An Introduction to Tkinter

Bruno Dufour &
Wen Hsin Chang

Friday, September 7th, 2001

McGill

What is Tkinter ?

- Tkinter is an open source, portable graphical user interface (GUI) toolkit designed for use in Python scripts.
- Tkinter is a python interface for the Tk GUI toolkit (originally developed for the Tcl language)

Advantages offered by Tkinter

- Layered implementation
- Accessibility
- Portability
- Availability

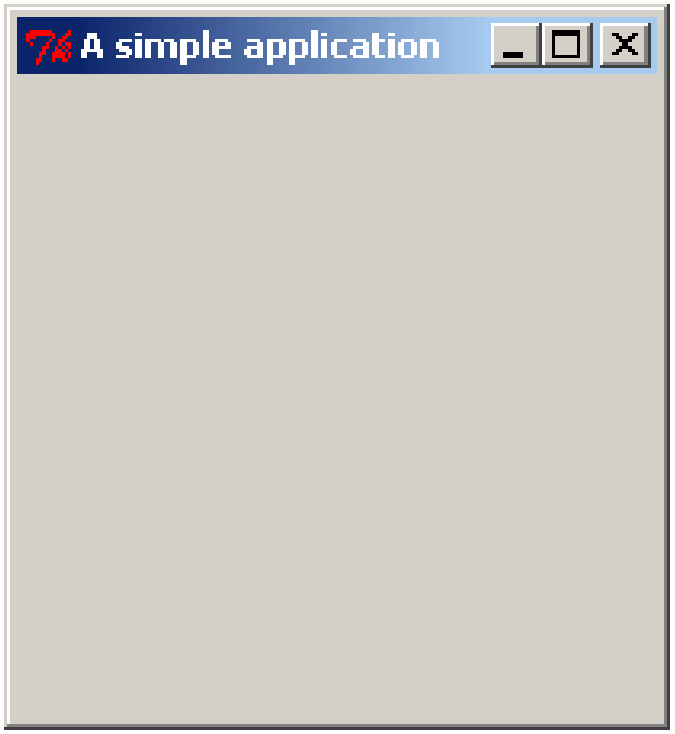
Drawback of Tkinter

- Due to the layered approach used in its implementation, execution speed becomes a concern.

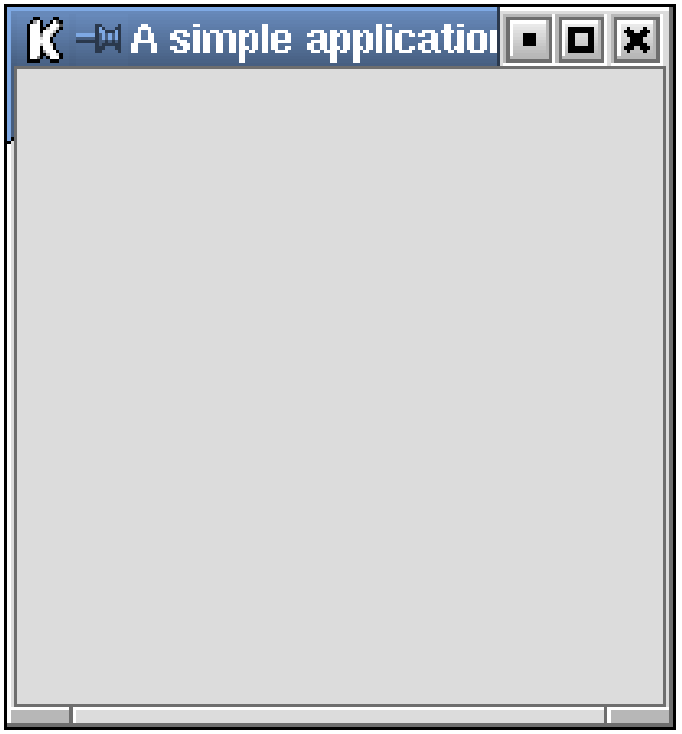
The smallest Tkinter program…

```
from Tkinter import *
root = Tk( )
root.title("A simple application")
root.mainloop( )
```

… and its ouput



Microsoft Windows 2000



Red Hat Linux 7.1 running KDE 2.0

Widgets and Tkinter

- Tkinter's components are called *Widgets*
- Widgets are equivalents to OOP's objects or components
- All widgets inherit from the Widget class

Widget Options

Options are attributes of the widget. Not all widgets have the same attributes. Some which are common to all widgets such as '**text**', specifying the text to be displayed, or '**Padx**', which specifies the space between itself and its neighbor widget. Other options like '**wrap**' for a text widget or '**orient**' for a scrollbar widget are widget specific.

