neue Position durch Steuerung
23:   // Wurde gerade eine Taste gedrückt?
24:   // boolean keyPressed: true oder false
25:   if (keyPressed) {
26:     if (key == 'w') {                                          // oben
27:       y = y-1;
28:     }
29:     if (key == 's') {                                        // unten
30:       y = y+1;
31:     }
32:   }
33:   // Bildschirm löschen
34:   background(255, 140, 0);
35:   // Ball zeichnen
36:   ellipse(x, y, 30, 30);
37: }
```

A 6.12. Zeichne einen Programmablaufplan und ein Struktogramm für die Funktion `draw()` aus Listing 20.

A 6.13. Verändern Sie das Programm 20 so, dass die Tasten a und d den Ball nach links und rechts bewegen.

7 Schleifen

7.1 while-Schleife

Listing 21. while1.pde

```
1: // While-Schleife
2: //============================================================================
3:
4: int i = 1;
5:
6: while (i < 3) {
7:   println(i);                                                    // Schleife
8:   i++;                                                       // Wert hochzählen
9: }
```

A 7.1. Verändere das Listing 21 so, dass das Programm bis 3 zählt.

A 7.2. Verändere das Listing 21 so, dass das Programm bis 100 zählt.

A 7.3. Schreibe ein Programm, dass alle Zahlen bis 100 ausgibt, die durch 7 teilbar sind.

A 7.4. Schreibe ein Programm, dass ein 640 mal 480 Pixel großes Fenster öffnet, in dem alle 10 Pixel eine horizontale Linie gezeichnet wird.

A 7.5. Schreibe ein Programm, dass ein 640 mal 480 Pixel großes Fenster öffnet und dort ein Gitter mit einer Kästchengröße von 10 Pixel zeichnet.

A 7.6. Schreibe ein Programm, dass so lange würfelt (Zufallszahl zwischen 1 und 6), bis eine 6 gewürfelt wurde.

7.2 for-Schleife

Listing 22. for1.pde

```
1: // for-Schleife
2: //============================================================================
3:
4: // Im Kopf der for-Schleife werden Startwert, Laufbedingung und Veränderung
5: // definiert.
6: for (int i = 1; i <= 3; i++) {                                    // Bis 3 zählen
7:   println(i);
8: }
```

A 7.7. Schreibe ein Programm, dass die Variable `zaehler` in 3er-Schritten von 0 bis 42 hochzählt.

A 7.8. Schreibe folgendes Programm:

- Fenstergröße 640 mal 480 Bildpunkte
- Zeichne um den Mittelpunkt des Fensters 10 Kreise. Der innerste Kreis hat einen Radius von 10 Pixel. Für jeden weiteren Kreis erhöht sich der Radius um 5 Pixel.

Listing 23. primzahl1.pde

```
1: // Primzahlen berechnen
2: //============================================================================
3:
4: int zahl = 1;                                                  // Zu prüfende Zahl
5:
6: void setup()
7: {
8:   // Hier gibt es nichts zu tun.
9: }
10:
11: void draw()
12: {
13:   boolean primzahl = true;              // Wir nehmen an, die Zahl ist eine Primzahl
14:
15:   zahl++;                                                     // Zahl um 1 erhöhen
16:
17:   for(int i = 2; i < zahl; i++) {
18:     if(zahl % i == 0) {
19:       primzahl = false;
20:     }
21:   }
22:
23:   if(primzahl) {
24:     println(zahl);
25:   }
26: }
```

8 String-Objekt

Listing 24. string1.pde

```
// String: Zeichenketten
//============================================================================

// String-Objekt anlegen
String mann = new String("Harry");                                    // Langform
String frau = "Sally";                    // Kurzform, die normal verwendet wird

// equals(): Strings vergleichen
if(mann.equals(frau)) {
  println("gleich");
} else {
  println("verschieden");
}

// length(): Anzahl der Zeichen im String
// substring(Start, Ende): Teilzeichenkette: Start inklusive, Ende exklusive
// Achtung: Nummerierung beginnt bei 0
for (int i = 0; i <= mann.length(); i++){
  println(mann.substring(0,i));
}

// contains(string): Ist der angegebene String enhalten
// indexOf(string): Position des gesuchten Strings oder -1
if (frau.contains("all")){
  println("Gefunden an Position " + frau.indexOf("all"));
} else {
  println("Nicht Gefunden");
}

