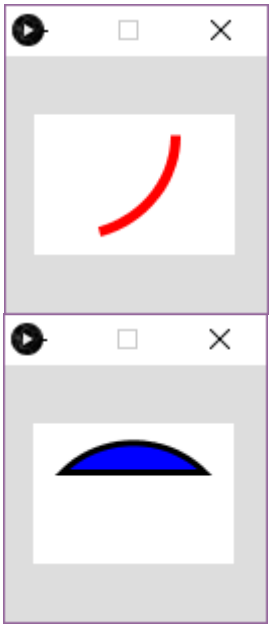
Kreissegmente

Du hast mit Hilfe der Anweisung `ellipse();` bereits einen Kreis gezeichnet.
Für das Zeichnen von Kreisteilen steht die Anweisung
`arc(x,y,Breite,Höhe,von,bis,Art)` zur Verfügung (vgl. Lerninhalte S. 6).

Kreisbogen
x:int y:int d:int wb:int rb:int gb:int bb:int
zeichne ()

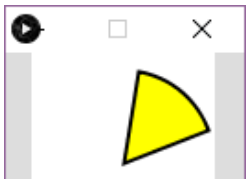
kb1:Kreisbogen
x=50
y=50
d=100
wb=75
rb=255
gb=0
bb=0

1. Ein roter Kreisbogen mit einem Radius von 100, dem Mittelpunktswinkel 75^0 im Uhrzeigersinn und der Linienstärke 5 soll gezeichnet werden. Die Linienenden sollen nicht abgerundet sein.
   Neben der Position des Mittelpunkts (`x`,`y`) und dem Durchmesser `d` wird ein Attribut `wb` für den Mittelpunktswinkel benötigt (vgl. Klassendiagramm rechts).
   - Ergänze in dem Objektdiagramm sinnvolle Werte.
   - Erstelle ein Programm dazu (`kreisbogen`).
     (*vgl. .\262-materialien\kreis\11-kreisbogen.htm*)



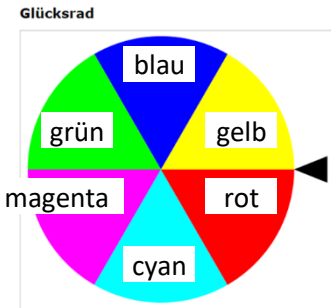
2. Ändere das Programm `kreisbogen` so ab, dass ein *Kreissegment* (Kreisbogen und Kreissehne) mit dem Anfangswinkel 45 Grad, dem Mittelpunktswinkel 90^0 **gegen** den Uhrzeigersinn, der Linienfarbe schwarz und der Linienstärke 3 gezeichnet wird (`kreissegment`).
   Hinweis: Winkel werden **im** Uhrzeigersinn gemessen.
   (*vgl. .\262-materialien\kreis\12-kreissegment*)

3. Ergänze das Programm `kreissegment` so, dass ein gelb gefüllter *Kreissektor* mit dem Anfangswinkel von 20 Grad, dem Mittelpunktswinkel 60^0 **gegen** den Uhrzeigersinn, der Linienfarbe schwarz und der Linienstärke 2 gezeichnet wird (`kreissektor`).
   (*vgl. .\262-materialien\kreis\13-kreissektor*)



Farben

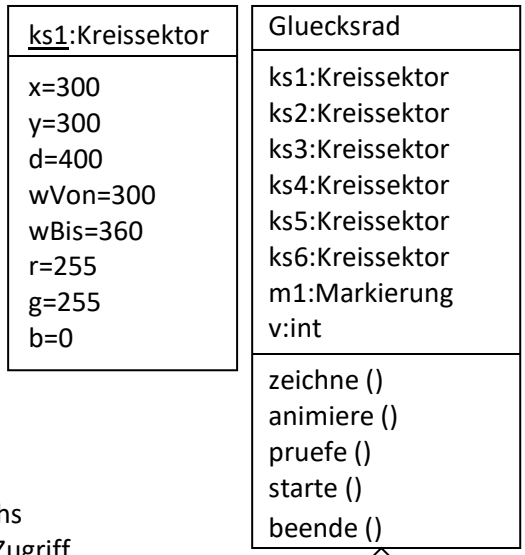
4. Mit einem Processing-Programm soll ein Glücksrad gezeichnet werden. Das Rad ist in sechs farbige Kreissektoren unterteilt; die Markierung zeigt nach der Drehung des Rads auf eines der sechs Felder.
   Ergänze das Programm `kreissektor` oder die Vorlagedatei `v13_kreissektor`.
   (*vgl. .\262-materialien\kreis\13-kreissektor*)



   Weitere Details zur Klassen- und Programmstruktur kannst du auch dem Programm `seifenkiste1` entnehmen (Vorlagedatei `v14_seifenkiste1`).

