CMPT 381 Assignment 3: Model-View-Controller

Due: Friday, February 15, 11:59pm

Overview

In this assignment, you will demonstrate your skills in developing applications that use the Model-View-Controller pattern, and that handle touch interaction and immediate-mode graphics using the Android platform. You will develop a simple graph editor with multiple views, where the user can interactively add to and manipulate vertices and edges in the graph.

Part 1: A Basic Android Graph Editor

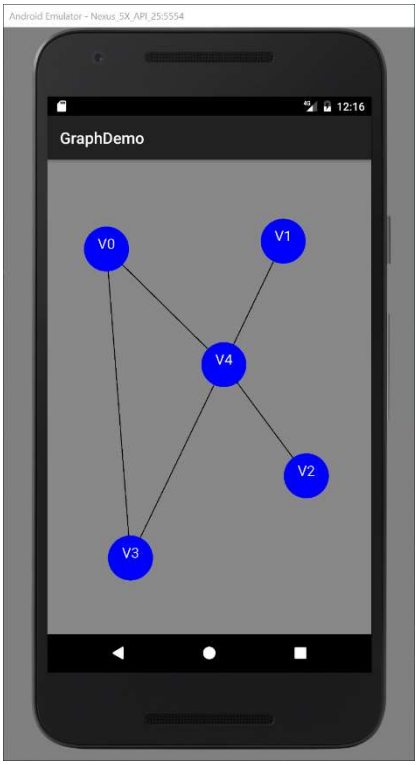
Build a simple GUI in Android that allows the user to create and manipulate a graph.

Interface requirements:

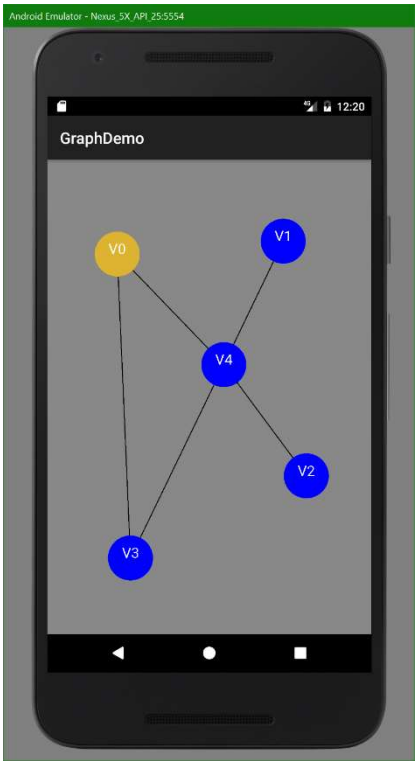
- A main panel that fills the screen (but does not scroll)
- The main panel is the area where the user will interact with the graph (see below)
- The main panel shows all graph vertices, labels, and edges

Interaction requirements:

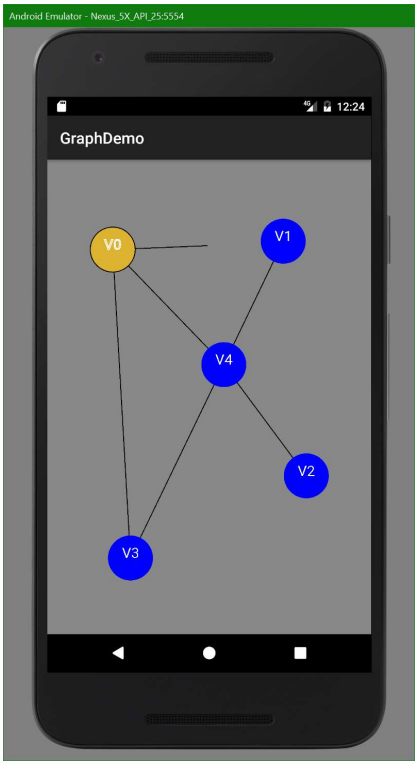
- Touching and releasing on the background of the main panel creates a new vertex, drawn as a blue circle with an integer label (V0, V1, etc.)
- Touching down on an existing vertex selects that vertex (shown by drawing the vertex in orange); dragging then moves the vertex (including any associated edges). Releasing the touch returns the vertex to blue
- If the user long-presses on a vertex, the orange circle gets a black border, and when the user drags their finger, a temporary edge is drawn to their finger location. If the user releases the touch on another vertex, the edge is added to the graph (if the user releases on the background, the edge is discarded).



