# PyGTK, PyQT, Tkinter and wxPython comparison

*Guilherme Polo*

Python offers a multitude of GUI toolkits, much more than described here, for assisting on development of graphical applications, and by having so many options available the chances are you will be unable to make a good decision. A good decision would be one that fulfill your requirements, and in order to achieve this it is necessary an understanding of choices available.

If you are just beginning GUI development, sometime you will need to select a toolkit and a lot of questions will eventually pop up, so I expect this article to help you making a sane decision. And if you already do GUI development and are considering learning another toolkit, or maybe you are moving from another language to Python, I, again, expect this text to help you choose your next tool.

Writing about every possible point of comparison is not possible (maybe it would fit in a book, a large one), so I have chosen to talk about some topics that you may face in your role as GUI developer. Options will be given, and you will balance them according to your requirements.

Each toolkit presented here has both strong and weak points, and it is, in fact, up to you to decide which one fits your needs. All the four major GUI toolkits available for Python are discussed on this text: PyGtk[1], PyQt[2], Tkinter, wxPython[3].

## Look and Feel

The main task of a GUI developer is to build applications that are easy to use and, to achieve that, they have to be designed to be familiar. Knowing your users is a big step towards success, and this is not really easy to achieve, it involves research, experience, dedication, effort, and goes on.... Here follows a good paragraph that will remind you why it is important to know your users, taken from "Designing Interfaces, By Jenifer Tidwell. Chapter 1: What Users Do":

> *It starts with an understanding of people: what they're like, why they use a given piece of software, and how they might interact with it. The more you know about them, and the more you empathize with them, the more effectively you can design for them. Software, after all, is merely a means to an end for the people who use it. The better you satisfy those ends, the happier those users will be.*

