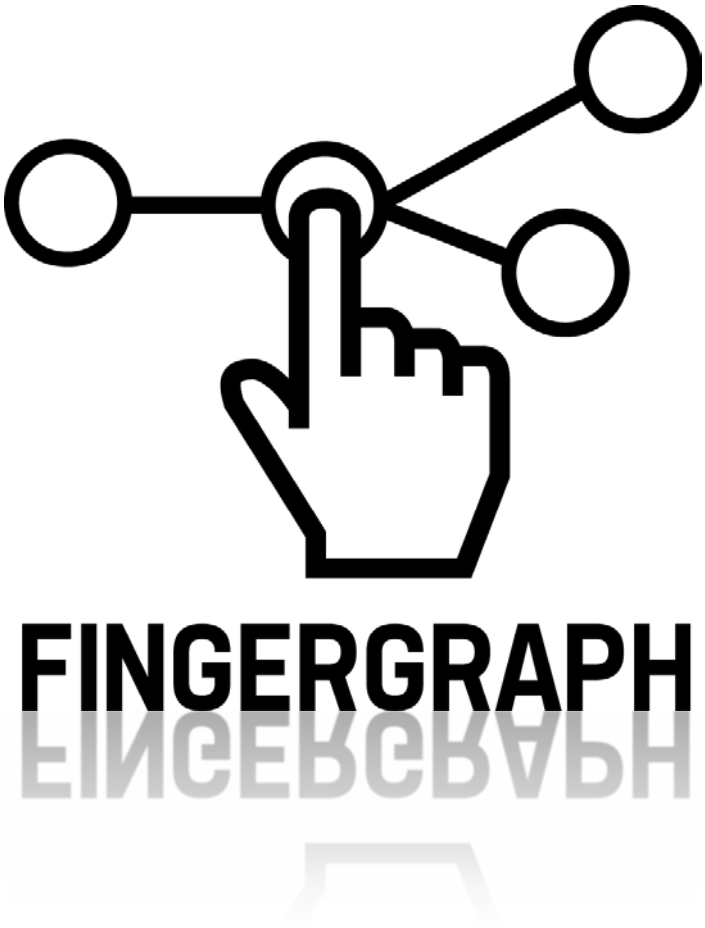
Master Visual Computing | Softwareprojekt

Multi-Touch Graph-Editor

Technische Dokumentation



Benjamin Schwarze
Matrikel-Nr.: 210207972

INHALT

1. EXPOSÉ

Ziel des Softwareprojektes war es, einen einfachen Grapheneditor für ein Multi-Touch-fähiges Eingabegerät unter Android zu realisieren. Der Editor sollte Funktionen zum Erstellen, Manipulieren und Attribuieren von Knoten und Kanten bereit stellen. Dabei lag das Hauptaugenmerk darauf, die Möglichkeiten gestenbasierter Interaktion für die nötigen Funktionalitäten zu nutzen und dadurch den typischen Arbeitsfluss einer mit den Fingern bedien baren Anwendung aufkommen zu lassen.

2. TECHNOLOGIEN

Programmiersprache:	**Java 1.6**
Programmier-Pattern:	**Model-View-Controller** (MVC)
Entwicklungsumgebung:	**Eclipse** Juno
Software development kit:	**Android SDK** Android SDK Tools (Rev. 20.0.1) Android SDK Platform-tools (Rev. 14) Android 4.0.3 (API 15) http://developer.android.com/sdk/index.html
Hardware:	**Sanei N10 Tablet** (10,1") Auflösung: 1920 x 1080 px Android 4.0.3 (Icecream Sandwich)

3. AUFGABEN UND ANFORDERUNGEN

A1.1: Knoten erstellen
Als Grundlage für das Kreieren von Graphen, soll es simpel möglich sein Knoten an beliebiger Stelle der Zeichenfläche zu erstellen.
→ Da die Fingerspitze stets einen Teil der Zeichenfläche verdeckt, wäre ein beispielsweise haptisches Feedback beim erfolgreichen Erstellen sinnvoll.

A1.2: Knoten löschen
Für das Entfernen von bestehenden Knoten sollte eine Bewegung gefunden werden, die zu der Metapher „Wegwerfen" passt und eindeutig ist.
→ Alle mit dem Knoten verbundenen Kanten sollten dabei ebenfalls entfernt werden.

A1.3: Knoten bewegen
Das Bewegen bestehender Knoten wird häufig eingesetzt werden und sollte somit auch kostengünstig durchführbar sein.

