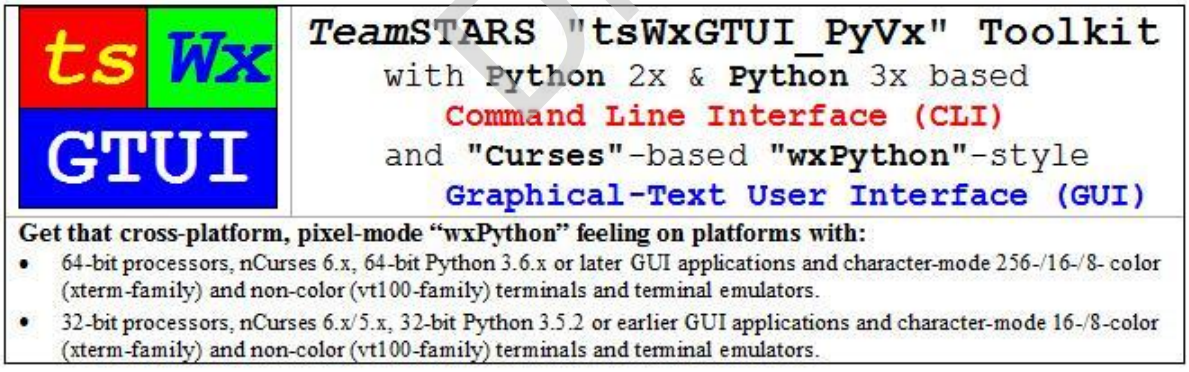
Software Gadgetry

Brochure

Vol. 1 - "tsWxGTUI_PyVx" Toolkit

Rev. 0.1.0 (Pre-Alpha)

Author(s): Richard S. Gordon



Printed on 16 February, 2017

Contents

1 INTRODUCTION

1.1 About (DRAFT)

The *Team*STARS "tsWxGTUI_PyVx" Toolkit's cross-platform Virtual Machine design and implementation supports a broad assortment of user selectable open and proprietary hardware and software components and platforms:

1 TeamSTARS "tsWxGTUI_PyVx-0.1.0" (Pre-Alpha) Toolkit may be run on any of those development and embedded systems which satisfy the following requirements

a) ***Hardware Requirements*** (on page 10)

b) ***Software Requirements*** (on page 18)

2 State-of-the-Art Platforms

With high performance 64-bit processors, nCurses-6.x (a character-mode terminal control library for Unix-like systems) and Python 3.x applications, it provides that cross-platform, pixel-mode "wxPython" feeling on character-mode 8-/16-/256-color (xterm-family) and non-color (vt100-family) terminals and terminal emulators.

a) POSIX-style Command Line Interface features for Python 3.0.1-3.6.x.

b) wxPython-style Graphical-style User Interface features supporting mouse and up to 256-colors with Python 3.6.x.

Computers having Python 3.6.x, nCurses 6.x and 64-bit processors (operating in 64-bit mode) support up to 256-colors with up to 65536 color pairs.

Computers having Python 3.6.x, nCurses 5.x or 6.x and having 32-bit or 64-bit processors (operating in 32-bit compatibility mode) support up to 16-colors with up to 256 color pairs.

3 Legacy Platforms

With medium or low performance 32-bit processors (or 64-bit processors operating in 32-bit compatibility mode), nCurses-5.x/6.x (a character-mode terminal control library for Unix-like systems) and Python 2.x/3.x applications, it provides that cross-platform, pixel-mode "wxPython" feeling on character-mode 8-/16-color (xterm-family) and non-color (vt100-family) terminals and terminal emulators.

a) POSIX-style Command Line Interface features for legacy Python 2.0.1-2.7.13 and transitional Python 3.0.1-3.5.2.

b) wxPython-style Graphical-style User Interface features supporting mouse and up to 16-colors with legacy Python 2.6.4-2.7.13 and transitional Python 3.1.5-3.5.2.

Computers having legacy Python 2.6.4-2.7.13, transitional Python 3.1.5-3.5.2 or Python 3.6.x, nCurses 5.x or 6.x and 32-bit or 64-bit processors (operating in 32-bit compatibility mode) support up to 16-colors with up to 256 color pairs.

Computers having legacy Python 2.6.4-2.7.13, transitional Python 3.1.5-3.5.2 or Python 3.6.x, nCurses 5.x and 64-bit processors (operating in 64-bit mode) support up to 16-colors with up to 256 color pairs.

4 Character-Mode Terminals (https://en.wikipedia.org/wiki/Text-based_user_interface)

Excerpt From Wikipedia, the free encyclopedia:

"From [character-mode] text application's point of view, a text screen (and communications with it) can belong to one of three types (here ordered in order of decreasing accessibility):

a) *A genuine text mode display, controlled by a video adapter or the central processor itself. This is a normal condition for a locally running application on various types of personal computers and mobile devices. If not deterred by the operating system, a smart program may exploit the full power of a hardware text mode.*

b) *A text mode emulator. Examples are xterm for X Window System and win32 console (in a window mode) for Microsoft Windows. This usually supports programs which expect a real text mode display, but may run considerably slower. Certain functions of an advanced text mode, such as an own font uploading, almost certainly become unavailable.*

c) *A remote text terminal. The communication capabilities usually become reduced to a serial line or its emulation, possibly with few ioctl()s as an out-of-band channel in such cases as Telnet and Secure Shell. This is the worst case, because software restrictions hinder the use of capabilities of a remote display device.*

Under Linux and other Unix-like systems, a program easily accommodates to any of the three cases because the same interface (namely, standard streams) controls the display and keyboard. Also, specialized programming libraries help to output the text in a way appropriate to the given display device and interface to it. ...."

5 Terminal Control Library Evolution

Highlights From Wikipedia, the free encyclopedia

a) *"Unix (trademarked as UNIX) is a family of multitasking, multiuser computer operating systems that derive from the original AT&T Unix, developed starting in the 1970s at the Bell Labs research center by Ken Thompson, Dennis Ritchie, and others.*

Initially intended for use inside the Bell System, AT&T licensed Unix to outside parties from the late 1970s, leading to a variety of both academic and commercial variants of Unix from vendors such as the University of California, Berkeley (BSD), Microsoft (Xenix), IBM (AIX) and Sun Microsystems (Solaris). AT&T finally sold its rights in Unix to Novell in the early 1990s, which then sold its Unix business to the Santa Cruz Operation (SCO) in 1995, but the UNIX trademark passed to the industry standards consortium The Open Group, which allows the use of the mark for certified operating systems compliant with the Single UNIX Specification (SUS). Among these is Apple's macOS, which is the Unix version with the largest installed base as of 2014."

b) *"BSD UNIX introduced the first curses terminal control library. It supported standout video highlighting. It supported neither a mouse nor color terminals. Most implementations of BSD Unix curses used a termcap database that can describe the capabilities of thousands of different terminals. The final 4.4 BSD Unix added support for bold and underline video highlighting."*

c) *The AT&T UNIX System III and System V introduced the terminfo database, enhanced highlighting (bold and underline) and line-drawing. It supported a mouse and 8-color terminals."*

d) *"The Free Software Foundation began development of open source nCurses for use in the open source Unix and GNU/Linux compatible operating systems. Its evolution continues to introduce numerous new capabilities and increase performance while still maintaining backward compatibility.*

6 Human Readable Toolkit Source Code

That part of computer software which most users don't ever see;.

a) It's the part computer programmers manipulate to change how a computer "program", "application" or "library" works.

b) Is written in a platform hardware and operating system independent Python programming language that is human readable.

c) Is processed by the platform's hardware and operating system specific Python Virtual Machine.

d) Is included in the release for you to freely use, study, modify and redistribute.

7 Python Virtual Machines

Python Virtual Machines (like operating systems, device drivers, programming libraries, compilers, linkers, loaders and application programs) are expected to be installed by an experienced system administrator before the computer platform is released for others to operate.

a) Python Virtual Machines compile (translate) Python source code into platform independent Virtual Machine readable byte-code. This occurs during application program launch. During program launch, Toolkit programming library component source code will also be translated into byte-code, unless it had been pre-compiled during the building and installation of one or more Python version specific site-packages.

b) Python version specific Virtual Machines execute (interpret) byte-code by invoking the associated platform's hardware and operating system specific library code.

8 Cross-Platform Design Verification and Validation

a) All Python 2x and 3x version specific Virtual Machines have been developed, tested, maintained, validated and released by the Python Software Foundation (PSF). The PSF also develops, tests, maintains, validates and releases equivalent version specific Python 2x and 3x Virtual Machines (and the source code to build them) for non-Intel processors and other operating system releases.

b) All Toolkit Python source code is compiled, interpreted, executed and validated with various Python 2x and 3x Virtual Machines using only 32-bit/64-bit Intel processors and representative GNU/Linux, Mac OS X, Microsoft Windows and Unix/Solaris operating system releases issued by third parties.

9 TeamSTARS "tsWxGTUI_Py2x-0.1.0" (Pre-Alpha) Toolkit Release Provides

a) Command Line Interface features have been developed, tested, maintained, validated and released for Python 2.0.1-2.7.13.

b) Graphical-style User Interface features have been developed, tested, maintained, validated and released for Python 2.6.4-2.7.13.

10 TeamSTARS "tsWxGTUI_Py3x-0.1.0" (Pre-Alpha) Toolkit Release Provides

a) Command Line Interface features which are released for Python 3.0.1-3.5.2 and 3.6.0.

b) Graphical-style User Interface features which are released for Python 3.1.5-3.5.2 and 3.6.0.

1.1.1 What is the Toolkit designed do?

The *Team*STARS "tsWxGTUI_PyVx" Toolkit facilitates the creation of cross-platform computer programs which:

1 ***Monitor and control local and remote mission-critical equipment*** (on page 6)

2 ***Require little, if any, local and remote platform specific modifications*** (on page 9)

1.1.1.1 Monitor and control local and remote mission-critical equipment

The application software which monitors and controls local and remote mission-critical equipment typically share the following features:

1 ***Supervisory Control And Data Acquisition (SCADA)*** (on page 6)

2 ***Operator-friendly User Interfaces*** (on page 6)

3 ***Embedded Systems*** (on page 8)

1.1.1.1.1 Supervisory Control And Data Acquisition (SCADA)

Supervisory Control And Data Acquisition (SCADA) computer programs are often associated with the following applications:

1 Automation (ex. robotics),

2 Communication (ex. network traffic),

3 Control (ex. supervisory and feedback),

4 Diagnostic (ex. hardware and software failure detection and analysis),

5 Instrumentation (ex. data acquisition and signal analysis),

6 Simulation (ex. flight)

1.1.1.1.2 Operator-friendly User Interfaces

Computer programs that must interface with a human operator are often associated with the following user interfaces:

1 ***Command Line-style User Interface (CLI)*** (on page 7)

2 ***Graphical-style User Interface (GUI)*** (on page 8)

1.1.1.1.2.1 Command Line-style User Interface (CLI)

This user interface is inspired by POSIX, the family of Portable Operating System Interface standards specified by the IEEE Computer Society for maintaining compatibility between operating systems. POSIX defines the Application Programming Interface (API), along with command line shells and utility interfaces, for software compatibility with variants of Unix and other operating systems.

The *Team*STARS "tsWxGTUI_PyVx" Toolkit uses each platform's Command-Line Interface (CLI) and Network Communication Interface to monitor and control the local and, if applicable, any remote platform hardware and software.

Application programs may support none, any, or all of these three major types of command line interface mechanisms:

1 Parameters

Most operating systems support a means to pass additional information to a program when it is launched. When a program is launched from an Operating System command line shell, additional text provided along with the program name is passed to the launched program.

The Toolkit's POSIX-compatible CLI supports use of:

a) key-word/value pair options

Example: python -m MODULE_NAME

b) positional arguments

Example: diff ./Python2xVersion/tsLogger.py ./Python3xVersion/tsLogger.py

2 Interactive command line sessions

After launch, a program may provide an operator with an independent means to enter commands in the form of text.

3 OS inter-process communication

Most operating systems support means of inter-process communication (for example; standard streams or named pipes). Command lines from client processes may be redirected to a CLI program by one of these methods.

Example: python tsLinesOfCodeProjectMetrics.py 2> error.log

1.1.1.1.2.2 Graphical-style User Interface (GUI)

This user interface is inspired by those found on televisions and modern computers (main frame, mini computer, workstation, desktop, laptop and smart phone). It allows users to interact with text-based computer terminals through graphical-like icons (character-mode symbols such as "<", ">", "^", "v") and visual indicators (such as side-by-side and overlapping boxes, colors and lines (horizontal "-", vertical "|" and intersecting "+" and corners), instead of text-based user interfaces, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of Command-Line Interfaces (CLIs), which require commands to be typed on a computer keyboard.

The *Team*STARS "tsWxGTUI_PyVx" Toolkit uses the local platform's GUI and CLI to monitor and control the platform's hardware and software. It can also use the available local and remote platform's network communication hardware and software to monitor and control the remote platform's hardware and software.

The Toolkit's character-mode emulation of the pixel-mode "wxPython" Graphical User Interface creates displays on terminals and terminal emulators with:

1 Display Output

The 8-/16-/256-Color (xterm-family) and non-color (vt100-family) displays may contain horizontal and vertical lines, side-by-side and overlapping windows with or without titles, window size and termination control buttons, menu bars, tool bars, scroll bars, status bars, task bars, buttons, checkboxes, radio boxes & buttons, gauges, date and time stamped event messages, text with or without color and intensity markup, and other GUI objects. Displays may be laid out with or without the help of box and grid sizer services. The basic 8-/16-Color capabilities require at least a 32-bit processor. The advanced 256-Color capabilities require at least a 64-bit processor.

a) alpha & beta release --- 8-bit US-ASCII characters (containing a repertoire of numbers 0 to 9, lowercase letters a to z, uppercase letters A to Z, basic punctuation symbols and control codes)

b) future release --- up to four 8-bit UNICODE characters (containing a repertoire of more than 128,000 characters covering 135 modern and historic scripts, as well as multiple symbol sets)

2 Keyboard Input

Input is submitted by the operator via a terminal keyboard device, The keyboard consists of a matrix of push buttons. The operator may press a letter, digit, punctuation symbol or function key. The use may also simultaneously press and hold a combination of the shift, control or alt keys.

3 Mouse, Trackball, Touchpad or Touchscreen Input

Input of GUI object selections is submitted by the operator via a terminal mouse device. A touchscreen, trackpad or trackball device may be substituted for a mouse device.

The input consists of the cursor position data needed to identify the selected GUI-object, the button identity (left, middle, right) and the type of selection (pressed and held, released, single click, double click and triple click).

1.1.1.1.3 Embedded Systems

The computer hardware and software which provides the intelligence needed to safely and effectively operate high-cost electrical, mechanical, chemical and hazardous equipment is often found in the following mission-critical applications:

1 Commercial (ex. building energy management),

2 Industrial (ex. power generation),

3 Medical (ex. CAT-scan) or

4 Military (ex. weapon control)

1.1.1.2 Require little, if any, local and remote platform specific modifications

The *Team*STARS "tsWxGTUI_PyVx" Toolkit is designed to require little, if any, local and remote platform specific modifications to accommodate:

1 Python programming language versions and releases

2 Hardware (processor, memory and input/output) make, model and performance features

3 Software (operating system, device driver, programming library and application program) make, model, release and performance features

1.1.2 What is the Toolkit NOT designed to do?

The *Team*STARS "tsWxGTUI_PyVx" Toolkit is NOT designed to do the following:

1 ***Directly Monitor and Control a Pixel-mode Graphical User Interface*** (on page 9)

2 ***Recognize and Respond to "Hot" or "Shortcut" Keys*** (on page 10)

3 ***Run Without Extensive Modification on Long Obsolete Platforms*** (on page 10)

4 ***Run Pre-compiled Machine Executable Code on Incompatible Platforms*** (on page 10)

1.1.2.1 Directly Monitor and Control a Pixel-mode Graphical User Interface

The *Team*STARS "tsWxGTUI_PyVx" Toolkit is NOT designed to directly monitor and control a pixel-mode display instead of a text-based character-mode display.

If a local or remote platform has a display terminal that physically operates in graphics-based pixel-mode, there must be an interfacing text-based terminal window application between the *Team*STARS "tsWxGTUI_PyVx" Toolkit and the display terminal:

1 "Terminal" on Linux, Mac OS X and Unix:

An operator selectable monospaced character-mode 8-color/64-color pair terminal emulator ("xterm") with press-release and multi-click mouse buttons

An operator selectable monospaced character-mode 16-color/256-color pair terminal emulator ("xterm-16color") with press-release and multi-click mouse buttons

An operator selectable monospaced character-mode 1-color/2-color pair terminal emulator ("vt100" and "vt220") with press-release and emulated single-click mouse buttons

2 "Cygwin Terminal" on Microsoft Windows (requires installation and use of "Cygwin", the free Linux-like plugin from Red Hat)

1.1.2.2 Recognize and Respond to "Hot" or "Shortcut" Keys

The *Team*STARS "tsWxGTUI_PyVx" Toolkit "alpha" stage releases are NOT designed to recognize and respond to any key or sequence of keys ("Hot" or "Shortcut" Keys) that when pressed might launch the command to perform the task normally associated with the use of the required Mouse button (or optional Trackball, Touchpad or Touchscreen button).

1.1.2.3 Run Without Extensive Modification on Long Obsolete Platforms

The *Team*STARS "tsWxGTUI_PyVx" Toolkit is NOT designed to easily back port to platforms whose hardware, software and documentation was discontinued long ago and is no longer readily available (such as DEC Alpha processor, BE Operating System and Python 1.6.1 Virtual Machine, Interpreter and Library).

Reflecting the evolution of computer technology, today's hardware and software platforms and Python programming language, virtual machines and run time libraries are significantly different from their predecessors.

Because of the level of effort to reverse-engineer, design, build and test a customized *Team*STARS "tsWxGTUI_PyVx" Toolkit, there needs to be sufficient justification before undertaking a back port of the Toolkit.

1.1.2.4 Run Pre-compiled Machine Executable Code on Incompatible Platforms

Once the human readable source code for the *Team*STARS "tsWxGTUI_PyVx" Toolkit has been compiled into machine executable code for a specific make, model and release of hardware (processor) and software (operating system) platform, the resulting machine executable code will only be usable on a compatible hardware (processor) and software (operating system) platform. Components within the following categories are typically NOT compatible with architecturally and functionally dissimilar components from another manufacturer:

1. Hardware --- Any proprietary Central Processing Unit (such as those having the Intel x86/x86_64 Complex Instruction Set computing architecture; the Motorola 68000 Complex Instruction Set Computing architecture or the IBM PowerPC Reduced Instruction Set Computing architecture).
2. Software --- Any program or application designed for an open source Operating System (such as GNU/Linux, FreeBSD and OpenSolaris) and proprietary Operating Systems (such as Apple's macOS (previously named Mac OS X), Microsoft's Windows and Oracle/Sun Microsystem's Solaris).

1.1.3 Hardware Requirements

Toolkit operation depends on the availability of the following hardware features:

1. ***Central Processing Unit (CPU)*** (on page 11)
2. ***Random Access Memory (RAM)*** (on page 11)
3. ***Non-volatile Memory (HDD or SSD)*** (on page 11)
4. ***Display (LCD or CRT)*** (on page 12)

5 ***Keyboard (KBD)*** (on page 16)

6 ***Mouse, Trackball, Touchpad or Touchscreen*** (on page 16)

7 ***Printer (PRN or LPT)*** (on page 16)

8 ***Input/Output Ports (IO)*** (on page 17)

9 ***Network Interface Unit (NIU)*** (on page 17)

10 ***Wireless Computer Networking (WiFi)*** (on page 18)

1.1.3.1 Central Processing Unit (CPU)

The Central Processing Unit required features:

1 Either a 32-bit or 64-bit data register width. Optionally, it may have one or more cores.

2 Usable with the Python 2x and/or Python 3x programming language.

During initial Toolkit development and testing, a single core 366MHz Pentium II was sufficient for Microsoft Windows XP and Ubuntu Linux 12.04 platforms.

1.1.3.2 Random Access Memory (RAM)

The Random Access Memory must have sufficient capacity for transient data and program storage during execution. The amount is both operating system and application specific.

During initial Toolkit development and testing, 384 MB was sufficient for Microsoft Windows XP and Ubuntu Linux 12.04 platforms.

1.1.3.3 Non-volatile Memory (HDD or SSD)

The Non-volatile Memory must have sufficient capacity for data and program storage during power shutdowns or outages. The amount is both operating system and application specific.

1 Popular and readily available computer systems typically come with one or more Hard Disk Drives (HDD) featuring an electro-mechanical mechanism and magnetic storage media. The primary characteristics of an HDD are its capacity, performance, form factor and interfaces.

   a) Capacity is specified in unit prefixes corresponding to powers of 1000: a 1-terabyte (TB) drive has a capacity of 1,000 gigabytes (GB; where 1 gigabyte = 1 billion bytes).

   b) Performance is specified by the time required to move the heads to a track or cylinder (average access time) plus the time it takes for the desired sector to move under the head (average latency, which is a function of the physical rotational speed in revolutions per minute), and finally the speed at which the data is transmitted (data rate).

   c) The two most common form factors for modern HDDs are 3.5-inch, for desktop computers, and 2.5-inch, primarily for laptops.

   d) HDDs are connected to systems by standard interface cables such as PATA (Parallel ATA), SATA (Serial ATA), USB or SAS (Serial attached SCSI) cables.

2 Growing in popularity and availability is the Solid State Drive (SSD) featuring electronic flash memory which has higher data transfer rates, higher reliability, and significantly lower latency and access times. SSDs are replacing HDDs where speed, power consumption and durability are more important considerations.

During initial Toolkit development and testing, 32 GB hard disk was more than ample for Microsoft Windows XP and Ubuntu Linux 12.04 platforms.

1.1.3.4 Display (LCD or CRT)

The Display is an electronic computer output surface and projecting mechanism that shows text and often graphic images to the computer user.

Consider the features of a basic "VGA" display (operating systems often provide the user with the means to improve image fit and readability by adjusting terminal window size and by overriding default font type and size).

1 Liquid Crystal Display (LCD)

Currently popular and readily available this kind of display features a thin flat panel that uses the light modulating properties of Liquid Crystals. Liquid Crystals do not emit light directly but modulate how much light passes through each picture element (pixel) from a bright, uniform background light source (typically white, green or orange in color).

Monochrome "VGA" displays must create an image that is 640 pixels wide by 480 pixels tall using only black and the background color (or in varying shades of it by modulating the applied voltage so that the intensity of each pixel produces 256 shades).

Color "VGA" displays must use three subpixels with red, green and blue color filters to create each color pixel. Combining the 256 shades of red, green and blue subpixels produces a possible pixel palette of 16.8 million (256 x 256 x 256) colors.

2 Cathode Ray Tube (CRT)

Once popular and readily available, this kind of display features an evacuated glass envelope (picture tube) which is large, deep (i.e. long from front screen face to rear end), fairly heavy, and relatively fragile. As a matter of safety, the face is typically made of thick lead glass so as to be highly shatter-resistant and to block most X-ray emissions, particularly if the CRT is used in a consumer product.

a) Monochrome "VGA" displays must create an image that is 640 pixels wide by 480 pixels tall using only the intensity of the electron beam. By modulating the applied voltage, the intensity of each pixel produces 256 shades of the phosphorescent screen coating (typically white, green or orange).

b) Color "VGA" displays must use three subpixels (three adjacent phosphorescent screen coatings in red, green and blue) to create each color pixel. Combining the 256 shades of red, green and blue subpixels produces a possible pixel palette of 16.8 million (256 x 256 x 256) colors.

The glass tube for monochrome contains one electron gun and the one for color contains three guns and the means to accelerate (modulate the intensity) and deflect the electron beam(s) onto the screen to create the image.

The entire front area of the glass tube is scanned repetitively and systematically in a fixed pattern called a raster. An image is produced by controlling the intensity of each of the three electron beams, one for each additive primary color (red, green, and blue) with a video signal as a reference. In all modern CRT monitors, the beams are bent by magnetic deflection, a varying magnetic field generated by coils and driven by electronic circuits around the neck of the tube.

CRTs have been largely superseded by newer display technologies such as LCD, plasma display, and OLED, which have lower manufacturing costs, power consumption, weight and bulk.

