ABSCHLUSS

Entweder etwas Eigenes:

Formuliere dazu in 1-3 Sätzen was dein Ziel ist / was du umsetzten möchtet und erstelle 1-4 Unterziele, also Zwischenetappen die du schaffen möchtest. Schick mir diese per E-Mail. Behalte sie aber auch und passe sie an, falls du merkst dass etwas nicht oder anders funktioniert oder unrealistisch war.

Oder eines dieser zwei Games:

BRICK BREAKER BASICS
Ein Ball der rumhüpft/an allen Seiten ausser der unteren abprallt und mit einer Plattform davon abgehalten werden soll unten raus zu fallen. Die Plattform kann mit der Maus oder den Pfeiltasten setitlich bewegt werden.
Unterziele:
- Ein Ball der sich bewegt und an drei Seiten abprallt
- Eine Plattform die vom Spieler bewegt werden kann
- Der Ball prallt auch von der Plattform ab
- Game Play Elemente wie: ein Score der für jede Berührung erhöht wird. Ein Game Over Screen etc
- Mögliche Erweiterung: Brick Breaker, der Ball trifft oben auf Steine, die zerstört werden und einen Punkt geben wenn sie berührt werden.

FLAPPY WHATEVER
Irgend ein Element (Whatever) welches über die Pfeiltaste Auftrieb bekommt und wieder runter fällt und so an Hindernissen vorbei gesteuert werden muss, die auf das Whatever-Element zukommen.

Erweiterung: Um den Fall nach unten realistischer wirken zu lassen, könnte ein Gravitationseffekt eingebaut werden. Die Gravitation wirkt sich auf die Beschleunigung aus: a = m*g, das heisst dass die Bewegung nach unten nicht linear verläuft sondern immer schneller wird, also beschleunigt. Für die Programmierung heisst das, dass wir die Position nicht wie bisher über einen konstanten Wert verändern:
setup: `int speed = 5;` **draw**: `y = y+speed;`
Sondern dass wir auch speed in jedem Schritt verändern:
setup: `int speed = 5; int gravity = 0.1;` **draw**: `speed = speed + gravity; y = y + speed;`
Wird speed für den Auftrieb nun umgekehrt wird weiterhin der gravity Wert dazu gerechnet, so dass das Whatever nach einer Zeit wieder nach unten fällt.

Unterziele:
- Das Whatever welches sich nach unten bewegt und über die Pfeiltaste wieder nach oben gebracht werden kann.
- Hindernisse welche merken, wen sie vom Whatever berührt werden
- Einen Mechanismus, der die Hindernisse auf das Whatever zu bewegt
- Erweiterung des Mechanismus, so dass immer neue Hindernisse erstellt werden die sich auf das Whatever zu bewegen und gelöscht werden, wenn sie aus dem Bild raus sind.
- Game Play Elemente wie: einen Score und einen Game Over Screen etc

