Quickly Creating Professional Looking Application Using wxPython, py2exe and InnoSetup

Miki Tebeka
mtebeka@qualcomm.com
About Me

- Software Process Engineer in Qualcomm Israel
- Started using Python around 1998
- Use Python wherever I can
  - Currently around 90%+ of my code is in Python
- Written from small scripts to a linker and a source level GUI debugger
- Little activity in Python development
  - Also wxPython, PLY, ...
Why Should You Provide GUI + Installer?

- Users expect certain way of using and installing programs
  - Shockingly, not all of them like the command line
- Good for your karma
  - It will make you look good with little effort
  - It will sell Python better for your next project
- Wrapping *is* important
  - Just make sure the content is good as well :)


About This Talk

- I'll try to show that it is very simple to create a professional looking application
- We'll write a demo “untar” program
- It can serve you as a template for similar projects
wxPython

- Can be found at www.wxPython.org
- Python bindings for www.wxwidgets.org
  - Cross platform GUI library
  - C++
  - Borland next GUI framework
- Very rich set of widgets
  - Keeps growing all the time
- Killer demo
  - Sometimes I just cut & paste from it
- Has several GUI designers
Library Tour

wxPython

http://wxPython.org/

Cross-Platform GUI Library
Basic Concepts

- Create a container window
- Use a sizer to place child widgets
  - Very much like Tcl/Tk `pack`
  - There are other layout options
- Bind event to actions using `Bind`
  - You are aware that GUI programming is event based?
- Create `wxPySimpleApp` to run the application
Demo Application - UnTar

- Open a tar file to a given directory
- Checks for user input validity ...
GUI Design

- Draw it in boxes by hand
sizer = wx.BoxSizer(wx.VERTICAL)  # Main sizer

# Tar File: _____ [Browse]
self._filename = ...
sizer.Add(self._filename, 0, wx.EXPAND)

# Output Directory: _____ [Browse]
self._outdir = ...
sizer.Add(self._outdir, 0, wx.EXPAND)

# ---------
sizer.Add(wx.StaticLine ...)
# [Extract] [Cancel]
hsizer = wx.BoxSizer(wx.HORIZONTAL)
b = ...
hsizer.Add(b)
hsizer.Add(((1,1), 1, wx.EXPAND))  # Spacer
hsizer.Add(wx.Button(self, wx.ID_CANCEL))
sizer.Add(hsizer, 1, wx.EXPAND)

# Layout window
self.SetSizer(sizer)
sself.SetAutoLayout(1)
sizer.Fit(self)
FileBrowseButton
(from wx.lib.filebrowsebutton)

FILEMASK = "Tar Files|*.tar;*.tgz;" \ 
  "*.tar.gz;*.tar.bz2|" \ 
  "All Files|*.*"

self._filename = 
  FileBrowseButton(self, -1, 
    labelText = "Tar File:", 
    fileMask = FILEMASK, 
    fileMode = wx.OPEN | 
      wx.FILE_MUST_EXIST, 
    size=(WIDTH, -1))
b = wx.Button(self, -1, "Extract")

self.Bind(wx.EVT_BUTTON,
    self.OnExtract, b)

def OnExtract(self, evt):
    '''Handle "Extract" click'''
    ...

Add An Icon

- So they will remember it
- Choose something easy and colorful
- Take a look at www.openclipart.org
  - More than 2489 images in public domain

```python
if isfile(iconfile):
    icon = wx.Icon(iconfile,
                   wx.BITMAP_TYPE_ICO)
self.SetIcon(icon)
```
# MAIN
if __name__ == "__main__":
    app = wx.PySimpleApp()
    dlg = UnTarDlg()
    dlg.ShowModal()
    dlg.Destroy()
py2exe

- Can be found at www.py2exe.org
- Packs all needed scripts and dynamic libraries in one place
- Adds an executable to run the program
- Not 100% compatible to running `python myscript.py`
  - See `appdir` in `untar.py`
- Can embed icon in executable
- Output is in `dist` directory
- Windows only :(
  - Check out `cx_Freeze` for other platforms
from distutils.core import setup
import py2exe

setup(
    windows = [
        {
            "script": "untar.py",
            "icon_resources": [(1, "tar.ico")]
        }
    ]
)
Use an Installer

- Why can't we just ship the a zip file?
  - You can
  - However users are used to a certain way of installing new applications

- There are many out there
  - InnoSetup (which we'll use)
  - NSIS (from the guys who gave us Winamp)
  - WiX (open source from Microsoft)
  - ...
InnoSetup

- Can be found at www.jrsoftware.org/isinfo.php
- Actively developed with a helpful community
- Pascal based scripting
- Can create uninstaller
- Can install from command line
InnoSetup Script

[Setup]
AppName = UnTar
AppVerName = UnTar version 0.1
DefaultDirName = {pf}\UnTar
DefaultGroupName = UnTar
OutputBaseFilename = UnTarSetup

[Files]
Source: "dist\*"; DestDir: {app}
Source: "tar.ico"; DestDir: {app}
[Icons]
Name: "\{group\}\UnTar"; FileName: "{app}\untar.exe"
Name: "\{group\}\UnInstall"; FileName: "{app}\{uninstalle.exe}"

Putting It all Together

- In a Makefile of course :)  
  - Can get one from
    - http://www.gnu.org/software/make/
    - http://www.cygwin.com/
    - http://unxutils.sf.net/
    - http://www.mingw.org/

- I use two utilities to find where python and InnoSetup are installed

- Also used for cleanup
To Sum Up

- Created a GUI based application with installer
- Took me less than two hours
- All in 281 LOC
  - Including comments
  - Including Makefile and other supplementary files
- Small price to impress your users
Resources

- WxPython
  - www.wxpython.org
- py2exe
  - http://starship.python.net/crew/theller/py2exe/
- InnoSetup
  - http://www.jrsoftware.org/isinfo.php
- OpenClipart
  - http://www.openclipart.org/
Questions?