Flattervogel

ein Flappy Bird-Klon

Autor: Benno Beispiel

Klasse: 10x

Schuljahr: 2020/21

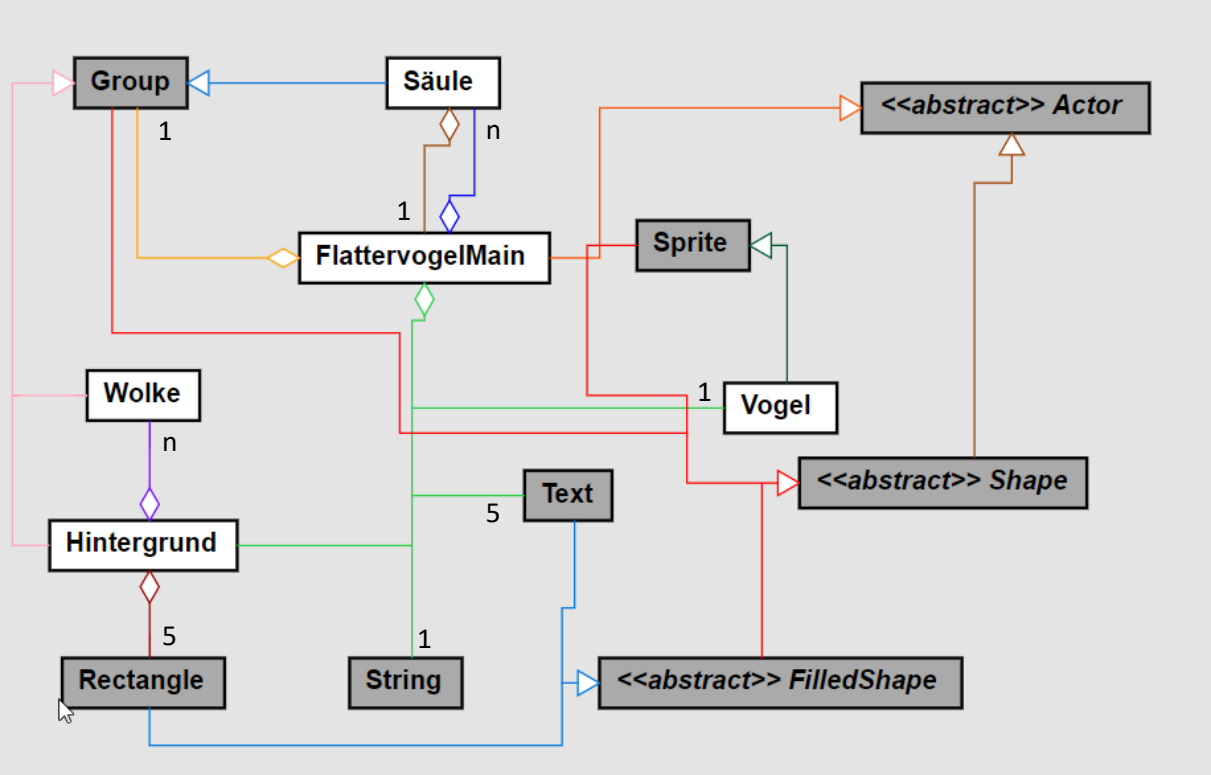
1. Projektvorhaben

„Flappy Bird ist ein Spiel des vietnamesischen Entwicklers Dong Nguyen aus dem Jahr 2013“[1], bei dem der Spieler einen kleinen Vogel steuert, der auf paarweise grüne Säulen zufliegt. Zwischen jedem Säulenpaar ist jeweils eine Lücke, durch die der Spieler den Vogel lenken muss. Berührt dieser die Säulen oder fliegt er oben oder unten aus dem Bildschirm, so ist das Spiel beendet.

Der Reiz des Spiels besteht darin, dass zur Steuerung nur die Leertaste verwendet wird. Bei jedem Drücken dieser Taste „flattert“ der Vogel und bewegt sich so ein Stück nach oben. Da ständig eine Beschleunigung in positive y-Richtung (d.h. nach unten) wirkt, verliert der Vogel in den Phasen, in denen die Leertaste nicht gedrückt wird, an Auftrieb und beschleunigt nach unten.