1.1.3.4.1 Display for Development of Documentation and Software

Writing, drawing, editing, publishing and Web research activities typically require a Graphical User Interface or GUI. This is a type of interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of Command-line Interfaces (CLIs), which require commands to be typed on the keyboard.

The actions in a GUI are usually performed through direct manipulation of the graphical elements. In addition to computers, GUIs can be found in hand-held devices such as smart phones and tablet computers.

Using an 8 (9 columns per inch) x 12 (6 lines per inch) pixel "Courier New" font, the basic 12" diameter "VGA" pixel-mode multi-font graphics display supports at least 80 col x 40 row (640 x 480 pixels) and 16.8 million colors.

Any one of the following display types may be substituted for the "VGA" display based on the user's need to perform various activities while concurrently viewing multiple software engineering documents and files.

The following table is derived from "https://en.wikipedia.org/wiki/Display_resolution":

Acronym	(usage)	Aspect ratio	Width (pixels)	Height (pixels)
VGA	(12" CRT)	4:3	640	480
SVGA	(14" CRT)	4:3	800	600
XGA	(15" Laptop)	4:3	1024	768
XGA+		4:3	1152	864
WXGA		16:9	1280	720
WXGA		5:3	1280	768
WXGA		16:10	1280	800
SXGA-	(UVGA)	4:3	1280	960
SXGA		5:4	1280	1024
HD		~16:9	1360	768
HD		~16:9	1366	768
SXGA+		4:3	1400	1050
WXGA+	(17" Laptop)	16:10	1440	900
HD+		16:9	1600	900
UXGA		4:3	1600	1200
WSXGA+		16:10	1680	1050
FHD		16:9	1920	1080
WUXGA		16:10	1920	1200
QWXGA		16:9	2048	1152
WQHD	(27" Desktop)	16:9	2560	1440
WQXGA		16:10	2560	1600

1.1.3.4.2 Display for Monitoring and Control of Mission Critical equipment

For those systems with:

1 A character-mode display adapter

Console-like terminal application programs output text directly via the display adapter.

2 A pixel-mode graphical display adapter

Console-like terminal application programs output text to a Curses-like Terminal Interface Control library which outputs to a character-mode terminal emulator which outputs directly via the display adapter.

A character-mode single font text display (whose size could be somewhere between that of a 3" smart phone and a 12" tablet) or optional pixel-mode graphical display (the same one used for ***Display for Development of Documentation and Software*** (on page 14)) that supports the application (or its splash screen) and industry standard terminal emulators:

1 **Legacy Systems** with 32-bit processors and AT&T System V Release 4.0 (SVr4) Curses or BSD 4.4 Curses:

a) vt100 (1-color ON/OFF, 2-color-pair NORMAl/REVERSE)

b) vt220 (1-color ON/OFF, 2-color-pair NORMAl/REVERSE)

c) xterm (8-color, 64-color pairs)

2 **Readily Available Systems** with 32-bit processors (or 64-bit processors running in 32-bit emulation mode) and nCurses 5.x/6.x:

a) vt100 (1-color ON/OFF, 2-color-pair NORMAl/REVERSE)

b) vt220 (1-color ON/OFF, 2-color-pair NORMAl/REVERSE)

c) xterm-16color (16-color, 256-color-pairs)

d) xterm-88color (16-color, 256-color-pairs)

e) xterm-256color (16-color, 256-color-pairs)

3 **Cutting Edge (State of the Art) Systems** with 64-bit processors and nCurses 6.x:

The Python Software Foundation has not announced plans and a road map for 64-bit nCurses 6.x support.

a) vt100 (1-color ON/OFF, 2-color-pair NORMAl/REVERSE)

b) vt220 (1-color ON/OFF, 2-color-pair NORMAl/REVERSE)

c) xterm (8-color, 64-color pairs)

d) xterm-16color (16-color, 256-color-pairs)

e) xterm-88color (88-color, 7744-color-pairs)

f) xterm-256color (181-color, 32761-color-pairs)

Applications may be as simple as a single frame window. A frame requires:

1 A title line with optional window control buttons;

2 Additional optional lines for a menu bar, tool bar, status bar, buttons, check boxes, radio boxes & buttons, gauges and scrolled text.

3 The display size for this is 35 col x 16 row (280 x 200 pixels).

More complex applications involve multiple side-by-side and overlapping frames which may optionally be arranged in a desktop consisting of:

1. A collection of application frame and dialog windows;
2. A scrolling operator event notification window ;
3. A task bar whose buttons enable the operator to raise hidden frames and dialogs from the invisible background to the visible foreground. For the sample splash screen, the display size can range from 60 col x 25 row (480 x 300 pixels) to over 80 col x 50 row (640 x 600 pixels).

1.1.3.5 Keyboard (KBD)

The Keyboard is an electro-mechanical panel of alpha-numeric, punctuation and shift keys such as used to operate a computer or typewriter. It must also include the standard ctrl, alt, del and function keys such as used to operate a typical laptop or desktop computer.

1.1.3.6 Mouse, Trackball, Touchpad or Touchscreen

The Mouse, Trackball, Touchpad or Touchscreen is a hand-operated electro-mechanical device that controls the coordinates of a cursor on the display as it is moved around.

1. The cross-platform, industry standard mouse or Trackball has:
   a) Two button (left & right)
   b) An optional scroll wheel, which can also act as a third (middle) button
2. The optional Touchpad or Touchscreen has:
   a) Software that can recognize one and two finger gestures such as tap, drag and scroll

1.1.3.7 Printer (PRN or LPT)

The optional printer is an electro-mechanical device which allows a user to print items on paper.

Marketed by numerous manufacturers, there are three types of printers:

1. An "impact" printer is a computer peripheral that strikes a print head against an ink ribbon to mark the paper. Common examples include dot matrix and daisy-wheel printers.
2. An "inkjet" printer is a computer peripheral that produces hard copy by spraying ink onto paper. A typical inkjet printer can produce copy with a resolution of at least 300 dots per inch (dpi). Some inkjet printers can make full color hard copies at 600 dpi or more.
3. A "laser" printer is a computer peripheral that produces good-quality printed material by using a laser to form a pattern of electrostatically charged dots on a light-sensitive drum, which attracts black (and sometimes red-yellow-blue) toner (or dry ink powder). The toner is transferred to a piece of paper and fixed by a heating process.

1.1.3.8 Input/Output Ports (IO)

The Input/Output ports must have sufficient plug-in connections for optional peripheral equipment. The amount is both operating system and application specific.

During initial Toolkit development and testing, 2 USB 2.0 ports were sufficient for Microsoft Windows XP and Ubuntu Linux 12.04 platform printer and backup disk connections.

During subsequent Toolkit development and testing, one or more of the following I/O Ports were used with Linux, Mac OS X, Microsoft Windows 10, Solaris and Unix platforms:

(1) Universal Serial Bus

USB 1.0 --- 1.5 Mbit/s (Low Bandwidth or Low Speed)

USB 2.0 --- 480 Mbit/s (High Speed or High Bandwidth)

USB 3.1 --- 10 Gbit/s (SuperSpeed)

(2) IEEE 1394 (FireWire)

FireWire 400 (IEEE 1394-1995) --- 100, 200, or 400 Mbit/s half-duplex

FireWire 800 (IEEE 1394b-2002) --- 786.432 Mbit/s full-duplex

(3) External Serial Advanced Technology Attachment (eSATA)

1.1.3.9 Network Interface Unit (NIU)

The Network Interface Unit is an optional electronic device that serves as a common interface (IEEE-802.3) for various other devices within a local area network (LAN), or as an interface to allow networked computers to connect to an outside network. A network interface card (NIC) is a type of NIU.

During initial Toolkit development and testing 10BASE-T (10 Mbps)/100BASE-T (100 Mbps) was sufficient for Microsoft Windows XP and Ubuntu Linux 12.04 platforms.

During subsequent Toolkit development and testing, one or more 1000BASE-T (1,000 Mbps) network interfaces were used with Linux, Mac OS X, Microsoft Windows 10, Solaris and Unix platforms.

1.1.3.10 Wireless Computer Networking (WiFi)

The Wi-Fi (or WiFi) is an optional local area wireless computer networking "WLAN" technology (IEEE-802.11) that allows electronic devices to network, mainly using the 2.4 GHz UHF and 5 GHz SHF ISM radio bands.

Many devices can use Wi-Fi, e.g. personal computers, smartphones and tablet computers. These can connect to a network resource such as the Internet via a wireless network access point. Such an access point (or hotspot) has a range of about 20 meters (66 feet) indoors and a greater range outdoors. Hotspot coverage can be as small as a single room with walls that block radio waves, or as large as many square kilometers achieved by using multiple overlapping access points.

During initial Toolkit development and testing 11 Mbps was sufficient for Microsoft Windows XP and Ubuntu Linux 12.04 platforms.

1.1.4 Software Requirements

Toolkit operation depends on the availability of the following software features:

1. ***Computer Operating System*** (on page 18)
2. ***Terminal Control Library*** (on page 19)
3. ***Python Interpreter / Python Virtual Machine*** (on page 21)

1.1.4.1 Computer Operating System

A multi-user (for local and remote access), multi-process (for interacting with multiple applications) and multi-threaded (for sharing platform resources) operating system such as:

1. GNU/Linux on Intel (x86 and x86_64) and Advanced Micro Devices (amd64) and other architectures

   Its POSIX-compatible CLI is provided by the GNOME Terminal, KDE Konsole, XTerm and UXTerm applications and a shell such as BASH.

   Its Curses-based GUI is provided by the ncurses terminal control library for Unix-like systems.

2. macOS/Mac OS X (Darwin Unix-based operating system) on Apple+IBM+Motorola alliance (PowerPC) and on Intel (x86 and x86_64) and Advanced Micro Devices (amd64) architectures

   Its POSIX-compatible CLI is provided by the GNOME Terminal and iTerm applications and a shell such as BASH.

   Its Curses-based GUI is provided by the ncurses terminal control library for Unix-like systems.

3. Microsoft Windows on Intel (x86 and x86_64) and Advanced Micro Devices (amd64) architectures with "Cygwin", the free GNU/Linux-like plug-in from Red Hat.

   (NOTE: Its DOS-like CLI is provided by the Microsoft "Command Prompt" and "Power Shell" terminal applications which CANNOT support a Curses-based GUI.)

   Its POSIX-compatible CLI is provided by the Cygwin Mintty ("Terminal") application and a shell such as BASH.

Its Curses-based GUI is provided by the ncurses terminal control library for Unix-like systems.

4 Unix on Apple+IBM+Motorola alliance (PowerPC), HP-UX (PA-RISC), IBM-AIX (RS/6000), Intel (x86 and x86_64), Advanced Micro Devices (amd64) , IRIX (SGI/MIPS), Solaris (Sun/SPARC) and other architectures

   Its POSIX-compatible CLI is provided by the GNOME Terminal and KDE Konsole applications and a shell such as BASH.

Its Curses-based GUI is provided by the curses or ncurses terminal control library for Unix-like systems.Except as noted below, The *Team*STARS "tsWxGTUI_PyVx" Toolkit is known to work with keyboard and mouse input under:

1 Linux (CentOS 7, Debian 8/8.5, Fedora 19-24, OpenSuSE 13.1, Scientific 7 and Ubuntu 12.04-16.04) using "ncurses 5.x/6.x" library, "Terminal" application with non-color (vt100, vt220), 8-color/64-color pair (xterm, xterm-color), 16-color/256-color pair xterm-16color, xterm-88color and xterm-256color) terminals/terminal emulators

2 macOS/Mac OS X (10.3-10.12) using "ncurses 5.x/6.x" library, third-party "iTerm2" application with non-color (vt100, vt220), 8-color/64-color pair (xterm, xterm-color), 16-color/256-color pair (xterm-16color, xterm-88color and xterm-256color) terminals/terminal emulators

3 Microsoft Windows (XP, 7, 8, 8.1 and 10 but GUI-style mode requires Cygwin, free Linux-like plugin from Red Hat) using "ncurses 5.x/6.x" library, "mintty" Terminal application with non-color (vt100, vt220), 8-color/64-color pair (xterm, xterm-color), 16-color/256-color pair (xterm-16color, xterm-88color and xterm-256color) terminals/terminal emulators

4 Unix (PC-BSD 10, OpenIndiana 151a8, OpenSolaris 11) using "curses" or "ncurses 5.x/6.x" library, "Terminal" application with non-color (mouseless vt100 and vt220), 8-color/64-color pair (xterm, xterm-color), 16-color/256-color pair (xterm-16color, xterm-88color and xterm-256color) terminals/terminal emulators

1.1.4.2 Terminal Control Library

To maintain upgradeability to "NCurses 6.0" and simplicity of user installation, configuration, application programming and troubleshooting, the TeamSTARS "tsWxGTUI_PyVx" Toolkit supports only the "Curses" text-mode Terminal Control Library from the Python Global Module Index.

It does NOT support Third Party alternatives to "Curses" such as "PDCurses", a public-domain "Curses" programming library implemented in the "C" programming language for "Microsoft DOS", "IBM OS/2", "Microsoft Windows", "X11" and "Simple DirectMedia Layer" and "UniCurses", a wrapper for Python 2.x/3.x that provides a unified set of "Curses" functions on all platforms ("Microsoft Windows", "GNU/Linux", "Mac OS X" and "Unix") with syntax close to that of the original "Curses" / "NCurses". To provide functionality on "Microsoft Windows", "UniCurses" wraps "PDCurses".

A cross-platform, industry-standard library of functions that manage an application's character-mode text display on character-cell terminals:

1 CURSES

The traditional library available on Unix-like systems. It provides a platform-independent Application Programming Interface (API) that enables the same application source code to work with proprietary hardware. Features (such as vt100/vt220 mouse and xterm-16color support) introduced with ncurses are not necessarily available.

a) BSD 4.4 Version Unix supports:

   Computers having 16-bit processors (supports up to 8-colors with up to 64 color pairs).

b) AT&T System V Release 4.0 (SVr4) Version Unix supports:

   Computers having 32-bit processors (supports up to 16-colors with up to 256 color pairs).

2 NCURSES (new Curses)

The updated library available on Linux-like systems. (including the Cygwin plug-in for Microsoft Windows). It provides a platform-independent Application Programming Interface (API) that enables the same application source code to work with proprietary hardware.

a) Version 5.x supports:

   Computers having 32-bit (or 64-bit processors operating in 32-bit compatibility mode) support up to 16-colors with up to 256 color pairs.

b) Version 6.x supports:

   Computers having 32-bit (or 64-bit processors operating in 32-bit compatibility mode) support up to 16-colors with up to 256 color pairs.

   Computers having 64-bit processors operating in 64-bit mode (support up to 256-colors with up to 65536 color pairs).

3 Industry Standard Terminal Emulators:

A terminal emulator, terminal application, term, or tty for short, is a program that emulates a video terminal within some other display architecture. Though typically synonymous with a shell or text terminal, the term terminal covers all remote terminals, including graphical interfaces. A terminal emulator inside a graphical user interface is often called a terminal window.

A terminal window allows the user access to a text terminal and all its applications such as command-line interfaces (CLI) and text user interface (TUI) applications. These may be running either on the same machine or on a different one via telnet, ssh, or dial-up. On Unix-like operating systems, it is common to have one or more terminal windows connected to the local machine.

Terminals usually support a set of escape sequences for controlling color, cursor position, etc. Examples include the family of terminal control sequence standards known as ECMA-48, ANSI X3.64 or ISO/IEC 6429.

Cross-platform applications require at least the following popular, readily available terminal emulators:

xterm	(8-color, 64-color pairs)
xterm-color	(8-color, 64-color pairs)s)
xterm-16color	(16-color, 256-color-pairs)
xterm-88color	(88-color, 7744-color-pairs)
xterm-256color	(256-color, 65536-color-pairs)

vt100 (1-color ON/OFF, 2-color-pair NORMAl/REVERSE)

vt220 (1-color ON/OFF, 2-color-pair NORMAl/REVERSE)

1.1.4.3 Python Interpreter / Python Virtual Machine

One or more cross-platform, industry-standard Python programing languages and associated Interpreter and Virtual Machine.

The *Team*STARS "tsWxGTUI_PyVx" Toolkit can be installed and will operate, without change, on any 32-bit or 64-bit computer processor and operating system that supports official Python releases from the Python Software Foundation. Though its compatibility has not been evaluated with other Python implementations, it is likely to be compatible with many of those listed here:

- ***Wikipedia's Python Implementation List*** (***https://en.wikipedia.org/wiki/Python_(programming_language)***)
- ***Wikipedia's ActiveState ActivePython Products*** (***https://en.wikipedia.org/wiki/ActiveState***)

The Toolkit's common User Interface and Application Programming Interface supports two implementations:

1. The Python 2x generation offers only legacy features, capabilities and limitations on historically available hardware are software platforms. The final 2.x version 2.7 release came out in mid-2010, with a statement of extended support for this end-of-life release. The 2.x branch will see no new major releases after that.
   a) Command Line Interface features are released for Python 2.0.1-2.7.13.
   b) Graphical-style User Interface features are released for Python 2.6.4-2.7.13.
2. The Python 3x generation offers enhanced features, capabilities and limitations. Python 3.x is under active development and has already seen over five years of stable releases, including version 3.3 in 2012, 3.4 in 2014, 3.5 in 2015 and 3.6.0 in 2016. This means that all recent standard library improvements, for example, are only available by default in Python 3.x.
   a) Command Line Interface features are released for Python 3.0.1-3.5.2 and 3.6.0
   b) Graphical-style User Interface features are released for Python 3.1.5-3.5.2 and 3.6.0.

      Computers having Python 3.6.x, nCurses 6.x and 64-bit processors (operating in 64-bit mode) support up to 256-colors with up to 65536 color pairs.

      Computers having Python 3.1.5-3.5.2 or Python 3.6.0, nCurses 5.x or 6.x and having 32-bit processors (or operating in 32-bit compatibility mode) support up to 16-colors with up to 256 color pairs.
3. The Python source code is pre-compiled, upon application launch, for the platform's processor and operating system by either the operating system or "Cygwin" plug-in manufacturer.
4. The Python building block library component source code is automatically compiled into platform independent byte-code during "site-package" installation or else upon Python application launching.

1.1.5 What is included in the release?

Toolkit development, maintenance, enhancement, porting, troubleshooting and user training depends on the availability of those release features described in the following sections:

1 ***Multi-Project Release*** (on page 22)

2 ***Consistent User Interface*** (on page 23)

3 ***Toolkit Components*** (on page 23)

1.1.5.1 Multi-Project Release

Unlike other "GitHub" repositories which contain a single project, the one for the *Team*STARS "tsWxGTUI_PyVx" Toolkit is organized into four collections of project-specific computer program source code files that the Toolkit recipient will need to install, operate, modify, port and re-distribute the Toolkit.

1 Site-Packages

These two projects are intended to be installed as a registered Python site-package (one for the mature Python 2x programming language and the other for the evolving Python 3x programming language).

Local or remote applications that have imported the appropriate Python 2x or Python 3x "site-package" can be launched from any convenient directory on the associated local or remote computer system.

Modifying a copy of one of these is the most direct way to port the Toolkit to a currently unsupported Python 1x, 2x or 3x platform.

a) Python-2x ("tsWxGTUI_Py2x")

b) Python-3x ("tsWxGTUI_Py3x")

2 Developer-Sandboxes

These two projects are NOT intended to be installed as a registered Python site-package (one for the mature Python 2x programming language and the other for the evolving Python 3x programming language).

Local or remote Python 2x or Python 3x applications can only be launched from the associated "tsWxGTUI_Py2x" or "tsWxGTUI_Py3x" developer-sandbox directory.

Modifying a copy of one of these is the least painful way to experiment with alternative software architectures and algorithms.

a) Python-2x ("tsWxGTUI_Py2x")

b) Python-3x ("tsWxGTUI_Py3x")

1.1.5.2 Consistent User Interface

The four *Team*STARS "tsWxGTUI_PyVx" Toolkit projects are released as a set so that (despite their Python 2x and Python 3x implementation differences) they retain the identical Application Programming Interface (API) and look & feel of their User Interfaces (UI):

1 ***Command Line User Interface (CLI)*** (see "***Command Line-style User Interface (CLI)***" on page 7)

2 ***Graphical User Interface (GUI)*** (see "***Graphical-style User Interface (GUI)***" on page 8)

1.1.5.3 Toolkit Components

The *Team*STARS "tsWxGTUI_PyVx" Toolkit include the following components:

1 **Documents**

The directory contains a collection of files which provide the Toolkit recipient with an understanding of the purpose, goals (capabilities), non-goals (limitations), terms & conditions and procedures for installing, operating, modifying and redistributing the Toolkit.

2 **Manual Pages**

The directory contains a collection of files which provide a form of online software documentation usually found on a Linux or Unix-like operating systems.

Topics covered include computer programs (library and system calls), formal standards and conventions, and even abstract concepts.

3 **Notebooks**

The directory contains a collection of files which provide commentaries that express opinions or offerings of explanations about events or situations that might be useful to Toolkit installers, developers, operators, troubleshooters and distributors.

Microsoft PowerPoint Presentation (with Adobe PDF copy) highlighting the context and most notable details of the Toolkit's Introduction, Project, Release and Use Case documents. It can be found via the path:

"**./Notebooks/EngineeringNotebook**":

"**./MS-PowerPoint-Files/tsWxGTUI_PyVx**"

"./**Adobe-PDF-Files/Adobe-PDF-Presentation-Files/tsWxGTUI_PyVx**"

4 **SourceDistributions**

The directory contains a collection of source code files organized by category:

a) Python programming language (Python 2x and Python 3x)

b) Operating Mode (Command Line Interface and Graphical User Interface)

c) Function (Building Block Libraries, Tools, Tests, Utilities and Examples)

d) Installation (registered Python "Site-Package" and non-registered Python "Developer-Sandbox")

1.1.6 How might you use the Toolkit?

The *Team*STARS "tsWxGTUI_PyVx" Toolkit can save you time by eliminating the need to re-invent, organize and integrate a collection of general purpose, re-usable software building block libraries, tools, tests, utilities and examples.

Here are a few usage situations and associated benefits:

1 ***Port your existing "curses" software*** (on page 24)

2 ***Port your existing "wxPython" or "wxWidgets" software*** (on page 24)

3 ***Adapt existing Toolkit software*** (on page 24)

4 ***Create new software from scratch*** (on page 25)

1.1.6.1 Port your existing "curses" software

Adapt your software so that an executable program can be created for computing environments that are different from the one(s) for which it was originally designed (e.g. different CPU, operating system, or third party library).

You should consider porting when the cost of porting it to a new platform is less than the cost of writing it from scratch. The lower the cost of porting software, relative to its implementation cost, the more portable it is said to be.

1 Existing GUI applications implemented in the low level "c" programming language which use the low level, character-mode "curses" terminal device interface library Application Programming Interface (API) are substantially more costly (in effort) to develop, maintain and enhance than those implemented in higher level "Python" with its somewhat higher level "curses" API.

2 Existing GUI applications implemented in the high level "Python" programming language which use the character-mode "curses"-based emulation of the high level pixel-mode "wxPython" API are substantially less costly (in effort) to develop, maintain and enhance.

1.1.6.2 Port your existing "wxPython" or "wxWidgets" software

1 Existing GUI applications implemented in the high level "Python" programming language which use the high level pixel-mode "wxPython" API will become portable after removal of those API activities involving icons and curved shapes.

2 Existing GUI applications implemented in the higher level "c++" programming language which use the high level, pixel-mode "wxWidgets" API will become portable after porting to "wxPython and removal of those API activities involving icons and curved shapes.

1.1.6.3 Adapt existing Toolkit software

1 Find an application among the Toolkit's various CLI and GUI tools, tests and tutorial examples having the structure and user interface closest to your needs.

2 Modify a copy of the Toolkit application to suit your needs.

1.1.6.4 Create new software from scratch

Creating new software from scratch involves iteration(s) through planning, requirement specification, system and component architecture design, documentation, development, integration, test and release.

The *Team*STARS "tsWxGTUI_PyVx" Toolkit's source code and various "Engineering Notebook" documents provided a starting point, road map and foundation to build upon.

