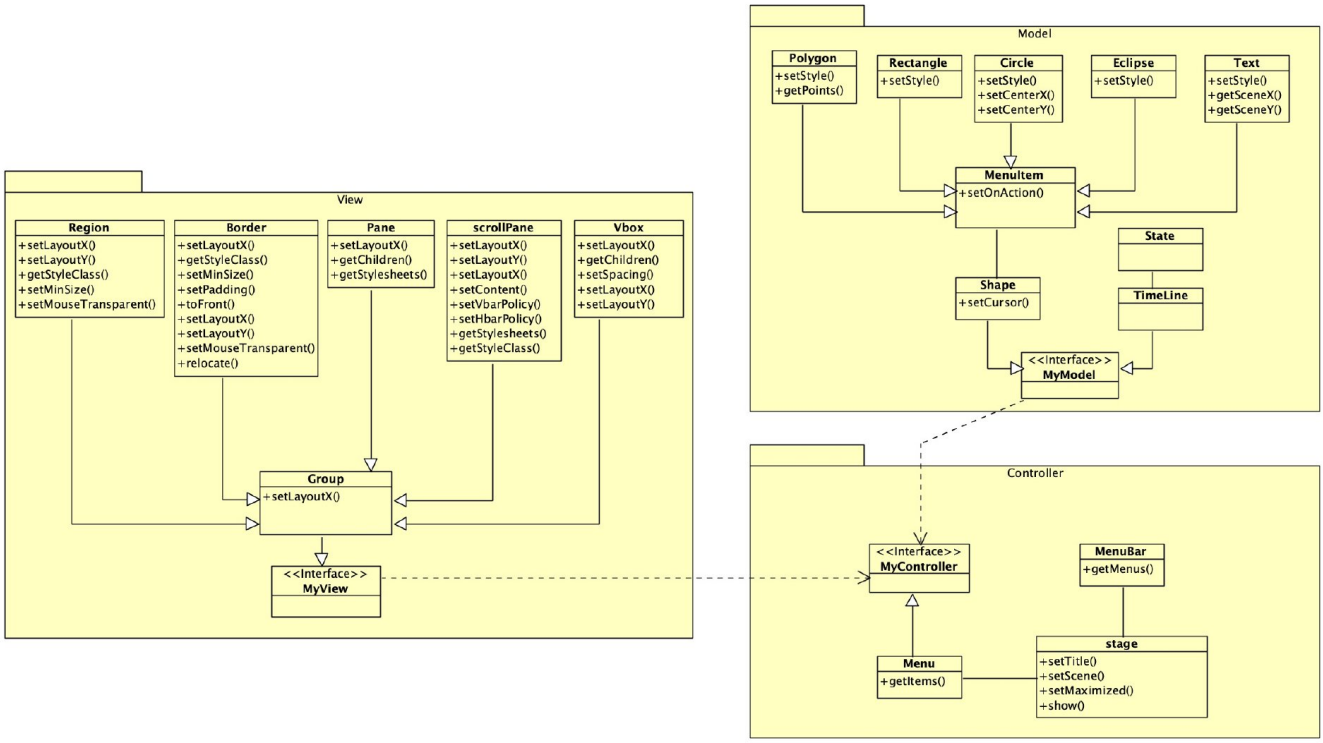
ENGI 9874 Software Specification and Design Design document group Inida