Die x-Koordinate des Vogels bleibt immer konstant. Trotzdem entsteht der Eindruck des Fliegens nach rechts, da von rechts her Säulen mit konstanter Geschwindigkeit herankommen sowie Wolken mit ebenso konstanter (aber etwas geringerer) Geschwindigkeit.

2. Klassendiagramm



FlattervogelMain ist die Hauptklasse des Programms. Im Hauptprogramm wird ein Objekt dieser Klasse instanziert und deren Konstruktor kümmert sich dann um die Instanzierung der weiteren per Aggregation enthaltenen Objekte:

- ein Hintergrund-Objekt (Unterklasse von Group), das mehrere Wolken-Objekte enthält sowie mehrere Rechteck-Objekte (für den Himmel und die Wiese)
- ein Vogel-Objekt (Unterklasse von Sprite)

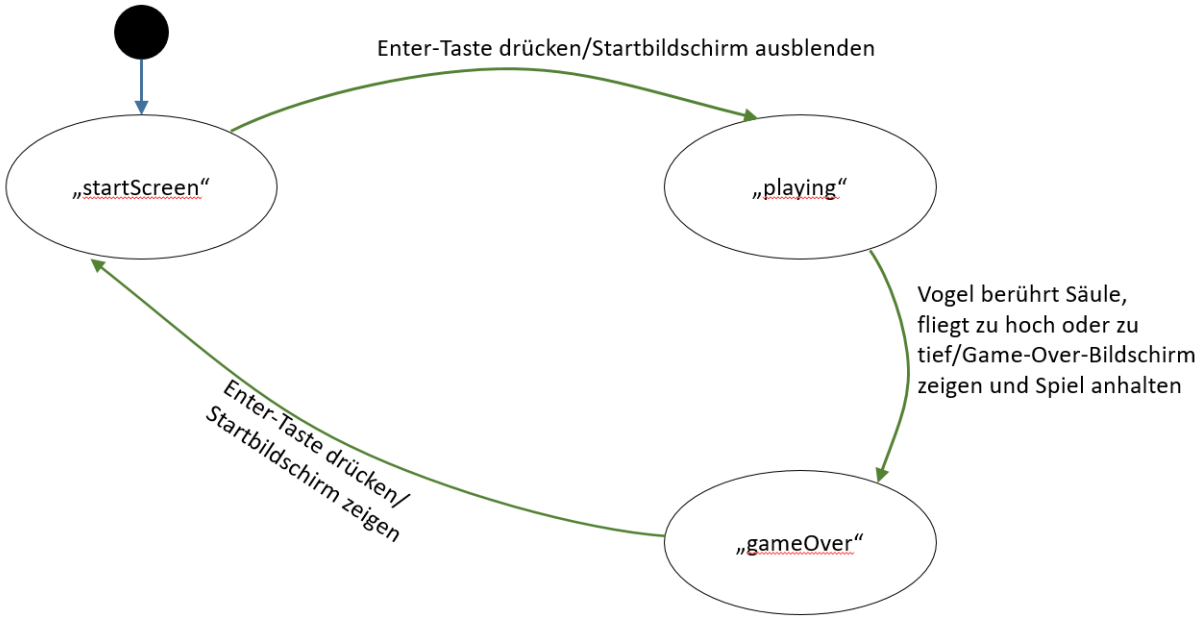
[1] Siehe https://de.wikipedia.org/wiki/Flappy_Bird

- 5 Text-Objekte (zwei für den Text auf dem Startbildschirm, eines zur Anzeige der Punktezahl und zwei für den Text auf dem Game-Over-Bildschirm)
- mehrere Säulen-Objekte (im Group-Objekt `säulen`)
- ein Text-Objekt `zustand` zur Speicherung des aktuellen Zustands (siehe Kapitel 3)

Die Säule-Klasse stellt ein Säulen-Paar dar. Sie ist Unterklasse von Group und enthält die Rechtecke, aus denen das Säulenpaar zusammengesetzt ist.

FlattervogelMain ist Unterklasse von Actor und überschreibt dessen Methode `act()`, die 30-mal je Sekunde vom System aufgerufen wird.

