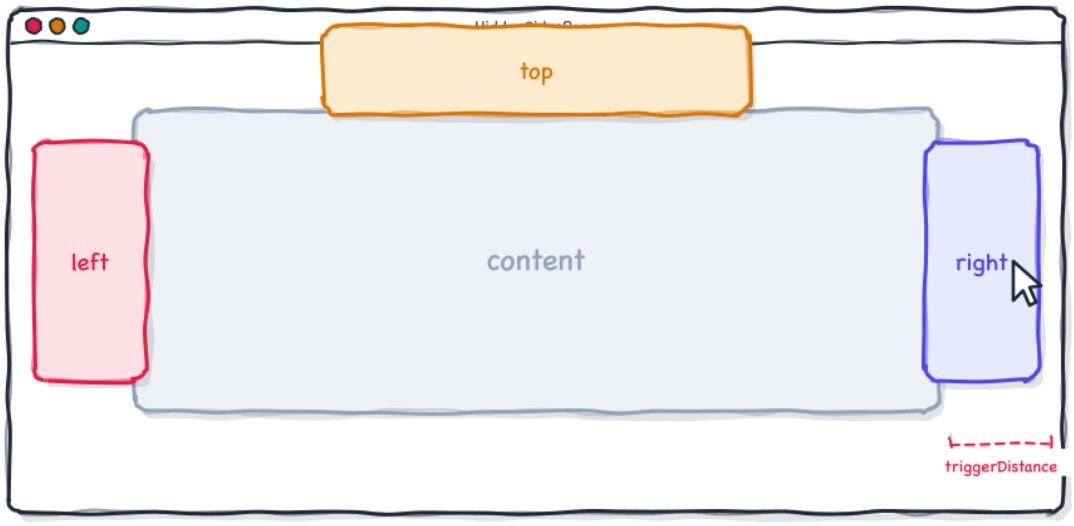
HiddenSidesPane

Developer Manual

GemsFX • com.dlsc.gemsfx • JavaFX Custom Controls



A generated cartoon overview of HiddenSidesPane.

HiddenSidesPane lays out a full-size content node and up to four hidden side nodes that slide in from pane edges on mouse proximity or when pinned.

Table of Contents

1. Introduction

HiddenSidesPane is a `Pane` with one full-size content node and up to four side nodes initially positioned outside the clipped pane bounds. The side nodes slide in when the mouse enters an edge trigger band or when a side is pinned.

1.1 Key features

- Content fills the complete pane and drives min / pref / max size.
- Top, right, bottom and left side nodes are optional overlays.
- `triggerDistance` defaults to 16 and disables hover when zero or negative.
- `pinnedSide` keeps one side visible.
- Show and hide use configurable delay and duration.

1.2 Maven dependency

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

Maven coordinates for the GemsFX control library.

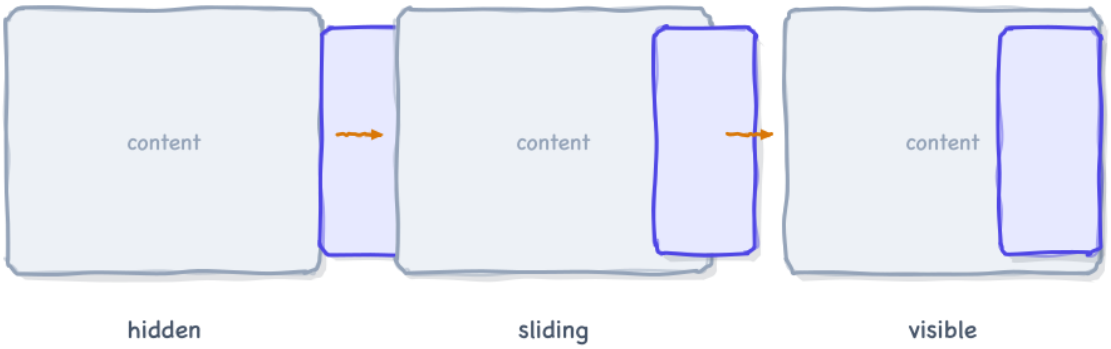
2. Getting started

Set a content node before the pane is laid out. Side nodes are cast to `Region` when installed, so use Region subclasses for the hidden sides.

```
HiddenSidesPane pane = new HiddenSidesPane();
Label content = new Label("Content");
content.setMaxSize(Double.MAX_VALUE, Double.MAX_VALUE);
pane.setContent(content);

Label right = new Label("Details");
right.setPrefSize(220, 200);
pane.setRight(right);
pane.setTriggerDistance(20);
```