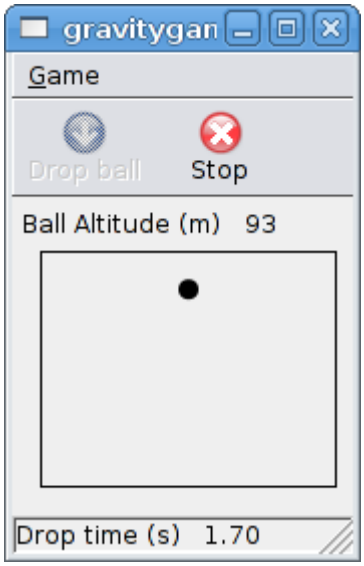
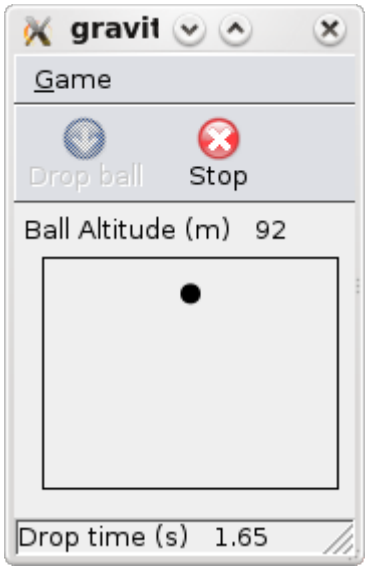
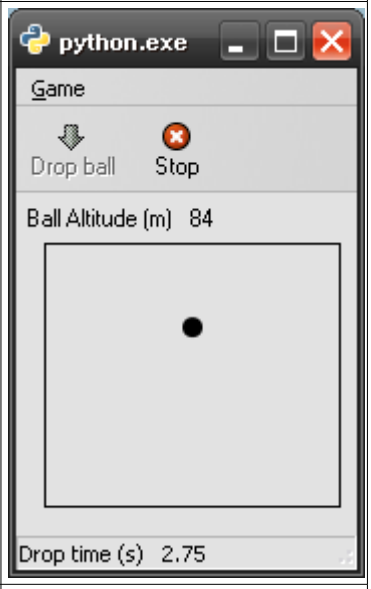
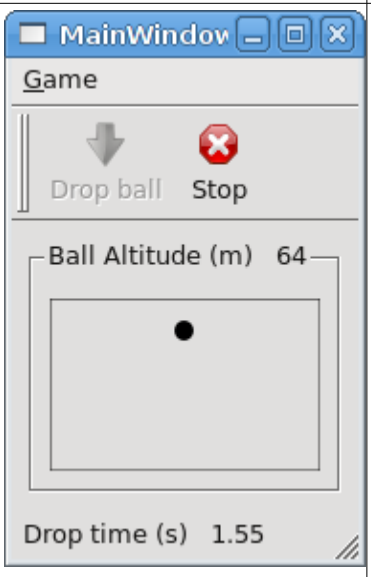
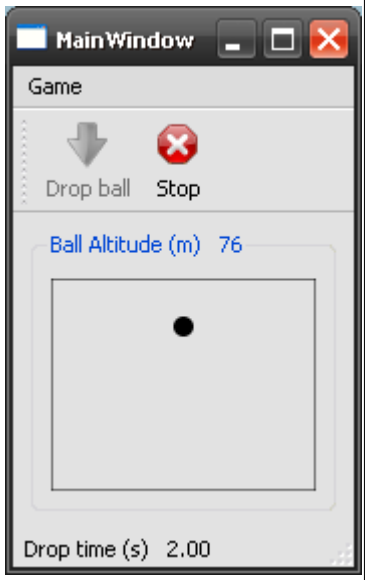
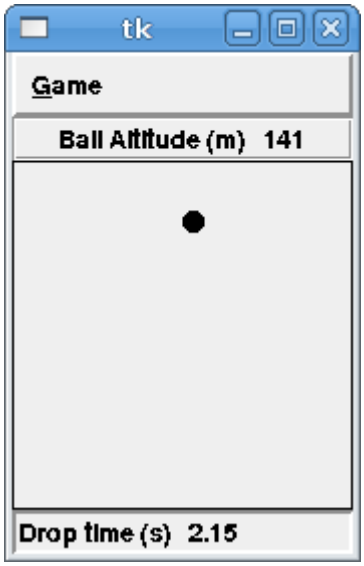
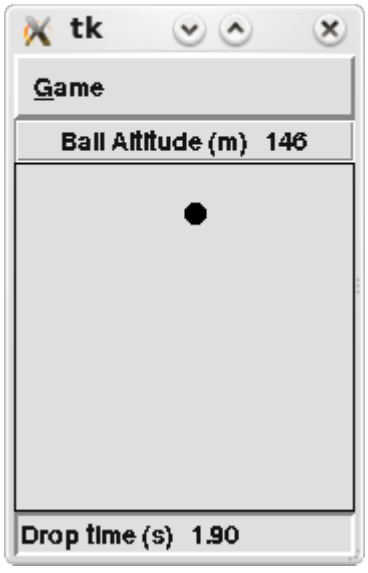
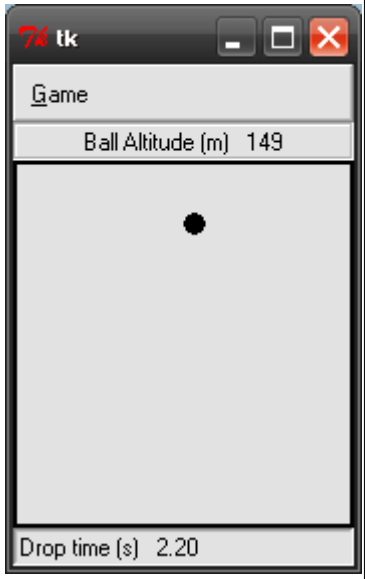
As a developer it should be interesting to follow some guidelines to make it easier to develop pleasant interfaces. Namely there is the Gnome HIG[4] (Human Interface Guidelines) and Apple HIG[5] that are widely used. Both describes good design tips that you should use while developing user interfaces, independent of what platform you are planning to focus on.

