UseCase_5_Graphical_User_Interface

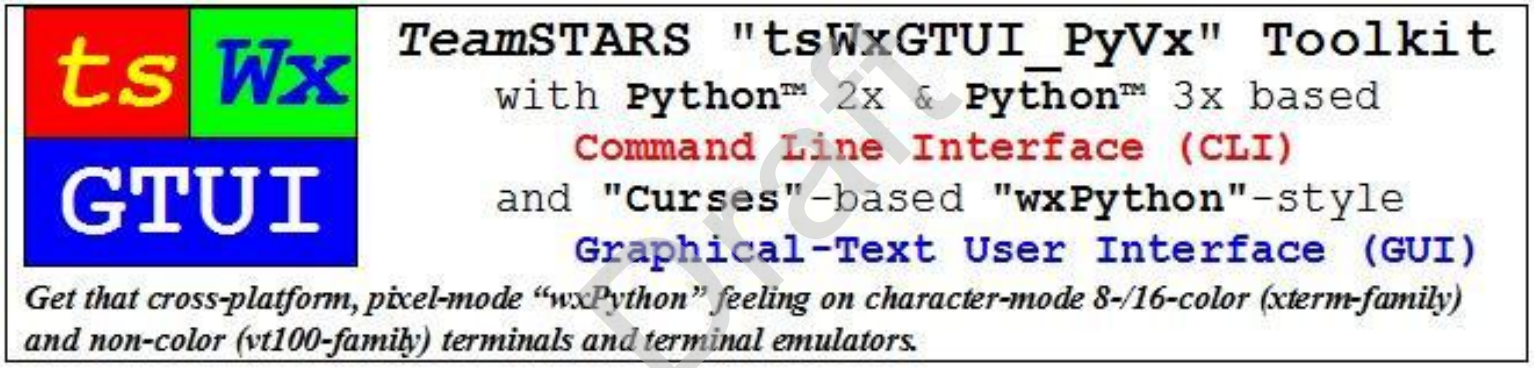


Table of Contents (*with slide show* Hyperlinks)

- Introduction
  - WEB browser link to role model "wxPython" interface to "wxWidgets"
  - "wxPython" High Level Pixel-Mode Graphical User Interface API
  - Curses Terminal Control Library and Low Level Graphical Widgets
  - Block Diagram Relationship Between GUI-Mode, CLI-Mode & User
  - "wxPython" API Classes
  - Sample "wxPython" Character-Mode Emulation Display (GUI)
- Differences between Pixel-Mode "wxPython" role model and its Character-Mode Emulation
  - Architecture Differences
  - Input & Output Differences
  - Import & Usage Differences
  - Virtual Platform Interface Differences Emulation
  - WEB browser links to Sample Pixel-Mode "wxPython" role model screenshots:
    - Second Generation
    - Third Generation

Draft

"wxPython" High Level Pixel-Mode Graphical User Interface API (Table of Contents)

- "wxPython" Releases:
  - Mature, second generation (2.0.0.0-2.9.5.0) was first released in 1999 and is no longer supported or available at "wxPython.org".
    - "wxPython" 2.8.9.2 API began as, and is still, the prototype for the Character-Mode Emulation.
  - Evolving, third generation (3.0.0.0-3.0.2.0) was first released in 2013 and is supported and available at "wxPython.org".
    - "wxPython" 3.x API will ultimately become the prototype for the Character-Mode Emulation.
- Excerpts From Wikipedia, the free encyclopedia:
  - "wxPython is a wrapper for the cross-platform GUI API (often referred to as a 'toolkit') wxWidgets (which is written in C++) for the Python programming language."
    - "In computer programming, an application programming interface (API) is a set of routines, protocols, and tools for building software applications. An API expresses a software component in terms of its operations, inputs, outputs, and underlying types. An API defines functionalities that are independent of their respective implementations, which allows definitions and implementations to vary without compromising the interface. A good API makes it easier to develop a program by providing all the building blocks. A programmer then puts the blocks together."
  - "It is implemented as a Python extension module (native code)."
  - "Like wxWidgets, wxPython is free software."