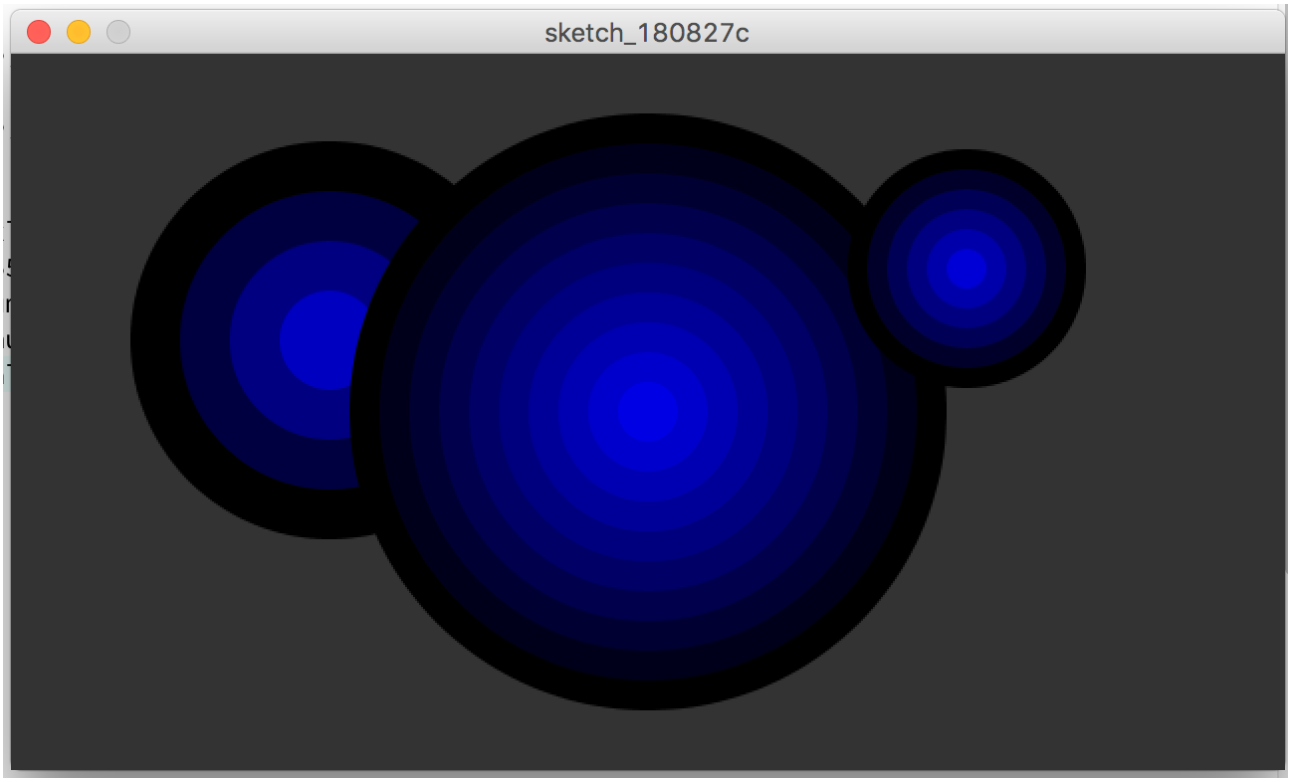
Oder repetiere alte Aufgaben und löse Aufgaben von hier:

METHODEN

Methoden sind dazu da Code zu strukturieren und ihn wiederverwendbar zu machen.

Aufgabe - Kreise

Schreibe und teste eine Methode, bei deren Aufruf wie im Bild Kreise gezeichnet werden sollen. Dabei soll bei jedem Aufruf die Position, Grösse (Gesamtgrösse in der alle Ringe enthalten sind) und Anzahl Ringe angegeben werden können.