Widget Methods

Methods are mostly widget specific, meaning some widgets such as scrollbar or list box have their own methods to help the user exploits widgets' full functionalities. Methods that is common to all widgets are methods such as 'Configure( )' or 'Keys( )'

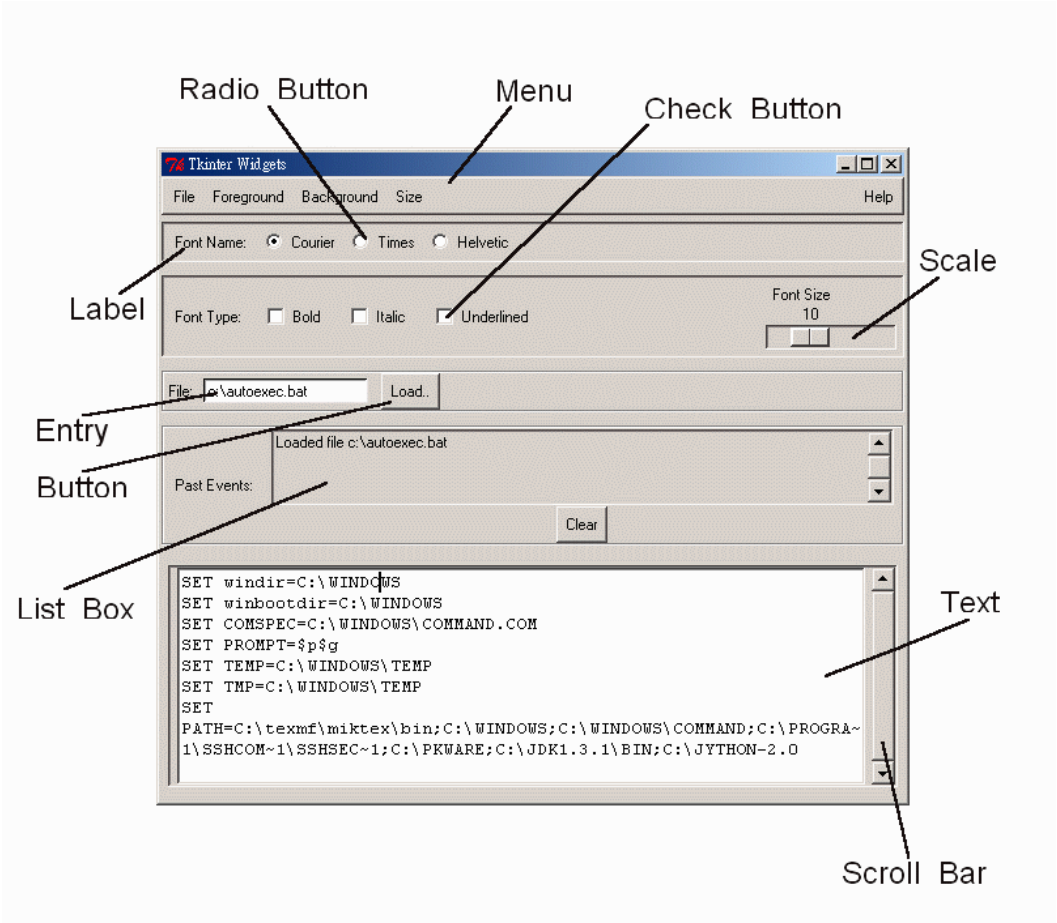
Widget Manipulation

- Widget manipulation is done via *options*
- Options can be set at creation time or later on by calling the `configure( )` method on the widget, with a list of valid widget option IDs and their respective values