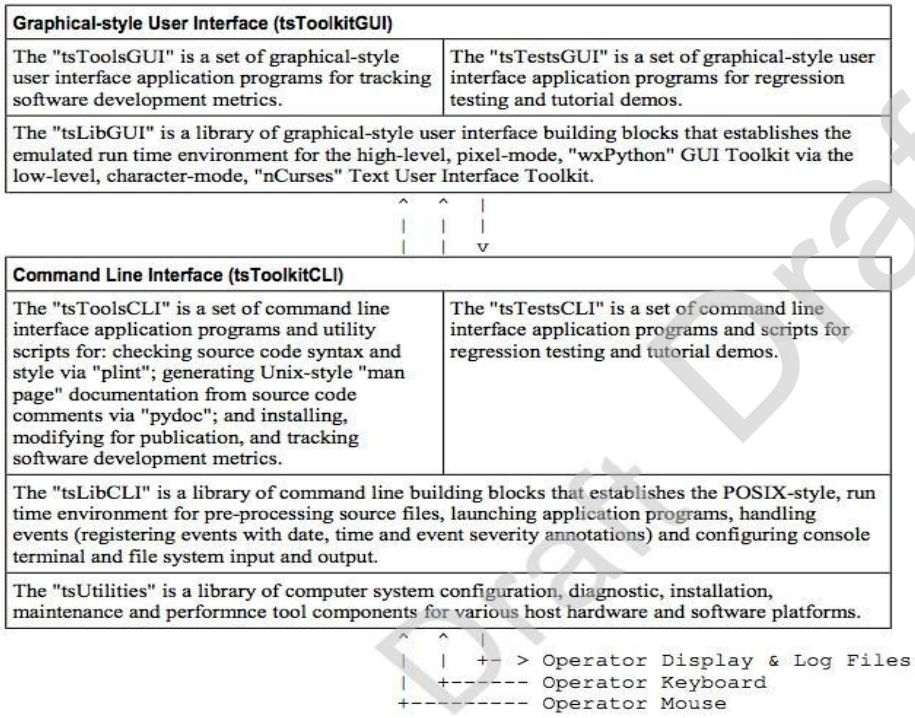
Draft

Curses Terminal Control Library and Low Level Graphical Widgets (Table of Contents)

- Excerpts From https://www.gnu.org/software/ncurses/:
  - "Announcing ncurses 6.0."
  - "Release Notes: These notes are for *ncurses* 6.0, released August 8, 2015."
  - "Extend the cchar_t structure to allow more than 16 colors to be encoded."
- Excerpts From Wikipedia, the free encyclopedia:
  - "The first curses library was written by Ken Arnold and originally released with BSD UNIX, where it was used for several games, most notably Rogue. Some improvements were made to the BSD library in the 1990s as "4.4BSD" curses, e.g., to provide more than one type of video highlighting. However, those are not widely used."
  - "Different lines of development started by imitating the AT&T curses, from at least three implementations: pcurses by Pavel Curtis (started in 1982), PDCurses (Public Domain curses) by Mark Hessling to support his editor THE (started in 1987) as well as Rexx/Curses, and PC curses (version 1.4 and earlier by Bjorn Larsson based inspired by Pavel Curtis' library before 1990.)"
  - "ncurses (new curses) "originated as pcurses ... and was re-issued as ncurses 1.8.1 in late 1993". ncurses is the most widely known implementation of curses, and has motivated further development of other variations, such as BSD curses in the NetBSD project."
  - "Curses-based software is software whose user interface is implemented through the Curses library, or a compatible library (such as New Curses)."
  - "Curses is designed to facilitate GUI-like functionality on a text-only device, such as a PC running in console mode, a hardware ANSI terminal, a Telnet or SSH client, or similar."
  - "Curses-based programs often have a user interface that resembles a traditional graphical user interface, including 'widgets' such as text boxes and scrollable lists, rather than the command line interface (CLI) most commonly found on text-only devices. This can make them more user-friendly than a CLI-based program, while still being able to run on text-only devices."

Draft

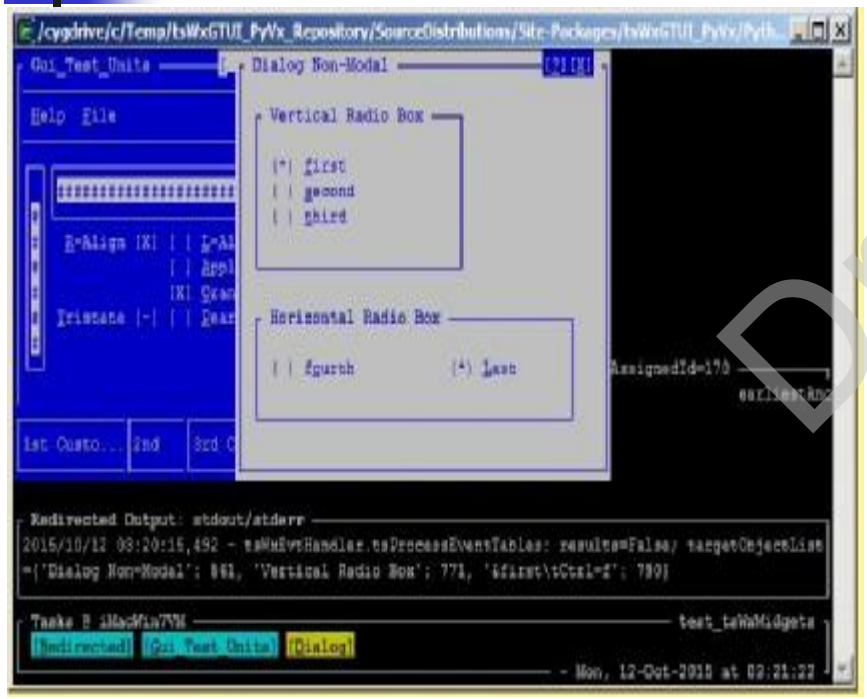
Block Diagram Relationship between GUI-Mode, CLI-Mode & User (Table of Contents)