1. Review the *Team*STARS "tsWxGTUI_PyVx" Toolkit's source code and various "Engineering Notebook" documents
   a) Review alternatives with Toolkit selected hardware technologies, products, patents and licensing
   b) Review alternatives with Toolkit selected software technologies, products, patents, copyrights and licensing
   c) Review alternatives with Toolkit selected goals, non-goals, functional and interface requirement specifications, system architecture, design and implementation changes
2. Research, gather and select candidate state-of-art changes since the *Team*STARS "tsWxGTUI_PyVx" Toolkit development began
   a) Hardware technologies, products, patents and licensing
   b) Software technologies, products, patents, copyrights and licensing
3. Tryout selected candidate state-of-art changes since the *Team*STARS "tsWxGTUI_PyVx" Toolkit development began
   a) Modify copies of the *Team*STARS "tsWxGTUI_PyVx" Toolkit "Engineering Notebook" documents for each Toolkit candidate state-of-art change
   b) Review modified copies of the *Team*STARS "tsWxGTUI_PyVx" Toolkit "Engineering Notebook" documents and select the candidate Toolkit state-of-art change to adopt
4. Implement the adopted candidate Toolkit state-of-art change

1.1.7 Platform Example Configurations

To maintain upgradeability to "NCurses 6.0" and simplicity of user installation, configuration, application programming and troubleshooting, the TeamSTARS "tsWxGTUI_PyVx" Toolkit supports only the "Curses" text-mode Terminal Control Library from the Python Global Module Index.

It does NOT support Third Party alternatives to "Curses" such as "PDCurses", a public-domain "Curses" programming library implemented in the "C" programming language for "Microsoft DOS", "IBM OS/2", "Microsoft Windows", "X11" and "Simple DirectMedia Layer" and "UniCurses", a wrapper for Python 2.x/3.x that provides a unified set of "Curses" functions on all platforms ("Microsoft Windows", "GNU/Linux", "Mac OS X" and "Unix") with syntax close to that of the original "Curses" / "NCurses". To provide functionality on "Microsoft Windows", "UniCurses" wraps "PDCurses".

1 Software Development System Configuration

A 12" to 60+" pixel-mode multi-font graphics display that supports at least 80 col x 40 row (640 x 480 pixels) and 16,777,216 colors and industry standard terminal emulators.

a) xterm (8-color, 64-color pairs)

b) xterm-16color (16-color, 256-color-pairs)

c) vt100 (1-color ON/OFF, 2-color-pair NORMAl/REVERSE)

d) vt220 (1-color ON/OFF, 2-color-pair NORMAl/REVERSE)

2 Embedded System Configuration

Applications may be as simple as a single frame window. A frame requires: 1) a title line with optional window control buttons; and 2) additional optional lines for a menu bar, tool bar, status bar, buttons, check boxes, radio boxes & buttons, gauges and scrolled text. The display size for this is 35 col x 16 row (280 x 200 pixels).

More complex applications involve multiple side-by-side and overlapping frames which may optionally be arranged in a desktop consisting of: 1) a collection of application frame and dialog windows; 2) a scrolling operator event notification window ; and 3) a taskbar whose buttons enable the operator to raise hidden frames and dialogues from the invisible background to the visible foreground. For the sample splash screen, the display size can range from 60 col x 25 row (480 x 300 pixels) to over 80 col x 50 row (640 x 600 pixels).

A graphic and/or character-mode single font text display (whose size could be somewhere between that of a 3" smart phone and a 12" tablet) that supports the application (or its splash screen) and industry standard terminal emulators:

a) xterm (8-color, 64-color pairs)

b) xterm-16color (16-color, 256-color-pairs)

c) vt100 (1-color ON/OFF, 2-color-pair NORMAl/REVERSE)

d) vt220 (1-color ON/OFF, 2-color-pair NORMAl/REVERSE)

3 Mouse or Trackball for Software Engineering Workstation and Embedded System Operator

4 Computer Operating System

A multi-user (for local and remote access), multi-process (for interacting with multiple applications) and multi-threaded (for sharing platform resources) operating system such as:

a) GNU/Linux on Intel (x86, x86_64 and amd64) and other architectures

Its POSIX-compatible CLI is provided by the GNOME Terminal, KDE Konsole, XTerm and UXTerm applications and a shell such as BASH.

Its Curses-based GUI is provided by the ncurses terminal control library for Unix-like systems.

b) macOS/Mac OS X (Darwin Unix-based operating system) on Apple+IBM+Motorola alliance (PowerPC) and Intel (x86, x86_64 and amd64) architectures

Its POSIX-compatible CLI is provided by the GNOME Terminal and iTerm applications and a shell such as BASH.

Its Curses-based GUI is provided by the ncurses terminal control library for Unix-like systems.

c) Microsoft Windows on Intel (x86, x86_64 and amd64) architectures with "Cygwin", the free GNU/Linux-like plug-in from Red Hat.

(NOTE: Its DOS-like CLI is provided by the Microsoft Command Prompt terminal application which CANNOT support a Curses-based GUI.)

Its POSIX-compatible CLI is provided by the Cygwin Mintty (Terminal) application and a shell such as BASH.

Its Curses-based GUI is provided by the ncurses terminal control library for Unix-like systems.

d) Unix on Apple+IBM+Motorola alliance (PowerPC), HP-UX (PA-RISC), IBM-AIX (RS/6000), Intel (x86, x86_64 and amd64), IRIX (SGI/MIPS), Solaris (Sun/SPARC) and other architectures

Its POSIX-compatible CLI is provided by the GNOME Terminal and KDE Konsole applications and a shell such as BASH.

Its Curses-based GUI is provided by the curses or ncurses terminal control library for Unix-like systems.

5 Terminal Control Library

A cross-platform, industry-standard library of functions that manage an application's character-mode text display on character-cell terminals:

a) curses

The traditional library available on Unix-like systems. It provides a platform-independent Application Programming Interface (API) that enables the same application source code to work with proprietary hardware. Features (such as vt100/vt220 mouse and xterm-16color support) introduced with ncurses are not necessarily available.

b) ncurses (new curses)

The updated library available on Linux-like systems. (including the Cygwin plug-in for Microsoft Windows). It provides a platform-independent Application Programming Interface (API) that enables the same application source code to work with proprietary hardware.

6 Python Interpreter / Python Virtual Machine

One or more cross-platform, industry-standard Python programing languages and associated Interpreter and Virtual Machine.

The *Team*STARS "tsWxGTUI_PyVx" Toolkit is implementation in both the mature Python 2x and evolving Python 3x interpreted programming languages.

a) It is precompiled for the platform's processor and operating system by either the operating system or "Cygwin" plug-in manufacturer.

b) It automatically compiles source code into platform independent byte-code during "site-package" installation or else upon Python application launching.

1.1.7.1 Professional Workstation and Guest “Embedded” System

1 2013 27" Apple iMac Desktop

a) 3.5 GHz Intel Quad Core i7 processor

b) 16 GB RAM

c) 27" 2560x1440 pixel LED display

d) 3 TB (7200 RPM) SATA 6 Gb/s internal hard drive with 128 GB Solid State Flash memory

e) Ethernet Network Adapter

f) WiFi Wireless Network Adapter

2 Development / Embedded Software

a) MAC OS X 10.12 Sierra

b) Wing IDE 5

c) LibreOffice

d) Microsoft Office 365 for Mac 15.3

e) XEmacs

3 Guest (non-optimized) Embedded Software

a) Parallels Desktop 12 Hypervisor for running GuestOS:

Linux (Centos 7.2, Debian 8.5, Fedora 25, OpenSuSE 13.2, Scientific 7.2 & Ubuntu 16.04 LTS & 16.10) with Wing IDE 5, LibreOffice and XEmacs

Microsoft Windows (XP, 7, 8, 8.1 & 10) with Wing IDE 5, AuthorIt-5, Office 2002, Office 365 & XEmacs

Unix (FreeBSD 11/PC-BSD 11, OpenIndiana 151a8 & OpenSolaris 11) with LibreOffice and Xemacs

b) VMware Fusion 7 Hypervisor for running GuestOS:

Linux (OpenSuSE 13.1)

Microsoft Windows (2000)

1.1.7.2 Professional Laptop and Guest “Embedded” System

1 2007 Apple 17" MacBook Pro

a) 2.33 GHz Intel Core 2 Duo processor

b) 4 GB RAM

c) 17" 1920x1200 pixel LED display

d) 160 GB (5400 RPM) SATA 1.5 Gb/s internal hard drive

e) 1.5 TB (7200 RPM) SATA 3 Gb/s external hard drive

f) Ethernet Network Adapter

g) WiFi Wireless Network Adapter

2 Development / Embedded Software

a) MAC OS X 10.7.5 Lion

b) Wing IDE 3-4

c) LibreOffice

d) XEmacs

3 Guest (non-optimized) Embedded Software

a) Parallels Desktop 8 Hypervisor for running GuestOS:

Linux (Fedora 20 32-bit, OpenSuSE 12.2 32-bit, Scientific (CentOS) 6.4-6.5 64-bit, Ubuntu 12.04 32-bit) with Python 2.7 and 3.2 with Wing IDE 3, LibreOffice and XEmacs

Microsoft Windows (XP, 7, 8 & 8.1 each with Cygwin 1.7.8) with Wing IDE 3, AuthorIt-5, Office 2002 & XEmacs

Unix (PC-BSD 9.2-10.0, OpenIndiana 151a3 & OpenSolaris 11) with LibreOffice and Xemacs

b) VMware Fusion 5 Hypervisor for running GuestOS:

Linux (OpenSuSE 13.1)

Microsoft Windows (3.1)

1.1.7.3 Budget Laptop and Pseudo "Embedded" System

1 1998 Dell 15.6" Inspiron 7000

a) 366 MHz Intel Pentium II processor

b) 384 MB RAM

c) 15.6" VGA (640x480) / SVGA (1024x768) pixel LED display

d) Two Interchangeable 32 GB (4200 RPM) ATA hard drives

Microsoft Windows XP

Ubuntu Linux 12.04 LTS

e) Xircom Ethernet and 3Com WiFi Wireless Plug-in Network adapters for Microsoft Windows XP

f) Linksys WiFi Wireless Plug-in Network adapter for Ubuntu Linux 12.04 LTS

2 Development / Pseudo (non-optimized) Embedded Software

a) Microsoft Windows XP

Cygwin 1.7

Office 2002

XEmacs

b) Ubuntu Linux 12.04 LTS

LibreOffice

Xemacs

1.1.8 Where to get further information?

Additional information is available:

1. Learn more about the *Team*STARS "tsWxGTUI_PyVx" Toolkit by browsing its collection of documents, manpages, notebooks (engineering) and source code (building block libraries, tools, tests and examples) at "**https://github.com/rigordo959/tsWxGTUI_PyVx_Repository**":

   a) For those who are initially only curious and seek more specifics about the features, capabilities and limitations of the *Team*STARS "tsWxGTUI_PyVx" Toolkit building blocks and applications, there are Microsift PowerPoint presentation slide shows and printable copies which can be found via the paths:

   "**./Notebooks/EngineeringNotebook/MS-PowerPoint-Files/tsWxGTUI_PyVx**"
   "**./Notebooks/EngineeringNotebook/Adobe-PDF-Files/Adobe-PDF-Presentation-Files/tsWxGTUI_PyVx**"

   b) For those seriously contemplating using the *Team*STARS "tsWxGTUI_PyVx" Toolkit and seek details (for applicable technologies, requirements, design, implementation, test and troubleshooting) the engineering notebooks can be found at:

   "**https://github.com/rigordo959/tsWxGTUI_PyVx_Repository/Notebooks/EngineeringNotebook**"

2. Get your own copy of the Toolkit repository (including its records of comments, issue tracking and revisions) via:

   "**git clone https://github.com/rigordo959/tsWxGTUI_PyVx-Repository**"

1.2 System Block Diagrams (Brochure)

1 ***Block Diagram*** (on page 34) - High level depiction of the organizational, functional and interface relationship between the *Team*STARS "tsWxGTUI_PyVx" Toolkit components and users (System Administrator, Software Engineer, System Operator and Field Service personnel):

a) A Command Line Interface ("tsToolkitCLI") - It provides the foundation for the Graphical User Interface ("tsToolkitGUI). Its components include:

tsApplicationPkg --- Base class to initialize and configure the application program launched by an operator. It enables an application launched via a Command Line Interface (CLI) to initialize, configure and use the same character-mode terminal with a Graphical-style User Interface (GUI).

tsCommandLineEnvPkg --- Class to initialize and configure the application program launched by an operator. It delivers those keyword-value pair options and positional arguments specified by the application, in its invocation parameter list. It wraps the Command Line Interface application with exception handlers to control exit codes and messages that may be used to coordinate other application programs.

tsCommandLineInterfacePkg --- Class establishes methods that prompt or re-prompt the operator for input, validate that the operator has supplied the expected number of inputs and that each is of the expected type.

t**sCxGlobalsPkg** --- Module to establish configuration constants and macro-type functions for the Command Line Interface mode of the "tsWxGTUI_PyVx" Toolkit. It provides a centralized mechanism for modifying/restoring those configuration constants that can be interrogated at runtime by those software components having a "need-to-know". The intent being to avoid subsequent manual searches to locate and modify or restore a constant appropriate to the current configuration. It also provides a theme-based mechanism for modifying/restoring those configuration constants as appropriate for various users and their activities.

tsDoubleLinkedListPkg --- Class to establish a representation of a linked list with forward and backward pointers.

tsExceptionPkg --- Class to define and handle error exceptions. Maps run time exception types into 8-bit exit codes and prints associated diagnostic message and traceback info.

tsLoggerPkg --- Class that emulates a subset of Python logging API. It defines and handles prioritized, time and date stamped event message formatting and output to files and devices. Files are organized in a date and time stamped directory named for the launched application. Unix-type devices include syslog, stderr, stdout and stdscr (the ncurses display screen). It also supports "wxPython"-style logging of assert and check case results.

tsOperatorSettingsParserPkg --- Class to parse the command line entered by the operator of an application program. Parsing extracts the Keyword-Value pair Options and Positional Arguments that will configure and control the application during its execution.

tsPlatformRunTimeEnvironmentPkg --- Class to capture current hardware, software and network information about the run time environment for the user process.

tsReportUtilityPkg --- Class defining methods used to format information: date and time (begin, end and elapsed), file size (with kilo-, mega-, giga-, tera-, peta-, exa-, zeta- and yotta-byte units) and nested Python dictionaries.

tsSysCommandsPkg --- Class definition and methods for issuing shell commands to and receiving responses from the host operating system.

b) A Graphical-style User Interface ("tsToolkitGUI") - It uses the services of the "tsToolkitCLI" when appropriate.

tsWxPkg --- Collection of approximately 100 Classes that use the services of the Python Curses module to create a character-mode emulation of their pixel-mode "wxPython" Class counterparts.

t**sWxGlobals** --- Module provides a centralized mechanism for modifying/restoring those configuration constants that can be interrogated at runtime by those software components having a "need-to-know". The intent being to avoid subsequent manual searches to locate and modify or restore a constant appropriate to the current configuration. It also provides a theme-based mechanism for modifying/restoring those configuration constants as appropriate for the character-mode emulation of the following pixel-mode "wxPython" features:

The collection includes widgets for frames, dialogs, panels, buttons, check boxes, radio boxes/buttons and scrollable text windows. It includes box and grid sizers.

It also includes classes to emulate the host operating system theme-based color palette management, task bar, scroll bar, mouse click and window focus control services used/expected by "wxPython".

The Application Programming Interface (API) retains those "wxPython" keyword-value pairs and positional arguments needed for pixel-mode application compatibility. It adds keyword-value pairs and positional arguments needed only for internal (non-application) Toolkit use.

2 ***Stand Alone System Architecture*** (on page 35) - High-level depiction and description of the components of and relationship between components of an isolated system operating by itself.

3 ***Stand Among System Architecture*** (on page 39) - High-level depiction and description of the components of and relationship between two or more networked systems operating in collaboration with each other.

1.2.1 Block Diagram

This Block Diagram depicts the organizational, functional and interface relationship between the *Team*STARS "tsWxGTUI_PyVx" Toolkit components and users (System Administrator, Software Engineer, System Operator and Field Service personnel):

1 the external System Operator interface to "tsToolkitCLI"

2 the internal "tsToolkitCLI" interface to "tsToolkitGUI"

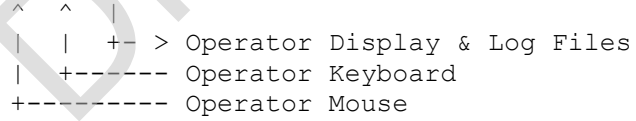
3 the internal System Operator interfaces:

a) to "tsUtilities", "tsToolsCLI" and "tsTestsCLI" via "tsToolkitCLI"

b) to "tsToolsGUI" and "tsTestsGUI" via "tsToolkitCLI" and "tsToolkitGUI"

Graphical-style User Interface (tsToolkitGUI)	
The "tsToolsGUI" is a set of graphical-style user interface application programs for tracking software development metrics.	The "tsTestsGUI" is a set of graphical-style user interface application programs for regression testing and tutorial demos.
The "tsLibGUI" is a library of graphical-style user interface building blocks that establishes the emulated run time environment for the high-level, pixel-mode, "wxPython" GUI Toolkit via the low-level, character-mode, "nCurses" Text User Interface Toolkit.	

Command Line Interface (tsToolkitCLI)	
The "tsToolsCLI" is a set of command line interface application programs and utility scripts for: checking source code syntax and style via "plint"; generating Unix-style "man page" documentation from source code comments via "pydoc"; and installing, modifying for publication, and tracking software development metrics.	The "tsTestsCLI" is a set of command line interface application programs and scripts for regression testing and tutorial demos.
The "tsLibCLI" is a library of command line building blocks that establishes the POSIX-style, run time environment for pre-processing source files, launching application programs, handling events (registering events with date, time and event severity annotations) and configuring console terminal and file system input and output.	
The "tsUtilities" is a library of computer system configuration, diagnostic, installation, maintenance and performance tool components for various host hardware and software platforms.	



1.2.2 Stand Alone System Architecture

The *Team*STARS "tsWxGTUI_PyVx" Toolkit provides:

1. Python version-specific components for its Command Line Interface ("tsToolkitCLI")

2. Python version-specific components for its Graphical-style User Interface ("tsToolkitGUI").

The following description uses the component names as depicted in the ***Block Diagram*** (on page 34)

This section depicts and describes the organization, function of and interface between various system hardware and software components and "tsWxGTUI_PyVx" Toolkit users (System Administrator, Software Engineer, System Operator and Field Service personnel):

1 the external System Operator interface to "tsToolkitCLI"

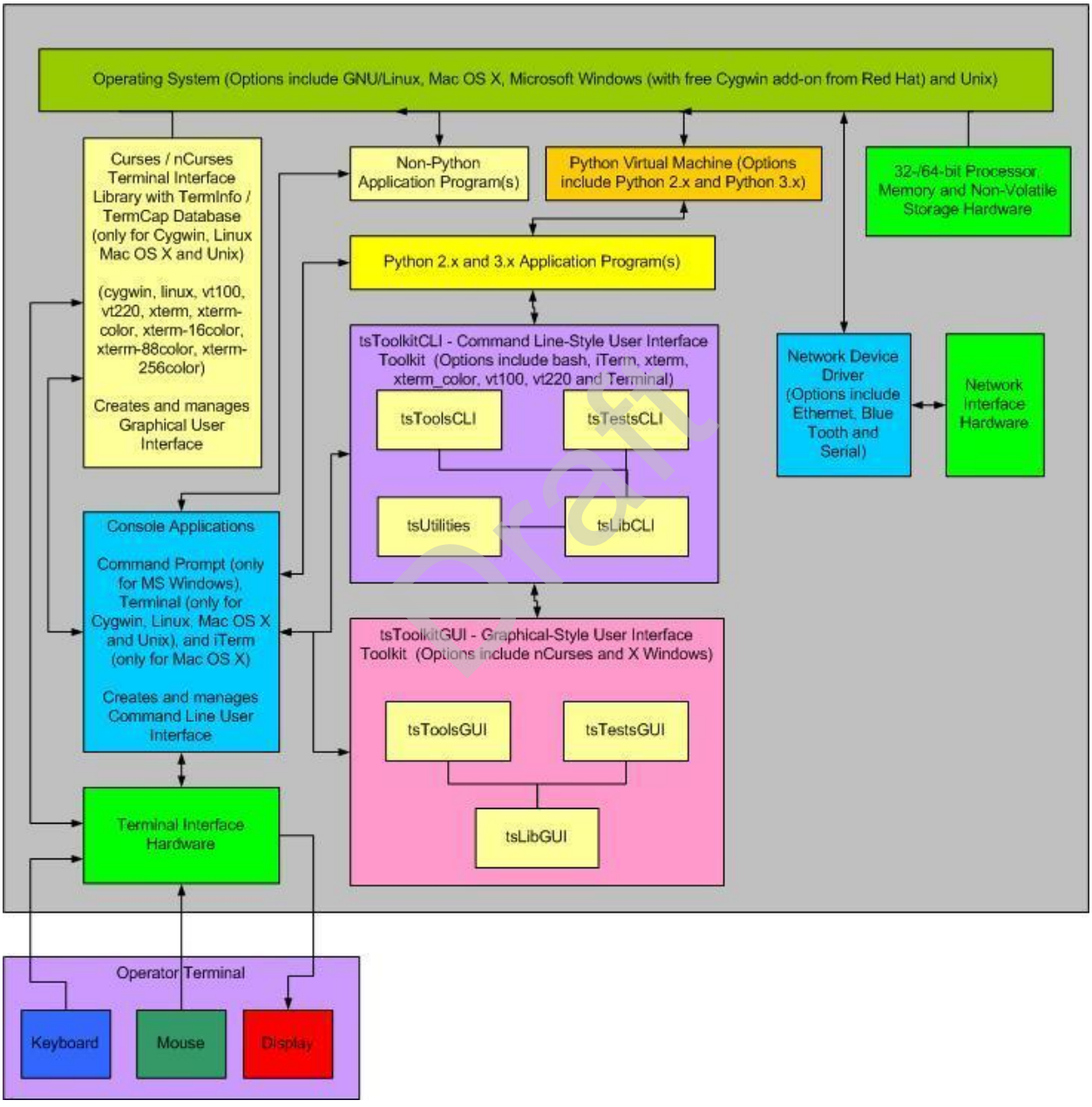
2 the internal "tsToolkitCLI" interface to "tsToolkitGUI"

3 the internal System Operator interfaces:

a) to "tsLibCLI", "tsToolsCLI" . "tsTestsCLI" and "tsUtilities" via "tsToolkitCLI"

b) to "tsLibGUI", "tsToolsGUI" and "tsTestsGUI" via "tsToolkitGUI" and "tsToolkitCLI"

This depiction represents a typical Stand Alone System configuration. In this configuration, the optional Network Hardware Interface and its associated Network Device Driver Interface should not be used, even if present, in order to avoid activities that adversely impact system performance.



1 Operating System - The platform specific software (such as Linux, MacOS X, Microsoft Windows and Unix) that coordinates and manages the time-shared use of a platform's processor, memory, storage and input/output hardware resources by multiple application programs and their associated users/operators.

2 **Operator Terminal** - A device for human interaction that includes:

a) A Keyboard unit for text input

b) A Mouse unit (mouse, trackball, trackpad or touchscreen with one or more physical or logical buttons) for selecting one of many displayed GUI objects to initiate an associated action.

c) A Display unit (1-color "ON"/"OFF" or multi-color two-dimensional screen) for output of text and graphic-style, tiled and overlaid boxes.

3 **Terminal Hardware Interface** - The platform specific hardware with connections to the device units of the Operator Terminal.

a) A PS/2 Port is a type of input port developed by IBM for connecting a mouse or keyboard to a Personal Computer. It supports a mini DIN plug containing just 6 pins.

b) An RS-232C or RS-422 port for connecting a mouse for position and button-click input.

c) A Universal Serial Bus (USB) port is an industry standard first developed in the mid-1990s that was designed to standardize the connection of computer peripherals (including keyboards, pointing devices, digital cameras, printers, portable media players, disk drives and network adapters) to personal computers, both to communicate and to supply electric power. It has effectively replaced a variety of earlier interfaces, such as serial and parallel ports, as well as separate power chargers for portable devices.

The USB 1.0 specification was introduced in January 1996. It defined data transfer rates of 1.5 Mbit/s "Low Speed" and 12 Mbit/s "Full Speed". The first widely used version of USB was 1.1 (now called "Full-Speed"), which was released in September 1998. Its 12 Mbit/s data rate was intended for higher-speed devices such as disk drives, and its lower 1.5 Mbit/s rate for low data rate devices such as joysticks.