A1.4: Knoten selektieren
Um den Arbeitsaufwand beim Editieren zu minimieren, soll es möglich sein, mehrere Knoten gleichzeitig zu verschieben oder mit Attributen zu versehen.
→ Es muss eine Möglichkeit bestehen, die Auswahl wieder aufzuheben.

A1.5: Knoten mit Attributen versehen
Um Knoten unterscheidbar zu machen und ggf. mit einer Bedeutung zu versehen, sollte es möglich sein, diese unterschiedlich einzufärben und/oder zu benennen.

A2.1: Kanten erstellen
Das Erstellen neuer Kanten sollte unkompliziert und in direktem Bezug auf die beiden beteiligten Knoten möglich sein.

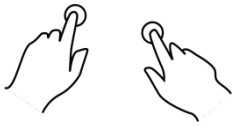
A2.2: Kanten löschen
Das Löschen von Kanten sollte durch eine eindeutige Bewegung geschehen.
→ Beteiligte Knoten bleiben dabei bestehen.

A3.1: Zeichenfläche bewegen
Da die Zeichenfläche endlos sein soll, sollte dem Benutzer eine Möglichkeit eingeräumt werden, den sichtbaren Ausschnitt zu verschieben.

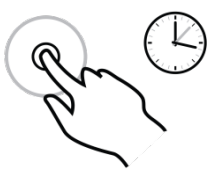
4. GESTEN

Gesten sind die Grundlage aller Interaktionen und dem Aufruf der damit verbundenen Funktionen des Editors. Im Folgenden werden alle verwendeten Gesten beschrieben.

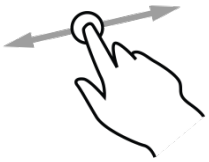
G1: Antippen (tap down)
Das Antippen erfolgt mit einem Finger durch kurzes gezieltes Berühren der Oberfläche.



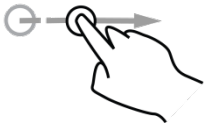
G2: Antippen mit zwei Fingern (two finger tap down)
Antippen der Oberfläche mit zwei Fingern einer oder beider Hände.



G3: Langes Drücken (long press)
Ein langes Drücken unterscheidet sich vom normalen Antippen dadurch, dass der Finger etwa 1,5 Sekunden auf der Oberfläche verweilen muss.



G4: Wischbewegung (scroll)
Für eine Wischbewegung wird der Finger aufgesetzt und anschließend bewegt, ohne die Oberfläche zu verlassen.



G5: Schleuderbewegung (fling)
Für eine Schleuderbewegung führt man ein schnelles Wischen aus, wobei der Finger am Ende der Geste die Oberfläche verlässt.

Illustrations provided by GestureWorks® (www.gestureworks.com)

5. REALISIERUNG DES EDITORS

Der Funktionsumfang des Editors erstreckt sich über das Erstellen, Manipulieren und Attribuieren von Knoten und Kanten. Entsprechend der Aufgaben und Anforderungen, wurden passende Gesten ausgewählt und implementiert. Die Funktionen und dafür jeweils auszuführenden Interaktionen werden im Nachfolgenden genauer erklärt.

Aufgabe	Geste	Beschreibung
A1.1	G3	Für das Anlegen eines neuen Knotens wurde die Geste „long press“ implementiert. Es genügt somit ein langer Druck auf einen freien Punkt der Zeichenfläche. Dies ermöglicht ein gezieltes und bewusstes Erstellen neuer Knoten. Der Erfolg wird durch haptisches Feedback in Form von kurzer Vibration bestätigt.
A1.2	G5	Das Löschen eines Knotens geschieht durch eine „fling“-Geste. Dazu führt man eine schnelle Wischbewegung aus und hebt den Finger von der Oberfläche. Es wirkt, als würde der zu löschende Knoten weggestoßen werden, was somit die „Wegwerfen“-Metapher trifft. Dabei werden auch alle mit dem Knoten verbundenen Kanten entfernt. Für das Löschen von Knoten wurden auch andere Gesten getestet und später verworfen: Das Herausschieben aus dem sichtbaren Zeichenbereich oder das Verschieben auf eine Art Papierkorb. Erstere war ungeeignet, weil man Knoten löschte, die eigentlich nur an eine neue Position verschoben werden sollten. Die Zweite überdeckte Teile der Zeichenfläche und sorgte für lange Wischbewegungen. Weiterhin wurde anfangs ein Löschen-Symbol in das Attribut-Menü integriert, was jedoch gegen den Grundsatz Gesten einzusetzen verstieß.
A1.3	G4	Einzelne Knoten lassen sich durch einfaches gedrückt halten auf der Zeichenfläche verschieben. Dabei wurde die „scroll“-Geste implementiert, die ein Wischen mit dem Finger bemerkt. Dies erfolgt direkt und ohne Verzögerung und entspricht somit der Anforderung kostengünstig zu sein.
A1.4	G1	Durch Antippen ist es möglich, mehrere Knoten zu markieren und gemeinsam

