WormGUIDES Desktop JavaFX Application

Code README

Purpose:

To provide a high level view of the logic of the package organization.

To detail the functionality of the major elements of the application.

To detail how communication between components works.

**GUI Element Breakdown**

1. Main GUI window

a. **3D window**

*Main Class*: Window3DController.java

*Class Location*: src/wormguides/controllers/

*Functionality*:

i. Manages scene data. Builds scene via RenderService in *buildScene()* line 1684, method triggered by change in BooleanProperty *rebuildSceneFlag*, listener at line 667

- RenderService refreshes scene i.e. clears current scene data with:

- *refreshScene()* line 1850, clears the *rootEntitiesGroup* Group (parent object of which all scene data is a child), removes sprites from the *spritesPane* Pane, and updates the *indicatorRotation* Rotation based on interpolated rotation values and frames

- *getSceneData()* line 1689, stores the cell names, positions and diameters based on time frame, builds the scene element geometry based on time frame, adds labels, notes

- *addEntitiesAndNotes()* line 1875, adds the entities loaded in getSceneData() to the *rootEntitiesGroup*, places labels and notes in the scene

ii. Time value

- *timeProperty* IntegerProperty listener defined in RootLayoutController.java line 780

b. **Below 3D window controls**

b1. Rewind/Play/Fast-forward buttons

i.Rewind button listener defined in Window3DController.java line 2869

ii. F-F button listener defined in Window3DController.java line 2878

b2. Time slider

i. *timeSlider* Slider (JavaFX component) listener defined in RootLayoutController.java line 787

b3. Zoom In/Out buttons

i.Zoom in button listener defined in Window3DController.java line 2845

ii.Zoom out button listener defined in Window3DController.java line 2862

b4. Info section

**-** Controlled by RootLayoutController.java *setSelectedEntityInfo()* line 853, sets the entity name and queries the description either from the active story in the case of a selected structure, or the partslist in the case of a cell lineage named entity

c. **Stories tab**

c1. Stories list view

*Main Class*: StoriesLayer.java

*Class Location*: src/wormguides/layers/

*Functionality:*

i. Manages and maintains Story.java objects (src/wormguides/stories/) with *stories* ObservableList<Story>

ii. Listens for Story change and triggers scene rebuild. Listener line 161

c2. Story Note

*Main Class*: Note.java

*Class Location*: src/wormguides/stories/

*Functionality*:

i. Stores note data and facilitates attachment type, location

ii. Note Visibility button controlled by *visibleProperty* SimpleBooleanProperty

c3. Story editor window (triggered by New Story, Edit Active Story buttons)

*Main Class*: StoryEditorController.java

*Class Location:* src/wormguides/controllers/

*Functionality*:

i. Captures story data via editor window, converts to internal Story object. See class for documentation

d. **Coloring and Display tab**

d1. Find Cells tab

*Main Class*: SearchLayer.java

*Class Location*: src/wormguides/layers/

*Functionality*:

i. Cells search field, controlled by *searchTextField* TextField, listener at line 761, queries the model based on the defined criteria on the SearchLayer. The results of the search are populated in the searchResultsList ObvservableList<String> using the *refreshSearchResultsList()* method line 771, and displayed in the layer’s list view. The scene is also updated to highlight the results of the search

ii. Add rule button *addRuleButton* adds a color rule using the *getAddButtonClickHandler()* method line 716, which calls the *addCallRule()* method line 550

d2. Find Structures tab

*Main Class*: StructuresLayer.java

*Class Location*: src/wormguides/layers/

*Functionality*:

i. Structures search field, controlled by *searchField* TextField, listener at line 99, queries the *sceneElementsList* (primary data structure for complex geometry in scene, located in src/wormguides/models/subscenegeometry) for search hits. The results are populated into the *searchStructuresResultsList* ObservableList<String> in StructuresLayer.

