CIS192 Python Programming

Graphical User Interfaces

Robert Rand

University of Pennsylvania

April 20, 2016

Outline

Tkinter

- The module `Tkinter` is a wrapper of the graphics library Tcl/Tk
- Why choose `Tkinter` over other graphics modules
  - It's bundled with Python so you don't need to install anything
  - It's fast
  - Guido van Rossum helped write the Python interface
- The docs for `Tkinter` aren't that good
  - The docs for Tk/Tcl are much better
  - Tk/Tcl functions translate well to `Tkinter`
  - It's helpful to learn the basic syntax of Tk/Tcl
- Tk/Tcl syntax $\rightarrow$ Python:
  - `class .var_name -key1 val1 -key2 val2` $\rightarrow$ `var_name = class(key1=val1, key2=val2)`
  - `.var_name method -key val` $\rightarrow$ `var_name.method(key=val)`

Bare Bones Tkinter

```
from Tkinter import Frame


class SomeApp(Frame):
    def __init__(self, master=None):
        tk.Frame.__init__(self, master)


def main():
    root = tk.Tk()
    app = SomeApp(master=root)
    app.mainloop()

if __name__ == '__main__':
    main()
```

Images

- By default, Tkinter only supports bitmap, gif, and ppm/pgm images
- More images are supported with Pillow
- Pillow is a fork of Python Imaging Library `pip install pillow`
- `from PIL import Image, ImageTk`
- Create a PIL image with
  `img = Image.open('path_to_img')`
- Make a Tk image with `tk_img = ImageTk.PhotoImage(im)`
- Set it as an attribute in `Tkinter` with `b['image'] = tk_img`

Widgets

- `Tkinter` has a bunch of widgets
  - Button, Label, Listbox, Radiobutton
- Create a widget with `b = Button(parent)`
  - `parent` is the containing widget
- Options can be accessed and set dictionary style
  - `b['text'] = 'Press Me'`
  - equivalently: `b = Button(parent, text='Press Me')`

Placing Widgets

- Just creating a widget will not display it
- The widget must be told where to go in the parent widget
- Grid placement: `my_widget.grid(row=r, column=c)`
  - Best to specify the grid layout beforehand.
- Absolute placement: `my_widget.place(x=c1, y = c2)`
  - May conflict with other objects.
  - Will ignore window resizing.
- Packing: `my_widget.pack(side = SIDE)`
  - Will be placed relative to other widgets.
  - Can specify `padx` and `pady` padding.

Handlers

- A Widget can have a registered Event Handler
- The handler is a function that gets called when the widget is used
- Register a handler for a widget:
  ```
  b['command'] = some_function
  ```
- Handlers do not automatically say what widget was used
  - Use a lambda to partially apply arguments
    - ★ `b['command'] = lambda w: button_handler(w)`
  - Or use `from functools import partial`
    - ★ `b['command'] = partial(button_handler, w)`

Other GUIs

- WxPython
  - Similar to `Tkinter` in that it wraps an existing library
  - Wraps the C++ wxWidgets
  - A little bit more user-friendly
- PyQT and PySide
  - Python bindings for the Qt cross-platform application and UI framework
  - PyQT is commercial software, PySide is open source
- PyGTK
  - Implements GTK+ (originally the “GIMP Toolkit”).
  - Requires a separate GTK+ install.

Outline

curses

- `curses` is a wrapper around the ncurses library
- ncurses is the standard for terminal graphics
  - Is terminal independent (XTerm, Command Prompt, ...)
  - Treats screen as a grid of characters
  - Pretty low-level
- An `curses` program runs in your current terminal
  - Not a new window
  - Debugging with `print` statements can cause weird behavior
  - Changes made in the program can persist after termination
  - `curses.wrapper` ensures that clean-up happens on termination

Bare Bones curses

```
import curses

class SomeApp(object):
    def __init__(self, stdscr):
        self.stdscr = stdscr

    def run(self):
        while True:
            key = self.stdscr.getch()

def main(stdscr):
    app = SomeApp(stdscr)
    app.run()

if __name__ == '__main__':
    curses.wrapper(main)
```

Wrapper Explained

- `wrapper(main)` executes `main(stdscr)` in a `try/except`
- `stdscr` is an initialized curses `WindowObject`
- The initialization includes:
  - cbreak: Buffering is turned off (But Ctrl-C still works)
  - no echo: Typed characters are not displayed on screen
  - colors: If the terminal supports colors they are initialized
- before exiting the settings are reset

Writting Strings

- A WindowObject is a uniform grid of characters
- Given a WindowObject `w`
- `w.addstr(row, column, some_string)` will write `some_string` to the window starting at `(row, column)`
- Overwriting a section of a window will only replace those characters
  - Use `w.clear()` to clear the entire window
- For the effects of a write or clear to take effect
  - `w.refresh()` repaints the window
  - `w.noutrefresh()` marks the window for update
  - `w.doupdate()` actually repaints the screen
  - `w.refresh()` marks the current window and repaints all marked windows

Creating Windows

- A window is basically a name for a rectangle of the screen
- `curses.newwin(height, width, r, c)`
  creates a window starting at row=r and column=c
- Windows allow parts of the screen to be refreshed separately
- Windows give a new coordinate system with (0, 0) in the top-left
- A panel is a window with depth
- You can overlap panels without overwriting other panels data