   Hinweise zu den Klassendiagrammen auf der nächsten Seite:
   - Das Attribut `v` steht für die Geschwindigkeit: Damit kann die Framerate variiert werden. Wähle zunächst `v=500`.
   - Die Winkel `wVon` und `wBis` in der Klasse `Kreissektor` geben Start- und Endwinkel an. Beispielsweise für den gelben Kreissektor gilt: `wVon=360-60` und `wBis=360`.

Rechts der Lösungsvorschlag für das Objekt `ks1`:



- Codiere die Objekte und speichere das Programm als `gluecksrad1`.
  Hinweis: Wenn beim Aufruf der Funktion `background()` nur ein Parameter angegeben wird, verwendet Processing anstatt des RGB-Farbmodells Grauwerte von 0 (Schwarz) bis 255 (Weiß). Möchtest du einen weißen Hintergrund, genügt also die Anweisung `background(255);`.

Arrays

Es ist etwas lästig, ständig identische Anweisungen für sechs Objekte zu schreiben. Mit Hilfe eines *Arrays* lässt sich der Zugriff auf mehrere Objekte ökonomischer erledigen:

- Ein **Array** (dt. **Feld**) ist ein Objekt, das mehrere Werte desselben Datentyps speichern kann.
  Ein eindimensionales Array kann man sich wie eine einzeilige Tabelle vorstellen, wobei die Spalten von 0 beginnend nummeriert sind:

[0]	[1]	[2]	[3]	[4]	[5]
(Wert)	(Wert)	(Wert)	(Wert)	(Wert)	(Wert)

  Damit können die sechs Objekte `ks[0]` bis `ks[5]` mit Hilfe einer Variablen angesprochen werden.
- Anweisung für die Deklaration eines Arrays:
  `Klasse[] Objekt = new Klasse [Anzahl];`
  Im Beispiel lautet die Anweisung:
  `Kreissektor[] ks = new Kreissektor[6];`
  Danach kann auf die Felder mit der Schreibweise `Objekt[Nummer]` zugegriffen werden,
  z. B. `ks[0]=new Kreissektor(300,300,400,360-60,360,255,255,0);`.
  Das ermöglicht den Zugriff mit Hilfe einer Variablen, z. B. auch mit `for` (vgl. Arbeitsblatt 04, S. 3):

```
for (int i = 0; i < 6; i++) {
  ks[i].zeichne();
}
```

- Ändere das Programm `gluecksrad1` wie beschrieben ab (`gluecksrad2`).

Zufallszahlen

- Die Funktion `random(a,b)` gibt eine zufällige Kommazahl im Intervall]a;b[zurück, die Funktion `floor()` **rundet** eine Zahl auf die nächstniedrige ganze Zahl **ab**.
  z. B. `Wurf=floor(random(1,7));` liefert eine ganze Zufallszahl von 1 bis 6.

- Eine ganze Zufallszahl von 720 bis 1080 erhält man folgendermaßen:

	erzeuge Zufallszahl (`von,bis+1`)	runde auf eine ganze Zahl ab
Wertzuweisung	`random(720,1080+1)`	`floor()`

```
Drehung=floor(random(720,1081));
```

5. In welchem Intervall werden im Beispiel ganze Zufallszahlen erzeugt?

	erzeuge Zufallszahl (`von,bis+1`)	runde auf eine ganze Zahl ab
Wertzuweisung	`random(1,2+1)`	`floor()`

