CMPT 381 Assignment 4: Ink, Selection, Translation, and Scaling

Due: Friday, March 10, 11:55pm

Overview

In this assignment, you will continue to practice your skills in Android touch interaction, and will build a system that supports interaction with ink-stroke-based input. Your system will support ink smoothing, stroke selection, and interactive scaling. You will also build on your experience with Model-View-Controller, and will work with the extended version of MVC that involves an InteractionModel class.

Part 1: Smoothed Ink Input

Build a simple sketching tool in Android that allows the user to draw "ink" strokes on a canvas using touch. The raw ink input will be smoothed, turning the points of the raw input into a Path object comprised of several quadratic curves.

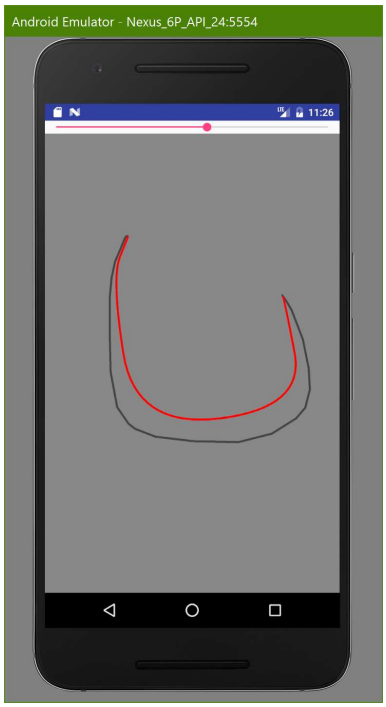
Interface requirements:

- A custom drawing panel that fills the main part of the screen (no scrolling required); this is where the user draws ink strokes
- A horizontal slider that appears above the drawing panel, to control the degree of smoothing (see below)

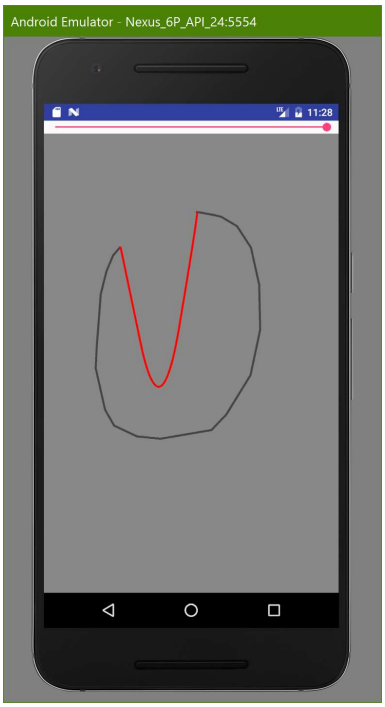
Interaction requirements:

- Touching and dragging on the main panel draws a grey line that follows the touch (i.e., the raw ink)
- When the touch lifts off, the ink is converted to a smooth stroke and is shown in red. The grey ink is still shown until the user starts a new stroke
- The slider ranges from 1-10 and controls the degree of "thinning" that occurs when the raw ink is converted to a smooth stroke. If the thinning value is 1, then every raw ink point is used in the smooth stroke; if the thinning value is 10, then every tenth point is used in the smooth stroke.

Smoothed stroke with thinning = 1



Smoothed stroke with thinning = 5



Smoothed stroke with thinning = 10

Software requirements:

- Implement the system using Model-View-Controller, with correct separation between these components
- Create separate classes for the Model, the View, and the Controller, following the examples given in class

- Implement publish-subscribe communication to notify Views of changes to the Model; other communication between the MVC components can be through direct links (i.e., instance variables)
- Your controller should implement a state machine to determine what actions to take at what times.
- Create a separate class for the interaction model, to store the raw ink and implement the smoothing capabilities
- Build the system using the following classes:
    - SketcherMainActivity: entry point and setup for the app
    - SketchModel: data model that stores all smoothed paths
    - SketchPath: class to represent one path in the model
    - SketchView: custom view to display both the raw ink and the smoothed paths in the model
    - SketchListener: interface to enable publish-subscribe communication between model and view
    - SketchViewController: controller class to handle touch events
    - InteractionModel: class to store the raw ink input and perform smoothing
- The system should work correctly on an emulated Nexus 5X, at Android API level 25

How to smooth raw ink input:

- The basic idea behind smoothing an ink stroke is to replace the lines connecting each input point with curves. We will use quadratic curves (which are easier to implement than cubic curves). There are two parts to the process: first, *thinning* to reduce the density of points in the ink stroke, and *smoothing* to create curves from the remaining points.
- *Thinning*: Given a list of raw input points **rawPoints** and an integer **thinning**, create a new list **thinnedPoints** that contains the first and last points of **rawPoints** plus each equally-spaced point indicated by **thinning**. For example, if **thinning** is 1, take every point from **rawPoints**; if **thinning** is 2, take every other point; if **thinning** is 10, take every tenth point, etc.
- *Smoothing*. From list **thinnedPoints**, create a new Path **smoothed** with the following constraints:
    - start at the first point in **thinnedPoints** (using the **Path.moveTo** method)
    - for each pair of points **p1** and **p2** in **thinnedPoints** (e.g., points 1 and 2, points 2 and 3, etc.):
        - assign **midX** to be the average of **p1.x** and **p2.x**, and assign **midY** to be the average of **p1.y** and **p2.y**
        - add a new quadratic curve to **smoothed** (using the **Path.quadTo** method) with arguments **p1.x**, **p1.y**, **midX**, **midY**
- The new path **smoothed** can then be added to the model.

Resources for part 1:

- Tutorial for custom Views in Android: https://developer.android.com/training/custom-views/index.html
- Tutorial for Android layout: https://developer.android.com/guide/topics/ui/declaring-layout.html
- API for Android SeekBar (slider widget): https://developer.android.com/reference/android/widget/SeekBar.html
- API for Android Path class: https://developer.android.com/reference/android/graphics/Path.html
- Simple sketchpad example from lab: in Examples folder on the course Moodle

Part 2: Selection, Translation, and Scaling

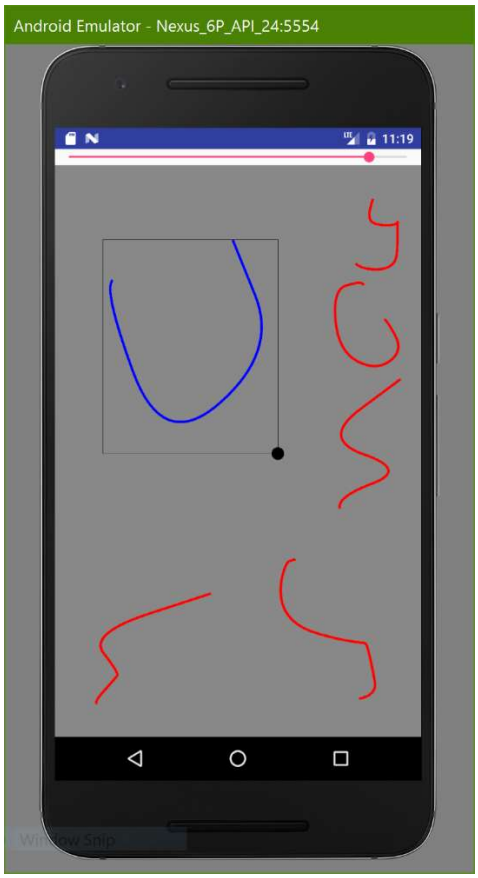
In the second part of the assignment you will extend your system from part 1 to provide the user with additional capabilities: selection of individual strokes, and the interactive translation and scaling of a selected stroke. The additional requirements are:

Additional interaction requirements:

- The capabilities described above (for part 1) all remain in the system
- When the user touches on an existing stroke, the stroke should be highlighted (drawn in blue instead of red) and should show a bounding box and resize handle (see picture at right).
- If the user drags the touch after selecting a stroke, the stroke translates according to the drag
- If the user touches on the background and releases without moving, then any selection is canceled (and no new ink is drawn)
- If the user drags the resize handle of a selected stroke, the stroke will resize according to the drag (see below)

Additional software requirements:

- The selection, bounding box, resize handle, and calculation of the resizing should all be managed by the InteractionModel class
- When resizing, you should create an Android Matrix object that can be passed in as an argument to Path.transform(…) method. Note that this method takes a reference to a path and changes that object, rather than returning a new Path.
- Resizing the Path object that you are also interacting with can often cause weird behaviour. To avoid this problem, you should create a copy of the Path when the resize begins, and calculate all resizing operations relative to this copied Path (i.e., relative to the original size of the stroke).
- Your controller should extend the state machine created for part 1 to handle the additional interactions.



This assignment is to be completed individually; each student will hand in an assignment.

What to hand in

- Android: a zip file of your Android Studio project folder for **either** part 1 (if that is as much as you have completed) **or** part 2. If you have completed part 2, you do not have to hand in a project file for part 1.
- A readme.txt file that indicates exactly what the marker needs to do to run your code. (Systems for 381 should never require the marker to install external libraries).

Where to hand in

Hand in your two files (one zip and one readme.txt) to the link on the course Moodle.

Evaluation

Marks will be given for producing a system that meets the requirements above, and compiles and runs without errors. Note that no late assignments will be allowed, and no extensions will be given, without medical reasons.