The Past, Present, and Future of SWT

Eric Williams

Email: ericwill@redhat.com
Twitter: @yyzericwill

About me: Eric Williams

- Studied computer science at the University of Toronto

- Intern at Red Hat from May 2015 - August 2016
  - Red Hat Eclipse team member in the Toronto office
  - full time SWT developer
  - became an SWT committer in February 2016

- After my internship I was hired back full time to work on SWT

- This is my first ever EclipseCon, both as an attendee and a speaker!

redhat.

What is SWT?

- SWT is a cross-platform GUI toolkit written in Java, originally developed by IBM in 2001 and now maintained by the Eclipse Foundation
- SWT allows for the creation of native GUI applications using Java
- The Eclipse IDE uses SWT, among a few other applications like Azureus (Vuze), Tuxguitar, and IBM Lotus Notes
- Supported platforms: Windows, Mac OS X, and Linux
- This means SWT needs to interact with the native GUI toolkit on each platform: on Windows it's Win32, on Mac OS X it's Cocoa, and on Linux it's GTK
- Provides a unified API across all platforms

redhat.

How does it work?

- Since SWT is cross-platform, it seeks to emulate the native look and feel on each operating system it supports

- This is accomplished via the Java Native Interface (JNI)
  - If you are interested in the specifics of JNI, Leo is hosting a talk on Thursday about this very topic

- Two types of widgets:
  - Native widgets: the majority of SWT widgets are OS specific, and will look different on each platform
    - These are created using JNI to communicate with the native toolkit

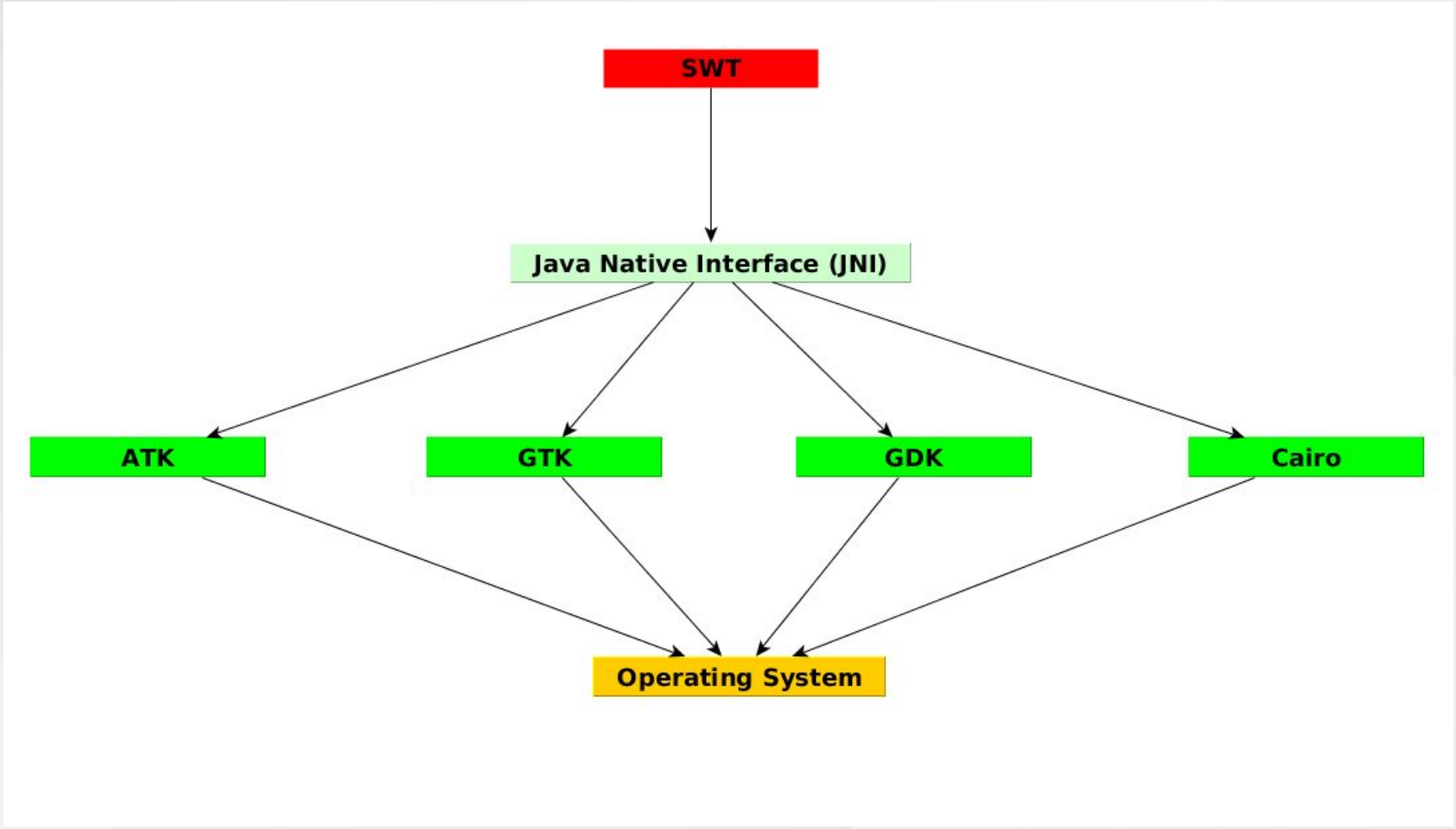
  - Common widgets: these look the same on all platforms
    - These are “pure Java” widgets, they have no platform specific interaction

