UNIVERSITÄT
LEIPZIG

Softwaretechnikpraktikum

Entwurfsbeschreibung

Gruppe: nw19a

Mitglieder: Thomas Pause, Sabine Lorius, Arik Korte, Martin George, Josephine Lange, Esther Prause, Anh Kiet Nguyen, Bärbel Hanle

Verantwortlich: Sabine Lorius

Betreuer: Dr. Nicolas Wieseke

Tutor: Martin Frühauf

Abgabedatum: 06.01.2020

Version: 0.5 (Vorprojekt)

Stand: 6. Januar 2020

Inhaltsverzeichnis

1 Visionen und Ziele

Ziel ist die Entwicklung einer Anwendung als *Proof-Of-Concept*, die es *Poi*-Spielern ermöglicht zu berechnen, auf welche Weise beliebige *Poi*-Figuren erzeugt werden können (beschränkt auf den 2-dimensionalen Raum). Als einfachste Stufe sollen im Vorprojekt mit beliebigen Eingabeparametern sog. *Flowers* berechnet werden.

2 Rahmenbedingungen und Produktübersicht

2.1 Rahmenbedingungen

Die Anwendung wird als Android App mit einer kleinsten unterstützten Display-Auflösung von 768x1366 Pixel (HD) und dem Minimum *API-Level* 25 (Android 7.1.1) umgesetzt. Als Programmiersprache kommt maßgeblich *Kotlin* zum Einsatz.

2.2 Produktübersicht

Die im Release 1 umgesetzte App bietet dem User die Möglichkeit, über die frei veränderlichen Eingabeparameter **radius** (Verhältnis der Radien Poikreis zu Handkreis), **freq1** (Frequenz des Handkreises), **freq2** (Frequenz des Poikreises) und **offset** (Drehung der *Flower*) beliebige *Flowers* zu generieren. Die zur Darstellung einer solchen *Flower* nötige Handbewegung eines *Poi*-Spielers kann darauf hin (zunächst ohne jede Berücksichtigung physikalischer Aspekte) berechnet und gemeinsam mit der *Flower* graphish präsentiert werden.

3 Grundsätzliche Struktur- und Entwurfsprinzipien

Im Rahmen des ersten Release konnten alle bis dahin implementierten Funktionalitäten noch in einem Paket gehalten werden. Mit steigender Komplexität der App und mit Hinzukommen neuer Funktionen werden die zu den verschiedenen Bereichen gehörenden Klassen jedoch in separaten Paketen zu organisieren sein.
Das *Flower*-Menü wird durch drei Klassen realisiert. Zum einen die Klasse `FlowerActivity`, deren Layout durch die `layout_flower_activity.xml` definiert wird. Hier werden alle UI-Elemente instantiiert und die für die User-Interaktion benötigten EventHandler angefügt. Des Weiteren werden hier Methoden der Klasse `FlowerCalc` aufgerufen, die auf Grundlage der vom User eingegebenen Werte für Radius, Hand- und Poifrequenz sowie Offset, die Koordinaten der entsprechenden *Flower* berechnen. Das Ergebnis wird in der Klasse `FlowerActivity` entgegengenommen und an eine Instanz der `FlowerCanvasView` weitergeleitet. Diese Klasse stellt einen Canvas auf der Benutzeroberfläche zur Verfügung, auf dem die *Flower* gerendert werden kann.