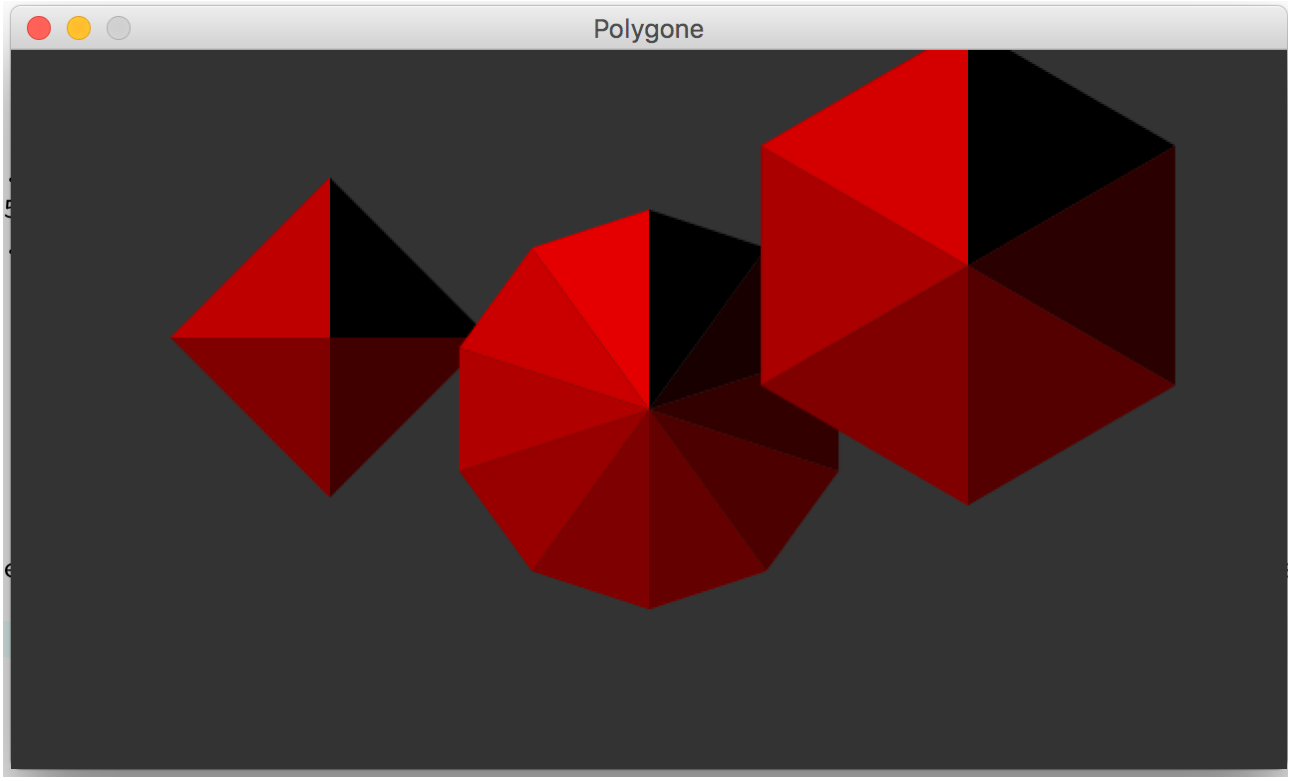


Tipp: Du hast eine gegebene Anzahl Ringe, benutze also eine for Schleife um die Kreise zu zeichnen. Wie soll sich der Farbwert für jeden Ring in der Schleife verhalten wenn blau (0, 0, **255**) und schwarz (0, 0, **0**) ist? Wie soll sich die Ring/Kreisgrösse für jeden Ring in der Schleife verhalten wenn du die Gesamtgrösse und die Anzahl Ringe kennst?

Rufe die Methode nun bei jedem Mausklick an Position der Maus mit einer zufälligen Grösse und einer zufälligen Anzahl Ringe auf.

Aufgabe - Polygone

Schreibe und teste eine Methode, bei deren Aufruf wie im Bild Polygone aus Dreiecken gezeichnet werden sollen. Dabei soll bei jedem Aufruf die Position, Grösse des Polygons und Anzahl Dreiecke angegeben werden können.



Tipp: Benutze die Parameterdarstellung für einen Kreis und eine Schleife um die Dreiecke zu zeichnen. Die cos() und sin() Methoden in processing nehmen den Winkel in Radiant entgegen.

Parameterdarstellung [Bearbeiten | Quelltext bearbeiten]

Eine andere Möglichkeit, einen Kreis durch Koordinaten zu beschreiben, bietet die Parameterdarstellung (siehe auch Polarkoordinaten):