3. Zustandsübergangsdiagramm

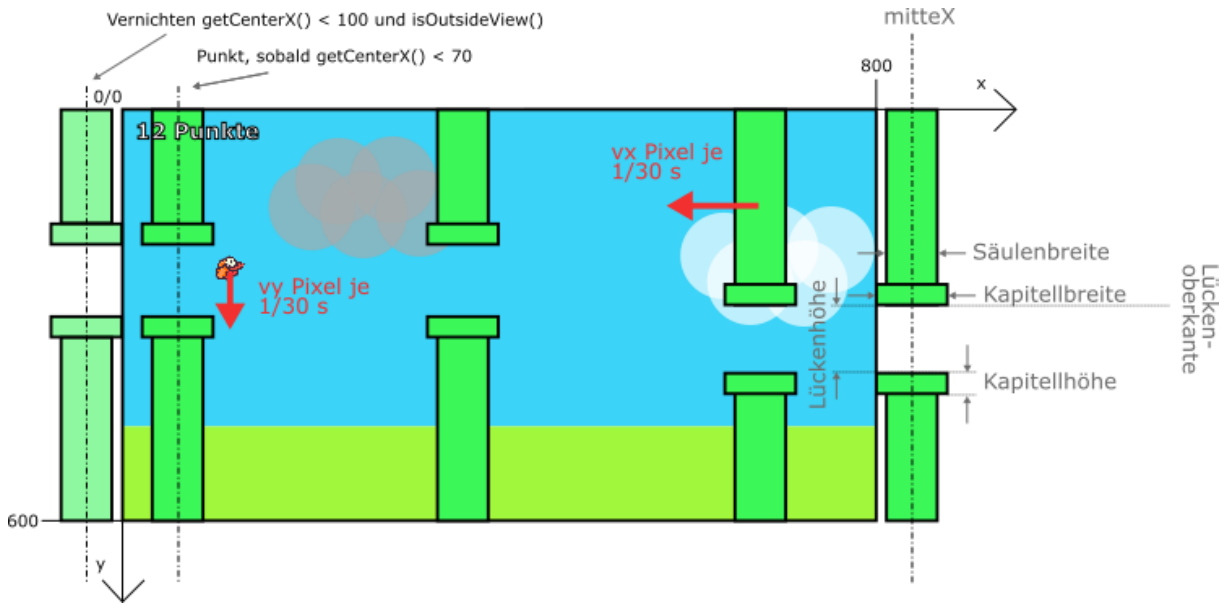


Der aktuelle Zustand wird im Attribut `zustand` der Klasse FlattervogelMain gespeichert. Zustandsübergänge werden jeweils durch Aufruf der Methode `setZustand` herbeigeführt, die sich um den Start der ausgelösten Aktionen kümmert.

4. Erzeugung/Vernichtung der Säulen/Punktevergabe

Die Säulen werden rechts außerhalb des sichtbaren Bereichs erzeugt. Sie bestehen aus jeweils zwei rechteckigen Säulenrümpfen und zwei rechteckigen Kapitellen. Alle vier Rechtecke sind anfangs in X-Richtung um den Wert `mitteX = 800 + Kapitellbreite/2` zentriert und so direkt rechts am sichtbaren Bereich angrenzend. Die Klasse Säule ist Unterklasse der Klasse Group, deren `act`-Methode überschrieben wird. Sie wird 30-mal je Sekunde vom System aufgerufen und kümmert sich um die gleichmäßige Bewegung nach links und die Vernichtung der Säule, sobald sie links aus dem Bildschirm herausgetreten ist. Als Kriterium hierfür kann nicht alleine die Methode `isOutsideView` verwendet werden, da die Säulen sonst direkt nach der Erzeugung vernichtet werden würden (denn zu diesem Zeitpunkt befinden sie sich ja rechts außerhalb des sichtbaren Bereichs). Pragmatischerweise wird daher auf `getCenterX() < 100 && isOutsideView()` getestet, denn wenn `getCenterX() < 100` ist, befindet sich die Säule sicher nicht rechts außerhalb des sichtbaren Bereichs.

Zur Erläuterung sowie zum Verständnis der Terme im Quellcode des Konstruktors der Klasse Säule siehe die nachfolgende Skizze.