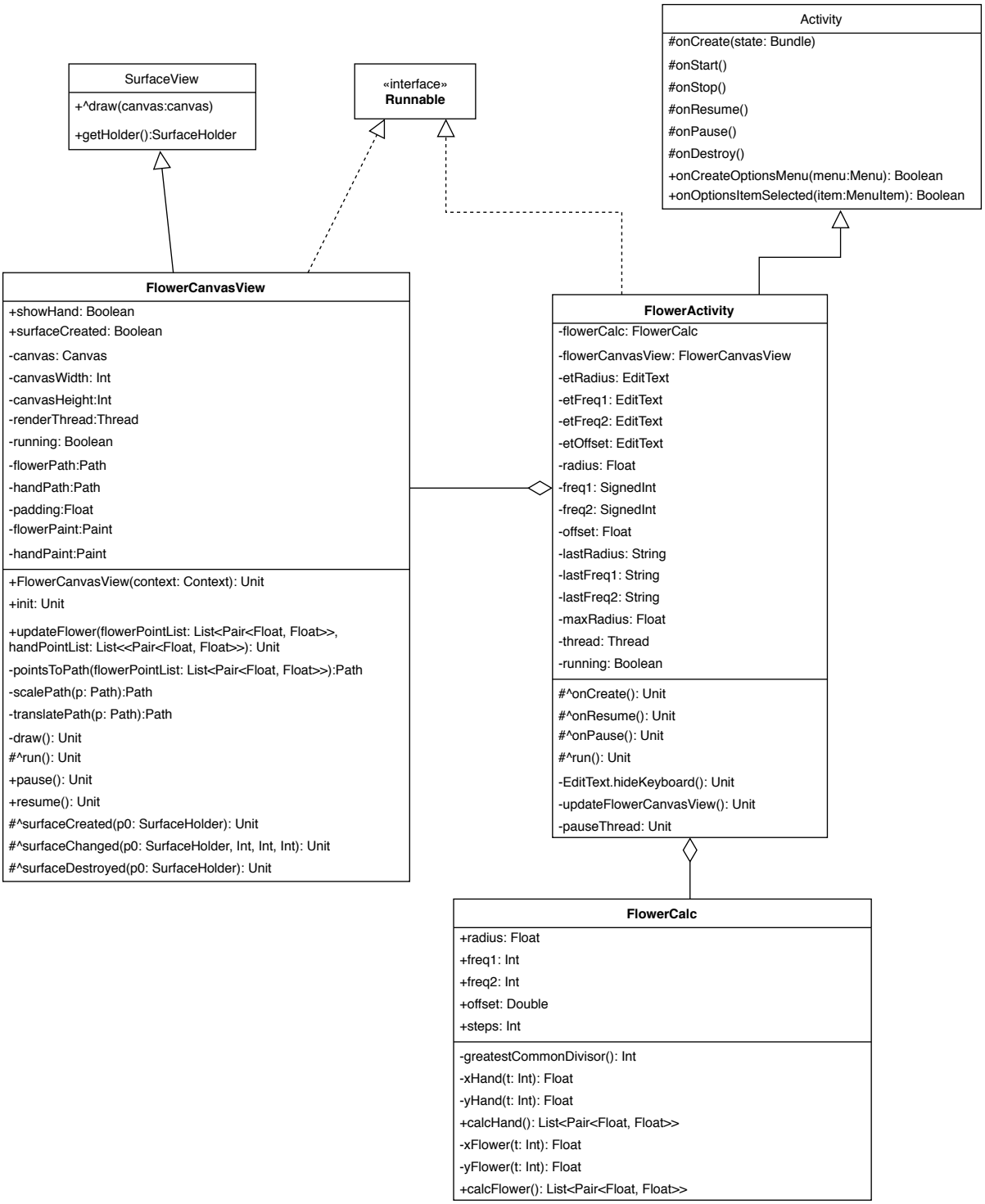

Abbildung 1: UML-Diagramm der oktopoi-App, Stand Release 1.

4 Struktur- und Entwurfsprinzipien einzelner Pakete

4.1 Struktur (Arbeitspaket 1)

Arbeitspaket 1 beinhaltet unter anderem die Erstellung von Activities. In Release 1 betrifft das die `FlowerActivity`-Klasse und die `layout_flower_activity.xml`, in der das zugehörige Layout festgelegt wird. Diese Activity nimmt als User-Eingaben die Parameter Radius, Handfrequenz, Poifrequenz und den Offset (Drehung der Flower) entgegen und stellt dazu eine *Flower* auf dem Bildschirm dar. Der Nutzer kann dabei zwischen einer Darstellung mit und ohne Handkreis wählen

Entscheidungen:

- Da interessante *Flowers* für Radien nahe 1 erwartet werden, schränken wir den Bereich für die Werte, die dieser Parameter entgegen nimmt, auf Werte zwischen 0 und 3 ein.