$$x = x_M + r\cos\varphi$$
$$y = y_M + r\sin\varphi$$

M = Mittelpunkt
r = Radius

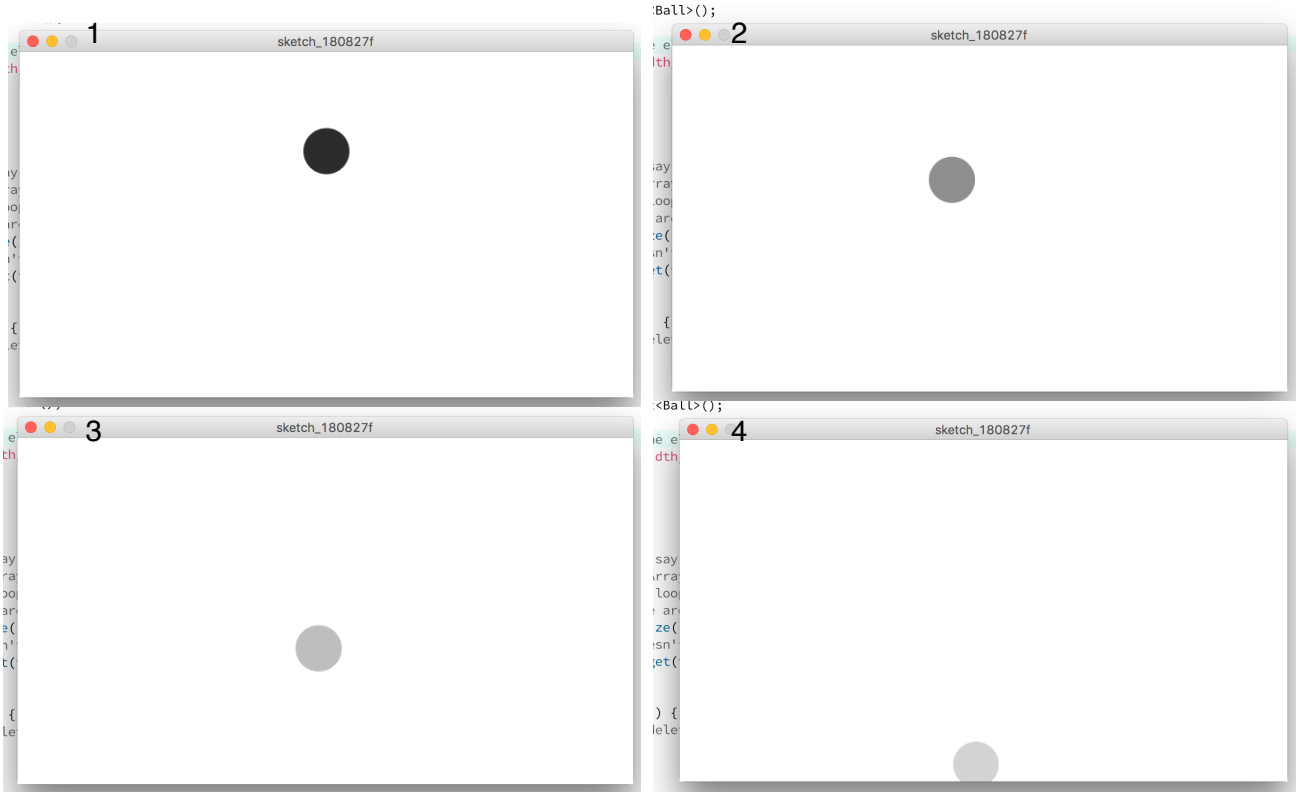
Hier werden die Koordinaten x und y durch den Parameter φ ausgedrückt, der alle Werte mit $0 \leq \varphi < 2\pi$ annehmen kann.

Quelle: https://de.wikipedia.org/wiki/Kreis

Rufe die Methode nun bei jedem Mausklick an Position der Maus mit einer zufälligen Grösse und einer zufälligen Anzahl Ringe auf.
Wenn du die Aufgabe mit den Kreisen auch gelöst hast, kannst du bei jedem Mausklick auch zufällig entweder einen Kreis oder ein Polygon zeichnen.

OBJEKTE UND ARRAY LIST

Wir wollen Bälle erstellen, die runterfallen und nach einer gewissen Zeit „sterben“. Um das sterben sichtbar zu machen sollen die Bälle ihre Farbe verändern. Sie sollen mit einer Anfangsfarbe starten und dann immer mehr wie der Hintergrund werden. Sobald sie die gleiche Farbe angenommen haben werden sie gelöscht.



BÄLLE DURCH MAUSKLICK

1.

Erstelle eine Ball Klasse, die über eine move() Methode verfügt, die den Ball nach unten fallen lässt. Und eine paint() Methode welche einen Kreis zeichnet.

Erstelle im Main script eine ArrayList für die Speicherung von Bällen. Füge dieser Liste für jeden Mausklick einen Ball hinzu. Rufe in der draw Methode für jeden Ball in der ArrayList move() und paint() auf.

2.

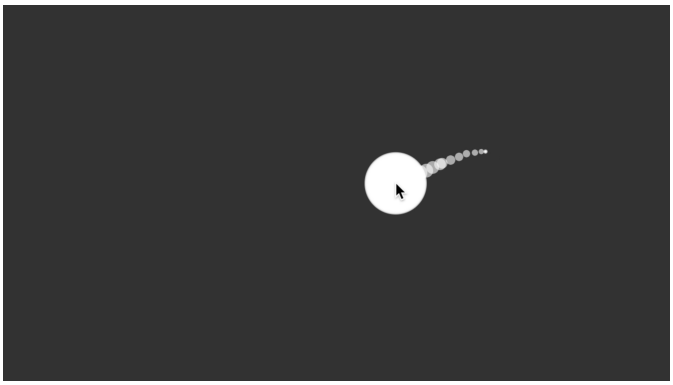
Füge eine lifeSpan Variable hinzu und initialisiere sie mit 255:

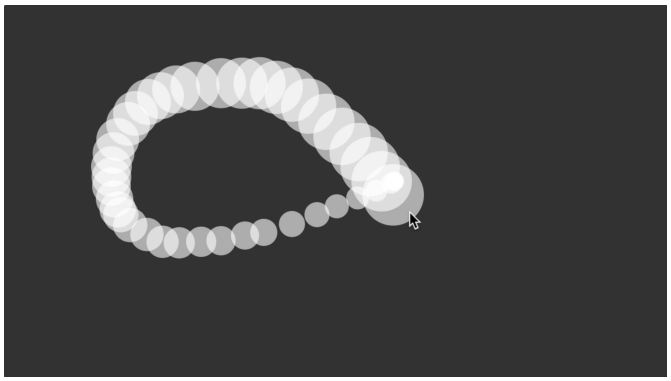
```
float lifeSpan = 255;
```

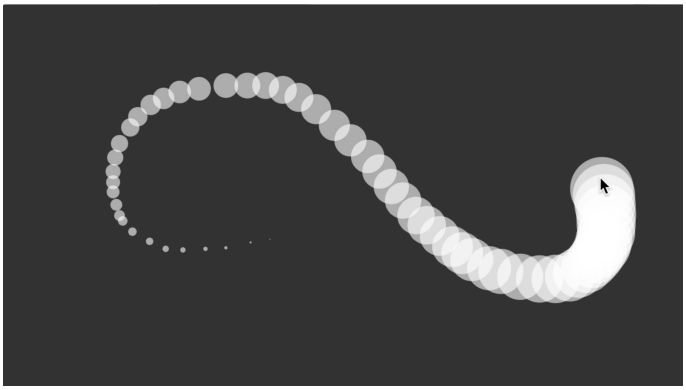
Zähle den lifeSpan für jeden move() Aufruf eins runter. Füge der Ball Klasse eine Methode `boolean isFinished()` hinzu, welche angibt ob das Leben des Balls abgelaufen ist. Sein Leben ist abgelaufen wenn lifeSpan 0 erreicht hat. Benutze den lifeSpan auch um die Farbe des Balls zu verändern, so dass man auch sieht wenn er „Tod“ ist. Lösche einen Ball aus der ArrayList, wenn er Tod ist.

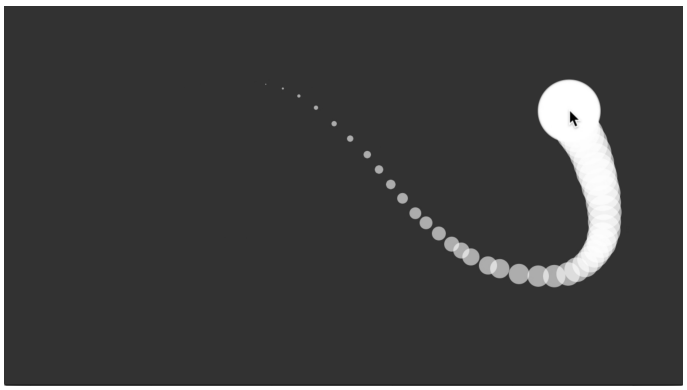
MAUS MIT BALL SCHWEIF

Wir wollen nun die gleichen Bälle verwenden aber so, dass sie wie ein Schweif der Maus folgen. Dabei sollen vorne bei der Maus immer neue entstehen und hinten am Schweif die alten sterben. Füge nun anstatt bei jedem Mausklick immer Bälle an der Mausposition hinzu. Lösche diejenigen Bälle die „Tod“ sind. Du kannst die lifeSpan Variable auch benutzen um die Transparenz des Balls zu verändern. Dazu kannst du die Farbe anstatt mit drei Werten (R, G, B) mit vier Werten (R, G, B, A) angeben. A steht für Alphakanal und somit die Transparenz.









Probiere ruhig aus, was du für Effekte erzielen kannst, wenn du zum Beispiel die Linien doch zeichnest, die Grösse abhängig von der Lebensdauer veränderst oder die Lebenslänge veränderst etc.

Erweitere dieses Programm so, dass du nicht einfach Bälle zeichnest sondern die Kreise oder Polygone aus der Aufgabe oben. Probiere hier auch wieder aus, was du noch für Effekte erzielen kannst.