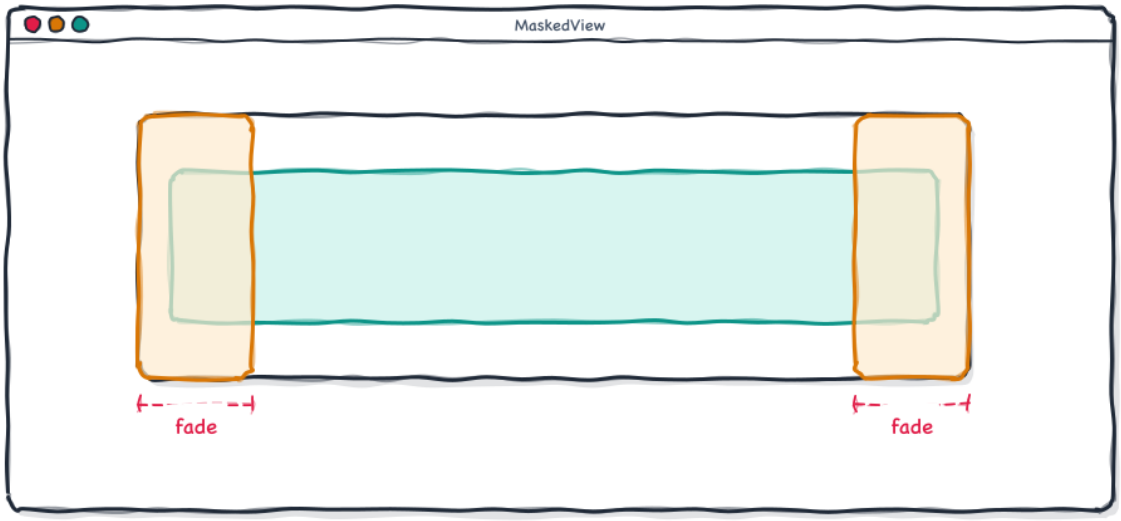
MaskedView

Developer Manual

GemsFX • com.dlsc.gemsfx • JavaFX Custom Controls



Generated cartoon overview of MaskedView.

MaskedView wraps one content node and clips it with left, center and right rectangles. The edge clips become gradients when translated content overflows.

Table of Contents

1. Introduction 2

1.1 Key features 2

1.2 Maven dependency 2

2. Getting started 2

3. Anatomy 3

4. Control API 3

4.1 Properties and callbacks 3

5. Behaviour 4

6. Layout and rendering 5

7. Fade-mask mechanics 5

8. Styling 6

8.1 Style classes and pseudo classes 6

8.2 Styleable CSS properties 6

9. Localization 7

10. Accessibility 7

11. Recipes 7

11.1 Programmatic configuration 7

11.2 Practical checklist 7

12. Integration notes 7

13. See also 8

1. Introduction

MaskedView MaskedView wraps one content node and clips it with left, center and right rectangles. The edge clips become gradients when translated content overflows.

1.1 Key features

- Stores a single content Node.
- Used internally by StripView for side-faded scrolling content.
- fadingSize defaults to 120 and is styleable.
- Left edge fades only when content.translateX is negative.
- Right edge fades only when content.translateX + prefWidth exceeds the view width.
- The content translateX is observed through a WeakInvalidationListener.

1.2 Maven dependency

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

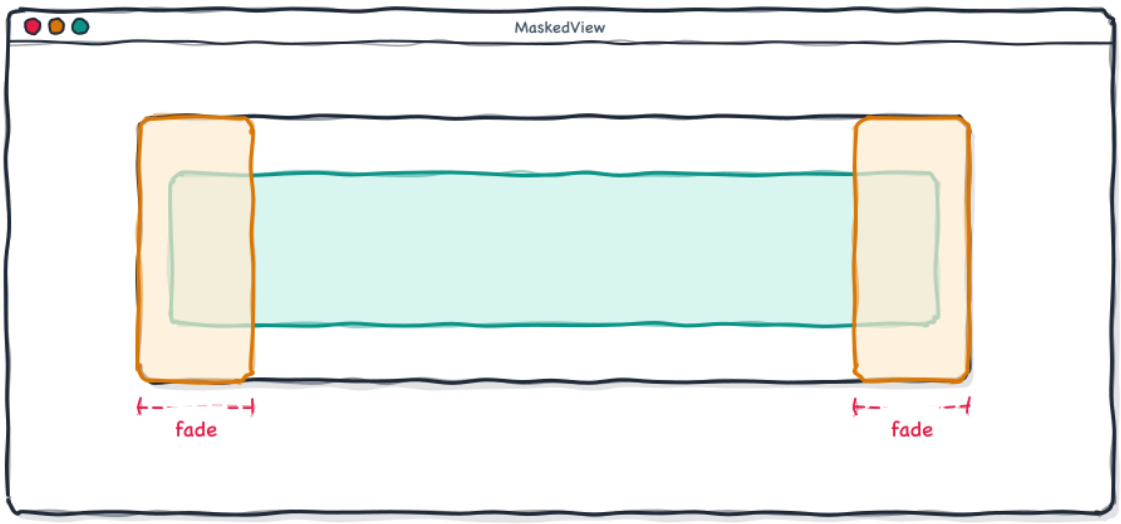
The control lives in module `com.dlsc.gemsfx`.

