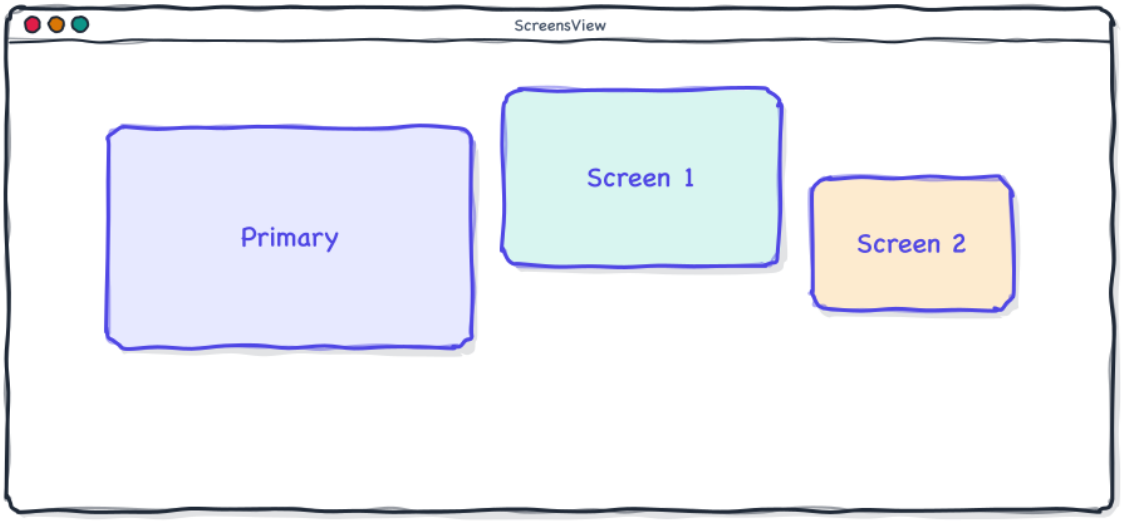
ScreensView

Developer Manual

GemsFX • com.dlsc.gemsfx • JavaFX Custom Controls



Generated cartoon overview of ScreensView.

ScreensView visualizes the current JavaFX Screen list using real screen bounds, optional wallpapers, visual-bounds overlays, live windows and arbitrary debug shapes.

Table of Contents

1. Introduction

ScreensView ScreensView visualizes the current JavaFX Screen list using real screen bounds, optional wallpapers, visual-bounds overlays, live windows and arbitrary debug shapes.

1.1 Key features

- Uses Screen.getScreens() and preserves relative virtual-desktop geometry.
- Default wallpaper is wallpaper.jpg from the control resources.
- showShadow, showReflection, showWallpaper and showWindows are true, true, true and false by default.
- Optional WindowView nodes bind to live Window x, y, width and height.
- enableWindowDragging defaults to true and lets users move live miniature windows.
- ScreensView.show() opens a utility Stage titled from the resource bundle.

1.2 Maven dependency

```
<dependency>
    <groupId>com.dlsc.gemsfx</groupId>
    <artifactId>gemsfx</artifactId>
    <version>4.4.1</version>
</dependency>
```