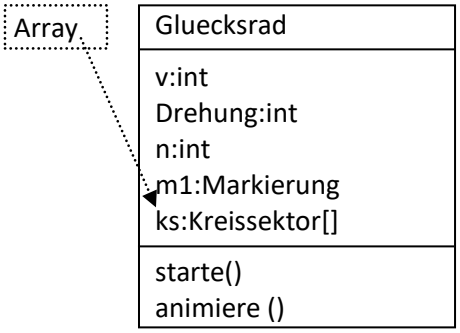
```
x=floor(random(1,3));
```

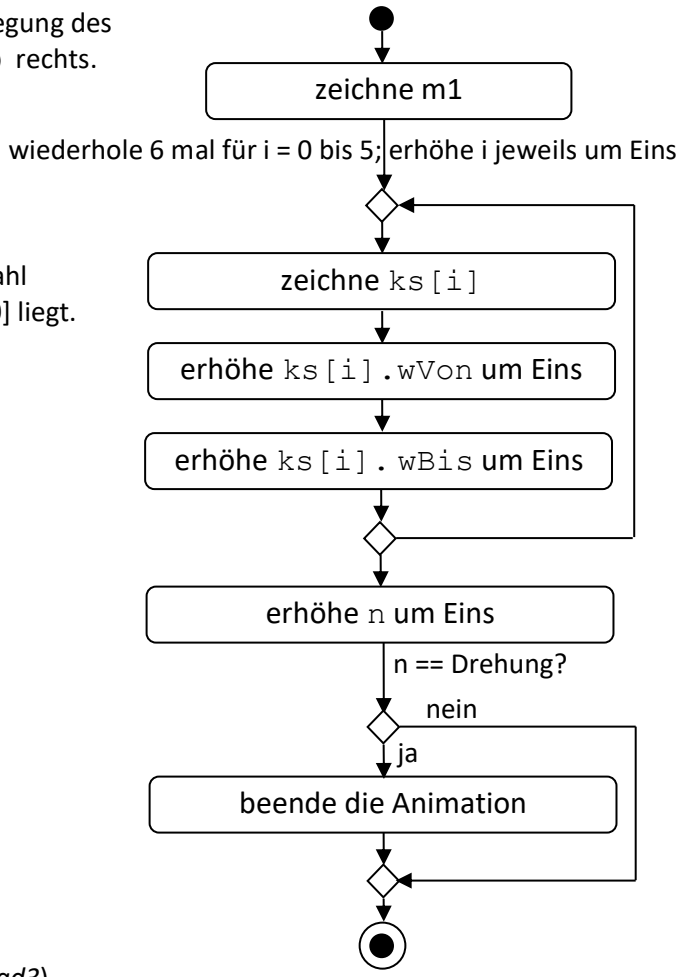
Intervall: *[1;2]*

6. Das Glücksrad aus Aufgabe 1 soll eine zufällige Anzahl im Bereich von zwei bis drei Umdrehungen ausführen. Die Framerate soll 500 betragen.

   Beachte das Aktivitätsdiagramm zur Bewegung des Glücksrads in der Methode `animiere()` rechts.

   Zum Klassendiagramm unten:
   - In der Methode `starte()` wird dem Attribut `Drehung` wird eine Zufallszahl zugewiesen, die im Intervall [720;1080] liegt.
   - In dem Attribut `n` wird die Anzahl der Drehungen mitgezählt.
   - Die Framerate wird zunächst auf 500 (`v`) gesetzt.





Aktivitätsdiagramm für die Methode animiere()

- Ergänzedas Programm `gluecksrad2`.
  (Vorlagedatei: `v15_gluecksrad2`)
  Speichere die Datei als `gluecksrad3`.
  (vgl. .\262-materialien\zufall\03-gluecksrad3)

- Teste das Programm, indem du es mehrfach startest.