Widget Types

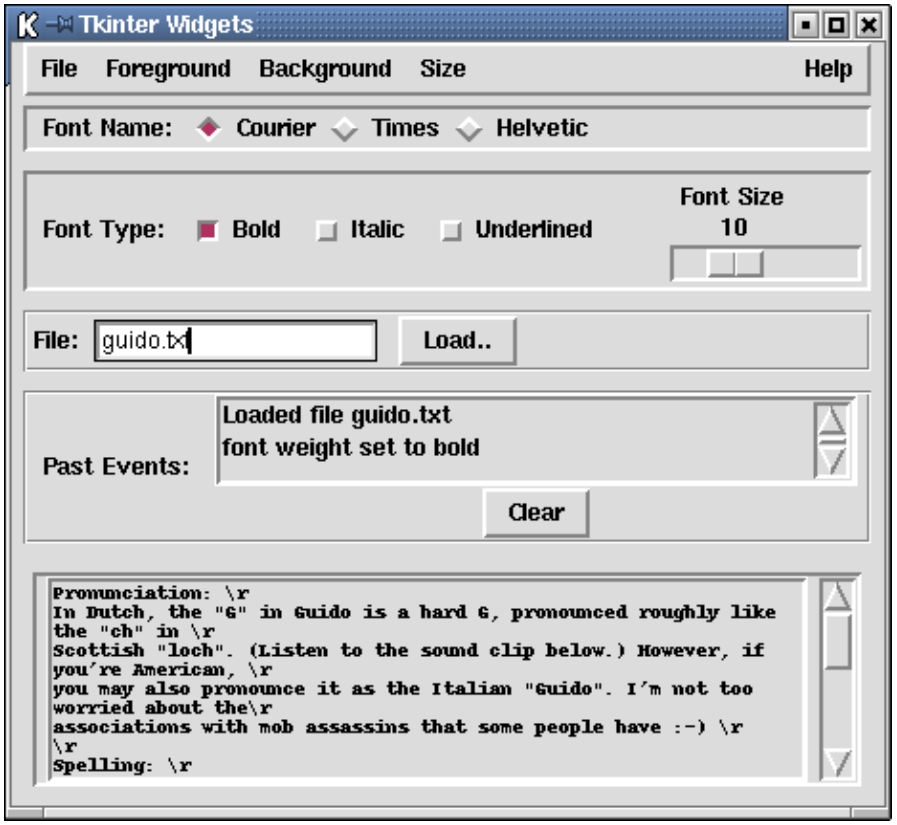
- Toplevel
- Frame
- Label
- Button
- Entry
- Radiobutton
- Checkbutton
- Menu
- Message
- Text
- Scrollbar
- Listbox
- Scale
- **Canvas**

What widgets look like…



Microsoft Windows 2000

… on different Operating Systems



Red Hat Linux 7.1 running KDE 2.0

Screen Layout in Tkinter

- Fonts: specified using a n-tuple
  - (family, size, option1, option2, …)
- Colors: specified using color names (“red”, “blue”, “peachpuff”, etc.) or RGB values in the form (hexadecimal)
  - #RGB
  - #RRGGBB
  - #RRRRGGGGBBBB

Tkinter variables

- Variables can be used as widget options to hold values associated with them (eg. the value option for Radiobuttons)
- Tkinter provides a way for the widget to adjust to a change in the value of such a variable
- This is not possible using standard variables

The need for Tkinter variables

- Tkinter provides the Variable (abstract) class.
- The Variable class provides the possibility of associating a callback method with a variable
- Thus, one could respond to a change in the value of such variables

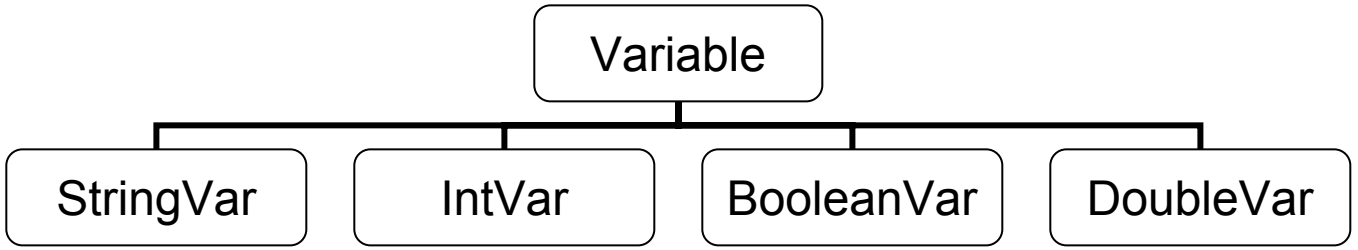
Methods of the Variable class

- Private Methods:
  - __init__(self, master=None)
  - __del__(self)
  - __str__(self)
- Public Methods
  - set(self, value)
  - trace(self, mode, callback)
  - Trace_vdelete(self, mode, cbname)
  - Trace_vinfo(self)

Particularities of Tkinter variables

- The Variable class does not implement the get( ) method (only a base class)
- The set( ) method does not do any type checking (do not expect to catch a type conversion error in a try..except construct)
- The get( ) method will fail if an erroneous data type has been stored in the variable

Subclasses of Variable



Always use one of the subclasses of Variable itself to manipulate data

Geometry management in Tkinter

- Geometry management consists of widget placement and sizing of the screen
- Geometry management increases the portability of the GUI toolkit.
- Tkinter provides 3 geometry managers: Pack, Grid and Place.