redhat.

GTK: the GIMP toolkit (GTK+)

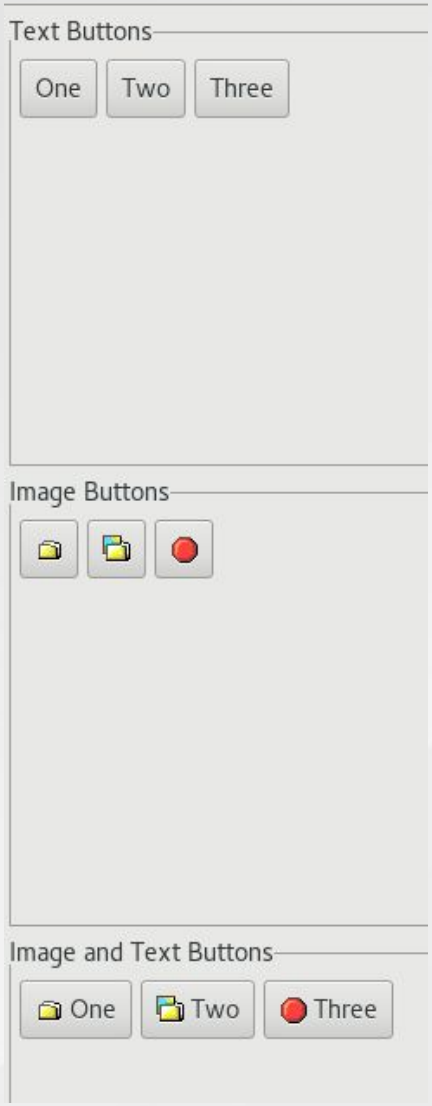
- Initially released in 1998 (GTK 1.0)
- Open source widget toolkit maintained by the GNOME foundation
- Native to Linux, has cross platform support for Mac OS X and Windows
- One of the most popular widget toolkits available for Linux
  - Written in C using GObject (object oriented C and part of GLib)
- GTK has other libraries that come packaged with it:
  - GIMP Drawing Kit (GDK): handles lower level GTK drawing and geometry, interacts with the OS display server (such as X11/Wayland)
  - Accessibility Toolkit (ATK): an API that allows GTK applications to be accessible for the hearing or vision impaired
  - Cairo: a library that provides a vector graphics and 2D drawing API -- doesn't come packaged with GTK per se, but GTK3 uses Cairo exclusively for custom drawing and rendering

How SWT and GTK interact

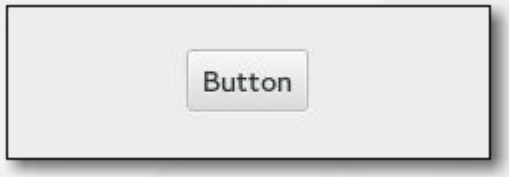


redhat.