- **Graphical User Interface**
  - High-level GUI
    - Character-mode emulation of popular pixel-mode **"wxPython" 2.8.9.2** GUI API
  - Low-Level Virtual Platform GUI
    - Popular, Python character-mode **Curses** GUI API
- **Command Line Interface**
  - High-level CLI
    - Popular, industry standard **POSIX** API
    - Available on **Linux**, **Mac OS X**, **Microsoft Windows** (with free nisKayuna959Cygwin plug-in from Red Hat) and **Unix**
  - Low-Level Virtual Platform CLI
    - Python "**argparse**", "**optparse**" or "**getopt**" API
- **Operator's Computer Terminal**
  - Output (Text or optional Graphics **Display** and optional **Printer**)
  - Input (**Keyboard** and **Mouse**)

Draft Draft

Sample "wxPython" Character-Mode Emulation Display (GUI) (Table of Contents)



- **Output to the User:**
  - An organized assortment of pictorial icons (buttons, checkboxes, radio boxes & buttons, scrollbar arrows & gauges etc.), visual indicators (border & separator lines, boxes, and textual titles, labels and content) which appear only when needed, as opposed to the scrolled chronological text-only output of a Command Line Interface.
    - **Blue Frame** (**Application partially hidden**) with Title, Menu Bar, Horizontal & Vertical Gauges, Check Boxes and Status Bar
    - **White Dialog** (**Application**) with Title, Window Control Buttons, Radio Boxes and Radio Buttons
    - **Black Redirected Output (stdout/stderr) Frame** (**Application Option**) **w**ith Title and Output of Date, Time & Severity-level () CRITICAL, ERROR, ALARM, INFO, DEBUG etc.) stamped event notifications (**optional colorization not shown**)
    - **Task Bar Frame** (**Application Option**) **w**ith Title, Application Frame & Dialog Focus Control Buttons and Output of Network (Name or IP-Address) & Program Name; Idle Time Spinner and Current Date & Time
- **Input from the User:**
  - **Mouse Button** pressed, released, clicked (single, double, triple) (**none in this example**)
  - **Keyboard Button** with text input (**none in this example**) inserted, within any application designated GUI object, as it is echoed in front of the moving cursor prompt character (under-line/block) with optional blink.

Draft

"wxPython" API Classes (tsLibGUI *full listing*) (Table of Contents)

Category: *Module File Name (Class Name)*

- **"wxPython" 2.8.9.2 Application Launchers**
  - tsWxPyApp.py (PyApp)
  - tsWxApp.py (App)
  - tsWxPySimpleApp.py (PySimpleApp)
- **Top-Level "wxPython" 2.8.9.2 Objects**
  - tsWxFrame.py (Frame)
  - tsWxDialog.py (Dialog)
  - tsWxPyOnDemandOutputWindow.py (Frame for RedirectedOutput)
  - tsWxTaskBar.py (Frame)
- **"wxPython" 2.8.9.2 Controls**
  - wxWxButton.py (Button)
  - tsWxCheckbox.py (CheckBox)
  - tsWxRadioButton.py (RadioButton)

Category: *Module File Name (Class Name)*

- **Lower-Level "wxPython" 2.8.9.2 Objects**
  - tsWxGauge.py (Gauge)
  - tsWxMenuBar.py (MenuBar)
  - tsWxPanel.py (Panel)
  - tsWxRadioBox.py (RadioBox)
  - tsWxScrollBar.py (ScrollBar)
  - tsWxStatusBar.py (StatusBar)
- **"wxPython" 2.8.9.2 Sizers**
  - tsWxBoxSizer.py (BoxSizer)
  - tsWxGridSizer.py (GridSizer)
- **"wxPython" 2.8.9.2 Events**
  - Keyboard/Mouse/Timer
- **"wxPython" 2.8.9.2 Configuration**
  - tsWx.py (Wx)
  - tsWxGlobals