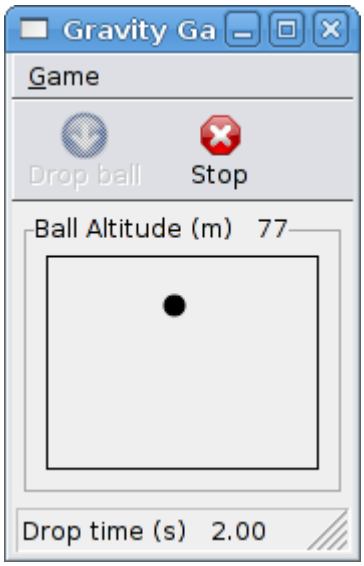
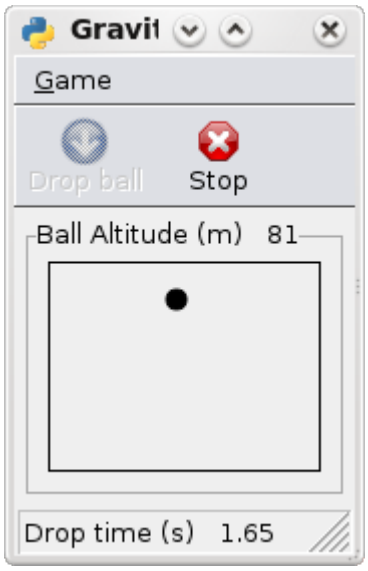
If you are going to use any of the toolkits presented here, it is good to know that your application will not look the same across platforms and may not feel the same. If you want this kind of thing, you should be looking for Lightweight GUI toolkits. They provide uniform behavior on all platforms with the disadvantage of slower execution.

By now, I believe it is important to show some screen shots. They were taken in three different platforms and should give an idea of how different your applications may look

1 PyGtk site: http://www.pygtk.org/
2 PyQt site: http://www.riverbankcomputing.co.uk/pyqt/
3 wxPython site: http://www.wxpython.org/
4 Gnome HIG documentation: http://developer.gnome.org/projects/gup/hig/
5 Apple HIG documentation: http://developer.apple.com/documentation/UserExperience/Conceptual/OSXHIGuidelines/XHIGIntro/chapter_1_section_1.html

across platforms. This is too a first chance to select which one is more likely to fit your users.

| | Gnome 2.20 | KDE 4 | Windows XP |
|---|---|---|---|
| PyGtk |  |  |  |
| PyQt |  |  |  |
| Tkinter |  |  |  |

| | | | |
|---|---|---|---|
| wxPython |  |  |  |

*Source code for shown applications are available at [http://gpolo.ath.cx:81/download/gravity_game](http://gpolo.ath.cx:81/download/gravity_game). This game is a simplified version of "The Gravity Game", you can find the original source at book "Physics for Game Programmers", Chapter 3 (in Java).*

### *GUI Customization*

The word "customization" can take different meanings according to the context, so, for this section, it's meaning will be defined as: "capacity to modify default behavior of the widgets contained within the toolkit, be these modifications beneficial or even harmful to users."

Said that, be sure customizations are needed before doing them. Remember it is important to keep your application familiar to the users, so there must be a really strong point to opt for customized look and feel. But depending on the kind of program you are developing, it may make sense to choose for this. For example, music and video players usually uses custom interfaces, therefore it is possible you could want customized look and feel too.

I classify customizations based on what they do: Compromising customizations, Pleasant customizations and Runtime customizations.

- Compromising customizations
  - Reasoning for this naming: Removes application's familiarity.
  - Typical changes: Shaped widgets, changing widget colors, custom fonts.
- Pleasant customizations
  - Reasoning for this naming: Any toolkit should provide in order to be useful and the user probably will feel better with these changes.
  - Typical changes: Possibility to change the label position relative to an image in a button, changing toolbar position, general changes that could be done to widgets available so they fit better your project.
- Runtime customizations
  - Reasoning for this naming: Customizations are done while the program is running.
  - Typical changes: Widget placement, applications's appearance.

Note that Runtime customizations lists as typical change applications's appearance, this would fall in Compromising customizations. But it was user's choice to do it, and he probably feels better with these changes, making it a good thing.

It is not interesting to discuss Pleasant customizations here since all toolkits presented provide them. Some of them supports more than others, mainly because they contain more widgets thus allowing more possibilities of customizations.

Lastly note that Compromising customizations may be useful too, they can help low vision users (for example). So they are not always dangerous or bad.

### *PyGtk customization*

- Compromising
  - Themes are the way to go for large customizations under PyGtk. Users can choose a new look for your application just by setting a different theme (requiring no code changes).
  - PyGtk supports creation of shaped widgets.
- Runtime
  - It is possible to change themes easily while the application is running.
  - There is no support for dock widgets, so the user cannot freely relayout his applications. If you need this kind of customization, there are wrappers for GDL library which supports this functionality.