SWT look and feel on GTK



These are SWT buttons



This is a native GTK button created in a C snippet

The look and feel in both screenshots is the same: this is the goal of SWT in a nutshell

redhat.

GTK versioning

- Two major versions supported: GTK2 and GTK3
- GTK2 is mostly in maintenance mode, only critical bug fixes are committed
  - 2.24 is the current stable release of GTK2
- GTK3 is the latest major version: 3.22 is the stable release
  - no new development happening for GTK3, only bug fixes
- SWT supports both versions, although new feature work for GTK2 has been scaled back significantly
- The design differences between GTK 2.24 and GTK 3.22 make supporting both versions in SWT a challenge, most due to:
  - API differences
  - internal machinery changes
  - aggressive deprecations in GTK3

redhat.

GTK2: an overview

- Originally released in 2002 with version 2.0: added support for new widgets and theming:
  - File choosers, palettes, combo boxes, spinners
  - GTK theming engines (Oxygen-GTK was a popular one)

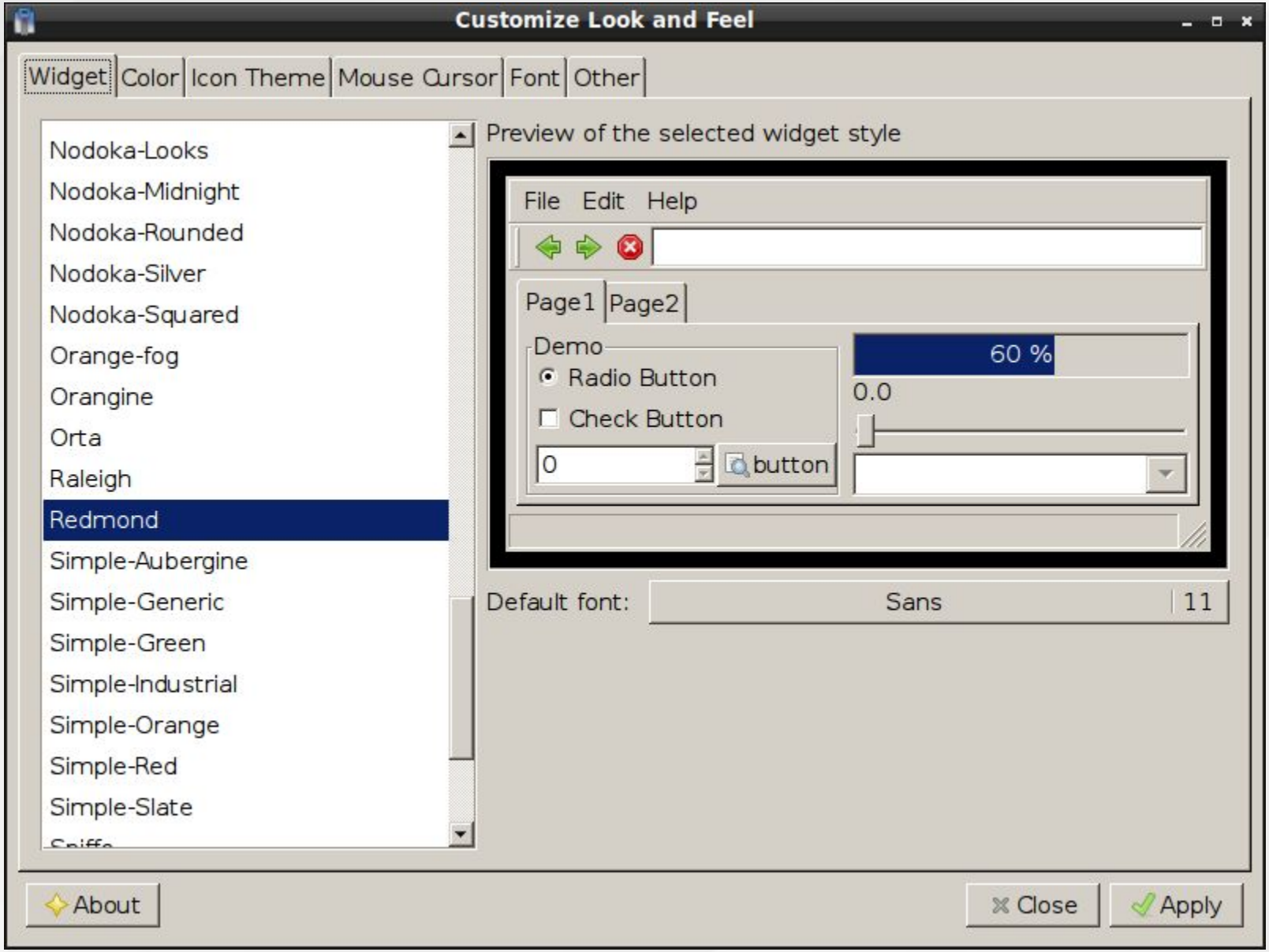
- New drawing methods, namely:
  - Cairo drawing for most widgets after GTK2.8
  - Pango for font rendering