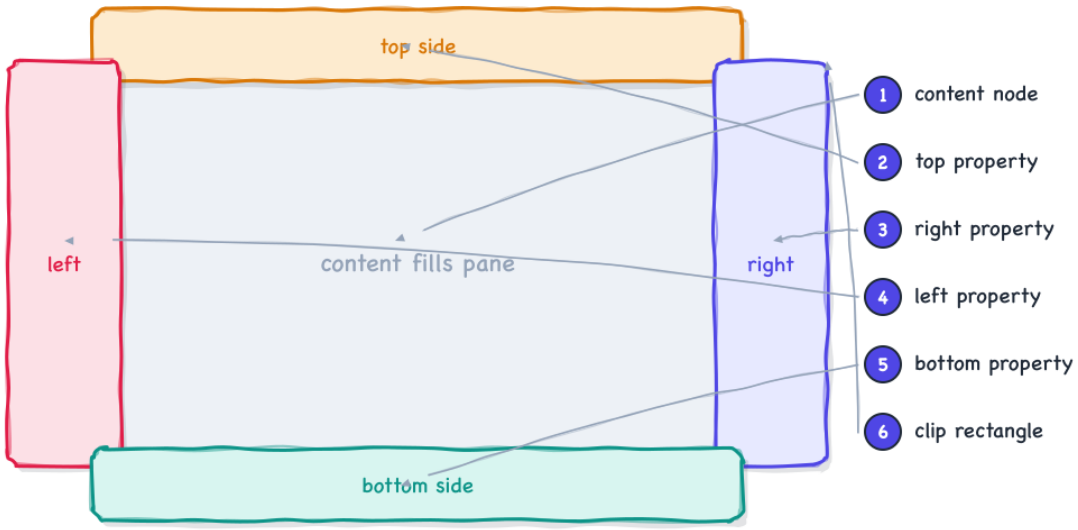
A complete right-side overlay setup.



Hidden, sliding and visible states for a side node.

> **Careful.** Layout calls `getContent().resizeRelocate(...)` without a null check. Do not leave content null once the pane is displayed.

3. Anatomy



The content node plus the four optional side nodes.

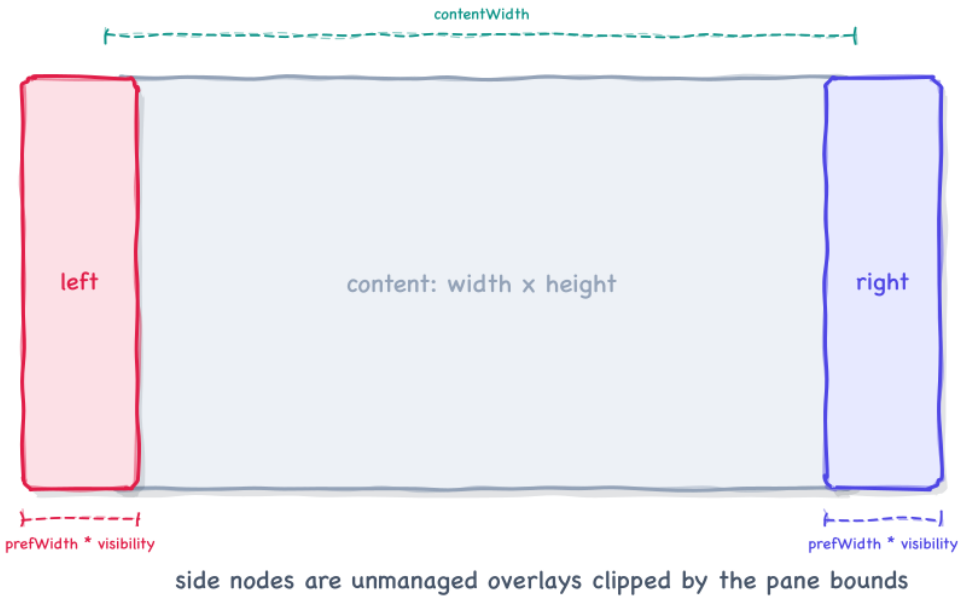
Part	Property	Description
Content	`content`	Full-size primary node and source of computed sizes.
Top	`top`	Region sliding down from above.
Right	`right`	Region sliding in from the right edge.
Bottom	`bottom`	Region sliding up from below.
Left	`left`	Region sliding in from the left edge.