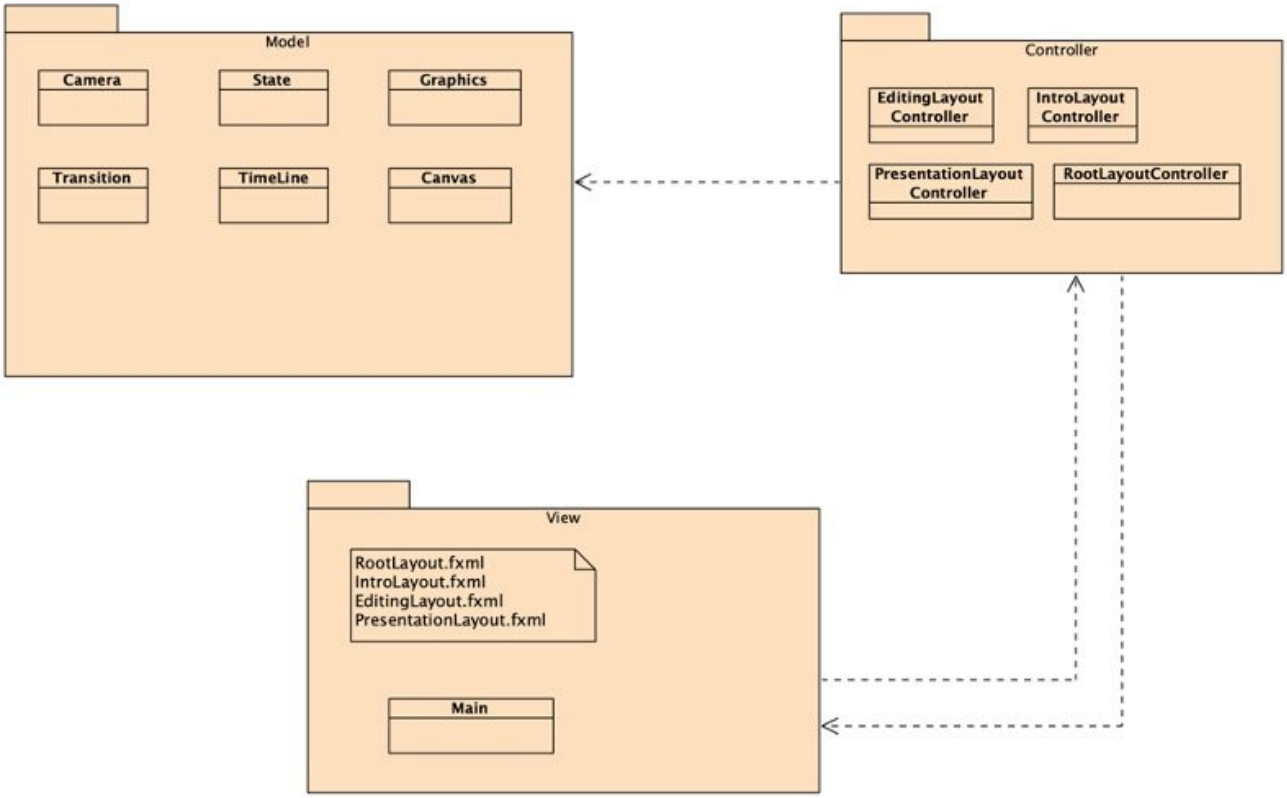
group member: Tong Zhao, Zirui Luo, Varun Yadav, Sachin Yadav, Deep Patel

class diagram



Packages

Package Diagram



Thrid party Packages summary

Following are the packages that we are using for developing the software. All these packages are part of JavaFX 1 libraries.

1. javafx.animation.FadeTransition
    - This package is useful for crating fade in and fadeout transition of the shape.
2. javafx.animation.FillTransition
    - This package is useful for changing colour of the shape.
3. javafx.animation.KeyFrame
    - Defines target values at a specified point in time for a set variable that are interpolated along a timeline.
4. javafx.animation.KeyValue
    - Defines a key value to be interpolated for a particular interval along the animation.
5. javafx.animation.ParallelTransition
    - For creating parallel transition of multiple animation.
6. javafx.animation.ScaleTransition
    - For creating zoom in and zoom out animation for the object.
7. javafx.animation.SequentialTransition
    - For crating multiple animation transitions in sequential transition.
8. javafx.animation.Timeline

- For creating a timeline, which will include sequential and parallel transitions perform.

9. javafx.event.ActionEvent
    - An event representing some type of action. This event type is widely used to represent a variety of things, such as when a button has been fired, when a keyframe has finished, and other such usages.
10. javafx.event.EventHandler
    - This is a functional interface and can therefore be used as the assignment target for a lambda expression or method reference.
11. javafx.geometry.Insets
    - A set of inside offsets for the 4 side of a rectangular area.
12. javafx.geometry.Orientation
    - Provides the set of 2D classes for defining and performing operations on objects related to two-dimensional geometry.
13. javafx.stage.Stage
    - Provides the top-level container classes for JavaFX content.
14. javafx.util.Duration
    - A class that defines a duration of time.
15. javafx.scene.Cursor
    - For including different aspects of the cursor.
16. javafx.scene.Group
    - To group multiple scene, pane, region and scroll pane into one module. A Group node contains an Observable List of children that are rendered in order whenever this node is rendered. A Group will take on the collective bounds of its children and is not directly resizable. Any transform, effect, or state applied to a Group will be applied to all children of that group.
17. javafx.scene.PerspectiveCamera
    - To include a perspective camera.
18. javafx.scene.Scene
    - The javafx Scene class is the container for all content in a scene graph.
19. javafx.scene.shape.*
    - This package includes multiple types of shapes like rectangle, line, circle and ellipse. Ans it also provides definitions of common properties for those objects.
20. javafx.scene.control.Button
    - A simple button control. The button control can contain text and/or a graphic.
21. javafx.scene.control.ContextMenu
    - For creating context menu for the shape. Which will be activated on right click on the shape.
22. javafx.scene.control.Menu
    - For creating the menu for the application.
23. javafx.scene.control.MenuBar

- For crating a menu bar, which will hold all menu.