Main graph view



User touches down on V0



User long-presses on V0 then moves

Software requirements:

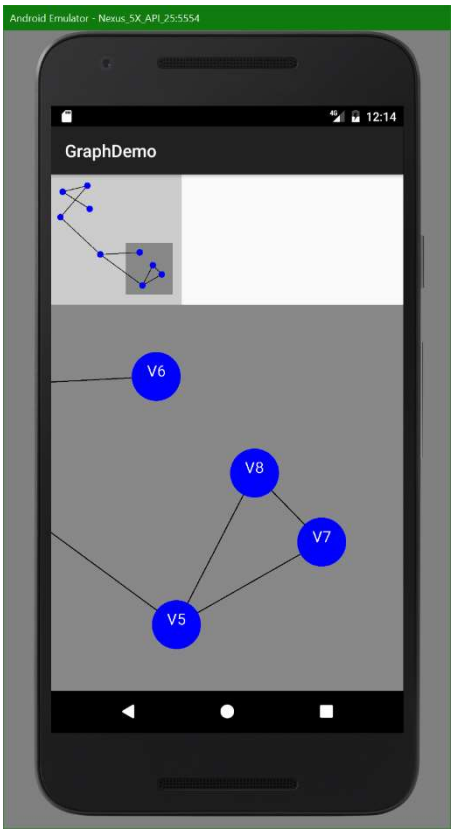
- You must implement the system using Model-View-Controller, with correct separation between these components

- Create separate classes for the Model, the View, and the Controller, following the examples given in class
- Create a class for the InteractionModel to store the selection and the coordinates of the temporary edge
- Implement publish-subscribe communication to notify Views of changes to the Model and to the InteractionModel
- Build the system using the following classes:
    - MainActivity
    - GraphModel
    - Vertex
    - Edge
    - InteractionModel
    - MainGraphView
    - MainGraphViewController

Resources for part 1:

- Custom Views in Android: https://developer.android.com/training/custom-views/index.html
- Custom drawing: https://developer.android.com/training/custom-views/custom-drawing
- Attaching events to custom views: https://developer.android.com/training/custom-views/making-interactive

Part 2: Two Views and View Navigation



In the second part of the assignment you will extend your system from part 1 to add another view (a mini view), and to enlarge the main view so that view navigation (i.e., swiping to move the viewport) is possible.

Additional interface requirements:

- The main panel is now 3000x3000 pixels in size, meaning that it is not all shown at once on the screen.
- There is now a second view (the mini view) at the top of the screen. The mini view shows a miniature version of the entire 3000x3000 main view. The mini view shows all vertices and edges, selection colour changes, and the temporary edge, but does not show vertex labels.
- The mini view can be any square size (specified when adding the view to the layout)
- The mini view also shows a grey viewfinder rectangle that indicates the extents of the main view within its 3000x3000 area. The viewfinder moves as the user swipes to change the location of the main view (see below).
- The white space to the right of the mini view is not used.

Additional interaction requirements:

- Touching and dragging on the main view's background now moves the main view to a new location in the 3000x3000 area.
- The main view's location and size within the 3000x3000 area is stored in the InteractionModel
- All manipulations of vertices and edges are the same as described above.

Additional software requirements:

- The mini view should use the existing publish-subscribe mechanism to receive notifications about model changes; in addition, the mini view will need to communicate with the InteractionModel in order to determine the extents of the viewport.
- Add one class for the new view: MiniGraphView
- When adding the mini view to the screen layout, the programmer can set any square size for the view using the LayoutParams argument to the layout's addView() method. For example:
    - vertical.addView(miniView,new LinearLayout.LayoutParams(400,400));
- Your controller must implement a state machine (as introduced in class) to determine what actions to take at what times (e.g., whether a move implies a swipe of the viewport, a move of a vertex, or drawing a temporary edge).

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.