2. Getting started

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

```
HBox strip = new HBox(cards);
MaskedView masked = new MaskedView(strip);
masked.setFadingSize(80);
strip.setTranslateX(-120);
```

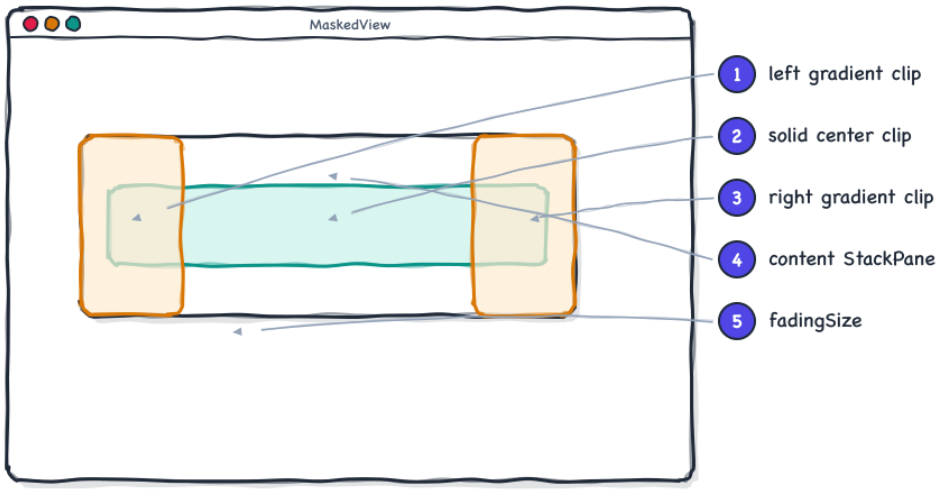
A compact setup for MaskedView.



A generated overview of MaskedView in use.

3. Anatomy

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



The parts of MaskedView.

Part	Verified detail
Root	Style class `.masked-view` is added by the constructor.
Stylesheet	No user-agent stylesheet is returned by the control source.
Clip group	Skin clips content with left, center and right rectangles.

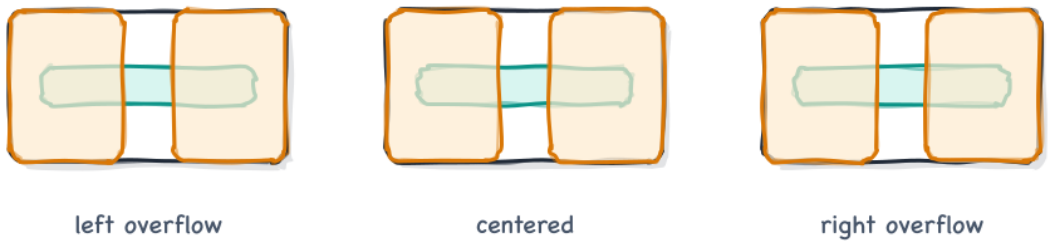
4. Control API

4.1 Properties and callbacks

Property	Type	Default	Description
content	SimpleObjectProperty<Node>	null	Node shown inside the masked view.
fadingSize	DoubleProperty	120	Width of left and right fade areas; styleable.

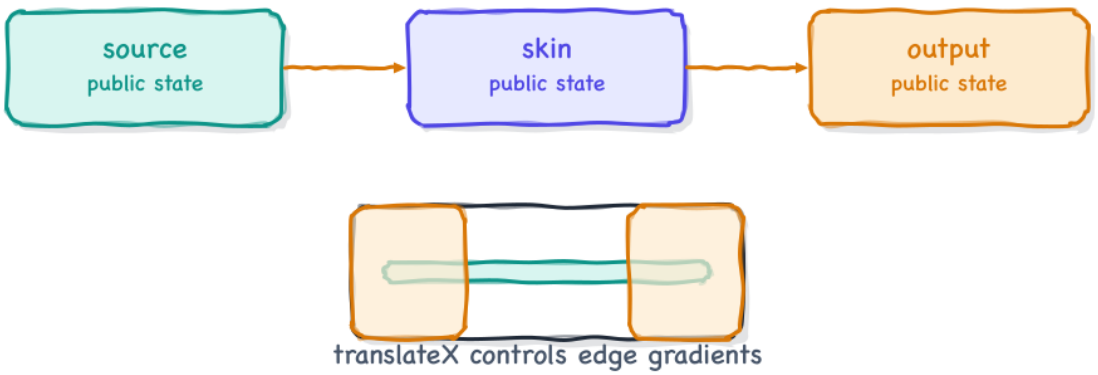
Note. Defaults and property names in this table were checked against the Java source for this batch.

5. Behaviour



Important runtime states of MaskedView.