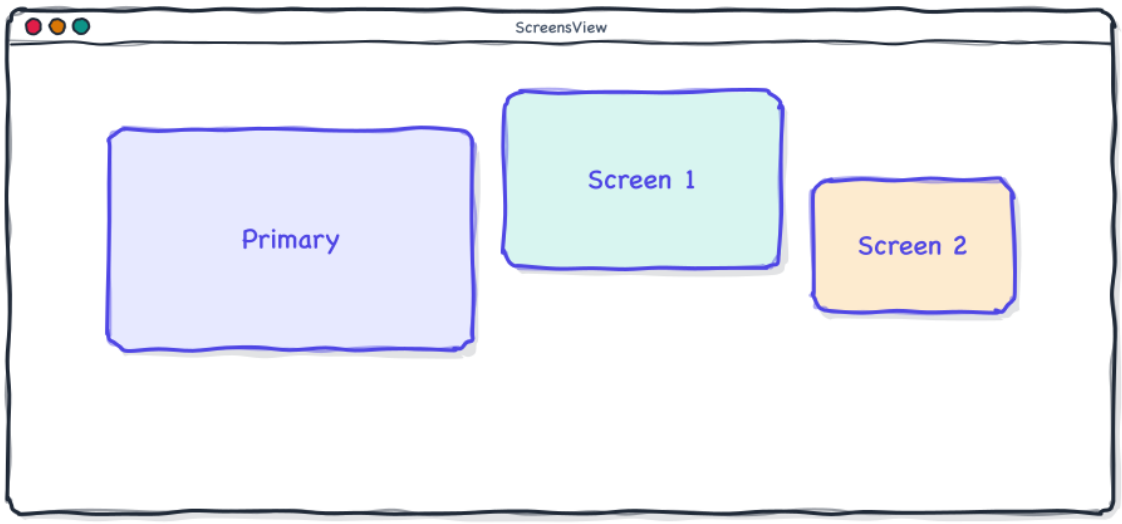
The control lives in module `com.dlsc.gemsfx`.

2. Getting started

The following snippet uses only public API verified in the control source.

```
ScreensView view = new ScreensView();
view.setShowWindows(true);
view.setShowWallpaper(false);
view.setWallpaperProvider(screen -> myWallpaperFor(screen));
view.getShapes().add(debugRectangle);
view.setEnableWindowDragging(true);
```

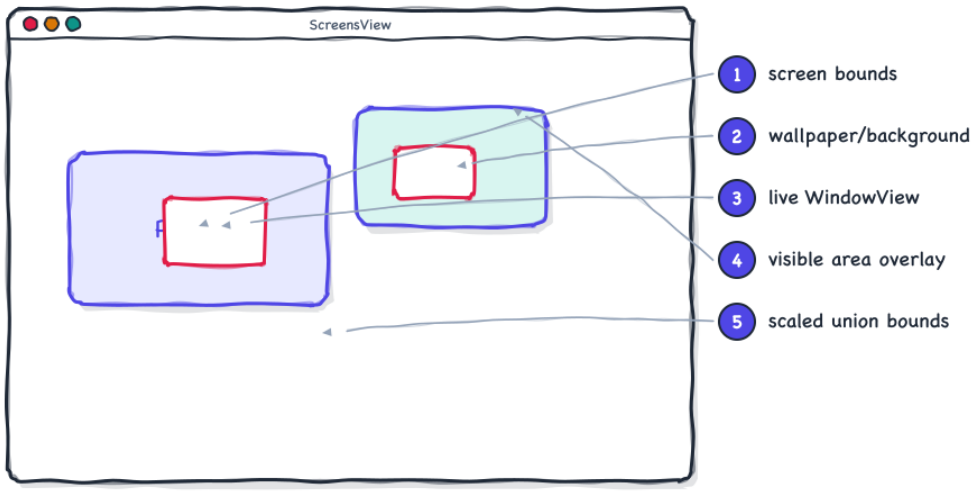
A compact setup for ScreensView.



A generated overview of ScreensView in use.

3. Anatomy

The anatomy diagram identifies the implementation pieces that matter when configuring, styling or debugging the control.



The parts of ScreensView.

Part	Verified detail
Root	Style class `.screens-view` is added by the constructor.
Stylesheet	User-agent stylesheet `screens-view.css` is returned by the control.
Screen model	Skin reads `Screen.getScreens()` and `Window.getWindows()`.