The Pack geometry manager

- Quickest and most common way to design interfaces
- Positioning is done relative to the container widgets (top, bottom, left, right)
- Widgets are packed from edge to center of the container, using space left available by previous pack operations.

Options to the `pack( )` method

Option	*Possible Values*
expand	YES / **NO**
fill	**NONE** / X / Y / BOTH
side	**TOP** / BOTTOM / RIGHT / LEFT
in_('in')	Widget
padx, pady	Integer values
ipadx, ipady	Integer values
anchor	N / S / E / W / NW / SW / NE / SE / NS / EW / NSEW / **CENTER**

Methods provided by the Packer

Method	*Effect*
pack(option=value,…), pack_configure(option=value,…)	Packs the widget with the specified options.
pack_forget( )	The widget is no longer managed by the Packer, but is not destroyed.
pack_info( )	Returns a dictionary containing the current options.
pack_slaves( )	Returns a list of widget IDs, in the packing order, which are slaves of the master widget.

The Grid geometry manager

- Used for more complex layouts
- Allows the container to be divided in rows and columns
- Similar to HTML's Table (columnspan + rowspan)
- Using the Packer, one would have to use multiple frames to achieve the same effect

Options to the `grid( )` method

Option	*Possible Values*
row, column	Positive integer values
rowspan, columnspan	Positive integer values
in_('in')	Widget
padx, pady	Integer values
ipadx, ipady	Integer values
sticky	N / S / E / W / NW / SW / NE / SE / NS / EW / NSEW (Note: Default is to center widgets *)

* CENTER is not supported with the sticky option

Methods provided by the Grid

Method	*Effect*
grid(option=value,…), grid_configure(option=value,…)	Places the widget in a grid, using the specified options.
grid_forget( ), grid_remove( )	The widget is no longer managed by the Grid, but is not destroyed.
grid_info( )	Returns a dictionary containing the current options.
grid_slaves( )	Returns a list of widget IDs which are slaves of the master widget.
grid_location(x, y)	Returns a tuple (column, row) which represents the cell in the grid that is closest to the point (x, y).
grid_size( )	Returns the size of the grid, in the form of a tuple (column, row)

Special notes about the Grid

- Empty rows and columns are not displayed by the grid geometry manager, even if a minimum size is specified.
- The grid manager cannot be used in combination with the pack manager, as this results in an infinite negociation loop.

The Place geometry manager

- Most powerful manager
- Allows exact placement of widgets in a container
- Allows placement of widgets using either exact coordinates, or as a percentage relative to the size of the master window (expressed as a float in the range [0.0, 1.0]).
- The same holds for the widget size.

Options to the `place( )` method

Option	*Possible Values*
anchor	N / NE / E / SE / SW / W / **NW** / CENTER
bordermode	INSIDE / OUTSIDE
in_('in')	Widget
relwidth, relheight	Float [0.0, 1.0]
relx, rely	Float [0.0, 1.0]
width, height	Integer values
x, y	Integer values

Event Handling in Tkinter

- Easy, convenient and flexible
- Allows callback functions to be associated with any event for any widget
- Event descriptors are used to identify events

Event Descriptors

- String representation of events
- Used for binding callbacks to events
- General form: <Modifier- Type - Qualifier>
- Not all 3 sections are required for an event descriptor to be valid (the type alone often suffices).