24. javafx.scene.control.MenuItem
    - MenuItem is intended to be used in conjunction with menu to provide options to users. Menuitem serves as the base class for the bulk of JavaFX menus API.
25. javafx.scene.control.RadioButton
    - For including radio button and its operation. So the user can choose one items from the list.
26. javafx.scene.control.ScrollBar
    - Either a horizontal or vertical bar with increment and decrement buttons and a "thumb" with which the user can interact. Typically not used alone but used for building up more complicated controls such as the ScrollPane and ListView.
27. javafx.scene.control.ScrollPane
    - For including scroll pane in the program. Scroll pane will hold the canvas so that user can scroll the canvas as its required.
28. javafx.scene.control.SeparatorMenuItem
    - To separate other menu items inside a menu.
29. javafx.scene.control.TextArea
    - To creating a text area which will hold the text and try to edit it.
30. javafx.scene.control.ToggleGroup
    - Toggle group is useful for crating a group for radio buttons. So that user can choose only one item from this group.
31. javafx.scene.control.ScrollPane.ScrollBarPolicy
    - To include different scroll bar policies to include in software.
32. javafx.scene.layout.GridPane
    - GridPane lays out its children within a flexible grid of rows and columns. If a border and/or padding is set, then its content will be layed out within those insets.
33. javafx.scene.layout.Pane
    - Base class for layout panes which need to expose the children list as public so that users of the subclass can freely add/remove children.
34. javafx.scene.layout.Region
    - Region is the base class for all JavaFX Node-based UI Controls, and all layout containers. It is a resizable Parent node which can be styled from CSS. It can have multiple backgrounds and borders. It is designed to support as much of the CSS3 specification for backgrounds and borders as is relevant to JavaFX.
35. javafx.scene.layout.StackPane
    - StackPane lays out its children in a back-to-front stack. The z-order of the children is defined by the order of the children list with the 0th child being the bottom and last child on top. If a border and/or padding have been set, the children will be laid out within those insets.

36. javafx.scene.layout.VBox
    - VBox lays out its children in a single vertical column. If the vbox has a border and/or padding set, then the contents will be laid out within those insets.
37. javafx.scene.paint.Color
    - The Color class is used to encapsulate colors in the default sRGB color space.

Test plan

1. The test cases in the classes in model folder.
    - In the /tstsrc/Prezoom/model/StateTest.java: included `addItemInState()`, `getCurrentNumString()`.
    - In the /tstsrc/Prezoom/model/TimeLineTest.java: included `addStateInTimeLine()`, `deleteStateInTimeLine()`.
2. The test cases in the main class launch test.
    - In the /tstsrc/Prezoom/MainTest.java: included the `main()` to launch the system

Complie and Execute code

To compile and execute the code here are few requirements and steps.

1. Install latest version of Java SE or JDK from the following link: https://www.oracle.com/java/technologies/javase-downloads.html
2. After installing the latest version of Java Developers Kit, we need to install the latest Eclipse IDE for Java package from the link below: https://www.eclipse.org/downloads/packages/release/oxygen/r/eclipse-ide-java-ee-developers
3. After installing Eclipse, we need to install JavaFX plugins by following these steps: “Eclipse IDE → Help → Eclipse Marketplace → type fx → install e(fx)clipse 3.6.0 → click accept terms and press the finish button.”
4. Our next step after following the above is to download and configure the latest version of JavaFX SDK from the following link: https://gluonhq.com/products/javafx/
5. Once you have downloaded the file, you can extract ‘javafx-sdk-15.0.1’ to a folder on your PC (Make sure you know the exact location of it).
6. After extracting, we have to create user libraries in Eclipse by following these steps: “Eclipse IDE → Window → Preferences→ Search & select user libraries under build path → Click new and name JavaFX→Click on Add external JARs→Go to ‘javafx-sdk-15.0.1’ folder and select all the files in lib folder expect src→Click on Apply and Close.”
7. After following the above step, we need to configure our build path and arguments using the following steps: “Go to Run configurations in Eclipse IDE→Click Arguments→In VM arguments type the following ‘—module-path “D:\javafx\javafx-sdk-15.0.1\lib” –add-modules javafx.controls,javafx.fxml’→Click on apply and run.”

8. Now you can create/load your existing project and able to successfully compile it Eclipse IDE.

contribution

1. Tong Zhao
    - Document: Domain Model diagram, package Diagram and Test Plan.
    - Source Code: Test cases and attributes of graphics set up.
2. Zirui Luo
    - Document: Lexicon, and class diagrams.
    - Source Code: Setting up the structure of project, graphics operations and state set up.
3. Varun Yadav Keshaboina
    - Document: Functional, Non-functional requirement and User-Case Stories.
    - Source Code: camera movement and its border layout.
4. Sachin Kumar Yadav
    - Document: Use-Cases Diagrams and State Diagrams.
    - Source Code: layout set up in view and animation set up.
5. Deep Maheshbhai Patel
    - Document: How to Run and Compile Code, and third party packages brif introductions.
    - Source Code: trasition,timeline and presentation mode set up.