In der Methode `act` der Säule wird auch geprüft, ob die Säule schon vom Vogel durchflogen wurde: Ist `getCenterX()` erstmals kleiner als 70, so wird die Methode `punktHinzuzählen` des FlattervogelMain-Objekts aufgerufen.

5. Bewegung des Vogels

Die Klasse Vogel ist Unterklasse der Klasse Sprite, deren `act`-Methode überschrieben wurde. Sie kümmert sich um die Tastatursteuerung und die Bewegung des Vogels.

30-mal pro Sekunde wird der Vogel um den Wert des Attributs `vy` (= Geschwindigkeit in y-Richtung) verschoben. Dieser ist anfänglich 0 wird aber 30-mal je Sekunde um 0,9 erhöht, wodurch die Beschleunigung des Vogels nach unten erreicht wird. Wird die Leertaste gedrückt, so wird `vy` auf den Wert -10 gesetzt. Diese schlagartige Änderung der Geschwindigkeit wird als „Flattern“ des Vogels sichtbar.

Die Neigung des Vogel-Bildes wird in der Methode `setBirdAngle` einfach abhängig von seiner Geschwindigkeit `vy` in y-Richtung vorgenommen:

```
Neigungswinkel = -vy * 3
```

Ein positiver Neigungswinkel entspricht einer Linksdrehung und korrespondiert daher mit einem negativen Wert von `vy` (d.h. Bewegung nach oben, da die y-Achse nach unten zeigt).

6. Die act-Methode der Klasse FlattervogelMain

Diese Methode kümmert sich im Zustand „playing“ um die Kollisionserkennung zwischen Vogel und Säulen und führt ggf. eine Zustandsänderung zum Zustand „gameOver“ herbei.

In den Zuständen „StartScreen“ und „GameOver“ wird überprüft, ob die Enter-Taste gedrückt wurde und entsprechend eine Zustandsänderung nach „playing“ bzw. „Startscreen“ veranlasst.

Jeweils nach der Zeit `zeitZwischenSäulen` * 1/30 s wird ein neues Säule-Objekt instanziert und der Gruppe `säulen` hinzugefügt.

Mit fortschreitender Spieldauer soll das Spiel schwerer werden. Dies wird dadurch erreicht, dass `zeitZwischenSäulen` sowie `minLückenhöhe` (minimale Höhe der Lücken zwischen den Säulen) ständig erniedrigt werden sowie die Betrag der Säulengeschwindigkeit `vx` erhöht wird:

7. Erzeugung der Wolken

Die Klasse Wolke ist Unterklasse der Klasse Group und besteht aus mehreren Circle-Objekten. Ihre act-Methode ist überschrieben und kümmert sich um die Bewegung der Wolke nach links sowie um ihre Vernichtung links außerhalb des sichtbaren Bereichs, ähnlich wie in der Säule-Klasse.

Erzeugt wird die Wolke rechts außerhalb des sichtbaren Bereichs. Dazu werden zunächst Richtwerte für ihre Helligkeit und ihre „Flughöhe“ zufällig bestimmt:

```
// Helligkeit und "Höhe" der Wolke
double helligkeit = Math.round(Math.random() * 128 + 128);
double y = -100 + Math.random() * 400;
```

Anschließend werden mehrere Kreise erzeugt, deren „Flughöhe“ und Helligkeit leicht um diese Richtwerte streuen. Die X-Koordinate ihres Mittelpunkts wird – ebenfalls in einem engen Intervall – zufällig bestimmt. Dadurch entstehen eng zusammenliegende Kreise ähnlicher Helligkeit, die zusammen eine Wolke bilden.

Die Instanzierung einer neuen Wolke erfolgt in der `act`-Methode der Klasse Hintergrund in zufälligen Zeitabständen. Alle Wolke-Objekte werden dann dem Hintergrund-Objekt (Unterklasse von Group!) hinzugefügt. Dadurch wird bewirkt, dass sie immer „hinter“ den Säulen und dem Vogel gezeichnet werden aber vor dem hellblauen Rechteck, das den Himmel darstellt und als allererstes (und damit „zuunterst“) der Hintergrund-Gruppe hinzugefügt wurde.