Draft

Input & Output Differences between "wxPython 2.8.9.2" API Pixel-Mode and Character-Mode Emulation (Table of Contents)

- **Output to the user via a terminal display device**:
  - **Pixel-mode Icons are pictograms** (more like a traffic sign than a detailed illustration of the actual entity it represents) which help the user monitor and control computer operation. Icons can optionally serve as an electronic hyperlink or file shortcut to trigger a program or access data
  - **Character-mode "wxPython 2.8.9.2" API Emulation** uses its host platform's native Curses services to input from terminal pointing & keyboard devices and to output icon characters and text composed of pre-defined fixed-size character cells (hardware generated array of pixels) to application-specified character cell column and row (line) fields on a computer terminal display.
- **Input from the user via a terminal device:**
  - **Pointing Device (mouse, trackball, touch pad or screen)**
    - Device is moved into a position by its operator before a mouse button is clicked to trigger a function button, radio button, checkbox or keyboard input operation.
  - **Keyboard Device:**
    - Input may be echoed as output to a reserved area of the display that is written from top to bottom and then scrolling off the top as each new line is written to the bottom of the reserved area
- **Comparison of Pixel-mode with Character-mode Emulation**
  - **Pixel-mode "wxPython 2.8.9.2" API u**ses its host platform's native GUI services to input from terminal pointing & keyboard devices and to output icons and text strings composed of dot-based picture elements (pixels) to application-specified pixel cell column and row (line) fields on a computer terminal display.
  - **Character-mode "wxPython 2.8.9.2" API Emulation c**onsumes significantly less memory, processor time and communication bandwidth than Pixel-mode. For a simplified example:
    - In Character-mode, one 8-bit-byte character code can designate one of 256 alphanumeric, punctuation or line –drawing characters (in an 8x12 pixel font) plus a second 8-bit-byte character can represent the foreground-background color pair in a 16-color palette. As a consequence, computer-terminal communication bandwidth consumes 2 Character-mode bytes per 8x12 pixel font character cell.
    - In Pixel-mode, one 8-bit-byte code sets the intensity of three individual Red-Green-Blue color subpixels (for a 16-million color palette) associated with each pixel in an 8x12 font cell. As a consequence, computer-terminal communication bandwidth consumes 288 Pixel-mode bytes per 8x12 pixel font character cell.
    - Retains names of original "wxPython 2.8.9.2" classes, methods, functions, constants and variables.
    - Uses "tsWx" prefix to file names and "ts" prefix to names of internal classes, methods, functions, constants and variables.

Draft

Import & Usage Differences between "wxPython 2.8.9.2" API Pixel-Mode and Character-Mode Emulation (tsLibGUI) (Table of Contents)

- Pixel-mode "wxPython 2.8.9.2-2.9.5.0" applications use a Python 2x / 3x version specific "wxPython 2.8.9.2" toolkit site-package:
  - "import wx".

- Applications reference "wxPython 2.8.9.2-2.9.5.0" objects:
  - Classes such as "Frame" via "wx.Frame"
  - Data such as "ID_ANY" via "wx,ID_ANY"
  - myApp = wx.PySimpleApp(redirect=True, filename=None)
  - myFrame = wx.Frame(None, title='Hello World')
  - myFrame.SetBackgroundColour("CYAN")
  - myFrame.SetForegroundColour("BLACK")
  - myFrame.Show()

- Character-mode "wxPython 2.8.9.2" applications use a Python 2x / 3x version specific "tsWxGTUI_PyVx" toolkit site-package:
  - "from tsWxGTUI_Py2x.tsLibGUI import tsWx as wx"
  - "from tsWxGTUI_Py3x.tsLibGUI import tsWx as wx"

- Applications reference "wxPython 2.8.9.2" objects:
  - Classes such as "Frame" via "wx.Frame"
  - Data such as "ID_ANY" via "wx,ID_ANY"
  - myApp = wx.PySimpleApp(redirect=True, filename=None)
  - myFrame = wx.Frame(None, title='Hello World')
  - myFrame.SetBackgroundColour("CYAN")
  - myFrame.SetForegroundColour("BLACK")
  - myFrame.Show()

Draft

Simplified Wrapper and Interface Generator (SWIG) (Table of Contents)

From Wikipedia, the free encyclopedia:

- *"Simplified Wrapper and Interface Generator (SWIG) is an open-source software tool used to connect computer programs or libraries written in C or C++ with scripting languages such as Lua, Perl, PHP, Python, R, Ruby, Tcl, and other languages like C#, Java, JavaScript, Go, Modula-3, OCaml, Octave, Scilab and Scheme. Output can also be in the form of XML or Lisp S-expressions."*

Draft

Virtual Platform Interface Relationship between "wxPython 2.8.9.2" API Pixel-Mode and Character-Mode Emulation (tsLibGUI *full listing*) (Table of Contents)

- **"wxPython 2.8.9.2 to 2.9.5.0" Pixel-Mode API**
  - **Python 2x Application**
    - Pixel-mode cross-platform "wxPython 2.8.9.2-2.9.5.0" classes, methods and functions issue service requests to and receive service responses using a **SWIG** interface to cross-platform "wxWidgets 2.8.9.2-2.9.5.0.
    - Cross-platform "wxWidgets 2.8.9.2-2.9.5.0" issues service requests to and receives service responses from a "wxWidgets" Virtual Platform Interface which interfaces to the host platform's specific GUI API.

- **"wxPython 3.0.0.0 to 3.0.2.0" Pixel-Mode API**
  - **Python 2x & 3x Application**
    - Pixel-mode cross-platform "wxPython 3.0.0.0-3.0.2.0" classes, methods and functions issue service requests to and receive service responses from the cross-platform "wxWidgets 3.0.0.0-3.0.2.0 Virtual Platform Interface which interfaces to the host platform's specific GUI API.

- **"wxPython 2.8.9.2" Character-Mode Emulation API**
  - **Python 2x & 3x Application**
    - Pixel-style "wxPython 2.8.9.2" classes, methods and functions issue service requests to and receive service responses using a cross-platform Python "Curses" Virtual Platform Interface which interfaces to the host platform's specific GUI API. The character-mode emulation uses "ts" prefix on file names and on internal methods, constants and variables. It otherwise retains same wxPython class, method and function names and parameters.

- **"wxPython 3.0.2.0" Character-Mode Emulation API (*Future*)**
  - **Python 2x & 3x Application**
    - Pixel-style (have "ts" prefix on file names but use same names and parameters) "wxPython 3.0.2".0 classes, methods and functions (have "ts" prefix on file names but use same names and parameters) issue service requests to and receive service responses using a cross-platform Python "Curses" Virtual Platform Interface which interfaces to the host platform's specific GUI API. The character-mode emulation uses "ts" prefix on file names and on internal methods, constants and variables. It otherwise retains same wxPython class, method and function names and parameters.

Draft

Virtual Platform Interface Relationship between "wxPython 2.8.9.2" API Pixel-Mode and Character-Mode Emulation (tsLibGUI *full listing*) (Table of Contents)

- **"wxPython 2.8.9.2" Pixel-Mode API**
  - Pixel-mode "wxPython 2.8.9.2" classes, methods and functions issue service requests to a "wxPython 2.8.9.2" Virtual Platform.
  - "wxPython 2.8.9.2-Host" Virtual Platform components:
    - Convert the "wxPython 2.8.9.2 service requests into the form used by the Host platform's Operating System and GUI.
    - Transmit the converted service requests to the Host Operating.
    - Receive the Host platform's service request response.
    - Convert the Host platform's response into the form used for the "wxPython 2.8.9.2" Virtual Platform's response.
    - Return the converted Host platform's response to "wxPython 2.8.9.2" classes, methods and function.

- **"wxPython 2.8.9.2" Character-Mode Emulation API**
  - Emulation-specific "wxPython 2.8.9.2" classes, methods and functions issue service requests to a "Python Curses-Host" Virtual Platform.
    - Use Python's popular, low-level character-mode "Curses" GUI API and run time library.
    - Convert API argument height and width dimensions from the number of pixels to the number of courier font mono-spaced character cells.
    - Associate mouse clicks with GUI objects by tracking parent-child relationships between objects.
    - Generate 68-color "wxPython 2.8.9.2" color palette when platform permits or else make substitutions from built-in "Curses" 8-/16-color palette.
  - "Python Curses-Host" Virtual Platform components:
    - Convert the "wxPython 2.8.9.2 service requests into the form used by the Python Curses-Host platform.
    - Transmit the converted service requests to the Python Curses-Host.
    - Receive the Python Curses-Host platform's service request response.
    - Convert the Python Curses-Host platform's response into the form used for the "wxPython 2.8.9.2" Virtual Platform's response.
    - Return the converted Python Curses-Host platform's response to "wxPython 2.8.9.2" classes, methods and function.