### *PyQt customization*

- Compromising
  - PyQt uses Styles instead of Themes to achieve large customizations.
  - This toolkit also supports creation of shaped widgets.
- Runtime
  - Like in PyGtk, it is pretty easy to change styles while the application is running.
  - PyQt contains a widget called QDockWidget allowing your users to freely move and reposition widgets.

### *Tkinter cstomization*

- Compromising
  - Tkinter allows widgets fonts and colors to be customized, but not as easily as using Themes or Styles.
  - Shaped widgets are not supported by this toolkit.
- Runtime
  - Not available at all.

### *wxPython Customization*

- Compromising
  - Like Tkinter, this toolkit supports changing widgets fonts and colors but not as easily as using Themes or Styles. wxPython may adapt its look and feel based on changes done on your platform.
  - Shaped widgets are supported.
- Runtime
  - wxPython provides a library called AUI (Advanced User Interface) allowing dockable floating frames (like PyQt dock widget) and much more.

### *Summary*

Designing good user interfaces requires a decent amount of effort, following some guidelines may help you and will make your users happier.

Customizations are not always welcomed by users, make sure you need it and your applications benefits from this before diving into this area. If you believe they are needed, both PyGtk and PyQt got more general solutions. If you are not planning changing the entire interface through themes, wxPython may be good enough to you as well. Lastly, if you do not need fancy customizations, Tkinter may be a choice.

## Web Integration

As time passes by, it gets harder to ignore how much the Web is part of our daily lives, and if it is not important for you yet, it is possible that in some time it will. It just keep growing (only Google indexes several billions of pages), all the time.

For this reason, it may be possible that you will want (or need) to integrate Web resources into your Desktop applications. Some programs already do it, be it a small part or even the largest and most important part of the GUI application.

If you are planning to use Web resources (HTML, CSS, Javascript, Flash, ...) in your Desktop applications, you need to be more careful when choosing among the toolkits just in case frustration, anger, and co. are not your friends. But, be warned: depending on the level of resources and portability desired, there may be no choices at all.

### *PyGtk*

PyGtk comes with Pango that uses a SGML-like markup language that allows you specify attributes with the text they are applied to by using a small set of markup tags. This is helpful for displaying applications's documentation or simple HTML pages, for example. With Pango it is possible to build a humble HTML renderer.

Next options are external libraries.

For rendering and/or printing simple HTML documents, you would need GtkHtml (python-gtkhtml bindings[6]). If you need CSS 1,2 and/or DOM 1,2 then GtkHtml2 would be more appropriate, but unfortunately I couldn't find any documentation for GtkHtml2. Even if you do not need documentation at all to learn how to use it, you should worry about its lifetime. It looks like both GtkHtml and GtkHtml2 are not being developed anymore (but it still works), so I wouldn't pick them. Platform supported (GtkHtml, GtkHtml2): Linux.

And there is GtkMozEmbed that allows you to embed a Mozilla browser window into your Gtk application. Its API is severely limited, it doesn't provide printing or viewing the page source (as examples of the limitation). But if you do not need this kind of service, I would opt for using it. The reason for this choice is the support for all Web resources your standard web browser does, and it's very easy to use. Lastly, it seems to have been used only in Linux for now.

### *PyQt*

PyQt's components QLabel, QGraphicsTextItem, QTextEdit and QTextBrowser supports HTML (a subset of HTML 3.2 and 4) rendering. It isn't intended to support all features a browser does, but at least it is multi-platform.

Furthermore PyQt supports ActiveX through QAxContainer making it possible to embed Internet Explorer ActiveX control into your application. You get all features you might need in exchange of being limited to Windows and it is only available for commercial versions of PyQt.

### *Tkinter*

Just like you could do in PyGtk (use Pango to build a simple HTML renderer), you can do one in Tkinter using its Text widget. Next to this there is the possibility of using Python's built-in module webbrowser[7].