The USB 2.0 (now called "Hi-Speed") specification was released in April 2000. It defined a higher data transfer rate, with the resulting specification achieving 480 Mbit/s, a 40-times increase over the original USB 1.1 specification.

The USB 3.0 specification (now called "SuperSpeed") was released in November 2008. It defined an even higher data transfer rate (up to 5 Gbit/s) and was backwards-compatible with USB 2.0. It added a new, higher speed bus called SuperSpeed in parallel with the USB 2.0 bus.

d) A Video Adapter is a computer circuit card that provides digital-to-analog conversion, video RAM, and a video controller so that data can be sent to a computer's display. It typically adheres to the de facto standard, Video Graphics Array (VGA). VGA describes how data - essentially red, green, blue data streams - is passed between the computer and the display. It also describes the frame refresh rates in hertz. It also specifies the number and width of horizontal lines, which essentially amounts to specifying the resolution of the pixels that are created. VGA supports four different resolution settings and two related image refresh rates. The maximum VGA resolution setting produces a display that is 640 pixels wide by 480 pixels high. For a character font that is 8 pixels wide by 12 pixels high, the longest line of text with be 80 characters wide and there can be up to 40 lines of text displayed at any moment. Higher resolutions, such as SVGA, are supported by more advanced Video Adapters. The higher resolution settings typically require use of proportionally larger displays in order to maintain the size and legibility of the displayed text.

4 **Terminal Device Driver** - The platform specific software for transforming data (such as single button scan codes, multi-button flags and pointer position) to and from the platform independent formats (such as upper and lower case text, display screen column and row and displayed colors, fonts and special effects) used by the Command Line Interface and Graphical User Interface software.

5 **Command Line-Style Interface ("tsToolKitCLI")** - The platform specific keywords arguments, positional arguments and their associated values and syntax of text used to request services from the Operating System and various Application Programs.

   a) **"tsLibCLI"** --- A library of command line building blocks that establishes the POSIX-/Unix-style, run time environment for pre-processing source files, launching application programs, handling events (registering events with date, time and event severity annotations) and configuring console terminal and file system input and output.

   b) **"tsToolsCLI"** --- A set of command line interface application programs and utility scripts for: checking source code syntax and style via "plint"; generating Unix-style "man page" documentation from source code comments via "pydoc"; and installing, modifying for publication and tracking software development metrics.

   c) **"tsTestsCLI"** --- A set of command line interface application programs and utility scripts for unit, integration and system level regression testing.

   d) **"tsUtilities"** --- A library of computer system configuration, diagnostic, installation, maintenance and performance tool components for various host hardware and software platforms.

6 **Graphical-Style User Interface ("tsToolKitGUI")** - The platform specific tiled, overlaid and click-to-select Frames, Dialogs, Pull-down Menus, Buttons, CheckBoxes, Radio Buttons, Scrollbars and associated keywords, values and syntax of text used to request services from the Operating System and various Application Programs.

   a) **"tsLibGUI"**--- A library of graphical-style user interface building blocks that establishes the emulated run time environment for the high-level, pixel-mode, "wxPython" GUI Toolkit via the low-level, character-mode, "nCurses" Text User Interface Toolkit.

   b) **"tsToolsGUI"**--- A set of graphical-style user interface application programs for tracking software development metrics.

   c) **"tsTestsGUI"** --- A set of graphical-style user interface application programs and command line interface utility scripts for unit, integration and system level regression testing.

7 **Python Application Program** - The application specific program that performs its service when executed by the Python Virtual Machine.

8 **Non-Python Application Program** - The application specific program that performs its service when its pre-compiled, platform specific machine code is executed. Typically, these services are used to analyze, edit, view, copy, move or delete those data and log files which are of interest or no longer needed.

9 **Python Virtual Machine** - The platform specific program that loads, interprets and executes the platform independent source code of a Python language application program.

10 **Processor, Memory, Storage and Communication Hardware** - Platform specific resources that are required by the Operating System and Application software.

11 **Network Hardware Interface** - The optional platform specific ethernet, blue-tooth and RS-232 serial port hardware for physical connections between the local system and one or more remote systems. It may also include such external hardware as gateways, routers, network bridges, switches, hubs, and repeaters. It may also include hybrid network devices such as multilayer switches, protocol converters, bridge routers, proxy servers, firewalls, network address translators, multiplexers, network interface controllers, wireless network interface controllers, modems, ISDN terminal adapters, line drivers, wireless access points, networking cables and other related hardware.

a) An RJ-45 Ethernet port connecting to a local or wide area network. This port is capable of conducting simultaneous (full-duplex,) two-directional input and output at speeds over 100 gigabits per second. This port can also provide the interface to an optional printer shared with other network users.

b) An RS-232C or RS-422 port for connecting a modem. Depending on the application, modems could be operated, at speeds over 56 kilobits per second, in either of two modes. In half duplex mode, each side alternated its sending and receiving roles. In full-duplex mode, each side could simultaneously send and receive.

12 Network Device Driver Interface - The optional platform specific software whose layered protocol suite (such as TCP/IP) enables the concurrent sharing of the physical connection between the local system and one or more remote systems.

NOTE: Concurrent local and remote login sessions are a convenience that must be used with caution. Time critical, resource intensive activities ought to be performed individually or sequentially (but NOT concurrently) on a Stand Alone System.

1.2.3 Stand Among System Architecture

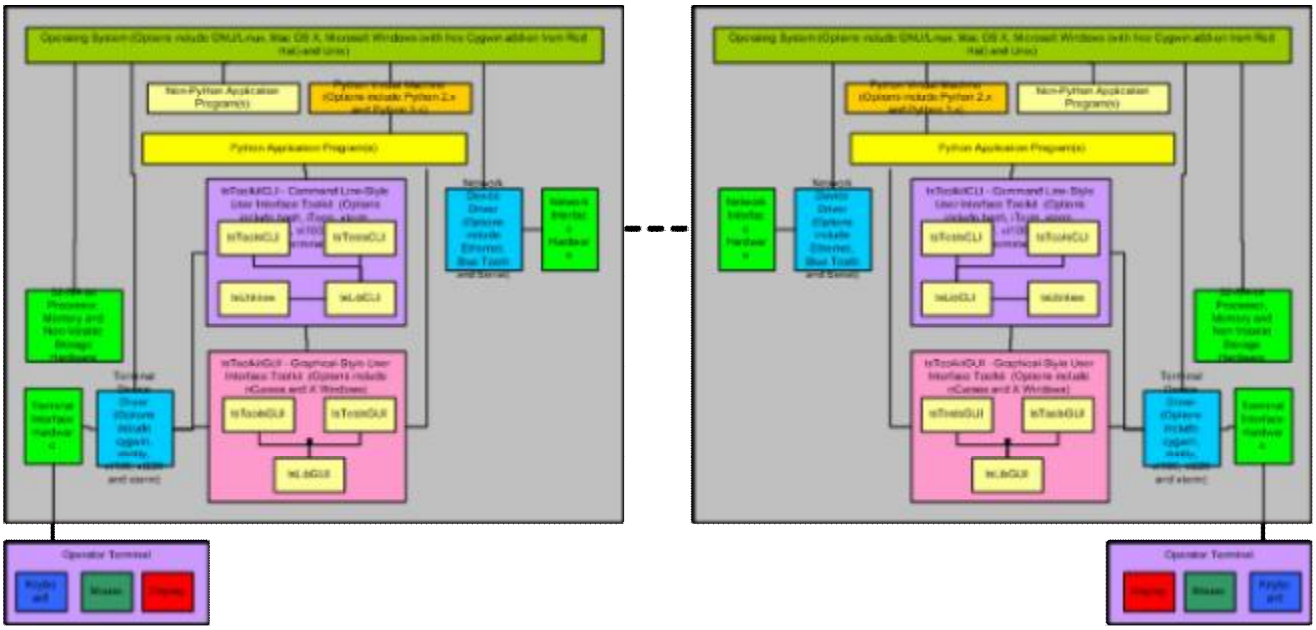
The *Team*STARS "tsWxGTUI_PyVx" Toolkit provides:

1. Python version-specific components for its Command Line Interface ("tsToolkitCLI")

2. Python version-specific components for its Graphical-style User Interface ("tsToolkitGUI").

The following description uses the component names as depicted in the ***Block Diagram*** (on page 34)

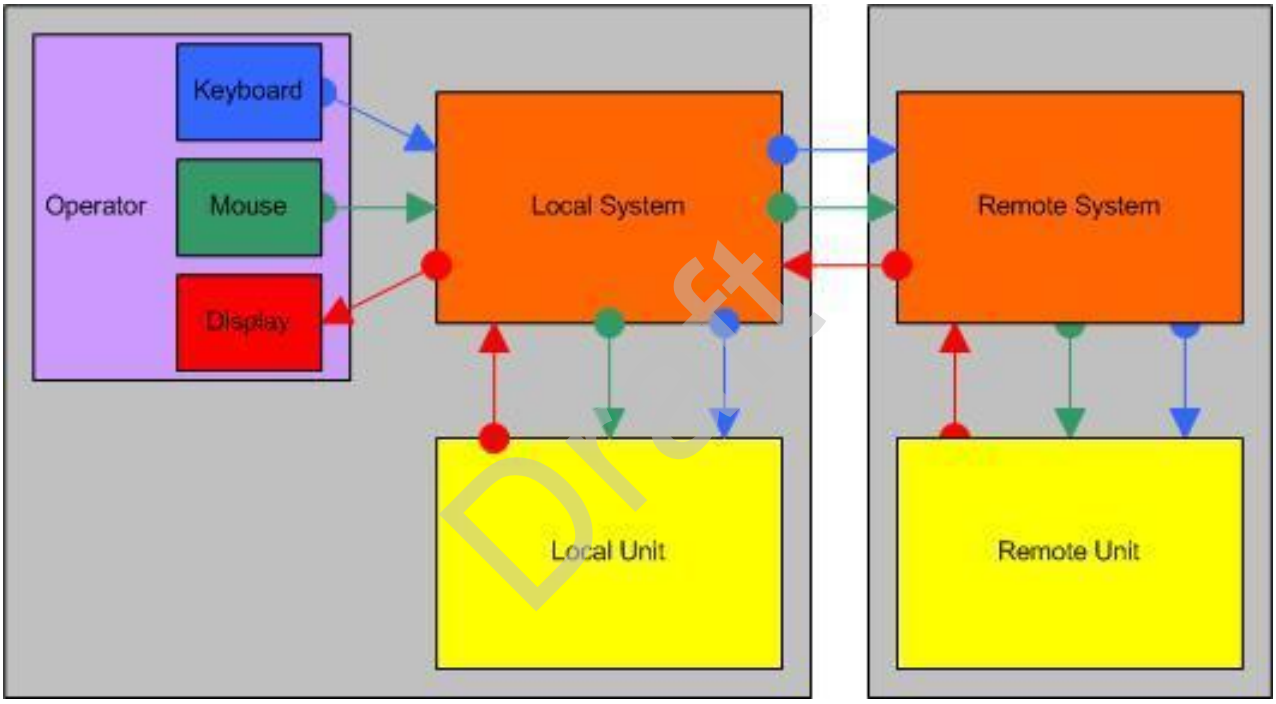
The ***Stand Alone System Architecture*** (on page 35), as previously described, may be extended to enable a single operator, working from a Local System, to interact with one or more Remote Systems.



In this configuration, the Local (Left) and Remote (Right) systems must first be networked via the available communication resources (Network Interface Hardware and Network Device Driver Interface).

Once networked, the local system operator must login to the Remote system via the "ssh user@Remote" command. The Local and Remote Terminal Device Interface then establishes a logical communication channel for exchanging keyboard, mouse and display information.

For each login Local and Remote session, the Operator may then select and run an Application Program. As depicted in the following figure. Application Programs run as Local Units on the system to which the Operator first logged in. Application Programs run as Remote Units on the systems to which the operator logged in via the "ssh user@Remote" command.



NOTE: Concurrent login sessions are a convenience that must be used with caution. Time critical, resource intensive activities ought to be performed individually or sequentially (but NOT concurrently) on a Stand Alone System. Only non-time critical, non-resource intensive activities may be performed concurrently on Stand Alone or Stand Among Systems.

1 **Local System (Left)** --- Operator opens one or more Command Line Interface Shells.

- One or more newly opened shells may be used to run a Python or non-Python Application Program as its **Local Unit (Left**.)
- One or more newly opened shells may be used to login to a Remote system (via the "ssh user@Remote command). Once Remote login has been successful, the operator may run a Python or non-Python Application Program as a **Remote Unit (Right)**.

2 **Local Unit (Left)** --- A Python or non-Python Application Program that has been launched in a Local Command Line Interface Shell.

3 **Remote System (Right)** - Same Operator opens one or more Command Line Interface Shells.

- One or more newly opened shells may be used to run a Python or non-Python Application Program as its **Remote Unit (Right)**.

  The ***Multi-Session Desktop*** (on page 51) figure depicts the desktop of a multi-user, multi-process, multi-threaded computer running the Professional Edition of Microsoft Windows 7. Among the background of desktop icons, there are two *Team*STARS "tsWxGTUI_PyVx" Toolkit sessions. The time each session displays synchronizes within its own one second refresh interval. The local session, on the left, is actively running Python 3.x on the Windows platform. The remote session, on the right, is actively running Python 2.x on the Mac OS X Yosemite platform which also serves as the Parallels 10 Hypervisor host for diverse Guest Operating Systems including:

  Linux (CentOS 7 64-bit, Fedora 21 64-bit, Scientific 6.5 32-bit and Ubuntu 12.04 32-bit)

  Windows (98 SE 16-bit, XP 32-bit, 7 32-bit, 8 32-bit and 8.1 32-bit)

  Unix (FreeBSD 9.2, PC-BSD 10, OpenIndiana 151a8 and OpenSolaris 11 32-bit)

- One or more newly opened shells may be used to login to a Remote system (via the "ssh user@Remote command). Once Remote login has been successful, the operator may run a Python or non-Python Application Program as a **Remote Unit (Right)**.

4 **Remote Unit (Right)** --- A Python or non-Python Application Program that has been launched in a Remote Command Line Interface Shell.

1.3 Usage Terms & Conditions

1 The *Team*STARS "tsWxGTUI_PyVx" Toolkit and its third-party components are copyrighted works that are licensed and distributed as free and open source software, in the hope that they will be useful but WITHOUT ANY WARRANTY; WITHOUT EVEN THE IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL THE COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

2 You may use, modify and redistribute individual copyrighted works only under the terms and conditions of the license designated by the *Team*STARS "tsWxGTUI_PyVx" Toolkit and its third-party component work's copyright holder(s).

2 SCREENSHOTS

From Wikipedia, the free encyclopedia

"A screen dump, screen capture (or screen-cap), screenshot (or screen shot), screengrab (or screen grab), or print screen[1] is an image taken by the computer user to record the visible items displayed on the monitor, television, or another visual output device. Usually, this is a digital image using the (host) operating system or software running on the computer, but it can also be a capture made by a camera or a device intercepting the video output of the display (such as a DVR). That latent image converted and saved to an image file such as to JPEG or PNG format is also called a screenshot.

Screenshots can be used to demonstrate a program, a particular problem a user might be having, or generally when display output needs to be shown to others or archived. For example, after being emailed a screenshot, a Web page author might be surprised to see how his page looks on a different Web browser and can take corrective action. Likewise with differing email software programs, (particularly such as in a cell phone, tablet, etc.,) a sender might have no idea how his email looks to others until he sees a screenshot from another computer and can (hopefully) tweak his settings appropriately."

From Wikipedia, the free encyclopedia:

"In computing, xterm is the standard terminal emulator for the X Window System. A user can have many different invocations of xterm running at once on the same display, each of which provides independent input/output for the process running in it (normally the process is a Unix shell).

xterm originated prior to the X Window System. It was originally written as a stand-alone terminal emulator for the VAXStation 100 (VS100) by Mark Vandevoorde, a student of Jim Gettys, in the summer of 1984, when work on X started. It rapidly became clear that it would be more useful as part of X than as a standalone program, so it was retargeted to X. As Gettys tells the story, "part of why xterm's internals are so horrifying is that it was originally intended that a single process be able to drive multiple VS100 displays."

After many years as part of the X reference implementation, around 1996 the main line of development then shifted to XFree86 (which itself forked from X11R6.3), and it is presently actively maintained by Thomas Dickey.

Xterm variants are also available.

Most terminal emulators for X started as variations on xterm."

Typically, xterms support keyboard and mouse input and a display whose character cells consist of an array of RED-GREEN-BLUE phosphors per pixel that are independently programmable in intensity so as to produce any of 8-colors (with their associated 64-color pairs) per character:

1 *BLACK*

2 *RED*

3 *GREEN*

4 *YELLOW*

5 *BLUE*

6 *MAGENTA*

7 *CYAN*

8 *WHITE*

Introduced long after the 8-color xterm, the xterm-16color produced any of the following 16-colors (with their associated 256-color pairs) per character:

1 BLACK

2 RED

3 GREEN

4 YELLOW

5 BLUE

6 MAGENTA

7 CYAN

8 WHITE

9 MAROON

10 OLIVE

11 NAVY

12 PURPLE

13 TEAL

14 SILVER

15 GRAY

16 LIME_GREEN

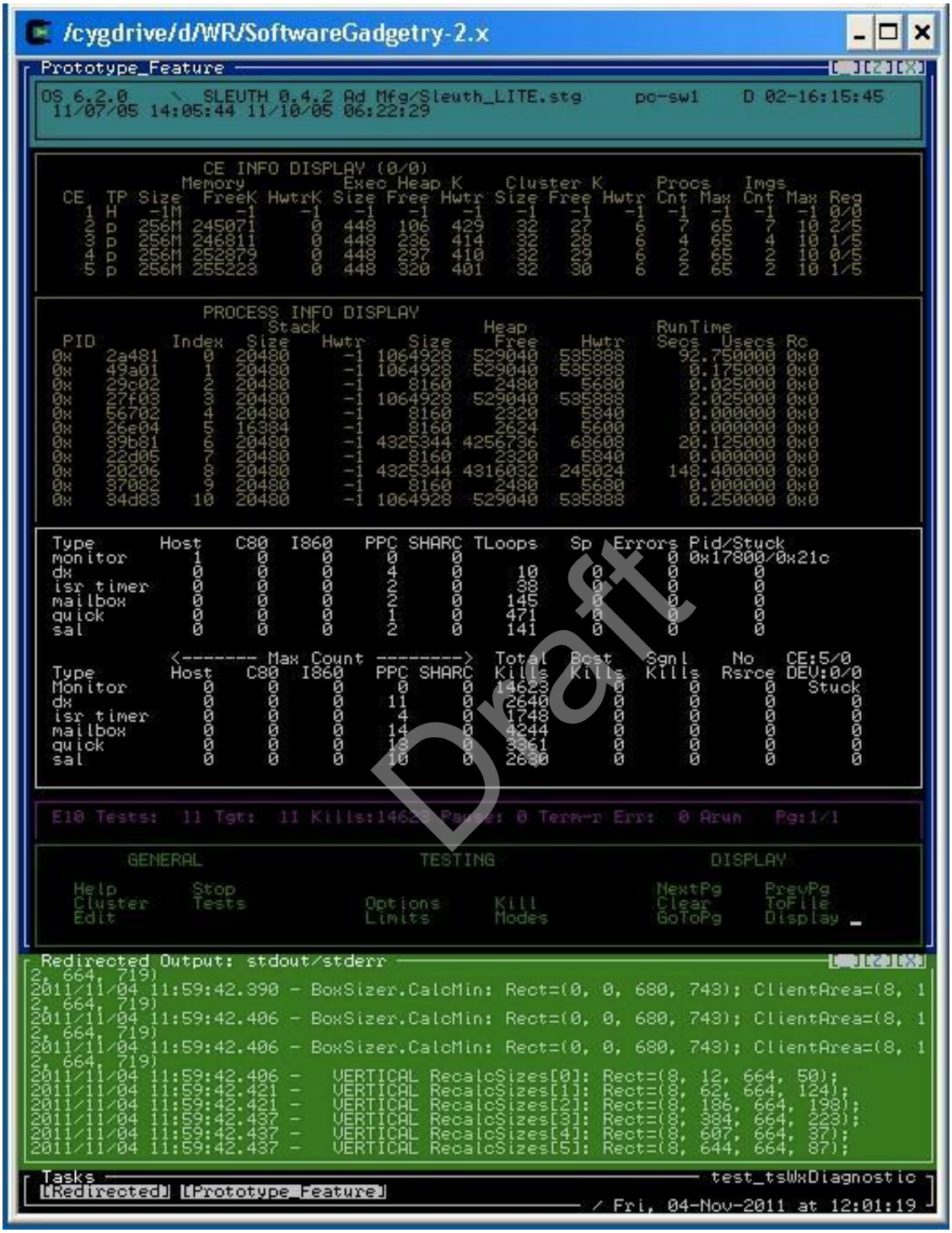
NOTE: The following screen shots demonstrate various widgets on a terminal that reports having colors. It is the same application that runs on terminals that report NOT having colors. The application remains unaware of the presence or absence of colors.

- ***2016 Samples of XTERM (8, 16 & 256 Color)*** (on page 45)
- ***2015 Samples of XTERM (16 color) & VT100 (1 color ON/OFF) Desktops*** (on page 50)
- ***2015 Samples of XTERM (16 color) & VT100 (1 color ON/OFF) Desktops*** (on page 50)
- ***2012 Samples of VT-100 (1 color ON/OFF)*** (on page 67)

2.1 2016 Samples of XTERM (8, 16 & 256 Color)

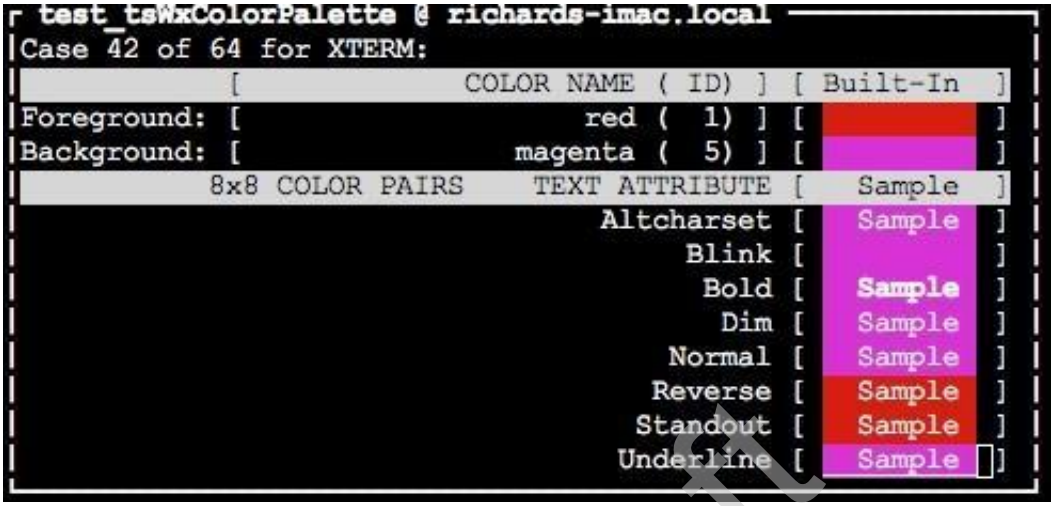
2.1.1 "Curses" API Sample 8-Color Output

The following figure is visually similar to the output-only one in the NCurses API. Adding input capability only requires adding a relatively small amount of event handling code to define and connect (bind) clickable GUI Objects (such as buttons, checkboxes etc) to the appropriate event handler.