Draft

Architecture Differences between "wxPython 2.8.9.2" API Pixel-Mode and Character-Mode Virtual Platform Interface (tsLibGUI *full listing*) (Table of Contents)

Application (Python language)

"wxPython 2.8.9.2" API Pixel-Mode

- Issue service requests to and receive responses from "wxWidgets 2.8.9.2" API ("C/C++" language)
  - Issue service requests to and receive responses from "wxWidgets" Virtual Platform Interface API ("C/C++" language)
    - Issue service requests to and receive responses from Host Platform Interface GUI API ("C/C++" language):
      - Linux Host OS GUI API
      - Mac OS X Host GUI API
      - Microsoft Windows Host GUI API
      - Unix Host GUI API

Application (Python language)

"wxPython 2.8.9.2" API Character-Mode Emulation

- Issue service requests to and receive responses from "tsLibGUI" "wxPython 2.8.9.2" API (Python language)
  - Issue service requests to and receive responses from Python "Curses" API (Python language)
    - Issue service requests to and receive responses from Host "Curses" API ("C/C++" language)
      - Linux Host OS GUI API
      - Mac OS X Host GUI API
      - Microsoft Windows Host GUI API
      - Unix Host GUI API
- Issue service requests to and receive responses from internal, "tsLibGUI" Virtual Platform Interface API (Python language) associated with:
  - Generation and mapping of "wxPython" 68-color and non-color palette into available terminal emulator color (8, 16 or 256) or non-color "curses" palette.
  - Generation of "wxPython" event notification by using parent-child relationships to map "curses" mouse clicks to the "wxPython" event triggering and event handling "wxPython" GUI objects.

Draft

Virtual Platform Interface
"wxPython 2.8.9.2" API Building Blocks (tsLibGUI *full listing*) (Table of Contents)

Category: *Module File Name (Class Name)*

- Curses (wxPython 2.8.9.2) Configuration
  - tsWxGlobals.py
- Curses Launcher
  - tsWxMultiFrameEnv.py
- Curses Supervisor
  - tsWxGraphicalTextUserInterface.py
  - tsWxScrollBarButton.py
  - tsWxScrollBarGauge.py
  - tsWxSplashScreen.py
  - tsWxTaskBar.py
- Curses Event Handler
  - tsWxEvtHandler.py
- Curses Utilities
  - tswxPython 2.8.9.2PrivateLogger.py
  - tsWxCursesServices.py
  - tsWxWindowCurses.py

Category: *Module File Name (Class Name)*

- Curses Keyboard
  - tsWxCursesKeyCodesDataBase.py
- Curses Mouse
  - tsWxCursesMouseButtonCodesDataBase.py
- Curses Display
  - tsWxDisplay.py
  - tsWxScreen.py
  - tsWxColorDatabase.py
  - tswxPython 2.8.9.2 Color16DataBase.py
  - tswxPython 2.8.9.2 Color16SubstitutionMap.py
  - tswxPython 2.8.9.2 Color256DataBase.py
  - tswxPython 2.8.9.2 Color88DataBase.py
  - tswxPython 2.8.9.2 Color8DataBase.py
  - tswxPython 2.8.9.2 Color8SubstitutionMap.py
  - tswxPython 2.8.9.2 ColorDataBaseRGB.py
  - tswxPython 2.8.9.2 ColorNames.py
  - tswxPython 2.8.9.2 ColorRGBNames.py
  - tswxPython 2.8.9.2 ColorRGBValues.py
  - tswxPython 2.8.9.2 MonochromeDataBase.py

Draft

"wxPython 2.8.9.2" API Character-Mode Emulation (tsLibGUI *full listing*) (Table of Contents)

- **tsWxFrame**
  - A window whose size and position can (usually) be changed by the user. It usually has thick borders and a title bar, and can optionally contain a menu bar, toolbar and status bar. A frame can contain any window that is not a Frame or Dialog. It is one of the most fundamental of the wxWindows components.
  - A Frame that has a status bar and toolbar created via the CreateStatusBar / CreateToolBar functions manages these windows, and adjusts the value returned by GetClientSize to reflect the remaining size available to application windows.
  - By itself, a Frame is not too useful, but with the addition of Panels and other child objects, it user interfaces are constructed.
- **tsWxDialog**
  - A window with a title bar and sometimes a system menu, which can be moved around the screen. It can contain controls and other windows and is often used to allow the user to make some choice or to answer a question.
  - Dialogs can be made scrollable, automatically, for computers with low resolution screens.
  - Dialogs usually contains either a single button allowing to close the dialog or two buttons, one accepting the changes and the other one discarding them (such button, if present, is automatically activated if the user presses the "Esc" key).