4.2 Berechnungen (Arbeitspaket 3)

Aus Arbeitspaket 3 wurden die Berechnung der *Flower*- und der Handkoordinaten für den *Flowers*-Bereich umgesetzt. Die entsprechenden Funktionen finden sich in der `FlowerCalc`-Klasse. Die folgenden Tabellen geben eine Übersicht über die Variablen eines `FlowerCalc`-Objekts, sowie die von ihm bereitgestellten Funktionen.

Variable	Bedeutung
`radius`	Verhältnis von Poikreisradius zu Handkreisradius
`freq1`	Frequenz des Handkreises (ganzzahlig)
`freq2`	Frequenz des Poikreises (ganzzahlig)
`offset`	Drehung der *Flower*
`steps`	Anzahl der berechneten Punkte auf der Kurve

Funktion	Ausgabe
`greatestCommonDivisor()`	größter gemeinsamer Teiler der Frequenzen (s.u.)
`xHand(t)`	x-Koordinate der Hand im Schritt t
`yHand(t)`	y-Koordinate der Hand im Schritt t
`xFlower(t)`	x-Koordinate des *Poi*-Kopfs im Schritt t
`yFlower(t)`	y-Koordinate des *Poi*-Kopfs im Schritt t
`calcHand()`	Liste der Koordinaten, die von der Hand durchlaufen werden
`calcFlower()`	Liste der Koordinaten, die vom *Poi*-Kopf durchlaufen werden

Entscheidungen:

- Wie implizit im Lastenheft festgelegt rechnen wir mit dem Radius des Handkreises als Längeneinheit.

- Aus mathematischen Gründen haben wir uns dafür entschieden, mit zwei ganzzahligen Frequenzen für Hand- und Poikreis zu rechnen, statt wie im Lastenheft vorgesehen mit einem reellen Parameter ω, der das Verhältnis beider Frequenzen darstellt (siehe dort `/LFMF3/`). Auf diese Weise können wir sicherstellen, dass wir immer periodische Kurven erhalten. Da für die Kurven von Hand- und *Poi*-Bewegung nur das Verhältnis dieser Frequenzen von Bedeutung ist, dividieren wir an jeder Stelle, wo diese Frequenzen verwendet werden, durch deren größten gemeinsamen Teiler. Dies betrifft die Funktionen `xHand`, `yHand`, `xFlower` und `yFlower`. Dadurch können wir sicherstellen, dass in der vorgegebenen Anzahl Schritte die *Flower* genau einmal durchlaufen wird.

4.3 Live-Modus (Arbeitspaket 4)

In der Klasse `FlowerCanvasView` werden Methoden definiert, die die statische graphische Darstellung einer parametrisierten Kurve auf einem Canvas ermöglichen. Diese Funktionalitäten bilden den Grundstein für eine Animation von Poi- und Handkurve.
`FlowerCanvasView` erbt von *SurfaceView*. Statt des Android Runtime-Systems kann ein separater Thread innerhalb der Anwendung die `onDraw`-Methode einer *SurfaceView* aufrufen. Dadurch lassen sich später die Zeitpunkte für die schrittweise graphische Anzeige der Animationen kontrollieren.
Die Funktion `pointsToPath` übersetzt eine Liste von Koordinaten in ein Path-Objekt. Dieses kann dann mit den klasseneigenen Methoden `scale` und `translate` flexibel auf die Größe des Canvas angepasst werden, ohne dass eigens komplexe Funktionen geschrieben werden müssen, die diese Berechnungen durchführen. Path stellt zudem nicht nur für Animationen eine Reihe von Instrumenten zur Verfügung, sondern bietet sich auch als mögliche Datenstruktur für das Abspeichern graphischer Objekte an.