- Improvements for internal machinery:
  - GObject released with version 2.0
  - UTF-8 support
  - JPEG 2000 compression support

- GTK2 is characterized as very stable, lots of new features added to a rock solid platform

- Still readily available but application maintainers are encouraged to move away from it

redhat.


Customize Look and Feel
Widget
Color
Icon Theme
Mouse Cursor
Font
Other
Nodoka-Looks
Nodoka-Midnight
Nodoka-Rounded
Nodoka-Silver
Nodoka-Squared
Orange-fog
Orangine
Orta
Raleigh
Redmond
Simple-Aubergine
Simple-Generic
Simple-Green
Simple-Industrial
Simple-Orange
Simple-Red
Simple-Slate
Preview of the selected widget style
File Edit Help
Page1
Page2
Demo
Radio Button
Check Button
0
button
60 %
0.0
Default font:
Sans
11
About
Close
Apply

GTK3: it all changes

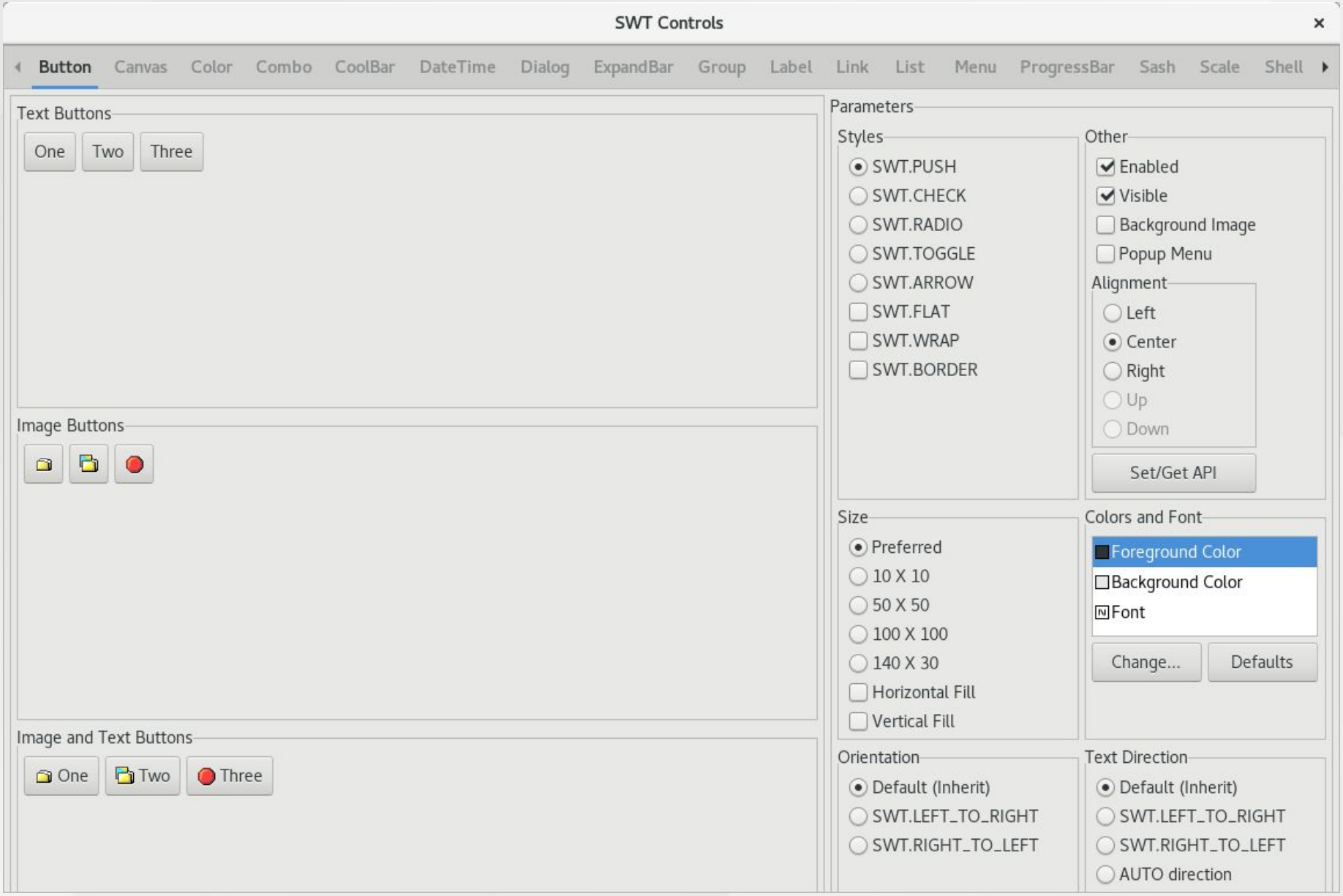
- First released in 2011 with the goal of "fixing" GTK2
  - GTK2 was very stable but also very X11 centric
  - Changes in accessibility support
  - Aggressive deprecation of widgets and functionality in favor of different internal design changes and methodologies