Draft

"wxPython 2.8.9.2" API Character-Mode Emulation (tsLibGUI *full listing*) (Table of Contents)

- **tsWxPanel**
  - A window on which controls are placed. It is usually placed within a frame.
  - Its main feature over its parent class wxWindow is code for handling child windows and TAB traversal.
- **tsWxStatusBar**
  - A narrow window that can be placed along the bottom of a frame to give small amounts of status information.
  - It can contain one or more fields, one or more of which can be variable length according to the size of the window.

Draft

"wxPython 2.8.9.2" API Character-Mode Emulation (tsLibGUI *full listing*) (Table of Contents)

- **wxWxButton**
  - A control that contains a text string. It is one of the most common elements of a GUI. It may be placed on a dialog box or panel, or indeed almost any other window.
- **tsWxCheckbox**
  - A labelled box that by default is either on (the checkmark is visible) or off (no checkmark). Optionally (When the wx.CHK_3STATE style flag is set) it can have a third state, called the mixed or undetermined state. Often this is used as a "Does Not Apply" state.
- **tsWxRadioButton**
  - A labelled box which by default is either on (the checkmark is visible) or off (no checkmark). When the operator turns one RadioButton on, all other Radio Buttons within the same RadioBox group are automatically turned off.
- **tsWxMenuBar**
  - Aa series of menus accessible from the top of a frame.
- **tsWxGauge**
  - A horizontal or vertical bar which shows a quantity (often time). wxGauge supports two working modes: determinate and indeterminate progress. The first is the usual working mode (see SetValue() and SetRange()) while the second can be used when the program is doing some processing but you do not know how much progress is being done. In this case, you can periodically call the Pulse() function to make the progress bar switch to indeterminate mode (graphically it is usually a set of blocks which move or bounce in the bar control).
- **tsWxScrollBar**
  - A control that represents a horizontal or vertical scrollbar. It is distinct from the two scrollbars that some windows provide automatically, but the two types of scrollbar share the way events are received.
- **tsWxTaskBar (buttons)**
  - A narrow window with a taskbar icon (button) for each top level window (frame, dialog etc.). A taskbar icon appears in the "system tray" and responds to mouse clicks, optionally with a tooltip above it to help provide information.

Draft

"wxPython 2.8.9.2" API Character-Mode Emulation (tsLibGUI *full listing*) (Table of Contents)

- **Keyboard**
  - An event class that contains:
    - key identifier
    - key pressed/released state
- **Mouse**
  - An event class that contains:
    - mouse identifier
    - button identifier
    - button state (pressed, released, single-clicked, double-clicked, triple-clicked)
    - current cursor position (display screen character cell row/column coordinates)
- **Timer**
  - An event class to allow you to execute code at specified intervals.
  - Its precision is platform-dependent, but in general will not be better than 1 millisecond nor worse than 1 second.

Draft

"wxPython 2.8.9.2" API Character-Mode Emulation (tsLibGUI *full listing*) (Table of Contents)

- **tsWxBoxSizer**
  - The basic idea behind a box sizer is that windows will most often be laid out in rather simple basic geometry, typically in a row or a column or nested hierarchies of either.
  - A wx.BoxSizer will lay out its items in a simple row or column, depending on the orientation parameter passed to the constructor.

- **tsWxGridSizer**
  - A grid sizer is a sizer which lays out its children in a two-dimensional table with all cells having the same size. In other words, the width of each cell within the grid is the width of the widest item added to the sizer and the height of each grid cell is the height of the tallest item.
  - An optional vertical and/or horizontal gap between items can also be specified (in pixels.)
  - Items are placed in the cells of the grid in the order they are added, in row-major order. In other words, the first row is filled first, then the second, and so on until all items have been added. (If necessary, additional rows will be added as items are added.) If you need to have greater control over the cells that items are placed in then use the wx.GridBagSizer.

Draft

"wxPython 2.8.9.2" API Character-Mode Emulation (tsLibGUI *full listing*) (Table of Contents)

