Ben Zeng

Swing++

What is it?

Swing++ is a beginner-friendly library that extends upon Java Swing and is optimized for the development of games and graphical-intensive apps. It aims to generalize complex graphical animations and user interface within JComponents by handling individual graphical elements separately and efficiently.

Swing++ aims to simplify Java Swing and create a more dynamic ecosystem for components, allowing free and customized collision detection, broken-down input handling, and the flexibility to efficiently and dynamically modify components in any way.

Swing++ is perfectly compatible with everything that regular Java Swing has to offer, including all of the built-in classes for GUI, as well as being able to utilize the currently existing framework.

How does it work?

Swing++ is composed of four main classes and an interface for collision detection.

HostApplication

The Content Pane of any application built using Swing++. The HostApplication is an extension to the custom MultiPanel that automatically and accurately simulates framerate by repainting its components to allow for animations.

MultiPanel

A container for the JComponent that handles screen-switching and the alternation of multiple views using the CardLayout layout on a JPanel.

GraphicalComponent

A more dynamic counterpart to the static JComponent, that aims to allow for abstraction of specific graphical components by representing any type of object on a screen using individual components. This can range from any type of graphical element, such as moving buttons, entities on a screen, animations, backgrounds, and can be used to simplify and organize nearly any form of GUI into multiple classes.

The GraphicalComponent allows for smooth animations and custom, dynamic hitboxes for collision-detection, allowing you to utilize components to represent anything - constantly being fired projectiles in a game, teleporting entities, individual items in drag-and-drop, and much more!

Each GraphicalComponent has multiple sets of event listeners, allowing you to have complete freedom over how you choose to handle input! You can utilize listeners that respect the hitboxes of itself and other components, or utilize listeners that fire whenever its parent container receives an event.

SPComponent

A general-purpose, enhanced JComponent that serves as a container for the GraphicalComponent. Input and graphics for this component can be handled directly by the SPComponent, but it is strongly recommended to utilize its GraphicalComponents instead!

GraphicalHitbox

The GraphicalHitbox is an interface that can be assigned to a specific GraphicalComponent to control collision-detection logic and has its own set of listeners for when its hitbox gets activated. The GraphicalHitbox directly utilizes mouse events to determine whether or not the hitbox has been activated, allowing you to create custom-shaped hitboxes for GraphicalComponents that you would not be able to do with a JComponent!

Resources

Source Code

Swing++.jar

https://drive.google.com/file/d/11RyO0tte9jUhAYLXGClMcAeigiSRKluG/view?usp=sharing

Changelogs:

Version 1.0:
- Initial Release.

Version 1.1:
- Added support for the addition of regular JComponents to ScreenPanels.

Version 2.0:
- **SPComponent**
  - Renamed ScreenPanel to SPComponent (Swing Plus Component)
  - Added support for using SPComponents as regular JPanels - Can both be contained within other JComponents, as well as act as a container
- **GraphicalComponent**
  - Renamed ScreenComponent to GraphicalComponent
  - Added support for multi-event handling and correctly firing mouse/key release events to the GraphicalComponents that receive them
  - Added a list of active events that this component has received
- **Hitboxes**
  - Added the Hitbox collision-detection system for GraphicalComponents
  - Separated mouse events into a per-hitbox basis and an event that gets triggered whenever its container receives the event
  - Deprecated denyComponents() to encourage usage of hitboxes and layering SPComponents