/cygdrive/d/WR/SoftwareGadgetry-2.x
Prototype_Feature
OS 6.2.0 \ SLEUTH 0.4.2 Ad Mfg/Sleuth_LITE.stg po-sw1 D 02-16:15:45
11/07/05 14:05:44 11/10/05 06:22:29
CE INFO DISPLAY (0/0)
Memory Exec Heap K Cluster K Procs Imgs
CE TP Size FreeK HwtrK Size Free Hwtr Size Free Hwtr Cnt Max Cnt Max Reg
1 H -1M -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0/0
2 p 256M 245071 0 448 106 429 32 27 6 7 65 7 10 2/5
3 p 256M 246811 0 448 236 414 32 28 6 4 65 4 10 1/5
4 p 256M 252879 0 448 297 410 32 29 6 2 65 2 10 0/5
5 p 256M 255223 0 448 320 401 32 30 6 2 65 2 10 1/5
PROCESS INFO DISPLAY
Stack Heap RunTime
PID Index Size Hwtr Size Free Hwtr Secs Usecs Rc
0x 2a481 0 20480 -1 1064928 529040 535888 92.750000 0x0
0x 49a01 1 20480 -1 1064928 529040 535888 0.175000 0x0
0x 29c02 2 20480 -1 8160 2480 5680 0.025000 0x0
0x 27f03 3 20480 -1 1064928 529040 535888 2.025000 0x0
0x 56702 4 20480 -1 8160 2320 5840 0.000000 0x0
0x 26e04 5 16384 -1 8160 2624 5600 0.000000 0x0
0x 39b81 6 20480 -1 4325344 4256736 68608 20.125000 0x0
0x 22d05 7 20480 -1 8160 2320 5840 0.000000 0x0
0x 20206 8 20480 -1 4325344 4316032 245024 148.400000 0x0
0x 37082 9 20480 -1 8160 2480 5680 0.000000 0x0
0x 34d83 10 20480 -1 1064928 529040 535888 0.250000 0x0
Type Host C80 I860 PPC SHARC TLoops Sp Errors Pid/Stuck
monitor 1 0 0 0 0 0 0x17800/0x21c
dx 0 0 0 4 0 10 0 0 0
isr timer 0 0 0 2 0 38 0 0 0
mailbox 0 0 0 2 0 145 0 0 0
qwick 0 0 0 1 0 471 0 0 0
sal 0 0 0 2 0 141 0 0 0
<-------- Max Count --------> Total Bcst Sgnl No CE:5/0
Type Host C80 I860 PPC SHARC Kills Kills Kills Rsrce DEV:0/0
Monitor 0 0 0 0 0 14623 0 0 0 Stuck
dx 0 0 0 11 0 2640 0 0 0 0
isr timer 0 0 0 4 0 1748 0 0 0 0
mailbox 0 0 0 14 0 4244 0 0 0 0
qwick 0 0 0 13 0 3361 0 0 0 0
sal 0 0 0 10 0 2630 0 0 0 0
E10 Tests: 11 Tgt: 11 Kills:14623 Pause: 0 Term-r Err: 0 Arun Pg:1/1
GENERAL TESTING DISPLAY
Help Stop NextPg PrevPg
Cluster Tests Options Kill Clear ToFile
Edit Limits Modes GoToPg Display
Draft
Redirected Output: stdout/stderr
2, 664, 719)
2011/11/04 11:59:42.390 - BoxSizer.CalcMin: Rect=(0, 0, 680, 743); ClientArea=(8, 1
2, 664, 719)
2011/11/04 11:59:42.406 - BoxSizer.CalcMin: Rect=(0, 0, 680, 743); ClientArea=(8, 1
2, 664, 719)
2011/11/04 11:59:42.406 - BoxSizer.CalcMin: Rect=(0, 0, 680, 743); ClientArea=(8, 1
2, 664, 719)
2011/11/04 11:59:42.406 - VERTICAL RecalcSizes[0]: Rect=(8, 12, 664, 50);
2011/11/04 11:59:42.421 - VERTICAL RecalcSizes[1]: Rect=(8, 62, 664, 124);
2011/11/04 11:59:42.421 - VERTICAL RecalcSizes[2]: Rect=(8, 186, 664, 198);
2011/11/04 11:59:42.437 - VERTICAL RecalcSizes[3]: Rect=(8, 384, 664, 223);
2011/11/04 11:59:42.437 - VERTICAL RecalcSizes[4]: Rect=(8, 607, 664, 37);
2011/11/04 11:59:42.437 - VERTICAL RecalcSizes[5]: Rect=(8, 644, 664, 87);
Tasks test_tsWxDiagnostic
[Redirected] [Prototype_Feature]
/ Fri, 04-Nov-2011 at 12:01:19

2.1.2 Built-in 32-bit "Curses" 8-Color Palette Sample Output

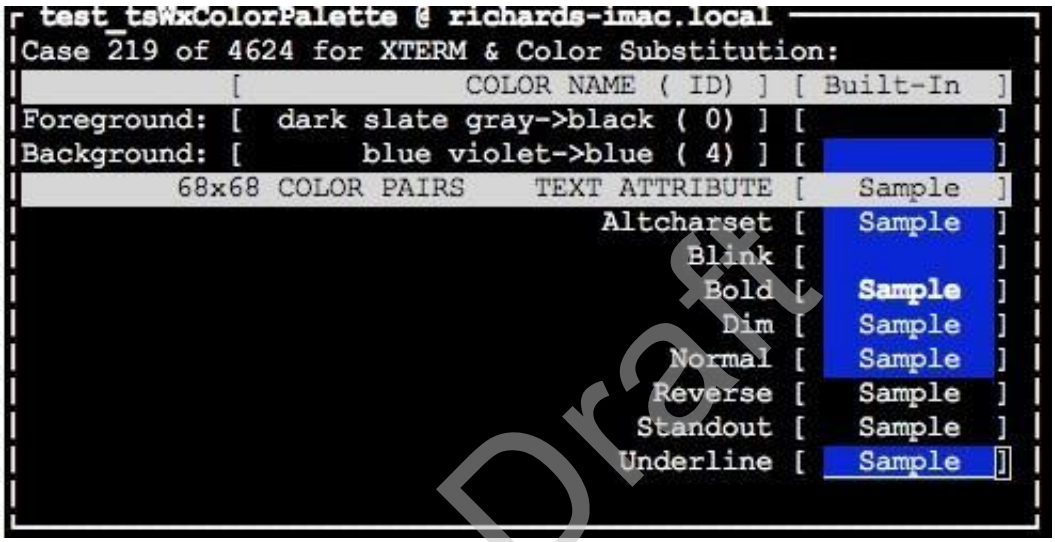
The following figure is produced by the "test_tsWxColorPalette.py" demo application. It displays each combination of foreground and background color-pair available in the built-in nCurses color palette with each available text character attribute.



2.1.3 Built-in 32-bit "wxPython" 68-Color Palette Sample Output

The following figure is produced by the "test_tsWxColorPalette.py" demo application.

- For 32-bit processors running Python 2.6.0-2.7.13, Python 3.0-3.5.2 or Python 3.6.0 GUI Applications (or 64-bit processors running those Application in 32-bit compatibility mode), it displays each combination of built-in "nCurses" foreground and background color-pair with each available text character attribute.
- *If the number of available "nCurses" colors is fewer than the number (68) of standard "wxPython" colors, a color substitution algorithm will map each "wxPython" color to an associated "nCurses" color.*



2.1.4 Built-in 64-bit "wxPython" 256-Color Palette Sample Output

The following figure is produced by the "test_tsWxColorPalette.py" demo application.

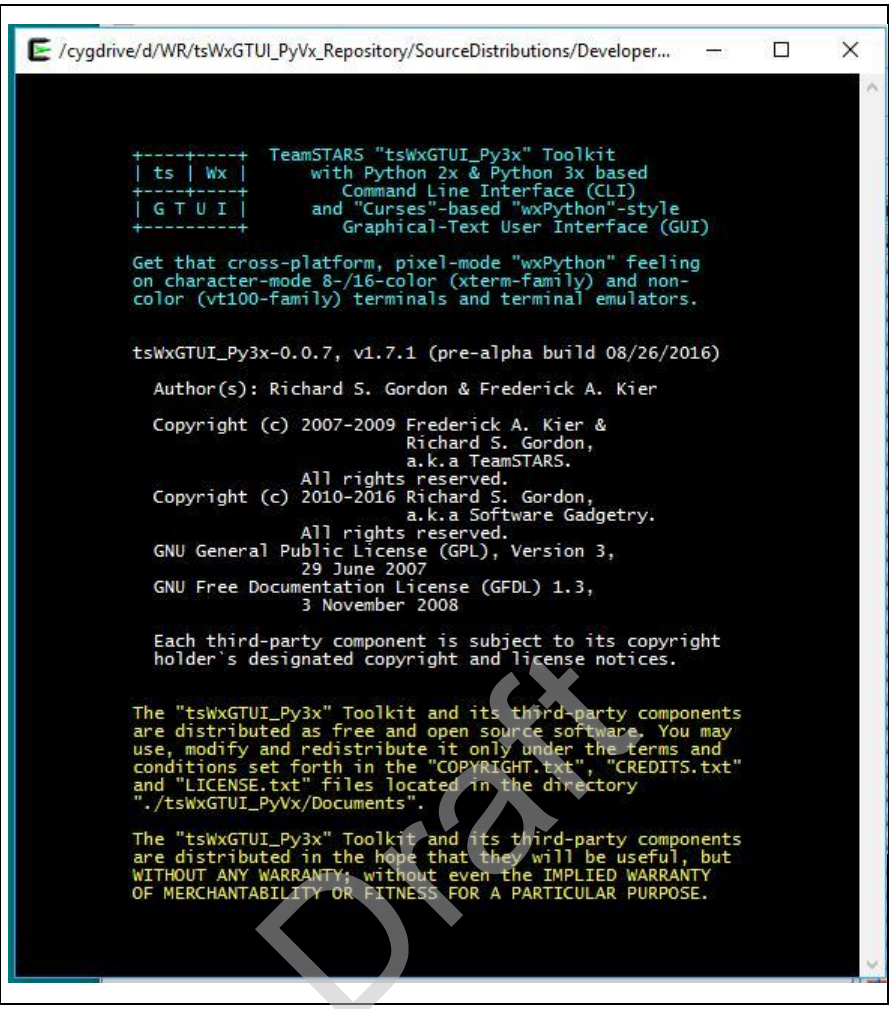
- For 64-bit processors running Python 3.6.0 or later GUI Applications, it displays each combination of the 68 Standard "wxPython" colors (plus up to application-specific ones) and associated foreground and background color-pair, by mixing up to 256 intensity-adjusted combinations of the three primary "nCurses" colors (Red-Green-Blue), in the "wxPython" color palette. with each available text character attribute.

Pending the Python Software Foundation's official release of 64-bit Python 3.6.0: ▪ There is no "test_tsWxColorPalette" figure.

2.1.5 Author's Default Splash Screen

Excerpt From Wikipedia, the free encyclopedia:

> *"A splash screen is a graphical control element consisting of window containing an image, a logo and the current version of the software. A splash screen usually appears while a game or program is launching. A splash page is an introduction page on a website. Splash screens cover the entire screen or web page, or simply a rectangle near the center of the screen or page. The splash screens of operating systems and some applications that expect to be run full-screen usually cover the entire screen.*
>
> *Purpose*
>
> *Splash screens are typically used by particularly large applications to notify the user that the program is in the process of loading. They provide feedback that a lengthy process is underway. Occasionally, a progress bar within the splash screen indicates the loading progress. A splash screen disappears when the application's main window appears.*
>
> *Splash screens typically serve to enhance the look and feel of an application or web site, hence they are often visually appealing. They may also have animations, graphics, and sound."*

Terminal ID	
Masthead / Trademark	
Copyright	
License	
Disclaimer	

2.2 2015 Samples of XTERM (16 color) & VT100 (1 color ON/OFF) Desktops

The appearance of the Graphical User Interface has evolved over time:

1 During development of support for 8-/16-color XTERMs, it seemed useful to support color markup for text messages rather than just for foreground/background window borders and client areas. The tsWxGlobals.py configuration file now includes options to enable or disable color markup in the Frame used for Redirected Output of stderr/stdout messages and/or in the test_tsWxScrolledWindow.py demo text messages.

2 During development of local and remote access demonstrations, it became obvious that the display needed to not only identify the launched application but also the local or remote host on which it was launched. The top-left of the Task Bar now displays the host name (and domain when available).

3 The Python Curses-based terminal device interface library supports proprietary terminals from manufacturers such as Data General, Hewlett-Packard, International Business Machines and Tektonix.

   Curses also supports widely-used industry-standard terminal emulators such as Digital Equipment Corporation's "vt100" and the X-Window System's "xterm".

   a) The "vt100" and "vt220" emulators generate each "White"-on-"Black" character by turning individual pixels in an 8x12 grid of pixels "ON" for foreground or "OFF" for background. Conversely, "Black"-on-"White" characters are generated by turning individual pixels in an 8x12 grid of pixels "OFF" for foreground or "ON" for background.

   b) The "xterm" emulators generate each character by adjusting the color intensity (from "OFF" at 0% to "ON" at 100%) of individual tri-color ("RED"-"GREEN"-"BLUE") pixels in an 8x12 grid of tri-color pixels. The "xterm" supports 8-colors for foreground or background with 64-color-pairs. The Toolkit includes an application that displays the built-in color palette: ***wxPython Color Palette (Built-In)*** (on page 55)

   c) Despite their name, the "xterm-16color", "xterm-88color" and "xterm-256color" emulators only support 16-colors with 256-color-pairs.

4 The TeamSTARS "tsWxGTUI_PyVx" Toolkit recognizes the "wxPython" 68-color palette but must use a mapped (substitute) color from the "xterm" or "vt100" color palette which may be a "likeness" in name only. The Toolkit includes an application that displays the color substitution palette: ***wxPython Color Palette (Mapped)*** (on page 55)

2.2.1 Multi-Session Desktop

From Wikipedia, the free encyclopedia

"In computer science, in particular networking, a session is a semi-permanent interactive information interchange, also known as a dialogue, a conversation or a meeting, between two or more communicating devices, or between a computer and user (see Login session). A session is set up or established at a certain point in time, and then torn down at some later point. An established communication session may involve more than one message in each direction. A session is typically, but not always, stateful, meaning that at least one of the communicating parts needs to save information about the session history in order to be able to communicate, as opposed to stateless communication, where the communication consists of independent requests with responses.

An established session is the basic requirement to perform a connection-oriented communication. A session also is the basic step to transmit in connectionless communication modes. However any unidirectional transmission does not define a session.[1]

Communication sessions may be implemented as part of protocols and services at the application layer, at the session layer or at the transport layer in the OSI model.

1 *Application layer examples:*

   a) *HTTP sessions, which allow associating information with individual visitors*

b) *A telnet remote login session*

2 *Session layer example:*

a) *A Session Initiation Protocol (SIP) based Internet phone call*

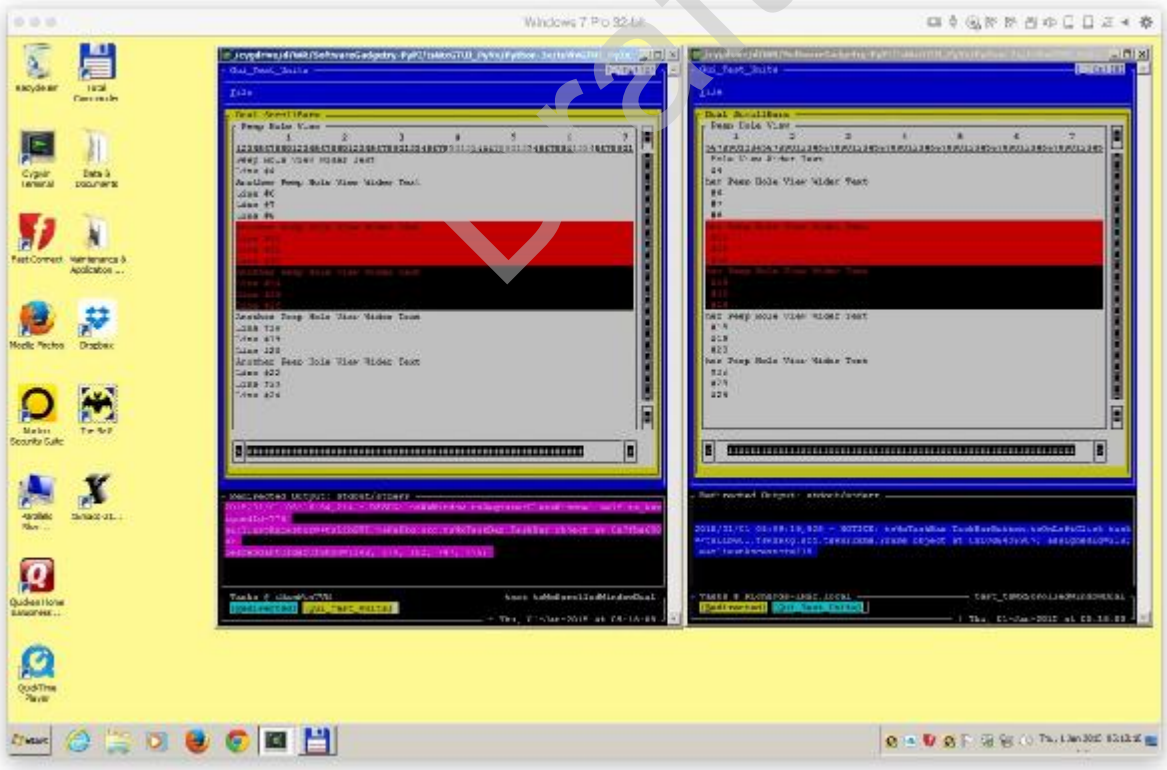
3 *Transport layer example:*

a) *A TCP session, which is synonymous to a TCP virtual circuit, a TCP connection, or an established TCP socket.*

In the case of transport protocols that do not implement a formal session layer (e.g., UDP) or where sessions at the application layer are generally very short-lived (e.g., HTTP), sessions are maintained by a higher level program using a method defined in the data being exchanged. For example, an HTTP exchange between a browser and a remote host may include an HTTP cookie which identifies state, such as a unique session ID, information about the user's preferences or authorization level.

HTTP/1.0 was thought to only allow a single request and response during one Web/HTTP Session. However a workaround was created by David Hostettler Wain in 1996 such that it was possible to use session IDs to allow multiple phase Web Transaction Processing (TP) Systems (in ICL[disambiguation needed] nomenclature), with the first implementation being called Deity. Protocol version HTTP/1.1 further improved by completing the Common Gateway Interface (CGI) making it easier to maintain the Web Session and supporting HTTP cookies and file uploads.

Most client-server sessions are maintained by the transport layer - a single connection for a single session. However each transaction phase of a Web/HTTP session creates a separate connection. Maintaining session continuity between phases required a session ID. The session ID is embedded within the <A HREF> or <FORM> links of dynamic web pages so that it is passed back to the CGI. CGI then uses the session ID to ensure session continuity between transaction phases. One advantage of one connection-per-phase is that it works well over low bandwidth (modem) connections. Deity used a sessionID, screenID and actionID to simplify the design of multiple phase sessions."



The Sample Screenshot above shows a Windows 7 Desktop with:

1 A local Windows host with Python 3x session on left; and

2 A remote Mac OS X host with Python 2x session on right.

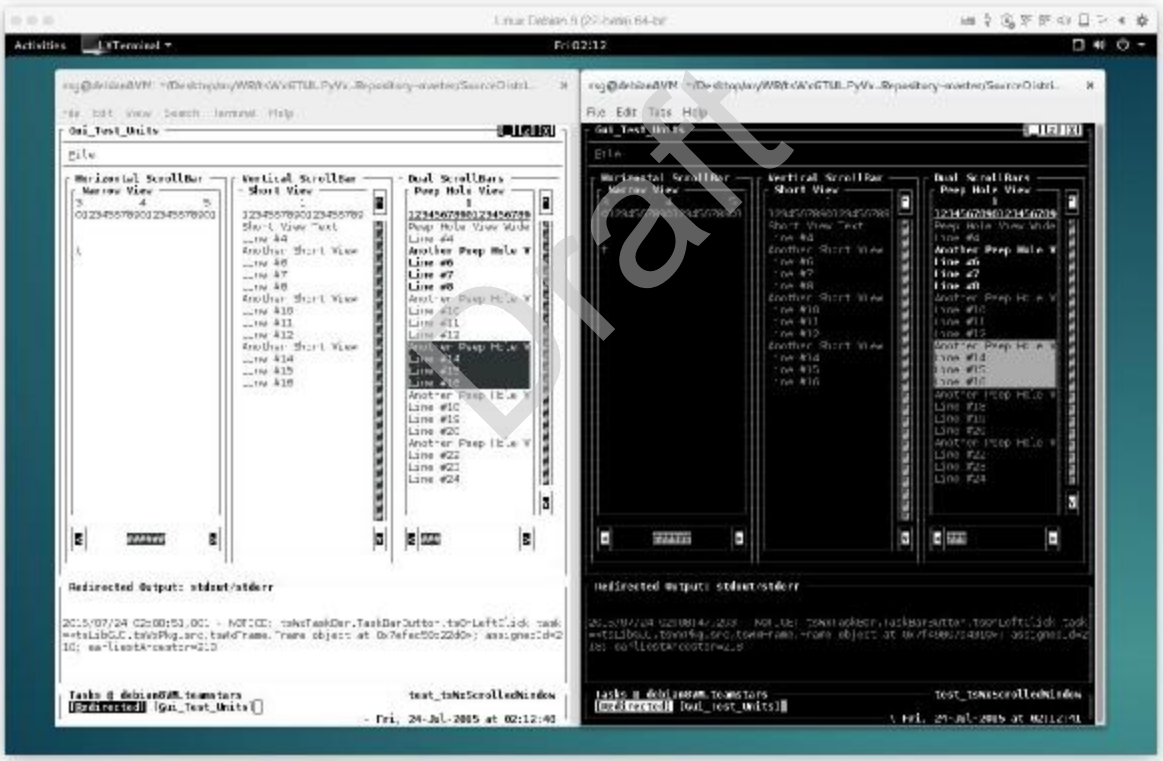
3 Each session consists of

a) a wxPython-style Frame named "Dual ScrollBars" containing scrollable text with optional color markup.

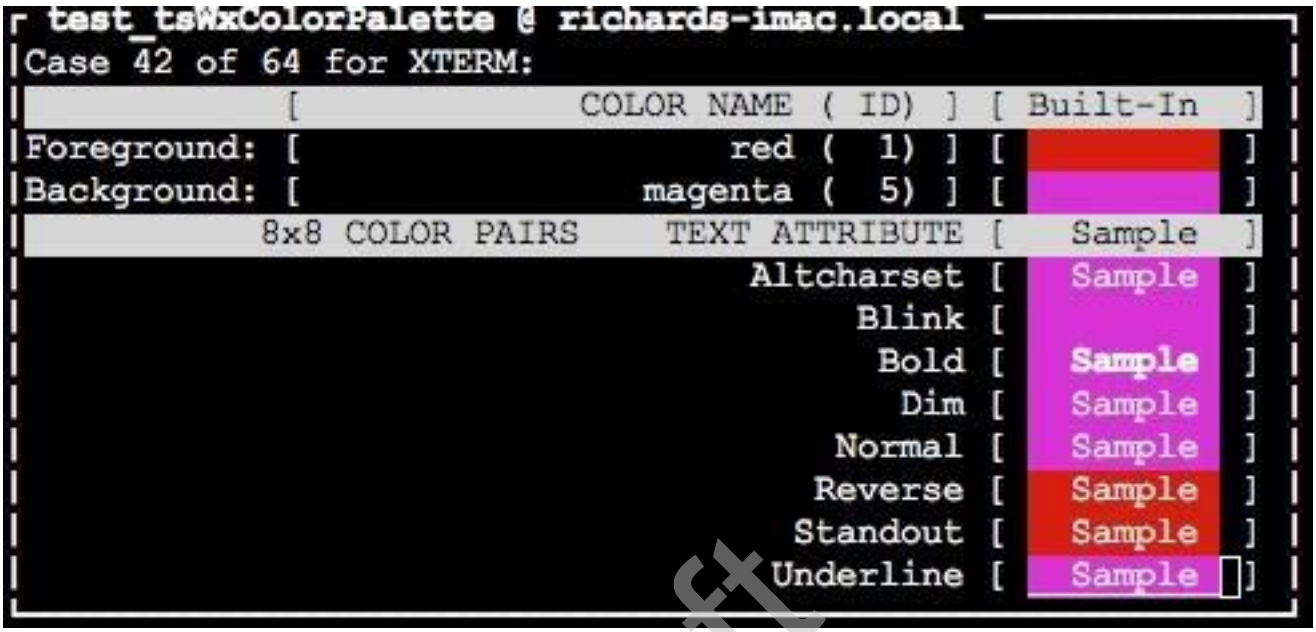
b) a wxPython-style Frame named "Redirected Output" containing date and time stamped event messages, with optional with color markup, that scroll up when new events are registered.

c) a Host Desktop-style Frame named "Tasks @ Host Name" and Application Name with buttons to shift focus from background to foreground. There is also a spinner (to indicate the frequency or absence of idle time) and the current date and time (to indicate when the display was last updated).

