CMPT 381 Assignment 3: Model-View-Controller

Due: Friday, February 16, 11:55pm

Overview

In this assignment, you will demonstrate your skills in developing applications that use the Model-View-Controller pattern. You will develop a simple graph editor with multiple views, where the user can interactively add to and manipulate objects from the graph. You will also provide basic view manipulation (panning around the visual workspace).

Part 1: A Basic JavaFX Graph Editor

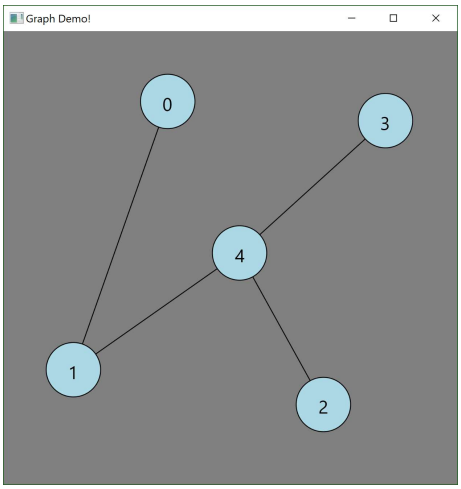
Build a simple GUI in JavaFX that allows the user to create and manipulate a graph.

Interface requirements:

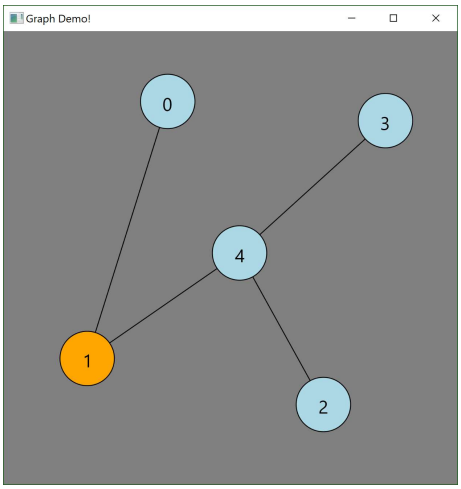
- A main panel with size 500x500; this is the area where the user will interact with the graph (see below)
- The main panel shows all graph vertices, labels, and edges

Interaction requirements:

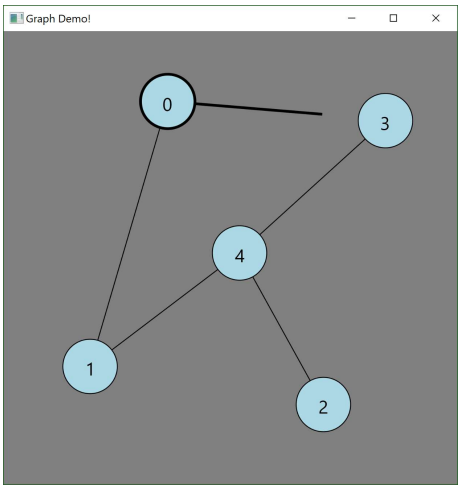
- Clicking on the background of the main panel creates a new vertex, drawn as a blue circle with an integer label
- Clicking down on an existing vertex selects that vertex (shown by drawing the vertex in orange); dragging then moves the vertex. Releasing the mouse returns the vertex to blue
- If the user Shift-Clicks a vertex, the vertex gets a thicker black border, and then when the user drags, an edge is drawn to the mouse location. If the user releases the mouse 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 clicks on vertex 1



User shift-clicks on vertex 0 then drags

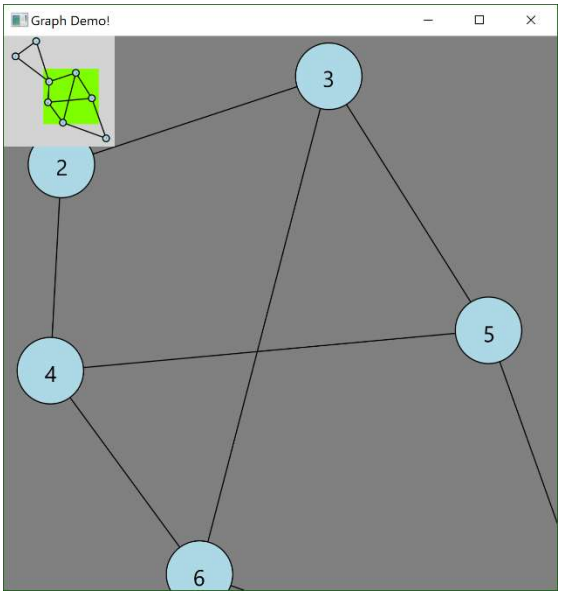
Software requirements:

- 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
- 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)
- Build the system using the following classes:
  - GraphDemo — JavaFX application class
  - GraphModel — Class to store the model
  - Vertex — Represents a vertex in the graph
  - Edge — Represents an edge in the graph
  - GraphView — A custom view for drawing the graph
  - GraphViewController — Controller for the GraphView

Resources for part 1:

- Blob MVC example from class: in Examples folder on the course Moodle

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 add interaction capabilities to both views to support navigation (panning the viewport).

Additional interface requirements:

- The main panel now has a *logical* size as well as a widget size, meaning that the main panel can now be a viewport onto a larger area than its own size
- For example, the workspace could be 2000x2000 pixels in size, even if the main panel is still only 500x500 pixels. The panel would show a 500x500 viewport onto the larger workspace.
- The logical and widget sizes of the main panel are passed into the constructor of the custom view (the main panel is always square, both logically and physically).
- There is now a second view (the mini view) at the top left of the UI. The mini view shows a miniature version of the *entire* workspace (e.g., the whole 2000x2000 workspace in the example above, but in miniature). The mini view shows all vertices and edges (including the selected vertex) and the temporary edge, but does not need to show vertex labels.
- The mini view can be any square size (the size is passed in to the constructor)
- The mini view also shows a green viewfinder rectangle that indicates the extents of the main view within the entire logical workspace. The viewfinder moves as the user navigates to change the location of the main view's viewport (see below).

Additional interaction requirements:

- Clicking and dragging on the main view's background now moves the main view to a new location in the logical workspace.
- Clicking and dragging on the green viewfinder in the mini view also moves the main view to a new location (and updates the viewfinder rectangle)
- All manipulations of vertices and edges are the same as described above.

Additional software requirements:

- Add a class for the new view: MiniView
- The mini view can be displayed overtop the main view by putting both custom views into a StackPane, and setting their alignment within the StackPane to be "top left". In addition, you will need to indicate that mouse events should be sent to the visible parts of the lower pane in the stack (i.e., the main view):
    - StackPane.setAlignment(mainView, Pos.TOP_LEFT);
    - StackPane.setAlignment(miniView, Pos.TOP_LEFT);
    - miniView.setPickOnBounds(false);
- Add a class for the mini view's controller: MiniViewController
- The mini view should use the existing publish-subscribe mechanism to receive notifications about model changes
- Add class InteractionModel to store the selection, the viewport location and extents, and the coordinates of the temporary edge
- The InteractionModel class will use a second publish-subscribe mechanism to notify views about changes
- Your controllers must implement state machines (as introduced in class) to determine what actions to take at what times (e.g., whether a drag action implies panning the viewport, moving 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

- A zip file of your **NetBeans** 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.