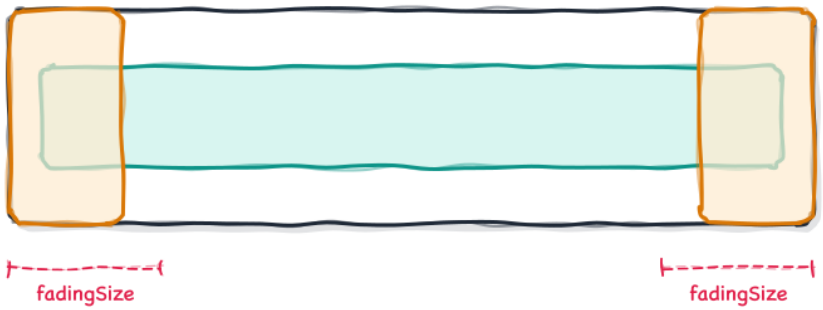
- The skin clips a StackPane with a Group containing left, center and right rectangles.
- The center clip is always solid black.
- The left clip is transparent-to-black when content is shifted left; otherwise solid.
- The right clip is black-to-transparent when content overflows the right edge; otherwise solid.
- layoutChildren caps fadingSize at half the content width.



How data and geometry flow through MaskedView.

6. Layout and rendering

Rendering uses a StackPane clipped by three rectangles. Edge rectangles switch to transparent gradients only when the content overflows that side.



Rendering and sizing rules for MaskedView.

Concern	Rule
Fade width	layoutChildren uses min(contentWidth / 2, fadingSize).
Left fade	Enabled when content.translateX < 0.
Right fade	Enabled when translated content extends past view width.

7. Fade-mask mechanics

MaskedView does not blur pixels and does not paint an overlay. It clips the content with three rectangles. In JavaFX clipping, black means opaque and transparent means clipped away, so the gradients become soft reveal masks.

Content position	Left clip	Right clip
content.translateX < 0	transparent → black gradient	depends on right overflow
content.translateX ≥ 0	solid black	depends on right overflow
translated right edge > view width	depends on left overflow	black → transparent gradient
translated right edge ≤ view width	depends on left overflow	solid black

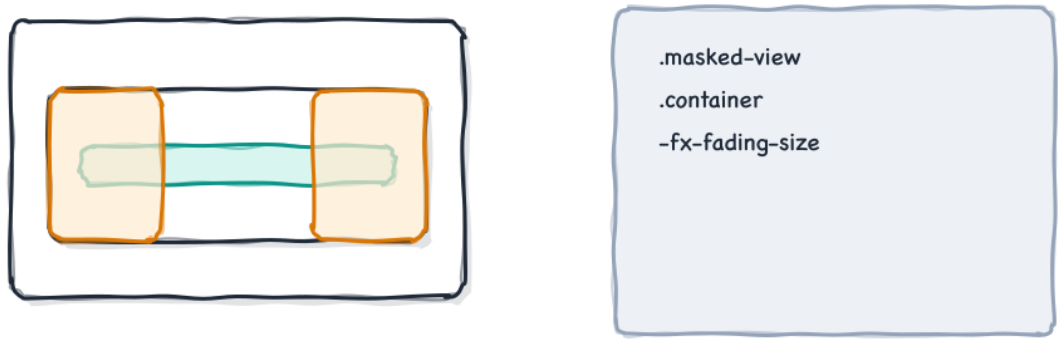
```
double fadingSize = Math.min(contentWidth / 2, getSkinnable().getFadingSize());
leftClip.resizeRelocate(x, y, fadingSize, contentHeight);
centerClip.resizeRelocate(x + fadingSize, y, contentWidth - 2 * fadingSize, contentHeight);
rightClip.resizeRelocate(x + contentWidth - fadingSize, y, fadingSize, contentHeight);
```

The three clip rectangles laid out by the skin.

> **Note.** Only horizontal overflow is handled. The verified skin listens to content.translateX and the control width, not to vertical movement.

8. Styling

MaskedView does not return a user-agent stylesheet in the verified source, but it still exposes a root style class and styleable CSS properties where listed below.



Style hooks for MaskedView.

8.1 Style classes and pseudo classes

Selector / pseudo class	Purpose
`.masked-view`	Verified in source, skin or CSS.
`.container`	Verified in source, skin or CSS.

8.2 Styleable CSS properties

CSS property	Type	Default
`-fx-fading-size`	size	120

```
.masked-view {
    -fx-fading-size: 80px;
}
```

CSS example using documented hooks.

9. Localization

The verified source has no ResourceBundleManager keys for MaskedView.

10. Accessibility

MaskedView sets AccessibleRole.IMAGE_VIEW and focusTraversable false. No accessible text is bound.

11. Recipes

11.1 Programmatic configuration

```
HBox strip = new HBox(cards);
MaskedView masked = new MaskedView(strip);
masked.setFadingSize(80);
strip.setTranslateX(-120);
```

11.2 Practical checklist

1. Move the content with translateX to activate edge fades.
2. Keep fadingSize no larger than half the useful viewport.
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.

12. Integration notes

No user-agent stylesheet and no resource bundle exist for MaskedView.

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.

13. See also

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