If using external libraries is not a problem, give TkHtml[8] and its Python wrapper[9] a try. I've

6 python-gtkhtml bindings site: http://www.fcoutant.freesurf.fr/python-gtkhtml.html
7 Python module webbrowser documentation: http://docs.python.org/lib/module-webbrowser.html
8 Tkhtml site: http://www.hwaci.com/sw/tkhtml/
9 Tkhtml Python wrapper: http://tix.sourceforge.net/Tixapps/src/Python/TkHtml.py

ran a sample that is included but didn't enjoy it all, it doesn't allow you to select text, among other things. Lastly there is TkHtml3[10] that gives cooler features, like CSS support, but there is no Python wrapper yet.

### *wxPython*

This toolkit provides some components that supports basic HTML. This includes HtmlListBox and HtmlWindow that serves as a basic and limited HTML render (like GtkHtml and QTextEdit). The good thing is that they are both multi-platform.

It also provides ActiveX support, and a component called IEHtmlWin that is an ActiveX IE window embedded in a wxWindow.

### *Summary*

Integrating web in your desktop application using any of these toolkits and expecting them to work everywhere with whatever you desire is just a hope nowadays. If you just need plain and simple HTML rendering, wxPython and PyQt will serve you well. If you are targeting Linux, PyGtk with GtkMozEmbed is your best shot. And if you are focusing on Windows users, wxPython will provide all features you might need through ActiveX (at no charge).

### *Future*

It seems WebKit[11] will save you (some time in the future). There is support for gtk, qt, wx and a project called wxWebKit[12] that already provides bindings for wxPython. And hopefully it will run in all platforms you expect to.

## Licenses

Licenses affect your work more than you can imagine, if you don't do already. You could want to hide the source, or keep any derivative works free, or maybe just put it in the public domain, or other uses, but to do so, you need to use the correct license. Choosing an appropriate license is not always easy, it may even require a lawyer, but this is out of scope of this article.

If you are worried about spending money, do not be. All the toolkits give you opportunity to not need to invest any money on licenses, but depending on your kind of application distribution and chosen toolkit, there may be some price to pay for.

### *PyGTK License*

PyGTK uses LGPL (Lesser General Public License), this license is a middle term between GPL and permissive licenses such as BSD and MIT licenses. The main difference between the GPL and the LGPL is that the latter can be linked to a non-(L)GPLed program, which may be free software or proprietary software. This means that you may choose to not distribute the source along with your program. If you are in doubt to use LGPL, check out http://www.fsf.org/licensing/licenses/why-not-lgpl.html.

### *PyQt Licenses*

PyQT follows Trolltech's license model, that means you will use different licenses based on how you use PyQT.

If your program is GPL compatible, then you do not need to buy a commercial PyQT license. Being compatible with GPL means, among other things, that:

10 Tkhtml3 site: http://tkhtml.tcl.tk/index.html
11 WebKit site: http://webkit.org/
12 wxWebKit site: http://wxwebkit.wxcommunity.com/pmwiki/index.php?n=Main.HomePage

- You will be providing the source code for your application
- All modified and extended versions of the program will continue being free
- Users are allowed to re-use, modify and re-distribute the code

Note that does not mean you can't sell copies of the program for money. There is a GPL FAQ that you should read in case of doubts: http://www.fsf.org/licensing/licenses/gpl-faq.html. Previous to PyQt v4, this free version was only available for Linux and Mac OSX.

In case your use of PyQt is not compatible with GPL then you will need a commercial PyQt license. Buying a license for PyQt does not include Qt licensing, so you must also purchase copies of the commercial edition of Qt from Trolltech. More informations on buying PyQt License can be found at http://www.riverbankcomputing.co.uk/pyqt/buy.php and for Qt at http://trolltech.com/products/qt/licenses/licensing.

#### *Tkinter License*

Tkinter is bundled with Python and also uses the PSFL (Python Software Foundation License). This is a permissive free software license which is compatible with the GNU General Public License (GPL). Its primary use is for distribution of the Python project software. Unlike the GPL the Python license is not a copyleft license, and allows modifications to the source code, as well as the construction of derivative works, without making the code open-source.

#### *wxPython License*

wxPython uses the same license as wxWidgets, the wxWindows License. This license is essentially the L-GPL (Library General Public License), with an exception stating that derived works in binary form may be distributed on the user's own terms. This is a solution that satisfies those who wish to produce GPL'ed software using wxWidgets/wxPython, and also those producing proprietary software.