  - Aggressive version-to-version changes: GTK3.4 is very much closer to GTK2 than it is to GTK3.22

- Theming system completely overhauled
  - All CSS based using CSS theme files
  - No more theming engines after GTK3.14

- Sizing mechanism changed completely:
  - widgets have preferred and natural sizes
  - invisible widgets have no size
  - shrinking a widget below its minimum size not supported

- More modern look and feel compared to GTK2: better styling

redhat.


SWT Controls
Button
Canvas
Color
Combo
CoolBar
DateTime
Dialog
ExpandBar
Group
Label
Link
List
Menu
ProgressBar
Sash
Scale
Shell
Text Buttons
One
Two
Three
Image Buttons
Image and Text Buttons
One
Two
Three
Parameters
Styles
SWT.PUSH
SWT.CHECK
SWT.RADIO
SWT.TOGGLE
SWT.ARROW
SWT.FLAT
SWT.WRAP
SWT.BORDER
Other
Enabled
Visible
Background Image
Popup Menu
Alignment
Left
Center
Right
Up
Down
Set/Get API
Size
Preferred
10 X 10
50 X 50
100 X 100
140 X 30
Horizontal Fill
Vertical Fill
Colors and Font
Foreground Color
Background Color
Font
Change...
Defaults
Orientation
Default (Inherit)
SWT.LEFT_TO_RIGHT
SWT.RIGHT_TO_LEFT
Text Direction
Default (Inherit)
SWT.LEFT_TO_RIGHT
SWT.RIGHT_TO_LEFT
AUTO direction