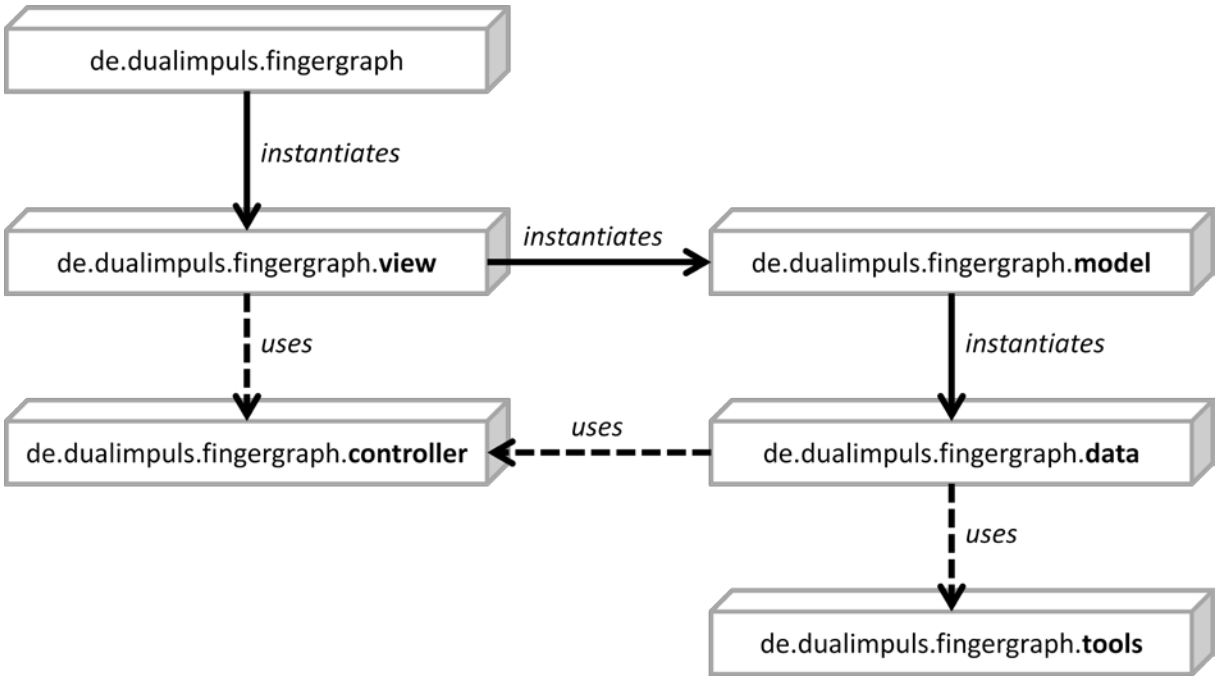
		zu bewegen oder mit Attributen zu versehen. Selektierte Knoten werden dabei durch einen zusätzlichen Ring gekennzeichnet. Es kam auch die Idee auf, die Markierung durch das Aufziehen eines Polygons zu realisieren. Dies durchbrach jedoch den Arbeitsfluss mit den Fingern, bei dem man einfach auf bearbeitende Knoten zeigen möchte und stand in Konflikt mit dem Verschieben der Zeichenfläche. Die Markierung lässt sich durch Antippen eines freien Punktes wieder aufheben.
A1.5	G1	Sobald einer oder mehrere Knoten ausgewählt, bzw. markiert wurden, erscheint das Attribut-Menü. Durch Antippen einer der Farbkreise, werden alle betreffenden Knoten neu koloriert. Auch alle anschließend neu erstellten Knoten, erhalten automatisch die zuletzt gewählte Farbe. Weiterhin können angewählte Knoten durch Anwählen des T-Icons mit einem Label versehen werden. Dafür öffnet sich ein Dialog und die Android-Tastatur. Das Menü wird stets zwischen allen betreffenden Knoten zentriert, um einen Bezug herzustellen.
A2.1	G2	Das Erstellen oder Löschen einer Kante, geschieht indem die beiden betreffenden Knoten jeweils mit einem Finger berührt werden. Dabei ist es auch möglich, auf einem Knoten zu verweilen, um mehrere Kanten neu zuzuordnen bzw. zu löschen. Eine Beschreibung der Implementierung dieser Geste findet sich unter Punkt 6.3. Diese Art und Weise unterstützt den Arbeitsfluss mit den Fingern und stellt unkompliziert eine Beziehung zwischen zwei Knoten her. Zuvor war auch das Durchstreichen von Kanten angedacht, um diese zu löschen. Dies hätte jedoch vorausgesetzt, dass es einen Moduswechsel zwischen Zeichenfläche verschieben und dieser Löschbewegung geben müsste.
A2.2	G2	
A3.1	G4	Durch eine Wischbewegung an einer weißen Stelle, wird die gesamte Zeichenfläche verschoben.