4. Control API

4.1 Properties and callbacks

Property	Type	Default	Description
shadow	ObjectProperty<DropShadow>	DropShadow radius 2, THREE_PASS_BOX	Shadow applied to the screen group when showShadow is true.
reflection	ObjectProperty<Reflection>	fraction .25, topOffset 5	Reflection applied when showReflection is true.
shapes	ObservableList<Shape>	empty list	Extra debug shapes added to the unified bounds.
showShadow	BooleanProperty	true	Enables shadow effect.
showReflection	BooleanProperty	true	Enables reflection effect.
showWallpaper	BooleanProperty	true	Uses wallpaperProvider images.
showWindows	BooleanProperty	false	Adds live WindowView nodes.
wallpaperProvider	ObjectProperty<Callback<Screen, Image>>	screen -> DEFAULT_WALLPAPER	Supplies wallpaper image per screen.
enableWindowDragging	BooleanProperty	true	Allows dragging live miniature windows.

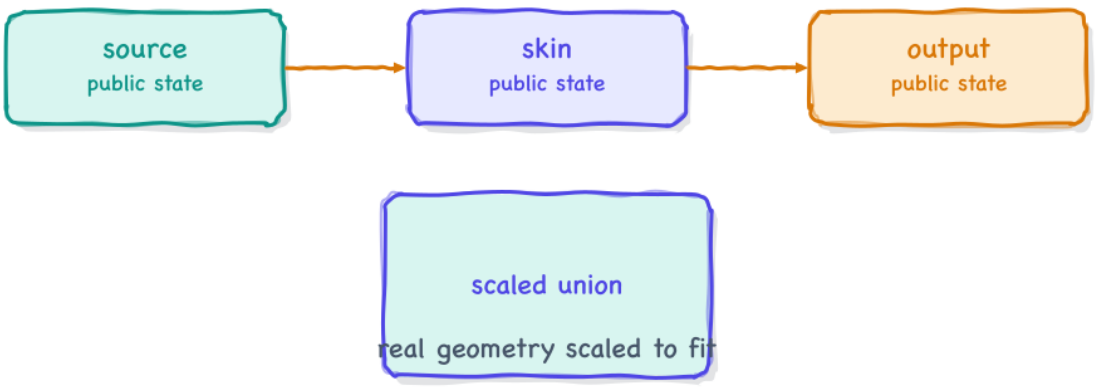
Note. Defaults and property names in this table were checked against the Java source for this batch.

5. Behaviour

Important runtime states of ScreensView.

- The skin computes the union of all Screen bounds plus extra shapes.
- Each screen group contains background, screen label, visible-area overlay and glass overlay.
- The scaling group uses min(width / totalWidth, height / totalHeight) * .75.

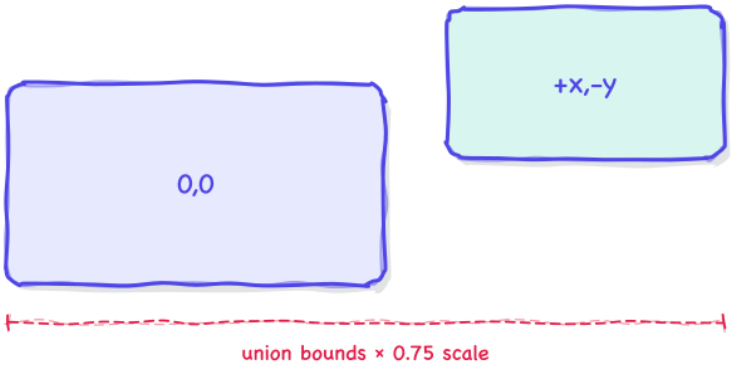
- If showWindows is true, WindowView nodes bind to live windows and stage titles.
- Dragging a WindowView moves the real Window by screen delta divided by the scale.



How data and geometry flow through ScreensView.

6. Layout and rendering

Rendering preserves virtual desktop geometry. Screens, optional shapes and optional windows are placed in real coordinate space, then scaled into the control.