8. Absturz des Vogels

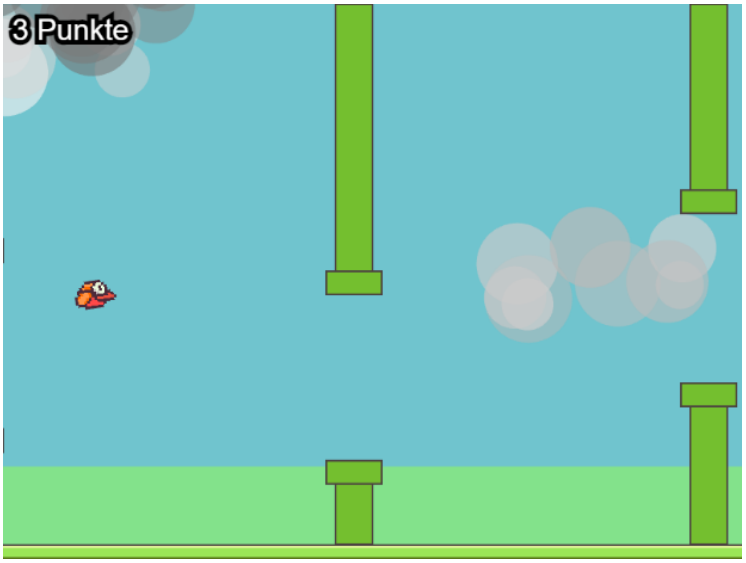
Das boolesche Attribut „`active`“ der Klasse Vogel legt fest, ob der Vogel gerade durch die Tastatur steuerbar ist oder nicht. Bei der Zustandsänderung von „playing“ nach „gameOver“ wird dieses Attribut auf `false` gesetzt. Danach wird die Methode `absturz()` des Vogel-Objekts aufgerufen, die `vy` auf 20 setzt. Damit bewegt sich der Vogel schnell nach unten und ist durch die Tastatur nicht mehr steuerbar.

9. Erneuter Spielbeginn

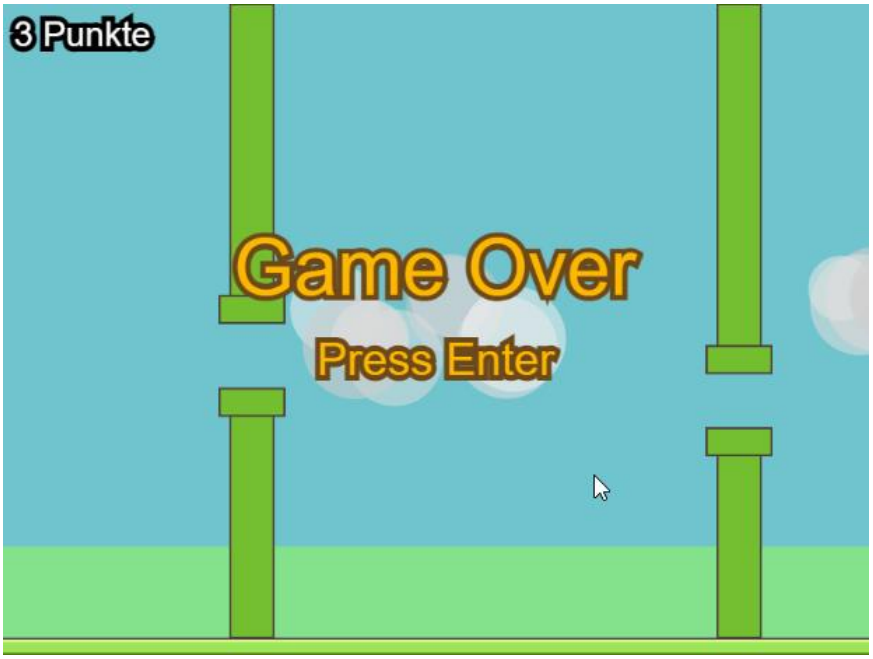
Bei der Zustandsänderung von „startScreen“ nach „playing“ wird das Attribut `active` des Vogel-Objekts wieder auf `true` gesetzt. Zudem werden durch Aufruf der Methode `reset` des Vogel-Objekts seine Position und seine Geschwindigkeit wieder auf die Ausgangswerte zurückgesetzt und alle Säulen-Objekte aus der Säulen-Gruppe gelöscht.

10. Screenshots

Startbildschirm



Während des Spiels



Game Over-Bildschirm