Event Types in Tkinter

Tkinter can handle the following event types:

- *Keyboard events:* KeyPress, KeyRelease
- *Mouse events:* ButtonPress, ButtonRelease, Motion, Enter, Leave, MouseWheel
- *Window events:* Visibility, Unmap, Map, Expose, FocusIn, FocusOut, Circulate, Colourmap, Gravity, Reparent, Property, Destroy, Activate, Deactivate

Event Qualifiers

- Can be either:
  - Mouse button index (1 to 5)
  - Keysym: the name of a particular key (eg: “backslash”, “backspace”)
- A type does not have to be specified when a qualifier is used (can still be done though)

Event Modifiers

- Possible Modifiers:
  - *Control, Shift, Alt, Meta*: Modifier keys
  - *B1 to B5*: Mouse button modifiers
  - *Double, Triple*: Repetition modifiers
  - *Any*: specifies to execute the callback regardless of the modifiers
- Any number of modifiers can be specified
- Order of modifiers is irrelevant (eg: <Control-Alt-Shift-A>)

Event Attributes

Attribute	*Description*
serial	Serial # of the event
num	number of the mouse button pressed (ButtonPress, ButtonRelease) (1=LEFT, 2=CENTER, 3=RIGHT, etc.)
focus	boolean which indicates whether the window has the focus (Enter, Leave)
height / width	Height / width of the exposed window (Configure, Expose)
keycode	keycode of the pressed key (KeyPress, KeyRelease)
state	state of the event as a number
time	Time at which the event occurred. Under Microsoft Windows®, this is the value returned by the GetTickCount( ) API function.

Event Attributes (cont.)

Attribute	*Description*
x / y	x / y – position of the mouse relative to the widget
x_root / y_root	x / y-position of the mouse on the screen relative to the root
char	pressed character (as a char) (KeyPress, KeyRelease)
keysym	keysym of the the event as a string (KeyPress, KeyRelease)
keysym_num	keysym of the event as a number (KeyPress, KeyRelease)
type	type of the event as a number
widget	widget for which the event occurred
delta	delta of wheel movement (MouseWheel)

Binding callbacks to Events

- 3 method calls:
  - `bind()`: can be called on any widget, and in particular a Toplevel widget
  - `bind_class()`: used internally in order to provide standard bindings for Tkinter widgets. Can be avoided by subclassing strategy.
  - `bind_all()`: binds events to the whole application

Callbacks and events

- Tkinter always uses the most specific event descriptor for a given event and a given widget
- Callbacks for the 4 different levels of Tkinter's event handling will be called in sequence, starting with the widget level, then the Toplevel, the class and then the Application.
- If, at any given level, one wants to stop the propagation of the event, simply return "break" in the callback associated with this event.

The Canvas widget

- Provides basic drawing facilities, as well as advanced drawing features
- Drawing is done by creating canvas *items*
- Items are **not** widgets, even though they are handled in a similar way
- Each item receives a unique ID upon creation
- Each item is enclosed in its bounding box specified by a top-left corner and a lower-right corner

Items supported by the Canvas

- Arc: arc, chord, pieslice
- Bitmap
- Image
- Line
- Oval: circle or ellipse
- Polygon
- Rectangle
- Text
- Window: used to place other **widgets** on the canvas (eg buttons)

Tkinter Canvas Options

Option	*Possible Values*
closeenough	Float
confine	Boolean
scrollregion	List of 4 coordinates
xscrollcommand, yscrollcommand	Function
xscrollincrement, yscrollincrement	Distance

Manipulating Items

- Item creation functions (create_line( ), create_oval( ), etc.) all return the item ID of the newly created canvas item
- The itemconfigure( ) method is used to configure canvas items after their creation

(See Tkinter documentation or Online Presentation for more information)

Tkinter coordinate systems

- 2 coordinate systems:
  - Canvas coordinate system: origin at the top-left corner of the canvas (may not be visible)
  - Window coordinate system: origin at the top-left corner of the visible portion of the canvas
- Event objects use the Window coord. sys.
- canvasx( ) and canvasy( ) methods can convert coords to Canvas coord. sys.

Canvas and Tags

- Tags are strings that can be associated with any canvas item
- More than one item can have the same tag, and a single item can have multiple tags
- This allows to create groups of items
- Canvas items can be interchangeably referenced by ID (integers) or tags.

Canvas and Tags (cont.)

- What happens to function which only take one item as parameter?
- Tkinter provides a very good approach: the first (lowest) item in the display list that matches the tag is used
- Binding can also be done on canvas items by using the tag_bind( ) and tag_unbind( ) functions

Special Tags

- CURRENT (“current”): the item that is currently situated under the mouse cursor is automatically assigned the CURRENT tag. (Note: don’t use this tag manually!!)
- ALL (“all”): this special tag matches all items in the canvas

To obtain more information:

- Presentation web site: http://pages.infinit.net/bdufou1/
- Official Python website: http://www.python.org/
- John E. Grayson. Python and Tkinter programming. Manning, 2000.