- **tsWxGraphicalTextUserInterface**
  - Class uses the Standard Python Curses API to initialize, manage and shutdown input (from a keyboard and mouse) and output (to a two-dimensional display screen).
  - Uses built-in or builds monochrome or "wxPython 2.8.9.2" 68-color palette for operator designated terminal or terminal emulator (vt100-family or xterm-family).
  - When appropriate, it substitutes available "Curses" 8-/16-colors for those "wxPython 2.8.9.2" 68-colors which are NOT available.
  - When appropriate, it synthesizes a single click "Curses" xterm-family mouse button input from a sequence of available press-release vt100-family mouse button input.
  - Builds platform-specific splash screen from configuration data extracted from txCxGlobals.
- **tsWxSplashScreen**
  - Class to show a window with a thin border, displaying a "bitmap" (text) describing your application. The splash screen is shown during application initialization. The application then either explicitly destroys it or lets it time-out.
  - Became deprecated when tsWxGraphicalTextUserInterface module began building splash screen from configuration data extracted from txCxGlobals.
- **tsWx (Wx)**
  - Module to load all symbols that should appear within the wxPython 2.8.9.2 wx emulation namespace.
  - Includes various classes, constants, functions and methods available for use by applications built with components from the "wxPython 2.8.9.2" emulation infrastructure.
  - The tsWx module is intended to be imported by each Toolkit user applications and Toolkit Test.
- **tsWxGlobals**
  - The tsWxGTUI_PyVx Toolkit emulates the definitions of and references to constant and variable names used by the C++ based "wxWidgets" API and the wxPython 2.8.9.2 interface wrapper used by Python programmers.
  - The tsWxGlobals module and emulated tswxPython 2.8.9.2 classes and methods are then imported by the tsWx module to emulate the "wxPython 2.8.9.2" wx module.

draft

"wxPython 2.8.9.2" API Character-Mode Emulation (tsLibGUI *full listing*) (Table of Contents)

- **tsWxGraphicalTextUserInterface**
  - Logs the following platform configuration information:
    - "Curses" GUI platform configuration constants
      - Keyboard, Mouse, Display
    - "Curses" GUI operator-designated run time configuration
      - Terminal Name / Type
      - Terminal Has Colors
        - Number of Colors
        - Number of Color-Pairs
        - Built-in Color Palette
        - Generated "wxPython 2.8.9.2"-style Color Palette
    - "wxPython 2.8.9.2"-style GUI Object configuration (including parent-child relationship(s), type, position, size, style, color-pair, title, label, border and curses GUI object handle etc.)
  - Logs the following run time event notification information:
    - Equipment Operator event notifications
    - Maintenance / Field Service event notifications
    - Developer Debug event notifications
- **tsWxLog** (Future Log)

Draft

"wxPython 2.8.9.2" API Character-Mode Emulation (tsLibGUI *full listing*) (Table of Contents)

- **tsWxPyApp (PyApp)**
  - PyApp class, based on tsWxEvtHandler, to represent the application and is used to:
    - Bootstrap the "wxPython 2.8.9.2"-style system and initialize the underlying GUI toolkit
    - Set and get application-wide properties
    - Implement the windowing system main message or event loop, and to dispatch events to window instances.
- **tsWxApp (App)**
  - App class, based on tsWxPyApp, to represent the application and is used to do the same things.
- **tsWxPySimpleApp (PySimpleApp)**
  - PySimple is a simple application class, based on tsWxApp.
  - You can just create one of these and then make your top level windows later, and not have to worry about OnInit.
- **tsWxPyOnDemandOutputWindow (PyOnDemandOutputWindow)**
  - PyOnDemandOutputWindow class that can be used for redirecting Python stdout and stderr streams.
  - It will do nothing until something is written to the stream at which point it will create a Frame with a text area and write the text there.
- **tsWxMultiFrameEnv**
  - MultiFrameEnv **c**lass to enable an application using a Command Line Interface (CLI) to launch and use the same character-mode terminal with a Graphical-style User Interface (GUI).
  - It uses application specified configuration keyword-value pair options to initialize any application specific logger(s)
  - It wraps the CLI, underlying the GUI, and the GUI with exception handlers to control the exit codes and messages used to coordinate other application programs.

Draft

“wxPython 2.8.9.2” API Character-Mode Emulation (tsLibGUI *full listing*) (Table of Contents)

- tsWxDisplay.py
- tsWxScreen.py
- tsWxColorDatabase.py
- tswxPython 2.8.9.2 Color16DataBase.py
- tswxPython 2.8.9.2 Color16SubstitutionMap.py
- tswxPython 2.8.9.2 Color256DataBase.py
- tswxPython 2.8.9.2 Color88DataBase.py
- tswxPython 2.8.9.2 Color8DataBase.py
- tswxPython 2.8.9.2 Color8SubstitutionMap.py
- tswxPython 2.8.9.2 ColorDataBaseRGB.py
- tswxPython 2.8.9.2 ColorNames.py
- tswxPython 2.8.9.2 ColorRGBNames.py
- tswxPython 2.8.9.2 ColorRGBValues.py
- tswxPython 2.8.9.2 MonochromeDataBase.py

Draft