SwtFixed: a custom container

- In GTK2, there was a container widget called GtkFixed
- It allowed users to place widgets inside of it with fixed coordinates/sizes
- In GTK3 this widget was changed, and no viable alternatives existed
  - in addition, GTK3 sealed off access to many private structs that were available on GTK2
  - this means SWT could no longer manipulate them directly
- Instead, SWT implements its own custom container called SwtFixed
  - allows for fixed size and positioning
  - implementation is at the C level
- Every SWT widget on GTK3 has an SwtFixed container as a parent
- Emulated widgets (aka, non-native widgets) are drawn onto a blank SwtFixed widget

redhat.

A closer look at GTK3

- **3.0:**
  - main contribution of GTK3, consolidation of several libraries
  - CSS theming added, new API
- **3.2 - 3.6:**
  - Wayland support
  - accessibility improvements
- **3.8 - 3.14:**
  - sizing changes
  - 7 widgets added, but just as many deprecated (including colors)
  - client side decorations, Hi-DPI, major overhaul of internal drawing
  - removal of settings, Motif DnD, tear off menu items
- **3.16 - 3.18:**
  - OpenGL support, touch pad scrolling, more CSS theming support
  - overlay scrolling enabled by default
- **3.20 - 3.22 (present):**
  - sizing changes, CSS theming added to API
  - CSS node support, changes in CSS selectors

redhat.

GTK3 changes and SWT: current efforts

- SWT has an API it is bound by: absolute behavior like setSize(x, y), setBackgroundColor(color), etc.

- Changes in GTK disrupt the functionality of the API
  - on top of this, all versions of GTK3 must be supported, but behavior may change from one major version of GTK3 to the next
  - this makes for multiple code paths depending on the GTK3 version

- Work can be broken down into categories
  - bugs: immediate breakages that need to be dealt with from one version to the next
    - crashes, color/rendering errors, sizing issues

  - big picture projects: as things are deprecated they need to be replaced with suitable alternatives and implementations
    - accessibility support for GTK3 (in progress)
    - WebKit2 port (in progress, 90% finished)
    - porting SWT colors from GdkColor to GdkRGBA (complete)

redhat.

Future work: Wayland

- SWT on Wayland is not fully stable
  - sizing challenges mean odd behavior
  - Wayland has no absolute coordinates like X11 does
  - all coordinates are relative to the parent window
    - calling Control.setLocation(100, 100) on Wayland places the Control relative to the parent, not the display

- Drag N Drop (DnD):
  - thanks to Ian Pun, Wayland DnD is in a better state than it was last year
  - however some improvements still remain

- Miscellaneous issues:
  - certain decorations broken due to client side decoration differences
  - double clicking broken in certain instances
  - events not firing when supposed to

- Ubuntu has dropped Mir support and will switch back to Wayland
  - more users, more use cases

redhat.

Future work: GTK4

- Deprecation changes will come into effect
  - functions/API/widgets deprecated in GTK3 will be removed in GTK4
  - we try our best to keep up with deprecations to save work in the future, but it is not possible to keep up with them all
    - in most cases we are still replacing things that were deprecated from GTK2 -> GTK3
  - notable areas include GNotifications, Accessibility, OpenGL drawing

- Base GtkWidget API is changing, which affects our implementation of SwtFixed
  - core GtkWidgetClass functions are changing
  - we override/implement these functions
  - SwtFixed will have to be adapted to these changes
  - in GTK4, all widgets will be able to act as containers -> more adaptation

- Drawing changes

- For anyone interested in GTK4 changes, the roadmap/plan can be found here: https://wiki.gnome.org/Projects/GTK+/Roadmap

redhat.

Thank you for listening!

eclipsecon Europe 2017

OSGi™ Community Event 2017

Evaluate the Sessions

Sign in and vote at **eclipsecon.org**

- 1 0 + 1