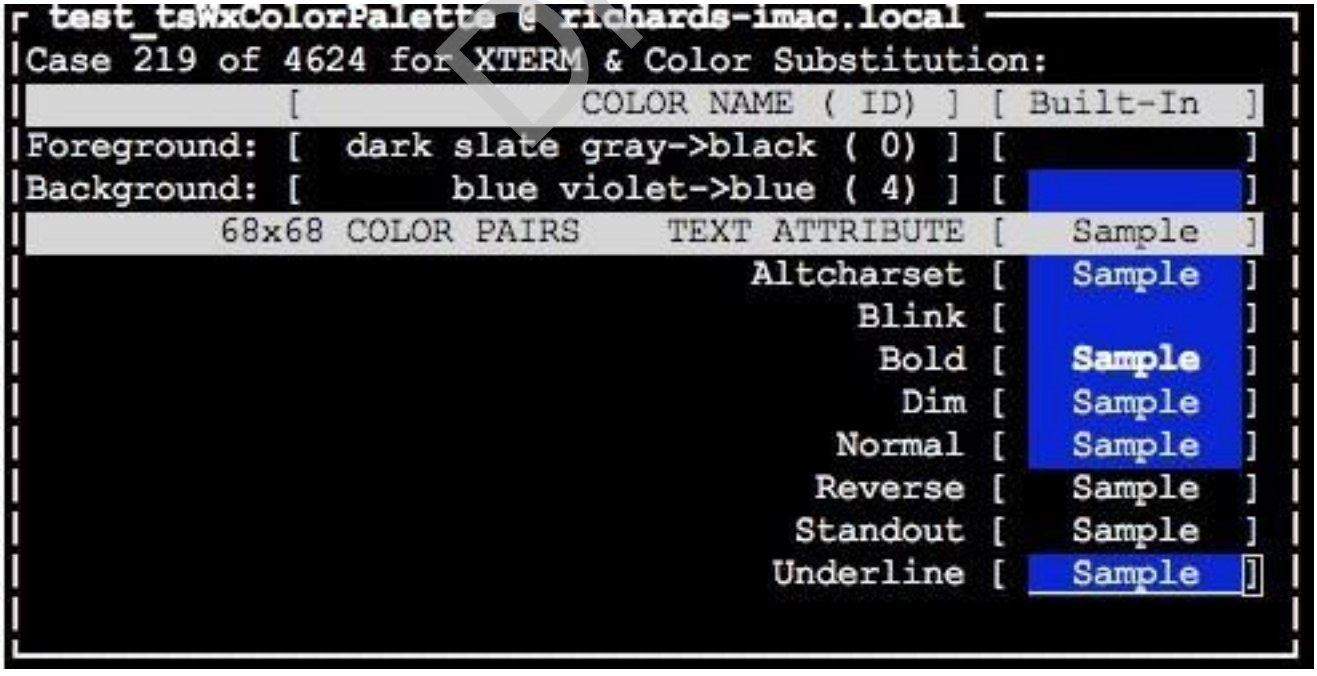
2.2.2 tsWxScrolledWindow using VT-100 (via Debian 8 Linux Terminal & LXTerminal)



2.2.3 wxPython Color Palette (Built-In)



2.2.4 wxPython Color Palette (Mapped)



2.3 2012 Samples of XTERM (8 color)

From Wikipedia, the free encyclopedia:

"In computing, xterm is the standard terminal emulator for the X Window System. A user can have many different invocations of xterm running at once on the same display, each of which provides independent input/output for the process running in it (normally the process is a Unix shell).

xterm originated prior to the X Window System. It was originally written as a stand-alone terminal emulator for the VAXStation 100 (VS100) by Mark Vandevoorde, a student of Jim Gettys, in the summer of 1984, when work on X started. It rapidly became clear that it would be more useful as part of X than as a standalone program, so it was retargeted to X. As Gettys tells the story, "part of why xterm's internals are so horrifying is that it was originally intended that a single process be able to drive multiple VS100 displays."

After many years as part of the X reference implementation, around 1996 the main line of development then shifted to XFree86 (which itself forked from X11R6.3), and it is presently actively maintained by Thomas Dickey.

Xterm variants are also available.

Most terminal emulators for X started as variations on xterm."

Typically, xterms support keyboard and mouse input and a display whose character cells consist of an array of RED-GREEN-BLUE phosphors per pixel that are independently programmable in intensity so as to produce any of 8-colors (with their associated 64-color pairs) per character:

1 *BLACK*

2 *RED*

3 *GREEN*

4 *YELLOW*

5 *BLUE*

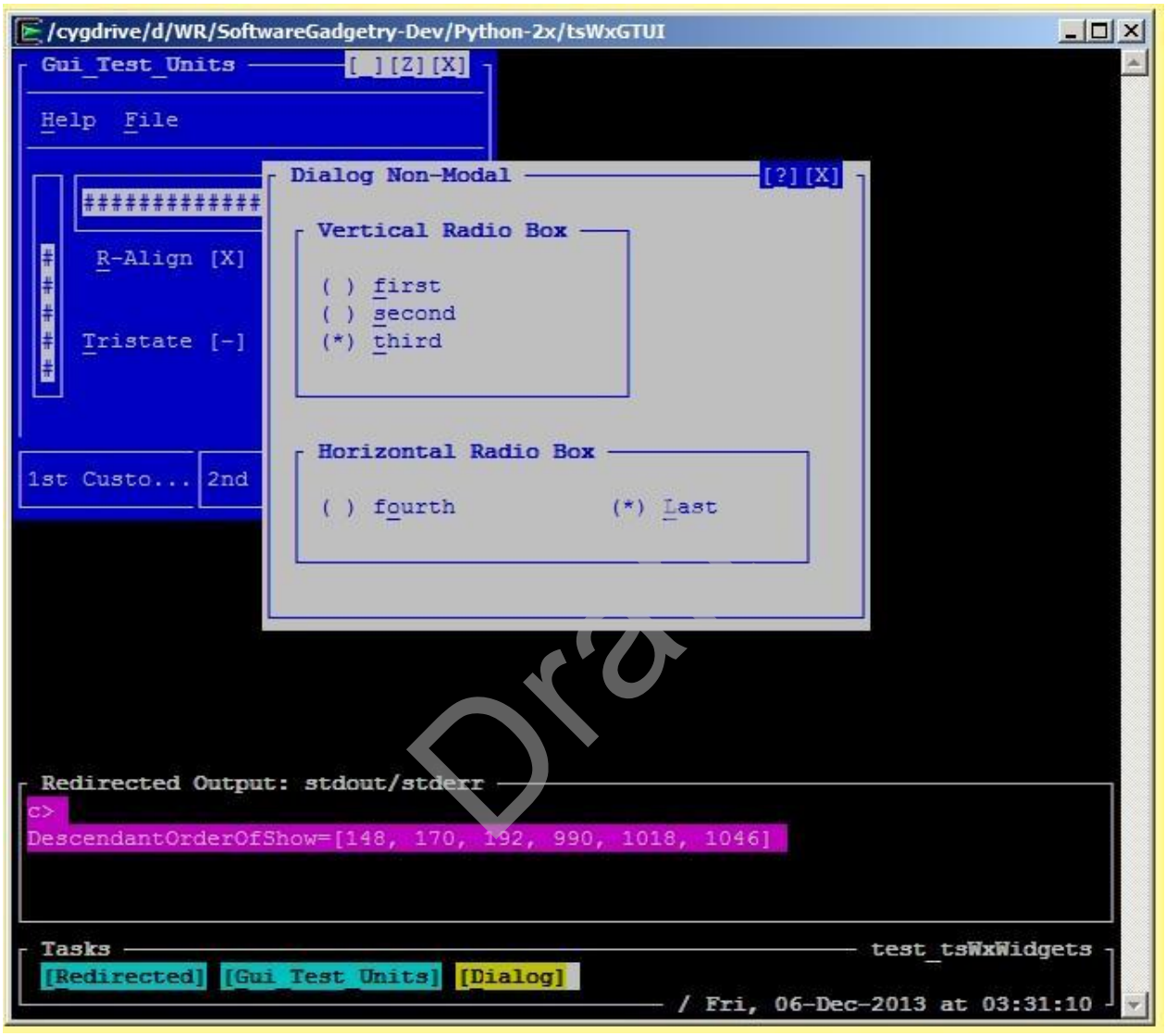
6 *MAGENTA*

7 *CYAN*

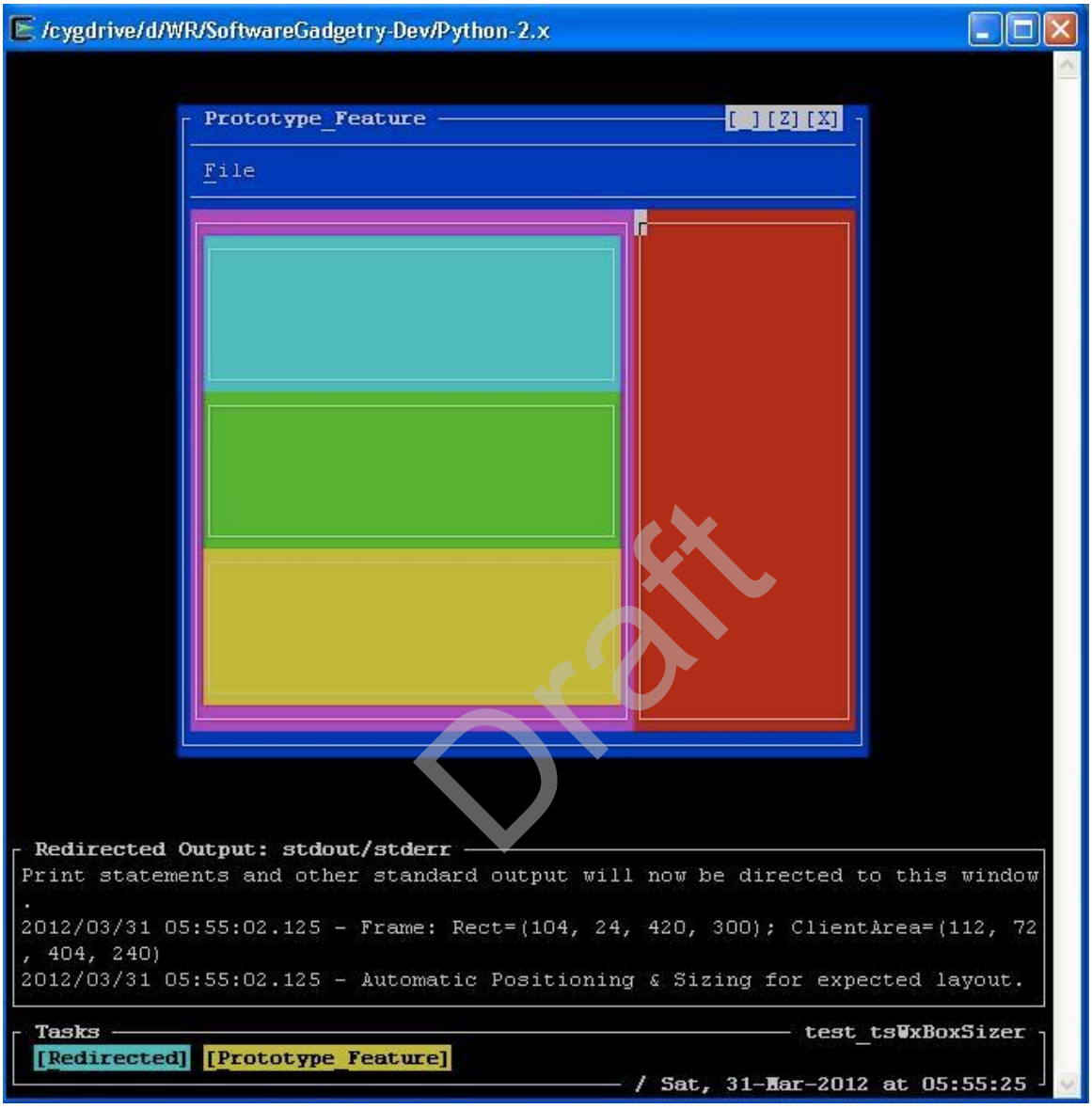
8 *WHITE*

NOTE: The following screen shots demonstrate various widgets on a terminal that reports having colors. It is the same application that runs on terminals that report NOT having colors. The application remains unaware of the presence or absence of colors.

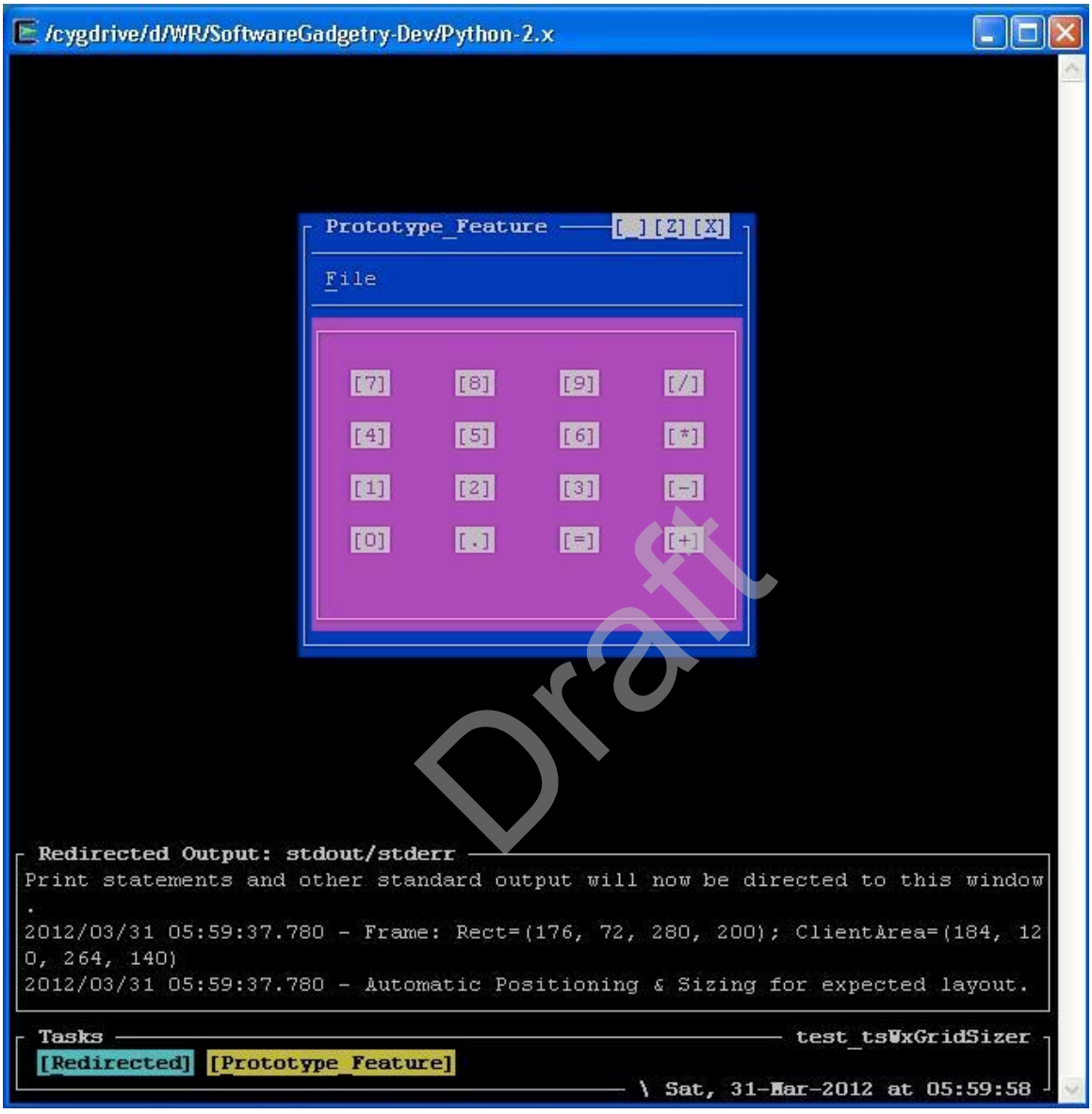
2.3.1 test_tsWxWidgets using xterm (via Cygwin mintty)



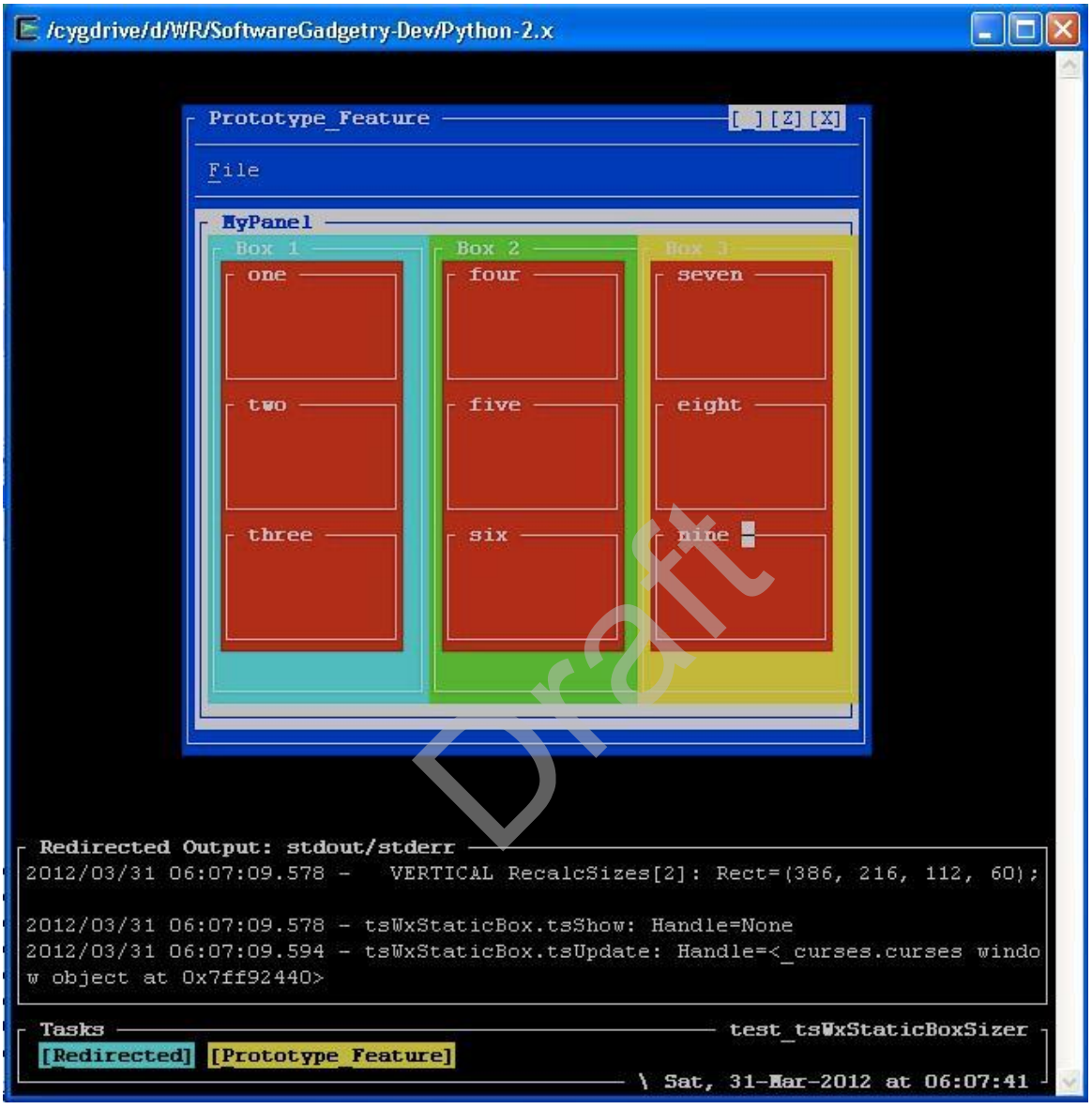
2.3.2 test_tsWxBoxSizer using xterm (via Cygwin mintty)



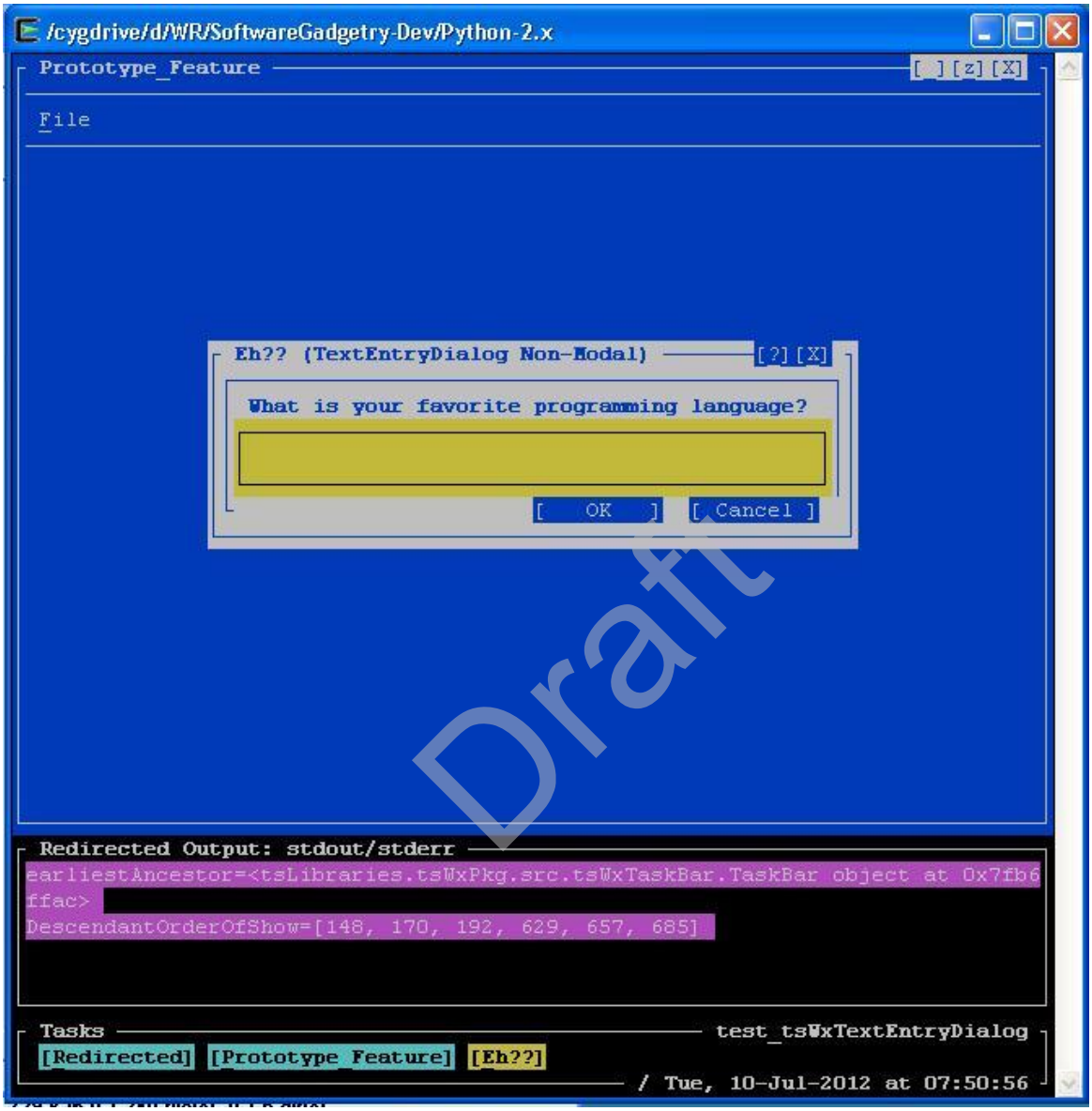
2.3.3 test_tsWxGridSizer using xterm (via Cygwin mintty)