#### *Summary*

If you don't even want to think about licenses in your way, Tkinter or wxPython would be the first option. Next to this is PyGTK, its license basically only forbids distribution of static linked libraries without neither source code nor linkable object files, so, if you are not considering this kind of distribution, it's all good. Finally there is PyQT. If you will be following GPL, you won't need to buy any licenses. But if that is not the case, you will need to invest some money in PyQT and QT before you start developing your application.

## GUI Designer Tools

As your applications grow, you will notice a lot of lines of code used just to create window components and by that time you will probably want to avoid unnecessary clutter in your code. That is where these tools should be used. GUI Designer tools also allows you to focus on the core development of your applications for the reason that GUI development will be a breeze.
Tools available for each toolkit will be rapidly described below. But you should familiarize yourself with them, or at least with the ones used by your chosen toolkit.

#### *PyGtk tools*

- Glade[13] (First release: April 18, 1998 ; Latest release: December 18, 2007)
- Gazpacho[14] (First release: June 30, 2004; Latest release: July 29, 2007)

Glade and Gazpacho are very similar from user perspective. They both requires you to understand how GTK containers work in order to construct interfaces. Using Glade there is a possibility of using a widget called "Fixed" that lets you drag and drop components into

13 Glade site: http://glade.gnome.org/
14 Gazpacho site: http://gazpacho.sicem.biz/

Toplevel widgets without caring about containers. Using this "technique" leads to a very problematic application, setting specific size and position for all widgets is a good recipe for headache. It is very likely your interface will not appear as correct in someone's else computer as it appeared on yours. For these reasons, you should learn how to use containers, it is not hard and your programs will achieve better results.
These tools generates .glade files that are then used by your application. Gazpacho also supports saving in gtkbuilder and gazpacho formats. GtkBuilder is said to take over glade format, making it the new format to use (it has been added in Gtk+ 2.12 and is available for PyGtk 2.12).
There are some reasons to pick Gazpacho over Glade, like the presence of a set of Kiwi[15] widgets and the possibility to use GtkUIManager[16].
And the reasons to pick Glade over Gazpacho are its development time and amount of users.

### *PyQt tool*

- Qt Designer[17] (First release: September 07, 2000 ; Latest release: October 03, 2007)

Qt Designer 4 is a very easy GUI designer to use, and very powerful as well. It is a tool that makes most people happy, you can just start dragging and dropping widgets and leave it like that or ask it to layout the widgets for you, or, you may choose to create Vertical/Horizontal/Grid Layout and drop widgets there.
Rarely you will need to change the tab order, but if you want to do it for the right reason, it provides a very attractive visual editor for that. Other nice visual features are edition of Signal/Slots and Buddies. The former provides a visual representation of the signal and slot connections in the form that can help other developers understand how the final component will behave. The later, associates a widget to its buddy. This association allows you to connect a QLabel to a QLineEdit (its buddy) and then focus the buddy widget by pressing QLabel's shortcut key combination.
This tool generates an .ui file, that you can choose to use through PyQT4.uic module or use the pyuic tool that will convert the .ui file to a .py file. It may also generate a .qrc file if you are using external resources, like images. To use the resources file, you need to use pyrcc tool so it converts to a python module with the external resources embedded into it.

### *Tkinter tool*

- GUI Builder[18] (Released into open source in November, 2006)

GUI Builder is a next-generation SpecTcl alternative, as its page says, but I didn't see much difference except that GUI Builder menu creation is more straightforward to use than SpecTcl's one.
It is pretty simple and somewhat intuitive, and enforces the use of Grid geometry manager which is good because it makes you write applications with a consistent layout, that is, you will not be hand positioning the widgets.
This tools generates two files, a _ui.py file and other .py file based on the name you gave when you saved the your .ui file.