Part	Property	Description
Clip	Rectangle	Bound to pane width and height to hide off-pane portions.

4. Control API

Property	Type	Default	Description
content	ObjectProperty<Node>	null	Full-size content node.
top / right / bottom / left	ObjectProperty<Node>	null	Optional side nodes. In practice use Region subclasses.
triggerDistance	DoubleProperty	16	Edge distance that triggers hover display; <= 0 disables mouse-triggered display.
pinnedSide	ObjectProperty<Side>	null	Side that remains visible.
animationDelay	ObjectProperty<Duration>	300 ms	Delay before show / hide timeline starts; null falls back to 300 ms.
animationDuration	ObjectProperty<Duration>	200 ms	Timeline duration; null falls back to 200 ms.

5. Layout geometry



Visibility transforms preferred side size into visible overlay thickness.

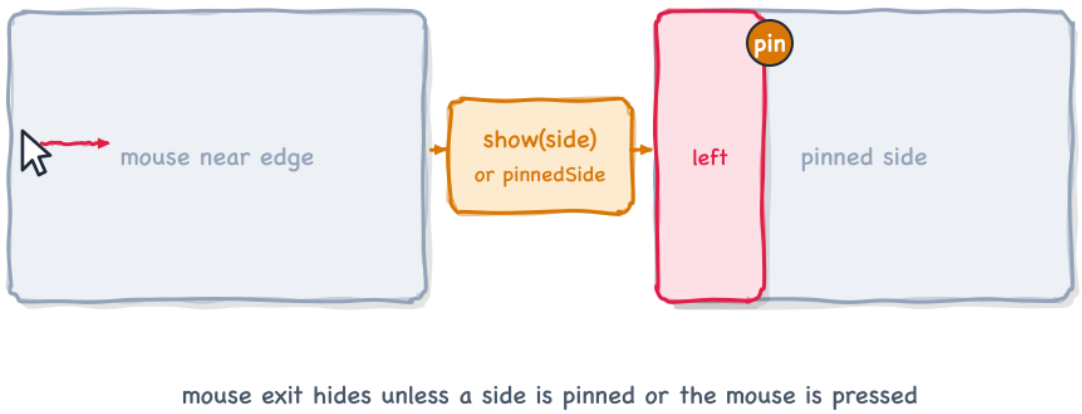
The content is resized to `0,0,width,height`. Horizontal side nodes use their preferred width and pane height; vertical side nodes use pane width and their preferred height.

```
rightPref = right.prefWidth(contentHeight)
offset = rightPref * visibility[RIGHT]
right.resizeRelocate(contentWidth - offset, 0, rightPref, contentHeight)
```

Simplified placement for the right side.

Each side has a visibility property in the range 0 to 1. Values greater than zero make the node visible; zero hides it.

6. Interaction and animation



Mouse activation and pinning.

Trigger	Result
Mouse in edge band	Shows that side if mouse is enabled and no side is pinned.
Mouse outside visible sides	Hides all sides.
Mouse exited side node	Hides unless a side is pinned or mouse is pressed.
Set pinnedSide	Shows the pinned side and suppresses hover switching.

Showing a side stops a running hide timeline and animates the selected side to 1 while all others animate to 0. Hiding animates every side to 0.