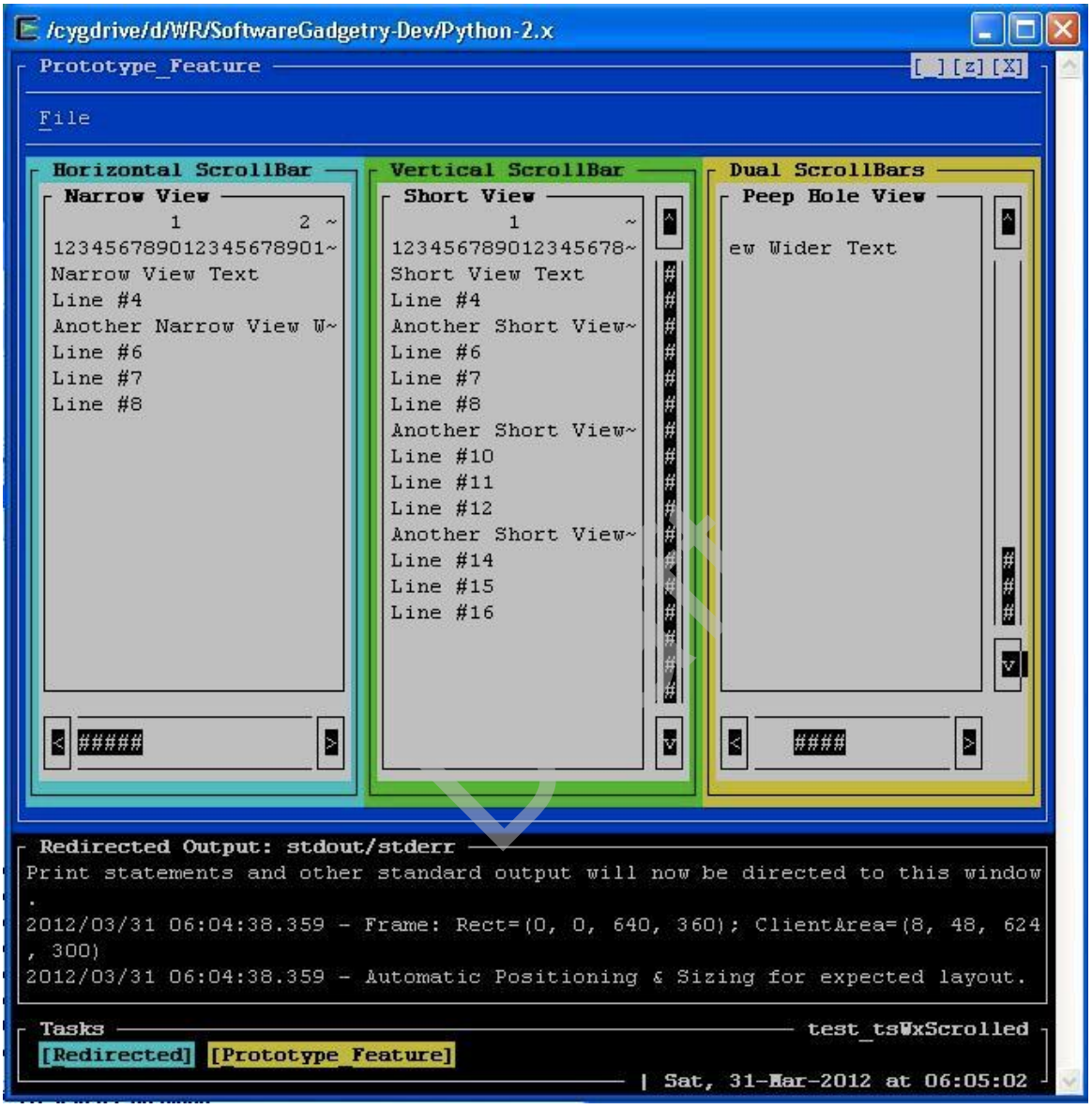
2.3.4 test_tsWxStaticBoxSizer using xterm (via Cygwin mintty)



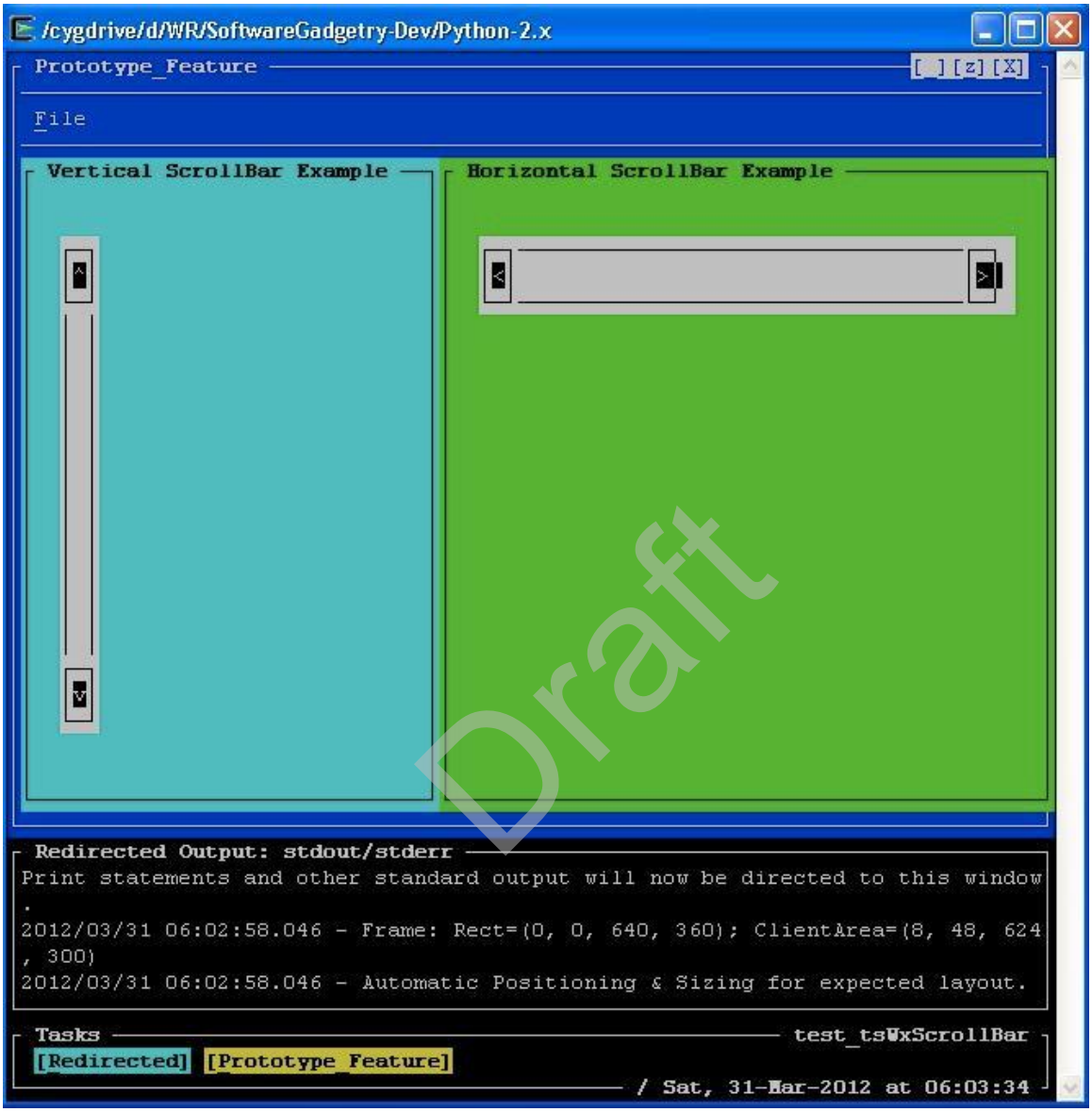
2.3.5 test_tsWxTextEntryDialog using xterm (via Cygwin mintty)



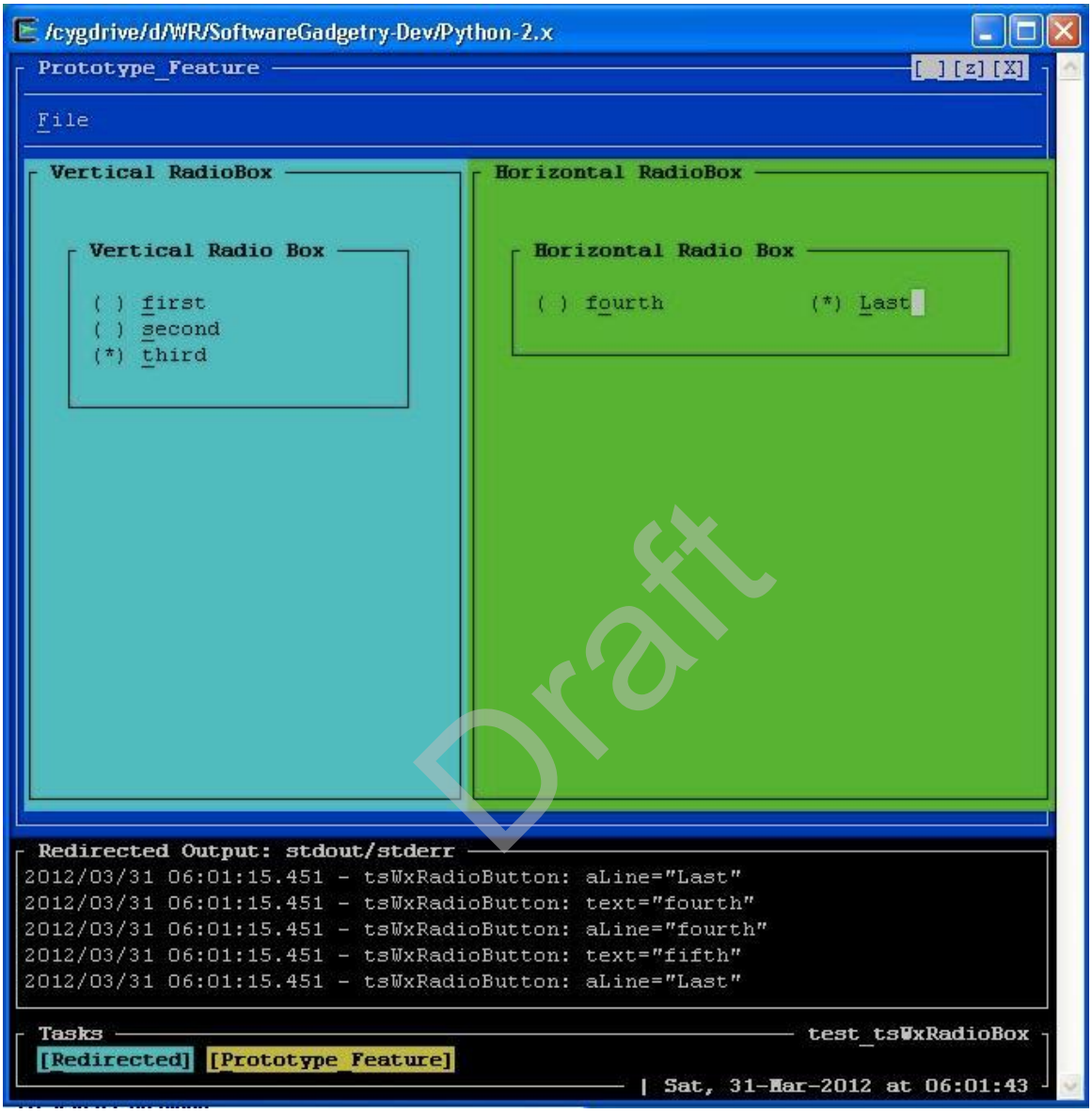
2.3.6 test_tsWxScrolled using xterm (via Cygwin mintty)



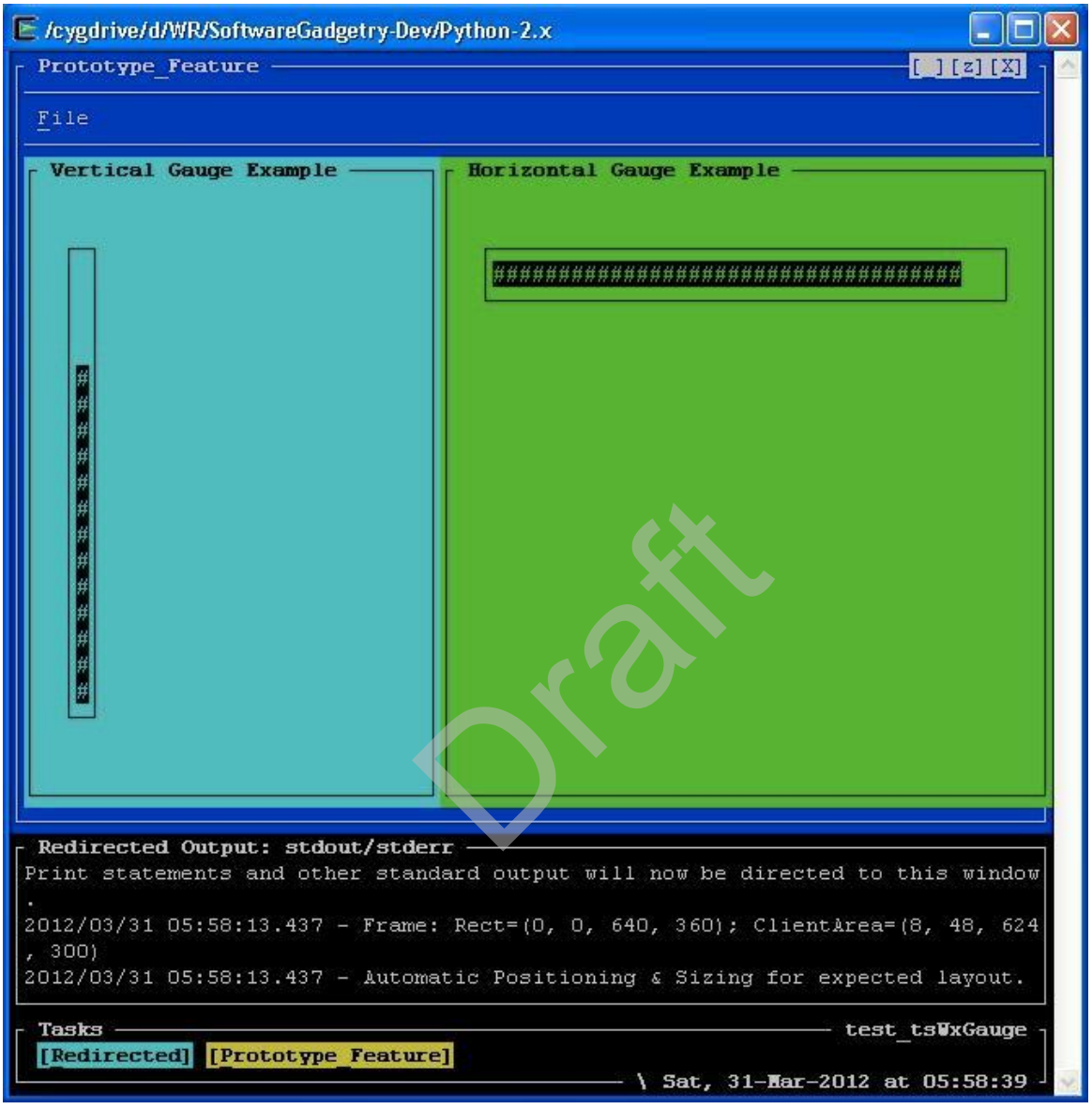
2.3.7 test_tsWxScrollBar using xterm (via Cygwin mintty)



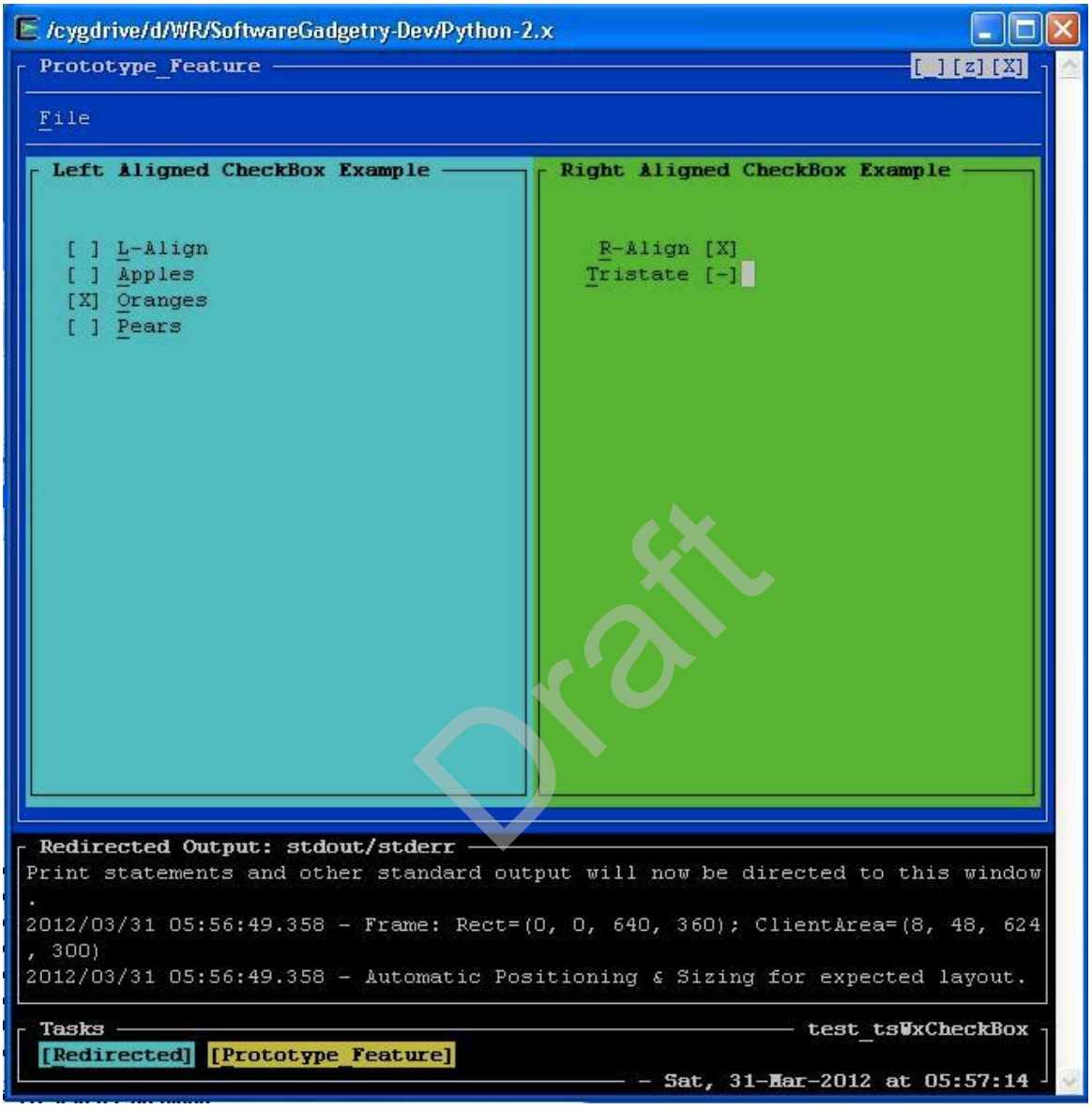
2.3.8 test_tsWxRadioBox using xterm (via Cygwin mintty)



2.3.9 test_tsWxGauge using xterm (via Cygwin mintty)



2.3.10 test_tsWxCheckBox using xterm (via Cygwin mintty)



2.4 2012 Samples of VT-100 (1 color ON/OFF)

From Wikipedia, the free encyclopedia:

"The VT100 is a video terminal that was made by Digital Equipment Corporation (DEC). Its detailed attributes became the de facto standard for terminal emulators to emulate.

It was introduced in August 1978, following its predecessor, the VT52, and communicated with its host system over serial lines using the ASCII character set and control sequences (a.k.a. escape sequences) standardized by ANSI. The VT100 was also the first Digital mass-market terminal to incorporate "graphic renditions" (blinking, bolding, reverse video, and underlining) as well as a selectable 80 or 132 column display. All setup of the VT100 was accomplished using interactive displays presented on the screen; the setup data was stored in non-volatile memory within the terminal. The VT100 also introduced an additional character set that allowed the drawing of on-screen forms.

The control sequences used by the VT100 family are based on the ANSI X3.64 standard, also known as ECMA-48 and ISO/IEC 6429. These are sometimes referred to as ANSI escape codes. The VT100 was not the first terminal to be based on X3.64—The Heath Company had a microprocessor-based video terminal, the Heathkit H-19 (H19), that implemented a subset of the standard proposed by ANSI in X3.64. In addition, the VT100 provided backwards compatibility for VT52 users, with support for the VT52 control sequences.

In 1983, the VT100 was replaced by the more-powerful VT200 series terminals such as the VT220.

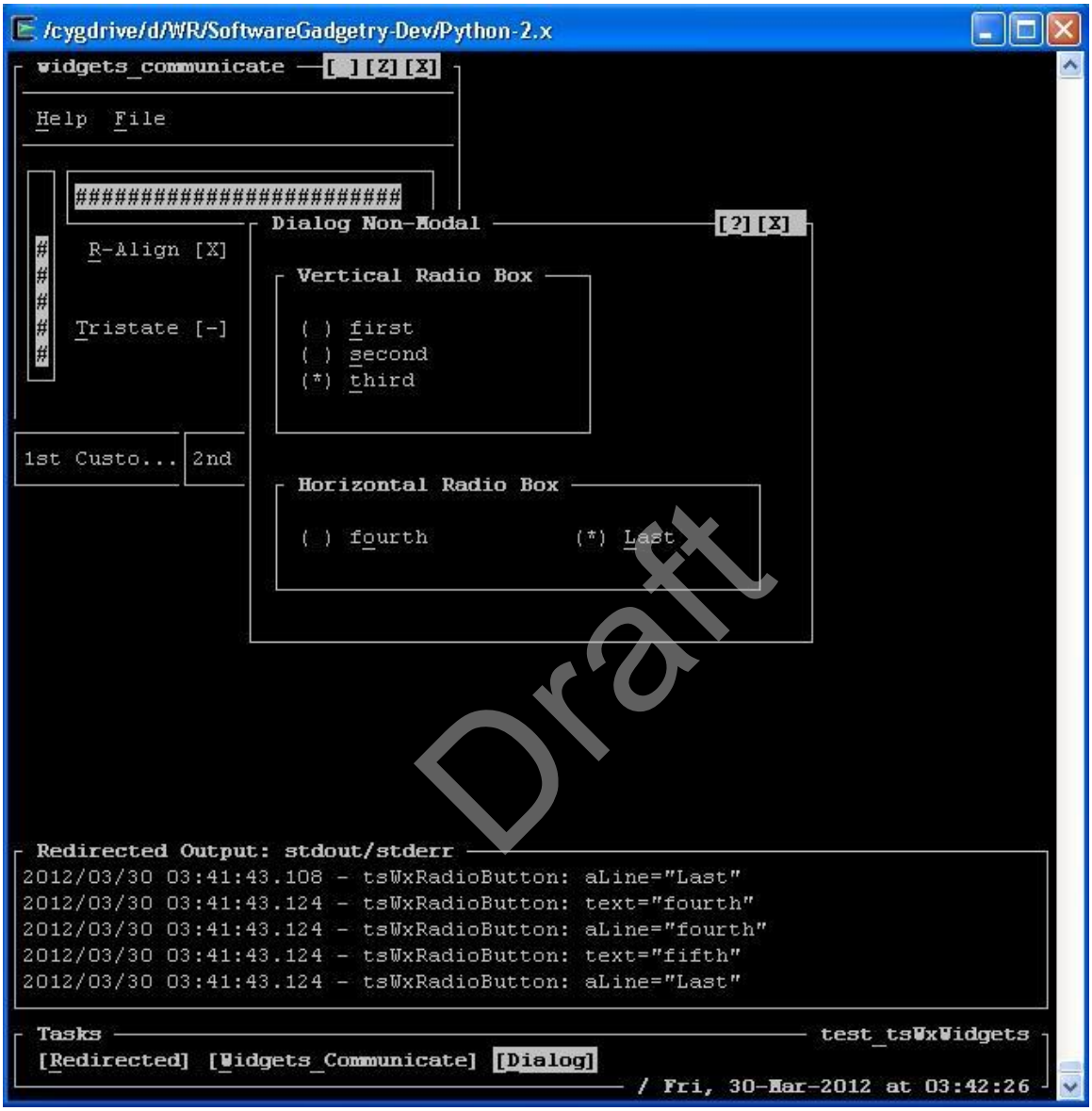
In August 1995 the terminal business of Digital was sold to Boundless Technologies."