6. PROJEKTSTRUKTUR UND DETAILS

Die Projektstruktur wurde gemäß dem Model-View-Controller Entwurfsmuster gestaltet, um eine klare Trennung von Daten- und Ausgabeschicht zu gewährleisten.

6.1 PACKAGES

Die folgende Darstellung zeigt die im Projekt verwendeten Packages und deren Abhängigkeiten.



de.dualimpuls.fingergraph
Enthält die Hauptklasse zur Initialisierung der Android-Anwendung.

de.dualimpuls.fingergraph.view
Enthält den View zur Realisierung eines Vollbildfensters, der Hauptschleife und des Zeichnens auf der bereitgestellten Oberfläche.

de.dualimpuls.fingergraph.model
Enthält die Businesslogik zur Datenverwaltung.

de.dualimpuls.fingergraph.data
Enthält alle benötigten Datenklassen.

de.dualimpuls.fingergraph.controller
Enthält Klassen zur Gestenverarbeitung und Nutzereingabe.

de.dualimpuls.fingergraph.tools
Enthält Hilfsklassen, wie beispielsweise Singleton-Handler.

6.2 KLASSEN UND METHODEN

Für eine detaillierte und kommentierte Übersicht aller Klassen und der jeweils implementierten Methoden, findet sich im Verzeichnis */doc* eine JavaDoc des Projekts.

6.3 VERARBEITUNG VON GESTEN

Die Klasse `DrawView` implementiert einen `OnTouchListener` der dafür sorgt, dass die Methode `onTouch` auf alle Berührungen der Oberfläche reagiert.

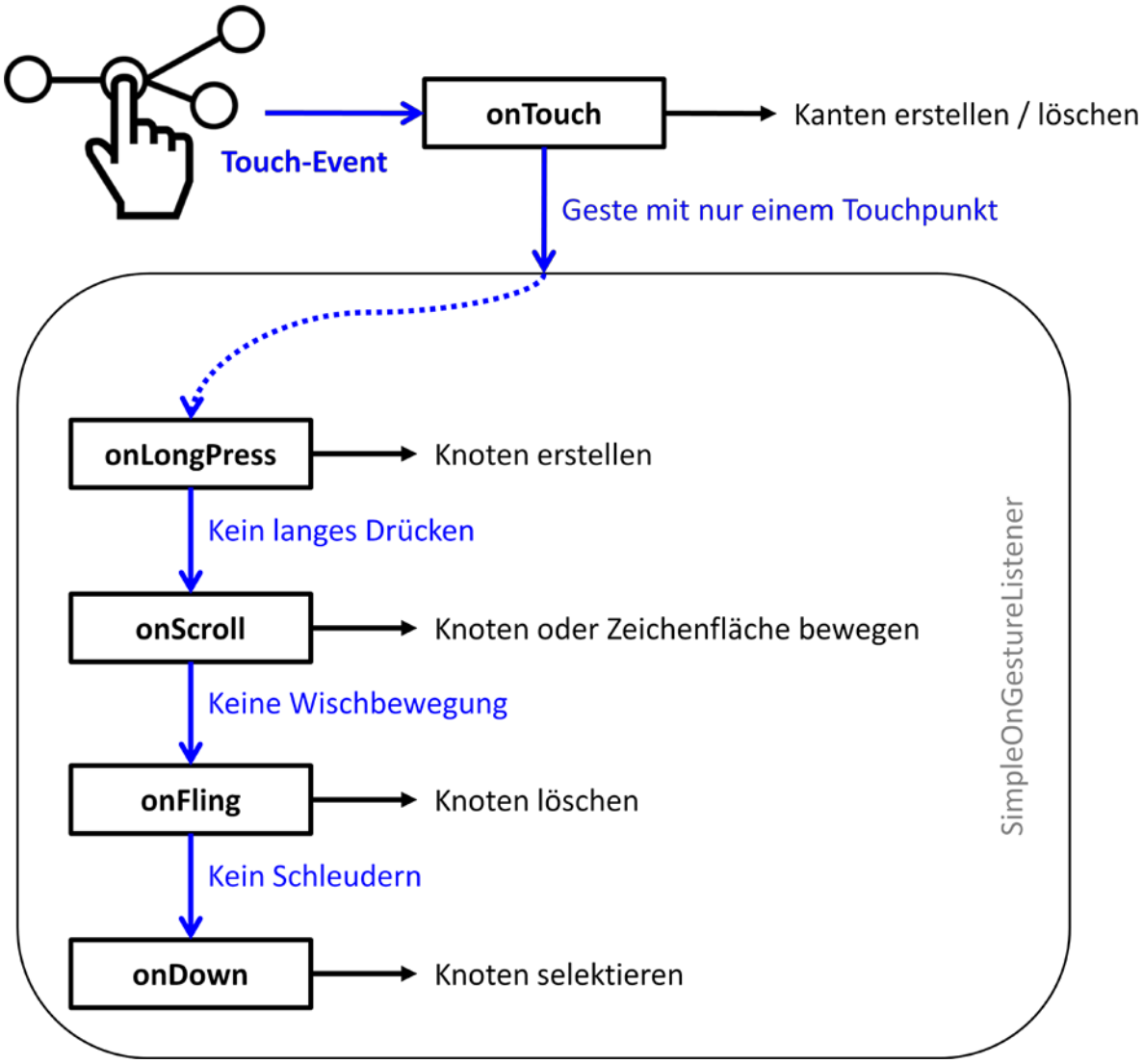
```
public boolean onTouch(View v, MotionEvent e) {
	Node node1, node2;

	if (gestureDetector.onTouchEvent(e)) {
		return true;
	}
	else {
		if (e.getPointerCount() == 2) {
			if(!edgeEvent) {
				node1 = model.touchedNode(e.getX(0), e.getY(0));
				node2 = model.touchedNode(e.getX(1), e.getY(1));
				if (node1 != null && node2 != null) {
					model.toggleEdge(node1, node2);
					edgeEvent = true;
					return true;
				}
			}
		}
		else {
			edgeEvent = false;
		}
		return false;
	}
}
```