ii. Select Structures list view, underlying model housed in *structuresTreeView* TreeView<StructureTreeNode>, facilitates expansion of tree structure and updates the shared variable *selectedStructureNameProperty* to populate sub 3D window panel with structure information. Listener at line 117

d3. Display Options tab

*Main Classes*: DisplayLayer.java, Rule.java, Window3DController.java

iv. Rules list view is populated by *currentRulesList* ObservableList<Rule> which houses color rules associated with the active story.

a. Rule fields

1. Rule editor window controlled by RuleEditorController.java (src/wormguides/controllers/)

2. Visibility button controlled by *setVisible()* Rule.java line 180

3. Delete button, controlled by listener DisplayLayer.java line 71

2. Lineage Tree window

*Main Class*: SulstonTreePane.java

*Location*: src/wormguides/view/popups/

*Functionality*:

Renders a lineage tree UI via *addLines()* line 511 given the root of the lineage tree and the pane on which to render the tree

b. **Cell Expansion click/Branch Click** controlled by *clickHandler* line 163, expands section of tree, and updates shared *timeProperty* shared variable which triggers scene rebuild

3. Primary Data windows (View → Primary Data dropdown)

a. **Cell Shapes Index**

Created by InfoWindowDOM.java line 78 (src/wormguides/views/infowindow)

b. **Parts List**

Created by InfoWindowDOM.java line 147 (src/wormguides/views/infowindow)

c. **Connectome**

Created by InfoWindowDOM.java line 184 (src/wormguides/views/infowindow)

d. **Cell Deaths**

Created by InfoWindowDOM.java line 244 (src/wormguides/views/infowindow)

e. **Experimental Data**

Created by InfoWindowDOM.java line 283 (src/wormguides/views/infowindow)

4. Cell Info Window (View → Cell Info Window)

*Main Class*: InfoWindow.java, InfoWindowDOM.java

*Location*: src/wormguides/view/infowindow/

*Functionality*:

Creates and maintains “Info” pages for selected entities in the 3D window.

i**.** “More Info” button in the Context Menu (right click Menu) triggers *addName()* in InfoWindow.java line 321 via a listener in RootLayoutController.java line 841

ii. “More Info” clickable text in info panel below 3D window triggers *addName()* in InfoWindow.java line 321 via a listener in RootLayoutController.java line 831

5. Rotation Control window (View → Rotation Control)

*Main Class*: RotationController.java

*Location*: src/wormguides/controllers/

*Functionality*:

i. All sliders and input fields (listed below) in the Rotation Controller are bound to the shared variables rotateXAngleProperty, rotateYAngleProperty, rotateZAngleProperty and update the scene accordingly when changed

a. **X Rotation Slider**

b. **Y Rotation Slider**

c. **Z Rotation Slider**

d. **X Rotation Input Field**

e. **Y Rotation Input Field**

f. **Z Rotation Input Field**

6. About window (Help → About)

*MainClass*: AboutPane.java

*Location*: src/wormguides/view/popups/

*Functionality*:

i. Creates a pane and sets the text to that listed in the class

**Shared Variables**

The following variables are initialized in RootLayoutController.java *initSharedVariables()* line 1239 and control functionality throughout the application’s classes and maintain important properties such as time. They are important to understand the control and flow of data within the application.

1. timeProperty - SimpleIntegerProperty

2. totalNucleiProperty- SimpleIntegerProperty

3. othersOpacityProperty - SimpleDoubleProperty

4. rotateXAngleProperty - SimpleDoubleProperty

5. rotateYAngleProperty - SimpleDoubleProperty

6. rotateZAngleProperty - SimpleDoubleProperty

7. translateXProperty - SimpleDoubleProperty

8. translateYProperty - SimpleDoubleProperty

9. zoomProperty – SimpleDoubleProperty

10. selectedEntityNameProperty - SimpleStringProperty

11. selectedNameLabelProperty - SimpleStringProperty

12. activeStoryProperty - SimpleStringProperty

13. cellClickedFlag - SimpleBooleanProperty

14. geneResultsUpdatedFlag - SimpleBooleanProperty

15. rebuildSceneFlag - SimpleBooleanProperty

16. usingInternalRulesFlag – SimpleBooleanProperty

17. bringUpInfoFlag – SimpleBooleanProperty

18. playingMovieFlag - SimpleBooleanProperty

19. capturingVideoFlag – SimpleBooleanProperty

20. colorHash – ColorHash

21. rootEntitiesGroup – Group

22. subscene – SubScene

23. rulesList - ObservableList

24. searchResultsList - ObservableList

**Package Organization**

The top level of the WormGUIDES directory is divided into the following components:

* *documentation/* contains **code leve**l, **Notes**, **URLs** and **Window3D parameters** documentation
* *lib/* stores external libraries that WormGUIDES leverages for functionality
  + *jmf.jar* – Java Media Framework JAR for taking screenshots and videos
* LICENCE – The GNU General Public License
* *META-INF/* contains the MANIFEST.MF file that IntelliJ uses to create an executable JAR (our suggested IDE for the WormGUIDES developer community)
* *out/* contains compiled source files
* README.md
* *src/* contains the source files, data files, and configuration files for WormGUIDES (described in detail below)
* WormGUIDES.iml– IntelliJ configuration file
* WormGUIDES.jar – the WormGUIDES executable file, of type Java Archive

The *src/* directory’s organizing principle is files that are specific to the WormGUIDES application and those that exist independently that are leveraged by the application. All files internal to the application are stored in the *wormguides/* directory and will be explained below. There are four directories external storing files external to WormGUIDES. They are:

* *acetree/* contains the lineage data, the interface for accessing this data, and the loader and database which implements the interface
  + LineageData.java – the data structure interface for querying the lineage data
  + *nucleifiles/* contains one file per time point during embryogenesis which defines cell positions
  + *tablelineagedata/* 
    - AceTreeTableLineageDataLoader.java – The loader that processes the nuclei files and places them in the TableLineageData
    - TableLineageData.java – Table that keeps the lineage data in **Frame** data structures
* *connectome/* contains the data file and classes for defining and querying the connectome
  + Connectome.java
  + ConnectomeLoader.java
  + NeuronalSynapse.java
  + NeuronConnect.csv
  + SynapseType.java
* *partslist/* contains the data files for the partslist and its accompanying class, with a subdirectory that contains a data file defining cells deaths and its accompanying class
  + *celldeaths/*
    - CellDeaths.csv
    - CellDeaths.java
  + PartsList.java
  + partslist.txt
* *search/* files used for searching the model and querying WormBase
  + SearchType.java
  + SearchUtil.java
  + WormBaseQuery.java

The *wormguides/* directory’s organizing principle follows the architecture of the WormGUIDES application. WormGUIDES is an MVC application (Model, View, Controller) which provides users with a graphical view of the model, and controllers that facilitate model manipulation that trigger view updates. The three main directories within *wormguides/* are:

* *controllers/* contains controllers classes that correspond to graphical controllers defined in view, and other controllers for functionality
  + ContextMenuController.java
  + InfoWindowLinkController.java
  + RootLayoutController.java
  + RotationController.java
  + RuleEditorController.java
  + StoryEditorController.java
  + Window3DController.java
* *models/* defines the models that form the backend of the application, in addition to the external lineage data
  + *analogouscell/*
  + *anatomy/*
  + *camerageometry/*
  + *cellcase/*
  + *colorrule/*
  + LineageTree.java
  + *obj\_files/*
  + *production\_info\_file/*
  + *shapes\_file/*
  + *subscenegeometry/*
* *view/* defines the views that present the underlying model
  + DraggableTab.java
  + *graphicalrepresentations/*
  + *icons/*
  + *infowindow/*
  + *layouts/*
  + *popups/*
  + *urlwindow/*

The other files and directories that define the WormGUIDES application within *wormguides/* are:

* *layers/* defines the functionality of the **Display**, **Search**, **Stories**, and **Structures** tabs on the RootLayout
* *loaders/* contains classes for loading elements for the 3D window and model parameters
  + GeometryLoader.java
  + IconImageLoader.java
  + NoteImageLoader.java
  + ParametersLoader.java
  + ProductionInfoLoader.javas
* MainApp.java
* *resources/*
* *stories/*
* *util/*