Typically, vt100/vt220 terminals support keyboard (without mouse) input and a display whose character cells consist of a single WHITE, GREEN or ORANGE color phosphor per pixel that are independently programmable in intensity so as to produce any of 2-colors (with their associated 2-color pairs) per character:

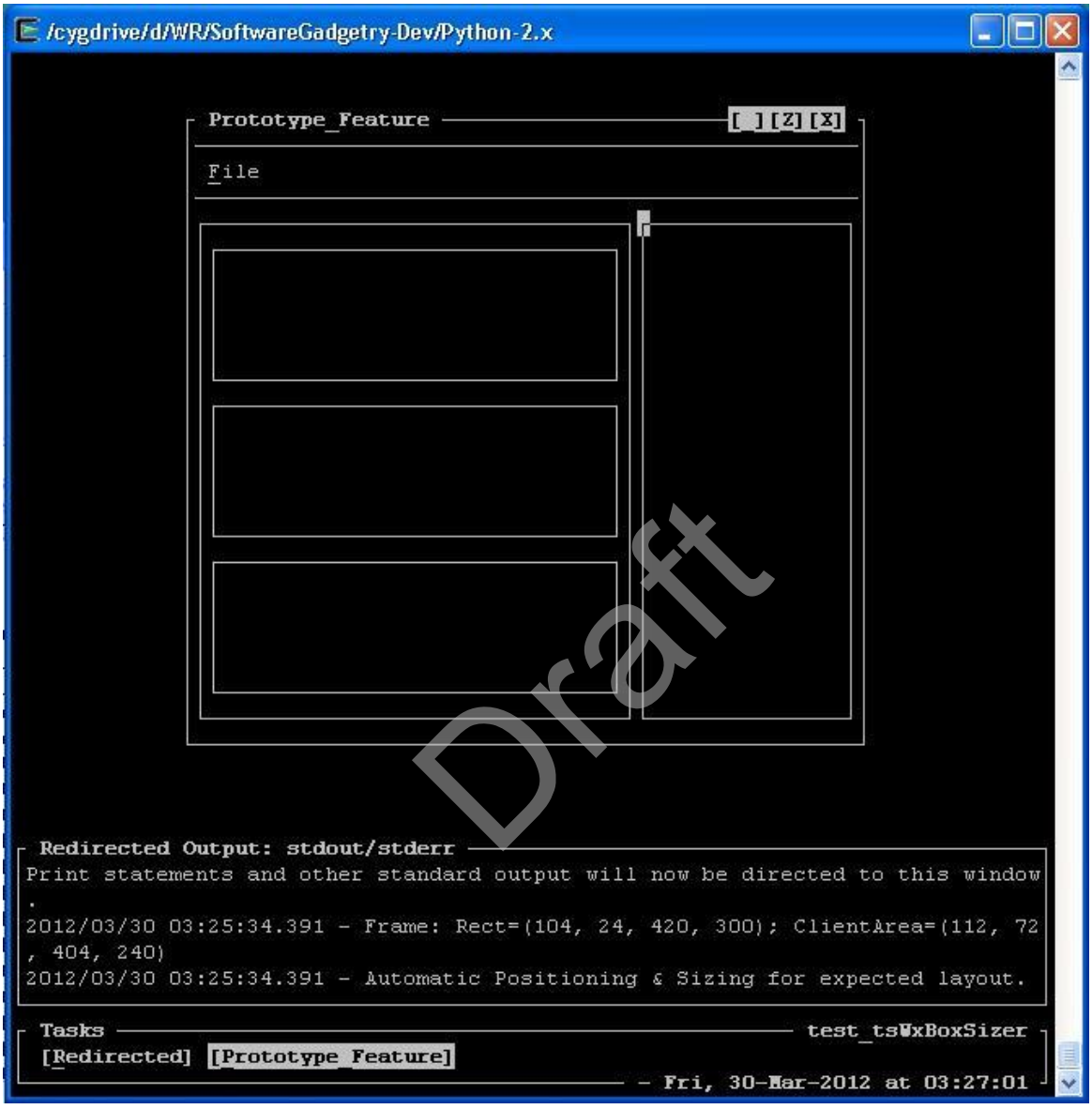
1. *BLACK (phosphor pixel "OFF")*
2. *WHITE, GREEN or ORANGE (phosphor pixel "ON")*

NOTE: The following screen shots demonstrate various widgets on a terminal that reports NOT having colors. It is the same application that runs on terminals that report having colors. The application remains unaware of the presence or absence of colors.

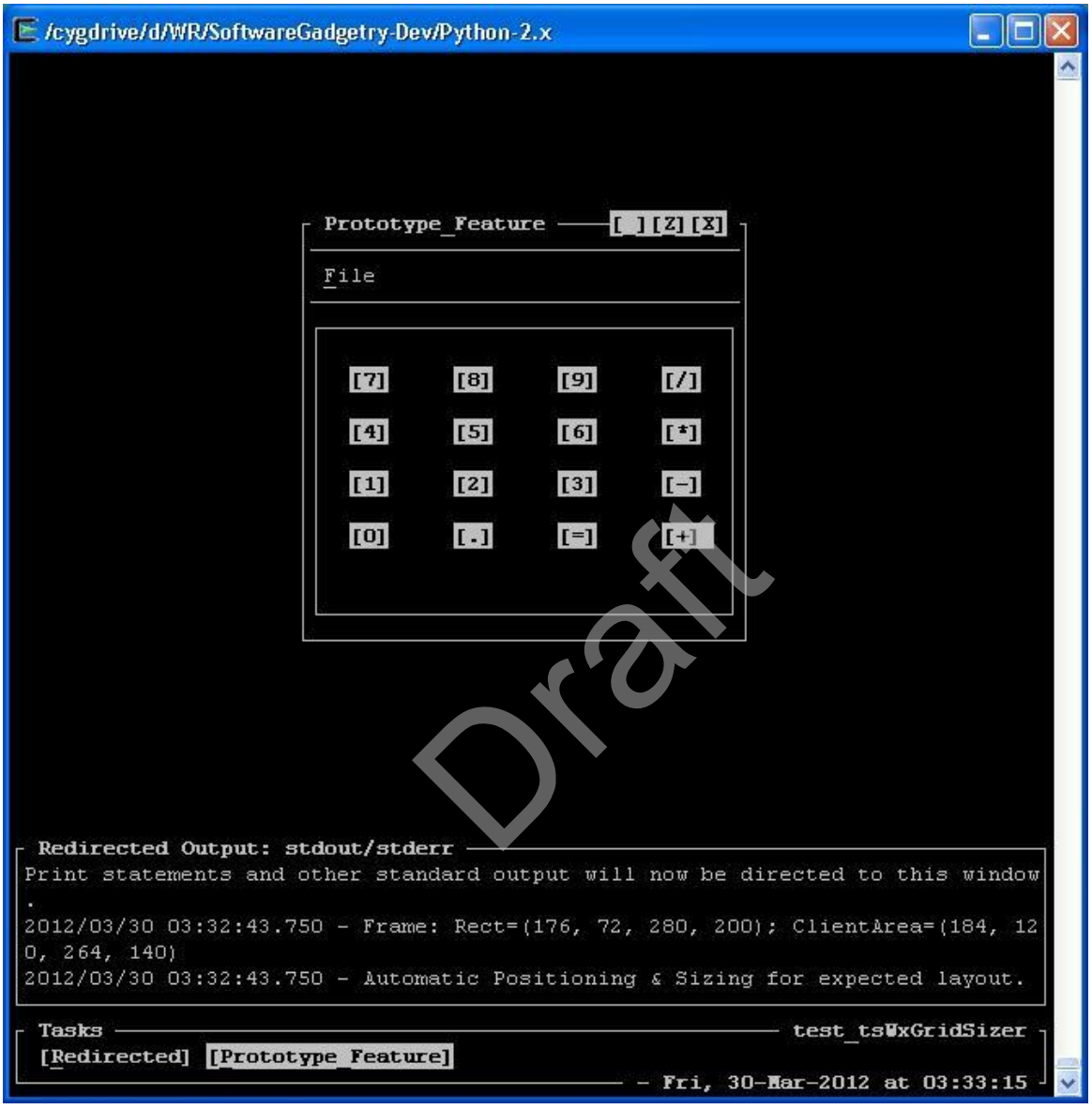
2.4.1 test_tsWxWidgets using vt100 (via Cygwin mintty)



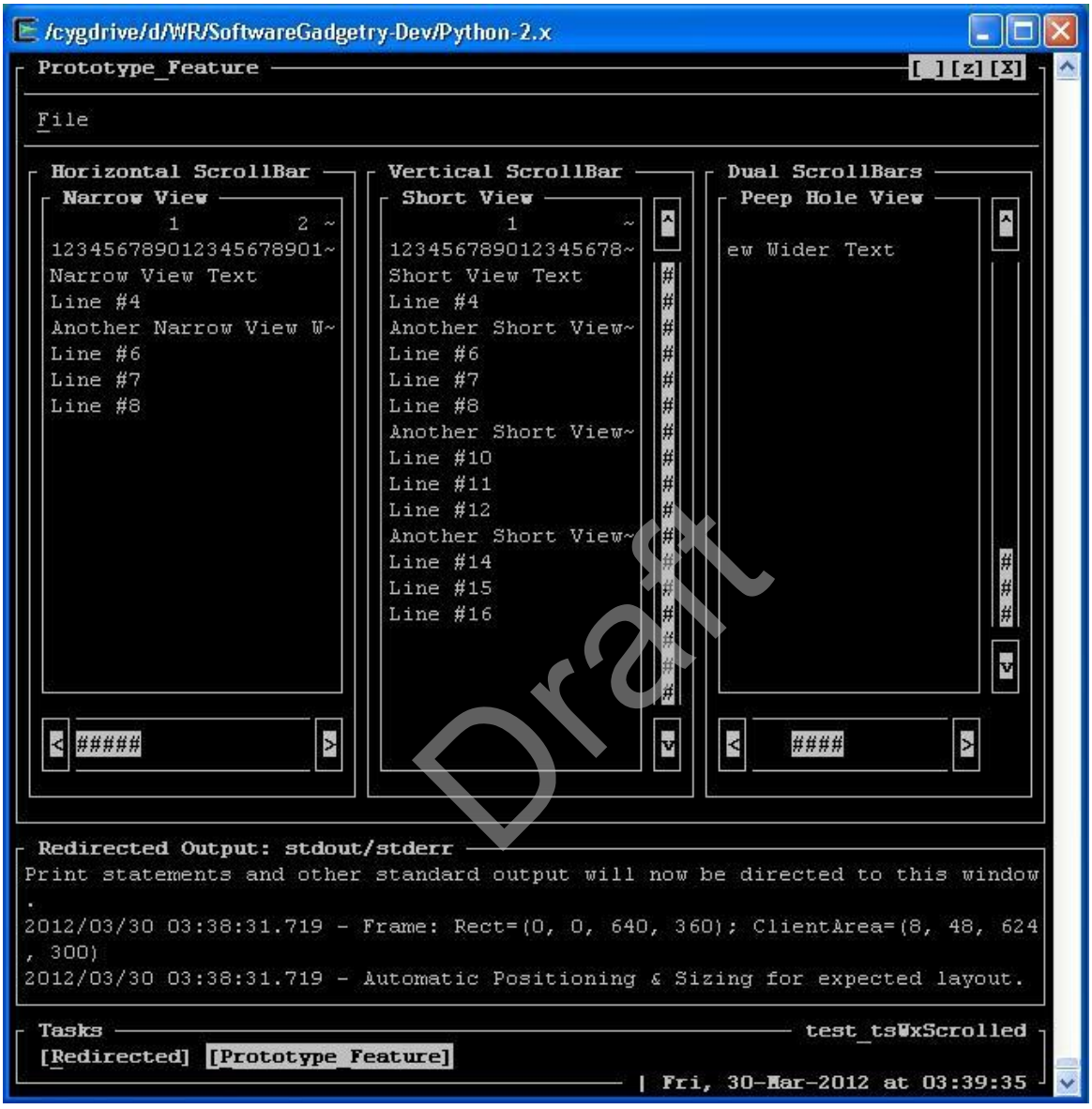
2.4.2 test_tsWxBoxSizer using vt100 (via Cygwin mintty)



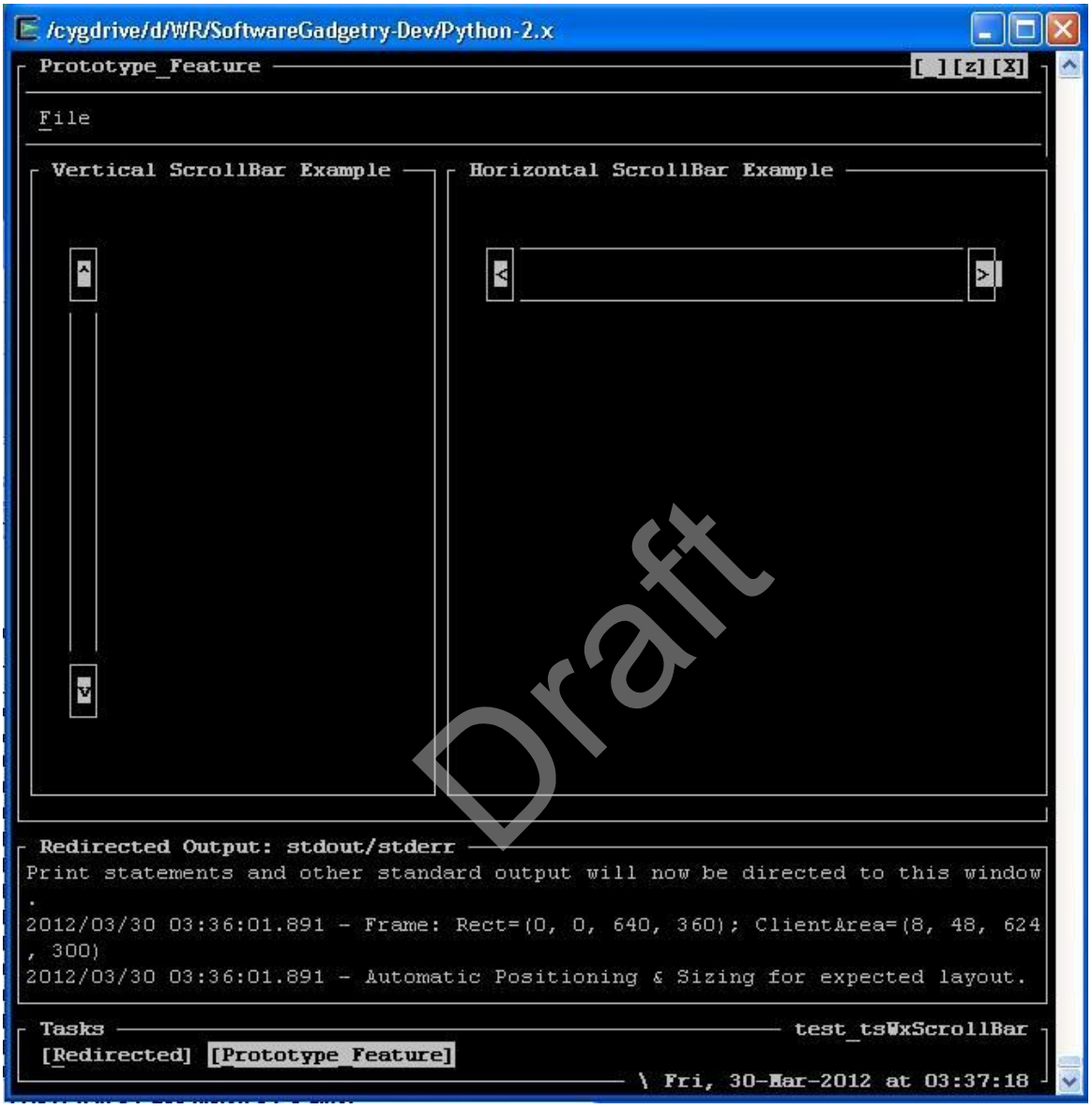
2.4.3 test_tsWxGridSizer using vt100 (via Cygwin mintty)-



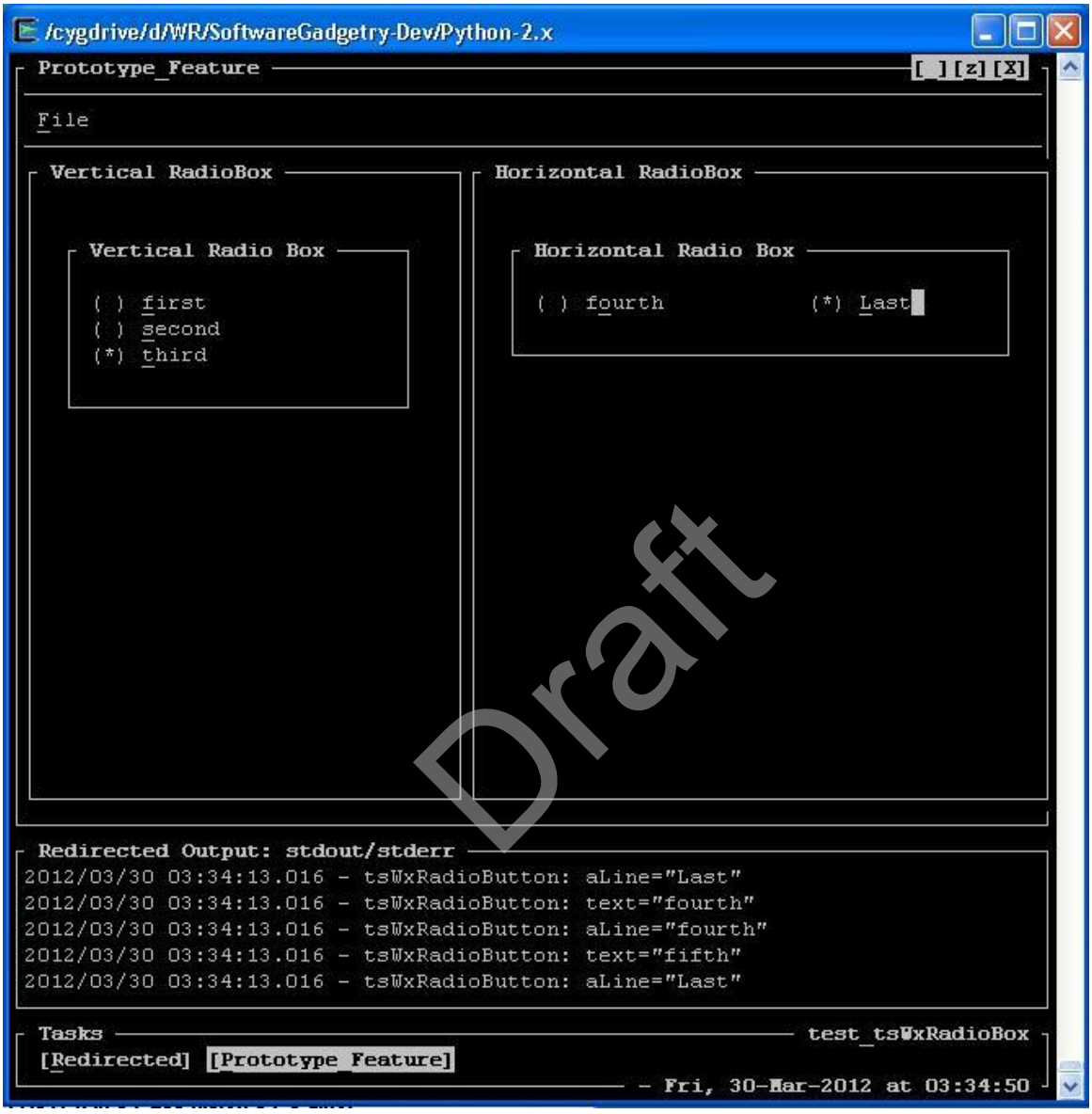
2.4.4 test_tsWxScrolled using vt100 (via Cygwin mintty)



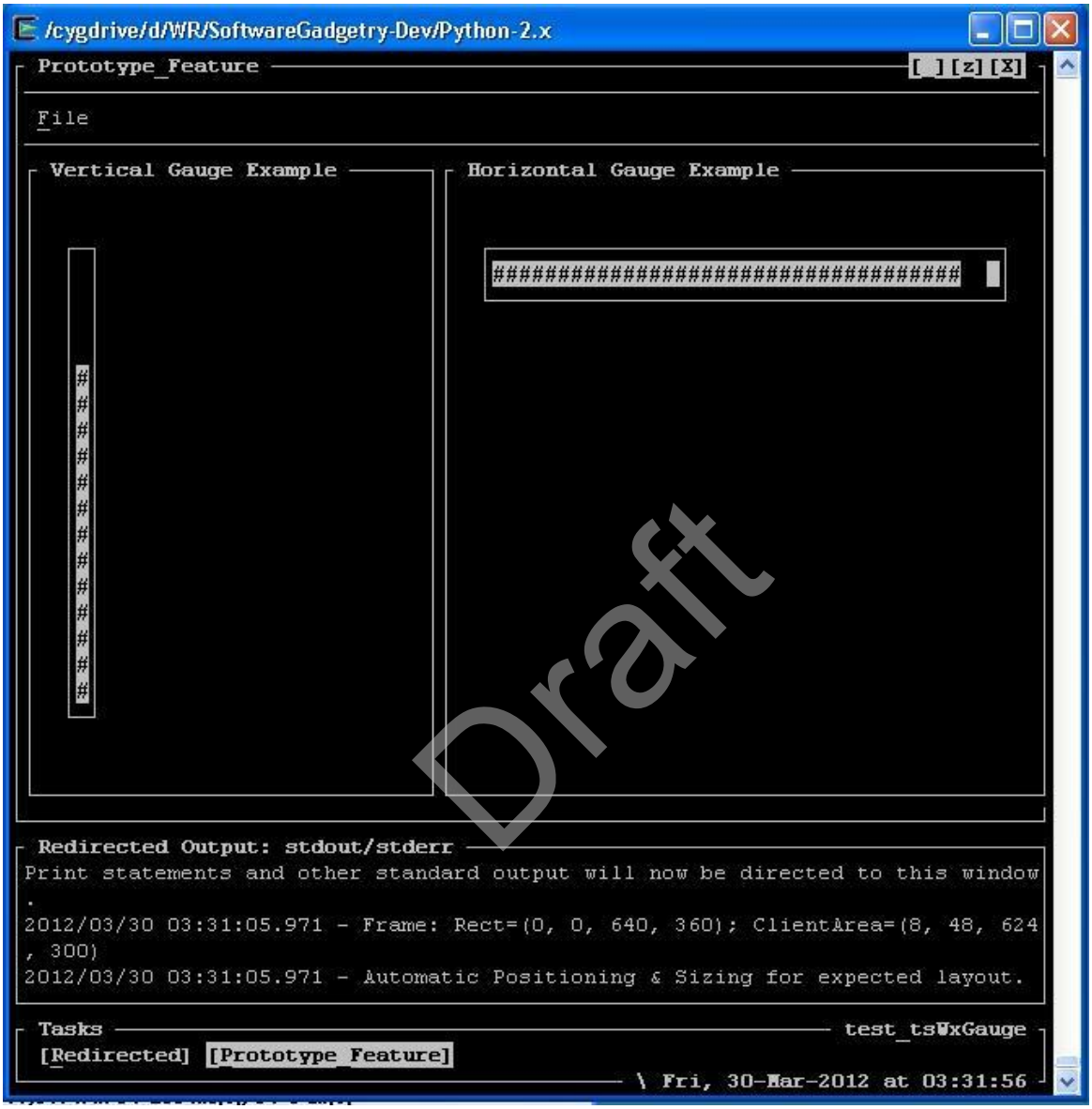
2.4.5 test_tsWxScrollBar using vt100 (via Cygwin mintty)



2.4.6 test_tsWxRadioBox using vt100 (via Cygwin mintty)



2.4.7 test_tsWxGauge using vt100 (via Cygwin mintty)



2.4.8 test_tsWxCheckBox using vt100 (via Cygwin mintty)