### *wxPython tools*

- wxGlade[19] (First release: July 31, 2002 ; Latest release: February 02, 2008)
- XRCed[20] (First release: ~ August 31, 2001 ; Latest release: March 10, 2007)

Before moving on let me say that there are more GUI builders available for wxPython, like: Boa Constructor[21] (IDE and GUI Builder), VisualWx[22] (Windows only), wxDesigner[23]

---

15 Kiwi site: http://www.async.com.br/projects/kiwi/
16 GtkUIManager reference: http://www.pygtk.org/pygtk2reference/class-gtkuimanager.html
17 Qt Designer site: http://trolltech.com/products/qt/features/designer
18 GUI Builder site: http://spectcl.sourceforge.net/
19 wxGlade site: http://wxglade.sourceforge.net/
20 XRCed site: http://xrced.sourceforge.net/
21 Boa Constructor site: http://boa-constructor.sourceforge.net/
22 VisualWx site: http://visualwx.altervista.org/
23 wxDesigner site: http://www.roebling.de/

(Commercial). But I have chosen to talk only about wxGlade and XRCed because they are under active development, both works cross-platform, and they are open source projects. Now we may continue.
wxPython uses Sizers, and for the same reasons you should understand containers if you are using PyGtk, you should invest some time understanding sizers. Sizers are a bit harder and you will probably need to read some tutorial in order to use them perfectly. As suggestion, read this page: http://wiki.wxpython.org/UsingSizers.
wxGlade looks like Glade-2, it is not as pretty as Glade-3 (but works), except that it is able to generate XML-based resource system (or just XRC) files. Note that it is also able to generate direct Python code, but the advantage of opting for XRC format is that they can be used by XRCed as well.
XRCed is actually a resource editor that supports creating and editing files in XRC format, but I am including it in this section anyway. It will be your preferred tool when you get accustomed with wxPython, it is the fastest tool to develop an UI if you know enough about wx. XRCed doesn't involve drag & drop of widgets into a beautiful window, on the other hand you will drag & drop components into a XML tree, making it a bad choice if you are not comfortable on wxPython.
If you prefer to separate the UI layout from the code, be sure to use a XRC file. Working with this file is like working with glade files, but this is wxPython so it has several differences in use.
Both wxGlade and XRCed misses a lot of widgets included in wxPython, a consequence of this toolkit having so many widgets.

***Summary***

If you know your tool, and if the name of that tool is wxPython then XRCed is waiting for you. If you are considering learning some GUI toolkit and it is important that it has a powerful and featureful GUI Designer application, QT Designer will be your favorite tool. Next to this there is Gazpacho, followed by Glade and wxGlade. Gazpacho takes the lead for supporting action based menus and toolbars (Qt Designer supports this too). Lastly, if you just want something simple and good looking is not important at all, GUI Builder is there for you.

## Built-in widgets

Unlike other sections this one does not contains subsections dedicated to each toolkit, all the discussion will take place here.
All toolkits discussed provides a set of basic widgets that can construct any GUI application. But as you start doing more advanced user interfaces the number of used widgets will tend to increase and it is very likely that you will see yourself creating new widgets, or at least using some external libraries, every time you have a different application to build, if you are using Tkinter. That is because this toolkit has a very limited set of widgets and the chances of it growing are rare (Tk 8.5 has new widgets, meaning they will eventually be available in Tkinter) because it is part of Python and that means it is stable, which is not bad, but it stays the same for a long time. This is also one of the reasons why it is unlikely that some other GUI toolkit will take Tkinter's place in Python, because they are changing too often and possibly causing some form of instability.
wxPython has the largest set of widgets between the four toolkits. This implies that the chances of needing to construct a new widget are lower than if you were using PyGTK, PyQT or Tkinter. Of course if you need a very specific widget, no toolkit will provide it, so, you can either check if someone has already done it, or do it yourself. wxPython can save a great amount of time depending on the application you are planning to do, one of the reasons being the addition of wx.AUI library (AUI stands for Advanced User Interface). This library instantly adds a huge amount of flexibility to your application, like: floating/docking frames, customizable look and feel, perspective saving and loading, optional transparent window effects while dragging and docking, and others.
PyGTK and PyQT provides a similar amount of widgets, but PyQT leads between the two because QTextEdit supports HTML rendering, it also supports OpenGL in PyQT applications through QtOpenGl (requiring PyOpenGL) and allows docking widgets, for example.

## Documentation Available

This last section points out some documents that you will be visiting while developing a GUI application. Note that there is much more "Documentation Available" than described here, this was just an attempt to collect most of the interesting documents existing relative to the toolkits discussed here.
Nevertheless, I hope these pointers will answer most of your question.