4.4 Sonstiges (Arbeitspaket 6)

Die Dateistruktur einer Android-App erlaubt es bestimmte Werte global zu definieren, um an ihnen leichter Anpassungen vornehmen zu können.

Dateien	Inhalt
`colors.xml`	speichert alle in der App benutzten Farben als Hex-Werte
`styles.xml`	definiert Themes(derzeit DarkTheme) und das Erscheinungsbild von InputBoxes und TextBoxes
`strings.xml`	setzt Strings für alle Textelemente des UI

Diese Dateien sind im Ordner **res/values** zu finden.

5 Datenmodell

Momentan werden von Methoden innerhalb der App weder Daten ausgelesen, noch abgespeichert. Im Zuge der Implementierung der Bibliothek, die es ermöglichen wird, *Flowers* und berechnete Handkurven zu laden und zu speichern, wird dieser Punkt jedoch relevant werden.

6 Glossar

Proof-Of-Concept Proof-Of-Concept oder Proof-Of-Principle ist ein Begriff aus dem Projektmanagement. Er ist ein Beleg dafür, dass ein Vorhaben prinzipiell realisierbar ist. Die Kriterien dafür können

in technischen oder betriebswirtschaftlichen Faktoren liegen. In der Regel ist mit dem Proof-of-Concept meist die Entwicklung eines Prototyps verbunden, der die benötigte Kernfunktionalität aufweist.

Poi Ein Poi besteht konzeptuell aus einer Kugel, an der sich eine Schnur mit einem Griff oder einer Schlaufe am anderen Ende befindet, von wo aus die Kugel im Kreis geschwungen werden kann. Durch zusätzliche Bewegungen der Hand kann eine Vielzahl an unterschiedlichen kreisähnlichen Bahnen erzeugt werden.

Flower Eine Flower ist eine spezielle parametrisierte Kurve, die durch eine Gleichung der Form

$$x(t) = cos(2\pi f_1 t) + r * cos(2\pi(f_2 t + \phi)), \quad y(t) = sin(2\pi f_1 t) + r * sin(2\pi(f_2 t + \phi))$$

dargestellt werden kann. Dabei sind r, f_1, f_2 und ϕ frei wählbare Parameter.

Ein *Poi*-Spieler kann solch eine Figur erzeugen indem er die Hand, welche den *Poi* hält, in einer zusätzlichen Kreisbahn bewegt. Dabei entstehen in Abhängigkeit von Rotationsrichtung und Geschwindigkeit unterschiedliche Muster. Sie werden Flowers genannt, da viele dieser Figuren schlaufenförmige Blütenblätter, sog. 'Petals', aufweisen.

API-Level Der API-Level gibt den Entwicklungsstand der eingebauten Funktionen (API, Application Programming Interface) an, die durch einen Entwickler, bspw. bei der Entwicklung von Apps, verwendet werden können. Die Angabe ist besonders dann entscheidend, wenn es um die Kompatibilität einer App mit einer auf einem Gerät installierten Android-Version geht. Verlangt die App einen höheren API-Level, als die Android-Version implementiert, bspw. weil spezifische Funktionen der Android-Version verwendet werden, so kann die App nicht installiert werden.

Kotlin Kotlin ist eine statische, typisierte Programmiersprache, welche sich in JavaScript-Quellcode transformieren und in Form von Bytecode für die JVM (Java Virtual Machine) übersetzen lassen lässt. Die wichtigsten Ziele bei der Entwicklung waren eine hohe Kompilier-Geschwindigkeit und möglichst wenig Code. Kotlin lässt sich außerdem zur Entwicklung von Android-Apps verwenden und wird dafür seit 2017 offiziell von Google unterstützt. Seit Mai 2019 ist Kotlin die von Google bevorzugte Programmiersprache für Android App-Entwicklung.