// toLowerCase(): Alles klein
// toUpperCase(): Alles GROSS
println(mann.toLowerCase() + " und " + frau.toUpperCase());

// charAt(pos): Gibt das Zeichen an Position pos aus.
for (int i = 0; i < frau.length(); i++) {
  println(frau.charAt(i));
}

// (int)string gibt die Zeichencodierung aus.
for (int i = 0; i < frau.length(); i++) {
  println((int)frau.charAt(i));
}

// Umgekehrt gibt (char)zahl das Zeichen mit der Codierung zahl zurück.
println((char)83);
```

8.1 Zeichencodierung

Listing 25. eingabe_zeichenmitcodierung.pde

```
 1: // Eingabe eines Zeichens und Ausgabe der Codierung
 2: // ============================================================================
 3: String input = "";
 4:
 5: void setup()
 6: {
 7:   size(800,600);                                          // Fenstergröße
 8:   textAlign(CENTER);                                     // Text zentriert
 9:   textSize(32);                                           // Textgröße
10:   fill(255);                                              // Textfarbe
11: }
12:
13: void draw()
14: {
15:   background(255,140,0);                                  // Frame löschen
16:   text(input,width/2,height/2);                          // Ausgabe in der Mitte
17: }
18:
19: void keyPressed()
20: {
21:   if((int)key > 10 && (int)key < 256) {       // Auf druckbare Zeichen begrenzen
22:     input = key + " : " + (int)key;              // Zeichen mit Zeichencodierung
23:   }
24: }
```

9 Funktionen

Funktionen können nur im aktiven Modus definiert werden. Es muss daher mindestens die Funktion `setup()` vorhanden sein.

9.1 void ohne Parameter

Listing 26. funktionvoid1.pde

```
// Funktion vom Typ void ohne Parameter
// ================================================================================
int x = 0;  // X-Koordinate
int y = 0;  // Y-Koordinate

void setup()
{
  size(800,600);                                                  // Fenstergröße
  x = 50;
  y = height/2;
}

void draw()
{
  background(255,140,0);                                        // Frame löschen
  paintFace();                                                 // Gesicht zeichnen
  x++;                                                // X-Koordinate um 1 erhöhen
}

// Zeichnet ein Gesicht um den Punkt x,y (globale Variablen)
void paintFace() {
  fill(255);
  stroke(0);
  ellipse(x,y,80,80);
  fill(0);
  ellipse(x-15,y-10,20,20);
  ellipse(x+15,y-10,20,20);
}
```

A 9.1. Schreibe eine Funktion `hallo`, die den Text "Hallo Welt!" in die Konsole schreibt.

A 9.2. Schreibe eine Funktion `mauszeiger()`, die an die aktuelle Mausposition einen Kreis mit dem Durchmesser 10 Pixel zeichnet. Rufe die Funktion in der Funktion `draw()` zum Testen auf.

A 9.3. Räume den rechts stehenden Code des Programms auf. Das Programm zeichnet ein Haus und eine Sonne. Erstelle für das Zeichnen des Hauses und der Sonne jeweils die Funktionen `sonne()` und `haus()` und rufe sie in der Funktion `draw()` auf.

```
void setup() {
  size(800, 450);
  noStroke();
}

void draw() {
  background(255,140,0);
  // Boden
  fill(255,255,200);
  rect(0,350,width,height);
  // Haus
  fill(128,0,0);
  rect(80, 260, 160, 160);
  triangle(80, 260, 240, 260, 160, 200);
  // Sonne
  fill(255,255,0);
  ellipse(700, 100, 160, 160);
}
```

A 9.4. Erstelle ein Programm mit den Funktionen `eins()`, `zwei()` und `drei()`, die die jeweilige Zahl auf der Konsole ausgeben. Rufe in der Funktion `setup()` die Funktion `eins()` auf, in der Funktion `eins()` die Funktion `zwei()` auf und in der Funktion `zwei()` die Funktion `drei()` auf.

9.2 void mit Parameter

Listing 27. funktionvoid2.pde