Zusatzaufgaben

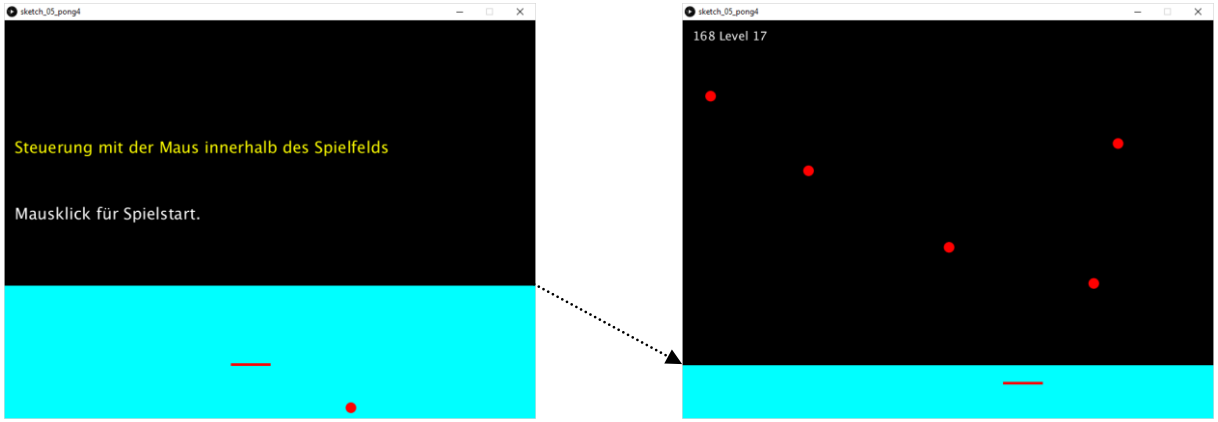
7. Das Spiel Pong kann etwas abwechslungsreicher gestaltet werden, indem der Ball mit einer zufälligen Bewegungsrichtung an einer zufälligen x-Position startet (Vorlagedatei: `v16_pong1`).
(*vgl. .\262-materialien\pong\02_pong2*)

   Hinweise:
   - Empfehlung für die Intervalle der Zufallszahlen: Für `x`]20;590[, für `dx`]0,2;0,8) und für `dy`]-2;-1[
   - Das Dezimalkomma ist als Dezimalpunkt zu schreiben, als Trenner ist ein Komma zu verwenden!

8. Entwickle eine Variante des Spiels Pong für zwei Spieler mit zwei Schlägern und zwei Bällen, die sich farblich unterscheiden.
(*vgl. .\262-materialien\pong\03_pong3*)

   Hinweise:
   - Als Steuerungstasten für den zweiten Schläger sind z. B. die Tasten <Y> und <X> geeignet.
   - Die Abfrage dieser Tasten erfolgt mit der Anweisung `if (key == 'y')` { nicht `keycode`!).

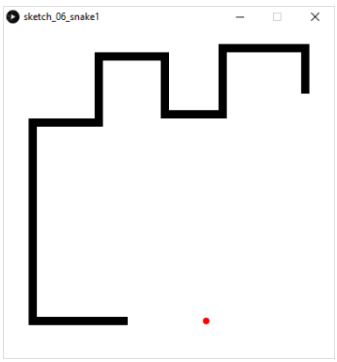
9. Bei einer weiteren Variante des Spiels Pong für einen Spieler könnte der Schwierigkeitsgrad bzw. der Level alle zehn Treffer erhöht werden. Der Schläger wird hier nicht mehr mit der Tastatur, sondern mit der Maus innerhalb eines farbig markierten Bereichs des Spielfelds bewegt.
(*vgl. .\262-materialien\pong\05_pong4*)



   - Zunächst wird die Anzahl der Bälle erhöht, maximal 5.
   - Daraufhin wird bis zu einem Spielstand 100 Treffern die Geschwindigkeit um 10 erhöht.
   - Ab einem Spielstand von mehr als 100 Treffern wird siebenmal das Feld, in dem die Maus bewegt werden kann, um 20 Pixel niedriger.
   - Bei einem Spielstand von 200 Treffern wird das Spiel beendet und der Kommentar "Gewonnen! Du Mauscrack!" angezeigt.

   Hinweise:
   - Es ist sinnvoll, die Ballobjekte in einem Array zu verwalten.
   - Die Mausposition innerhalb des Zeichenfensters kann aus den Variablen `mouseX` und `mouseY` ausgelesen werden.

10. Entwickle ein anderes Computerspiel. Wie wäre es mit Snake? Du kannst Teile des Spiels Pong verwenden.
(*vgl. .\262-materialien\pong\06_snake1*)



   Hinweise:
   - Zur Steuerung benötigt man zusätzlich zu den Tasten Pfeil nach links und Pfeil nach rechts noch die Tasten Pfeil nach oben und Pfeil nach unten („`UP`“ und „`DOWN`“).
   - Für die Körperteile der Schlange ist ein Array erforderlich.