7. Styling

HiddenSidesPane adds no root style class, exposes no styleable CSS properties and has no user agent stylesheet. Style the content and side nodes directly.

```
VBox drawer = new VBox();
drawer.getStyleClass().add("edge-drawer");
pane.setLeft(drawer);
```

```
.edge-drawer {
    -fx-background-color: white;
    -fx-effect: dropshadow(gaussian, rgba(0,0,0,.25), 14, 0, 0, 6);
}
```

8. Sizing contracts and node requirements

The pane delegates all size computations to the content node. Minimum, preferred and maximum width / height return the corresponding value of `content`, or zero when content is null. Side nodes do not contribute to these computations because they are transient overlays.

Computation	Source
`computePrefWidth(height)`	`content.prefWidth(height)`
`computePrefHeight(width)`	`content.prefHeight(width)`
Side thickness	The side node preferred width or preferred height during layout.
Content bias	First managed child content bias, preferring horizontal if present.

When side properties change, `updateStackPane()` clears and rebuilds the children list. For every side node it also sets the maximum size in the sliding direction to `Region.USE_PREF_SIZE` and the cross-axis maximum to `Double.MAX_VALUE`.

```
((Region) getRight()).setMaxWidth(Region.USE_PREF_SIZE);
((Region) getRight()).setMaxHeight(Double.MAX_VALUE);
```

Right and left sides keep preferred width and stretch vertically.

> **Careful.** The side-node casts mean a plain Canvas or Group is not a safe side node unless wrapped in a Region such as StackPane.

9. Event ordering and edge precedence

The private `getSide(MouseEvent)` method checks the left edge first, then right, then top, then bottom. In a corner where trigger bands overlap, horizontal sides therefore win over vertical sides.

Mouse condition	Chosen side
x <= triggerDistance	LEFT
x > width - triggerDistance	RIGHT
y <= triggerDistance	TOP
y > height - triggerDistance	BOTTOM
outside bounds	none

Mouse release re-evaluates the pointer position. This avoids a side staying open after a press / drag sequence ends away from all trigger bands.

```
pane.setTriggerDistance(24);
pane.setPinnedSide(null);
```

10. Production usage notes

Hidden side panes are most useful for temporary tools, inspectors and contextual palettes. They are not modal: the content remains visible underneath and the side overlays do not reserve layout space.

Concern	Recommended practice
Preferred sizes	Set explicit preferred sizes for important children so the layout maths has stable inputs.
Insets and padding	Remember that pane insets are part of the available area calculation or child placement.
Managed state	Only managed children should be expected to participate in layout decisions.
Runtime changes	Property invalidation calls requestLayout, so batch related changes where possible.

For touch-first or keyboard-first applications, prefer pinning from explicit buttons over relying on the mouse trigger band. A trigger distance of zero gives a completely programmatic pane while preserving the same slide-in layout.

```
pane.setTriggerDistance(0);
openInspector.setOnAction(evt -> pane.setPinnedSide(Side.RIGHT));
closeInspector.setOnAction(evt -> pane.setPinnedSide(null));
```

11. Recipes

11.1 Pin from a button

```
pinLeft.setOnAction(evt -> pane.setPinnedSide(Side.LEFT));
unpin.setOnAction(evt -> pane.setPinnedSide(null));
```

11.2 Disable hover activation

```
pane.setTriggerDistance(0);
```

11.3 Checklist

1. Set content before showing the pane.
2. Use Region subclasses for side nodes.
3. Keep side preferred sizes realistic.
4. Use pinnedSide for programmatic trays.

12. See also

- Demo application: `com.dlsc.gemsfx.demo.HiddenSidesPaneApp` (run with `mvn javafx:run -f gemsfx-demo/pom.xml -Dmain.class=com.dlsc.gemsfx.demo.HiddenSidesPaneApp`)
- `PowerPane` - composes a HiddenSidesPane.
- `DrawerStackPane` - bottom drawer alternative.
- API documentation: https://dlsc-software-consulting-gmbh.github.io/GemsFX/api/