```
 1: // Funktion vom Typ void mit Parameter
 2: // ================================================================================
 3: int x = 0;  // X-Koordinate
 4: int y = 0;  // Y-Koordinate
 5:
 6: void setup()
 7: {
 8:   size(800,600);                                                  // Fenstergröße
 9:   x = 50;
10:   y = height/2;
11: }
12:
13: void draw()
14: {
15:   background(255,140,0);                                          // Frame löschen
16:   paintFace(x,y-80);                                            // Gesicht zeichnen
17:   paintFace(2*x,y+80);                                           // Gesicht zeichnen
18:   x++;                                                 // X-Koordinate um 1 erhöhen
19: }
20:
21: // Zeichnet ein Gesicht um den angegebenen Punkt
22: // mittex und mittey sind lokale Variablen
23: void paintFace(int mittex, int mittey) {
24:   fill(255);
25:   stroke(0);
26:   ellipse(mittex,mittey,80,80);
27:   fill(0);
28:   ellipse(mittex-15,mittey-10,20,20);
29:   ellipse(mittex+15,mittey-10,20,20);
30: }
```

A 9.5. Schreibe eine Funktion `malZwei(x)`, die als Parameter eine Zahl `x` als `int` bekommt und das doppelte von `x` auf der Konsole ausgibt.

A 9.6. Schreibe eine Funktion `sterne(n)`, die als Parameter eine ganze Zahl `n` bekommt und dann `n` Sterne auf der Konsole ausgibt.

A 9.7. Schreibe eine Funktion, die als Parameter einen Text und eine ganze Zahl erwartet. Der übergebene Text soll mit einer Cäsarverschlüsselung verschlüsselt werden. Der Schlüssel ist dabei die übergebene ganze Zahl.

9.3 Funktion mit Rückgabewert

Listing 28. wuerfel.pde

```
1: // Funktion vom Typ int
2: // ============================================================================
3:
4: int wurf = 6;
5:
6: void setup()
7: {
8:   size(200,200);                                          // Fenstergröße
9:   textSize(64);
10:  textAlign(CENTER);
11: }
12:
13: void draw()
14: {
15:   background(255,140,0);                                 // Frame löschen
16:   text(wurf,width/2,height/2);
17: }
18:
19: int werfe_w6()
20: {
21:   return (int)random(1,7);
22: }
23:
24: void keyPressed()
25: {
26:   wurf = werfe_w6();
27: }
```

A 9.8. Schreibe eine Funktion `werfe_2w6()`, die das Ergebnis eines Wurfes von zwei Würfeln zurückgibt.

A 9.9. Schreibe eine Funktion `summeAllerZahlenBis(int zahl)`, die die Summe aller Zahlen von 1 bis zur angegeben Zahl zurückgibt.

A 9.10. Entwickelt werden soll eine Funktion `istGerade(int zahl)`, die zurückgibt, ob eine Zahl gerade ist.

a) Nenne einen sinnvollen Typ für die Funktion.
b) Nenne den Operator, der den Rest einer Division ausgibt.
c) Entwerfe die Funktion.

Inhaltsverzeichnis

1 Grundformen zeichnen **1**
1.1 Punkte, Strecken und Polygone . . . 1
1.2 Ellipsen zeichnen . . . 2
1.3 Hintergrund und Konturen . . . 3
1.4 Farben und Transparenz . . . 4
1.5 Text . . . 5

2 Statischer und Dynamischer Modus **6**

3 Position des Mauszeigers **7**

4 Eventfunktionen für Mausklick und Tastendruck **8**

5 Variablen **9**
5.1 Grundlagen . . . 9
5.2 Hoch- und Runterzählen . . . 10
5.3 Modulo . . . 11
5.4 Variablentypen . . . 12
5.5 Zeichenketten . . . 13

6 Entscheidungen **14**
6.1 if . . . 14
6.2 Der bewegte Ball . . . 15
6.3 if-else . . . 16
6.4 Tasten abfragen . . . 17
6.5 Anwendung π-Bestimmung . . . 18
6.6 Verschachtelte Entscheidungen . . . 19

7 Schleifen **20**
7.1 while-Schleife . . . 20
7.2 for-Schleife . . . 21

8 String-Objekt **22**
8.1 Zeichencodierung . . . 23

9 Funktionen **24**
9.1 void ohne Parameter . . . 24
9.2 void mit Parameter . . . 25
9.3 Funktion mit Rückgabewert . . . 26