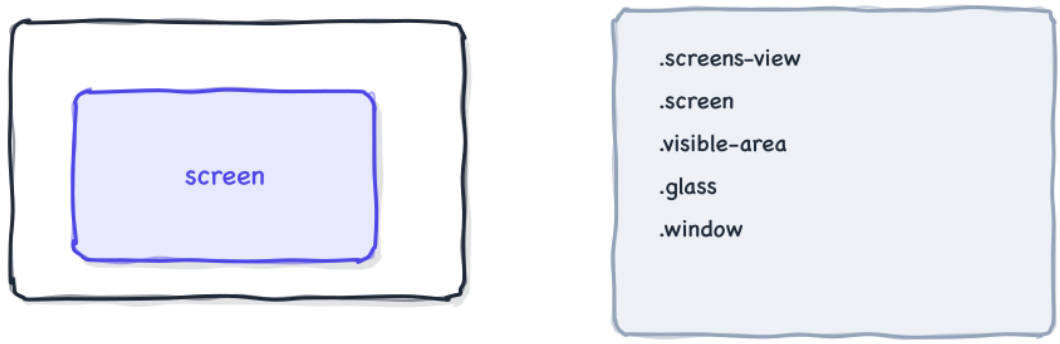


Rendering and sizing rules for ScreensView.

Concern	Rule
Bounds	Union includes all screens and custom shapes.
Scale	min(width / totalWidth, height / totalHeight) * .75.
Windows	WindowView binds layout and size to live Window properties.

7. Styling

The user-agent stylesheet is screens-view.css. The table lists selectors and pseudo classes that exist in source, skin or stylesheet.



Style hooks for ScreensView.

7.1 Style classes and pseudo classes

Selector / pseudo class	Purpose
`.screens-view`	Verified in source, skin or CSS.
`.container`	Verified in source, skin or CSS.
`.screen`	Verified in source, skin or CSS.
`.screen.no-wallpaper`	Verified in source, skin or CSS.
`.screen .label`	Verified in source, skin or CSS.
`.glass`	Verified in source, skin or CSS.
`.visible-area`	Verified in source, skin or CSS.
`.window`	Verified in source, skin or CSS.
`.window .label`	Verified in source, skin or CSS.

7.2 Styleable CSS properties

This control declares no additional styleable CSS properties beyond inherited JavaFX properties.

```
.screens-view .screen.no-wallpaper {
    -fx-background-color: lightblue;
}
.screens-view .window {
    -fx-border-style: dashed;
}
```

CSS example using documented hooks.

8. Localization

Key	English text
`window.title`	Screens
`label.screen-index`	Screen {0}
`label.primary`	Primary
`accessible.role-description`	screens

9. Accessibility

ScreensView sets AccessibleRole.NODE with localized role description "screens". No automatic accessible text for individual screens is bound.

10. Recipes

10.1 Programmatic configuration

```
ScreensView view = new ScreensView();
view.setShowWindows(true);
view.setShowWallpaper(false);
view.setWallpaperProvider(screen -> myWallpaperFor(screen));
view.getShapes().add(debugRectangle);
view.setEnableWindowDragging(true);
```

10.2 Practical checklist

1. Enable showWindows only for debugging or tooling views.
2. Use wallpaperProvider for per-screen backgrounds.
3. Use the public properties listed in the API chapter.
4. Style only through documented selectors and styleable properties.
5. Do not depend on private skin node structure except for documented CSS selectors.

11. Integration notes

ScreensView declares no styleable CSS properties; its options are JavaFX properties only.

Topic	Recommendation
Threading	Keep image loading and expensive rendering off the UI path when the control exposes a background option.
Styling	Scope selectors under the documented root style class.
Accessibility	Preserve the source-defined accessible role and add app-specific text when the control does not bind it.
State	Prefer public properties over skin node lookup.

12. See also

- Demo application: `com.dlsc.gemsfx.demo.ScreensViewApp` (run with `mvn javafx:run -f gemsfx-demo/pom.xml -Dmain.class=com.dlsc.gemsfx.demo.ScreensViewApp`)
- Related GemsFX media controls: `AvatarView`, `PhotoView`, `SVGImageView`, `BeforeAfterView`, `MaskedView`, `ScreensView`.
- API documentation: https://dlsc-software-consulting-gmbh.github.io/GemsFX/api/