***PyGtk***

| Document | Audience |
|---|---|
| PyGtk 2.0 Tutorial[24] | Anyone starting in PyGtk |
| PyGTK FAQ[25] | If you use PyGtk, but don't know everything about it yet, this FAQ will solve most of your questions that are likely to arise |
| PyGTk Wiki[26] | Both beginners and more experienced users will find this useful. Its main purpose is to collect links for several other documentations. |

- Others
  - PyGTK 2.0 Reference Manual[27]
  - Articles and Tutorials about PyGTK[28]
  - Beginning Python, Chapter 13[29]

***PyQt***

| Document | Audience |
|---|---|
| Introduction to PyQT4[30] | Anyone starting in PyQt |
| Rapid GUI Programming with Python and QT[31] | PyQt 4 book, one of the best options available for learning how to program with this toolkit |
| The PyQT and PyKDE community Wiki[32] | Good link collections, anyone interested in PyQt will find this useful |
| Qt Quarterly[33] | For those already developing in Qt/PyQt interested in learning more about this toolkit. |

- Others
  - PyQT v4 - Python Bindings for Qt v4[34]
  - PyQt's Classes[35]

***Tkinter***

| Document | Audience |
|---|---|
| An Introduction to Tkinter[36] | Anyone starting in Tkinter |

24 PyGtk 2 Tutorial: http://www.pygtk.org/pygtk2tutorial/index.html
25 PyGtk FAQ: http://faq.pygtk.org/index.py?req=index
26 PyGtk Wiki: http://live.gnome.org/PyGTK
27 PyGtk 2 Reference Manual: http://pygtk.org/docs/pygtk/index.html
28 Articles and Tutorials about PyGtk: http://pygtk.org/articles.html
29 Beginning Python book: http://www.amazon.com/Beginning-Python-Programmer-Peter-Norton/dp/0764596543
30 Introduction to PyQt4 site: http://www.rkblog.rk.edu.pl/w/p/introduction-pyqt4/
31 Rapid GUI Programming with Python and QT book: http://www.qtrac.eu/pyqtbook.html
32 The PyQT and PyKDE community Wiki: http://www.diotavelli.net/PyQtWiki
33 Qt Quarterly site: http://doc.trolltech.com/qq/
34 PyQT v4 - Python Bindings for Qt v4 reference: http://www.riverbankcomputing.com/Docs/PyQt4/pyqt4ref.html
35 PyQt's Classes reference: http://www.riverbankcomputing.com/Docs/PyQt4/html/classes.html
36 PyQt's Classes: http://www.pythonware.com/library/tkinter/introduction/index.htm

| | |
|---|---|
| Python and Tkinter Programming[37] | A good read, in form of a book, for those wanting to learn Tkinter |
| Tkinter Wiki[38] | Contains links for some tutorials, reference documentation, and others. Better suited for intermediate users |

- Others
  - Tkinter reference: a GUI for Python[39]
  - Tkinter -- Python interface to Tcl/Tk[40]

***wxPython***

| *Document* | *Audience* |
|---|---|
| The wxPython tutorial[41] | Anyone starting in wxPython |
| wxPython in Action[42] | Very good book, just a bit dated but still serves as one of the best ways, if not the best, to learn wxPython |
| wxPyWiki[43] | I would say this is the best wiki among the others mentioned here. Beginners, intermediate and experienced users will find good and interesting informations here |

- Others
  - wxPython API[44]

***Summary***

There is a decent amount of documentation for all the toolkits. But it is important to ask yourself if you are willing to learn. It does not matter how many books, tutorials, articles, etc.. may exist for a toolkit (or anything else) if you just do not take time to learn what you want/need. The excuse of documentation not being good enough, in this case, is hardly a reason for preventing you from learning any of these toolkits.

37 Python and Tkinter Programming book: http://www.manning.com/grayson/
38 Tkinter Wiki: http://tkinter.unpythonic.net/wiki/
39 Tkinter Reference site: http://infohost.nmt.edu/tcc/help/pubs/tkinter/index.html
40 Tkinter Python module documentation: http://docs.python.org/lib/module-Tkinter.html
41 wxPython Tutorial site: http://www.zetcode.com/wxpython/
42 wxPython in Action book: http://www.manning.com/rappin/
43 wxPython Wiki: http://wiki.wxpython.org/
44 wxPython API site: http://www.wxpython.org/docs/api/