Zuerst wird der ausgelöste Touch-Event an den `GestureDetector` weitergeleitet, bei dem es sich um die Ableitung des `SimpleOnGestureListeners` handelt. Liefert dieser ein positives Feedback, gilt die Berührung als erkannt und verarbeitet.
Die Geste G2 gehört nicht zum Standardrepertoire des verwendeten Listeners in Android und wurde deshalb selbst implementiert. Sie dient zur Manipulation von Kanten und soll auslösen, sobald eine gleichzeitige Berührung zweier Knoten erkannt wird. Aufgrund dessen erfolgt als Erstes eine Überprüfung des `PointerCounts`, der angibt, wie viele Berührungspunkte simultan auf der Oberfläche gemessen wurden. Handelt es sich um zwei Punkte, wird mit Hilfe der Methode `touchedNode` getestet, ob sich Knoten an diesen Stellen befinden. Ist dies der Fall, erstellt die Methode `toggleEdge` eine neue Kante oder löscht eine bestehende. Abschließend wird die Geste entprellt und der Touch-Event als verarbeitet markiert.

6.4 VERARBEITUNGSFLUSS

Neben der Implementierung einer eigenen Geste für das Erstellen von Kanten, erfolgt die Verarbeitung der gängigsten Gesten mit Hilfe einer Ableitung der vom Android-SDK bereitgestellten Klasse `SimpleOnGestureListener`. Diese bietet Methoden für einzelnes Tippen, langes Drücken, Wisch- und Schleuderbewegungen. Bei der Verarbeitung eines Touch-Events gibt jede der Methoden an, ob eine erkannte Geste verarbeitet wurde oder nicht. Andernfalls wird das Ereignis weitergegeben. Die folgende Abbildung zeigt den Fluss eines Touch-Events durch die Anwendung. Eine ausführliche Beschreibung der einzelnen Funktionen findet man unter Punkt 5.



7. FAZIT UND AUSBLICK

Alle geplanten Funktionen des Multi-Touch Graph-Editor zum Erstellen, Manipulieren und Attribuieren von Knoten und Kanten wurden realisiert und mit Gesten in den Interaktionsfluss integriert. Die Anwendung bietet eine gute Grundlage für Erweiterungen in der Zukunft. Diese könnten u.a. weitere Attribute wie z.B. Form der Knoten, Funktionen zum Laden und Speichern von Graphen oder eine Minimap zur besseren Übersicht sein.