Setting Attributes

- When writing a string you can specify Attributes
- Background/Foreground color pairs:
  - `red = curses.COLOR_RED`
  - `black = curses.COLOR_BLACK`
  - `curses.init_pair(1, red, black)`
  - `w.addstr(0, 0, 'some_text', curses.color_pair(1)`
- Bolding and Highlighting:
  - `w.addstr(0, 0, 'other_text', curses.A_BOLD)`
  - `w.addstr(0, 0, 'other_text', curses.A_BLINK)`
- Attributes are not guaranteed to mix well but multiple can be specified
- Attributes can be applied to entire windows
  - `w.bkgd(0, curses.A_STANDOUT)`

Handling Input

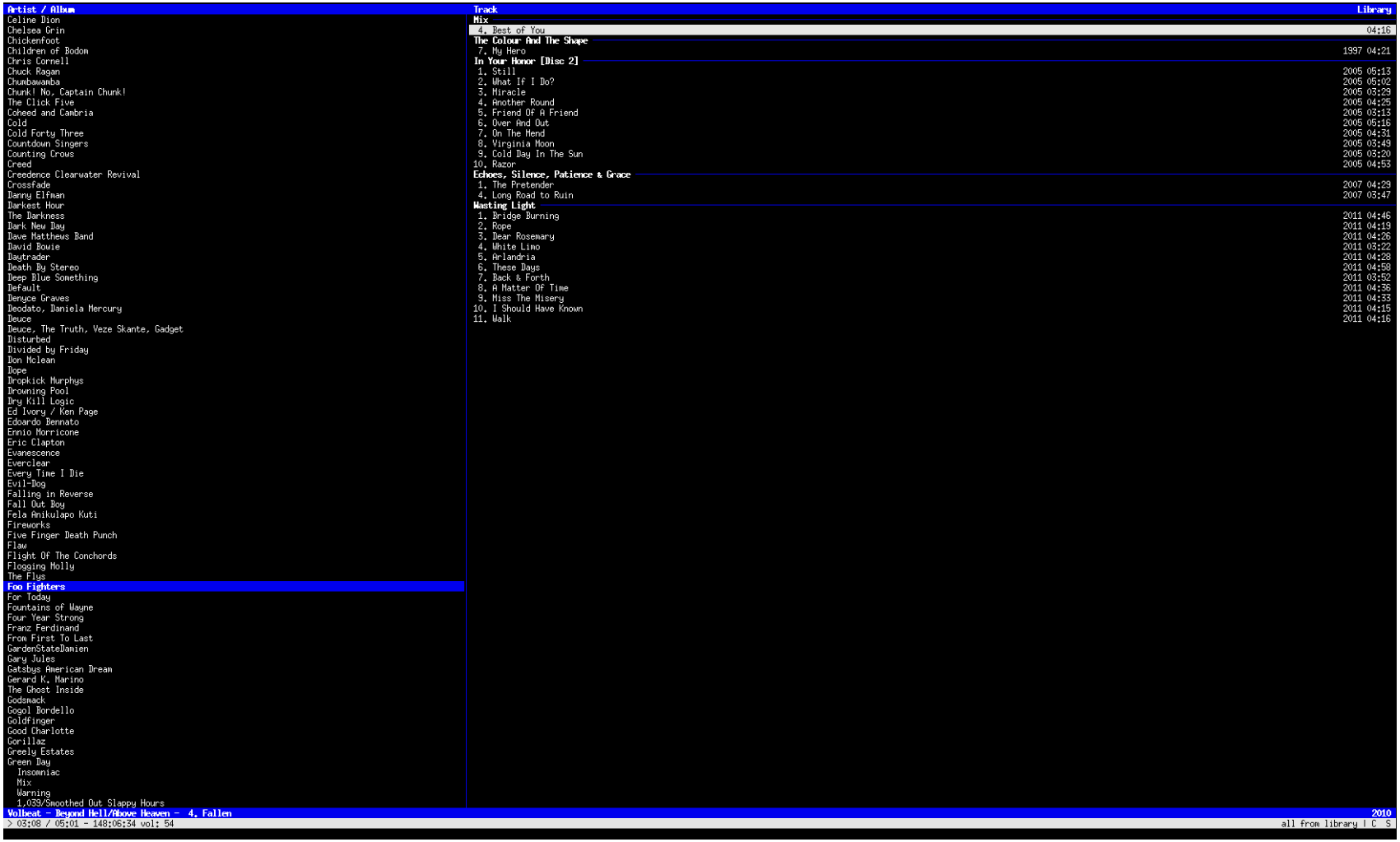
- `key = w.getch()` waits for a key to be pressed
- The return value is an integer representing the character
- Compare against constants to detect special keys
  - `if key == curses.KEY_RIGHT:`
- `w.getkey()` will return a string instead of an integer

Control Logic

- `curses` is very low-level
- Minimal abstraction (Rectangles of characters)
- No notion of event handlers
  - All key-presses and mouse clicks must be explicitly directed
- You are in charge of all state

Example: cmus

- a small fast music player written in curses



Urwid

- Urwid is a text widget library
- Has more abstraction (Widgets instead of blocks of text)
- If your UI is that complex just use a GUI