PyGTK

czyli jak przestać się bać i polubić GUI

Jarek Zgoda, Sensisoft
http://zgodowie.org/

GTK

- popularna biblioteka GUI
- 10 lat rozwoju
- oryginał w języku C, dowiązania dla wielu innych języków (w tym Pythona)
- wieloplatformowa
  - obecna we wszystkich (chyba) dystrybucjach linuksa
  - MS Windows 2000 i nowsze
  - OSX (podobno nie wymaga już XServera!)

PyGTK

- „cienka” nakładka na GTK
- ręczna robota (żadnych automatów typu SWIG)
- upythoniczniona gdzie się dało
  - np. widgety przyjmują `unicode` lub `str` w utf-8
    - ale zwracają zawsze `str` w utf-8
- dostępna na platformy, na których jest GTK
  - choć nie wszędzie jednakowo łatwo dostępna

Młotek i gwoździe

- klient usługi sieciowej
- wymagana wysoka responsywność aplikacji
- integracja z desktopem (GNOME, XFCE, ...)
- wtyczki do istniejących programów
  - w C: gEdit, Rhythmbox
  - w Pythonie: QuodLibet

Elementy układanki

- na samym dnie: GObject (`gobject`)
- baza dla widgetów: GDK (`gtk.gdk`)
- widgety: GTK (`gtk`)
- dostępność: ATK (`atk`)
- dynamiczne ładowanie okien: GLADE (`gtk.glade`)
- układ tekstu: Pango (`pango`)
- rysowanie na płótnach: Cairo (`cairo`)

Poza GTK: GNOME

- integracja z desktopem GNOME
- gnome-python, gnome-python-desktop, gnome-python-extras, PyORBit
- umożliwia pisanie programów i apletów zintegrowanych z komponentami GNOME

Poza GTK: XFCE

- biblioteka pyxfce (http://pyxfce.xfce.org/)
- desktop (`xfce4.gui`)
- panel (`xfce4.panel`)
- narzędzia (`xfce4.util`)

Widgety

- wszystkie dostępne w GTK
  - niezwykła elastyczność
  - za cenę sporego skomplikowania (np. `gtk.TreeView`)
  - Glade się przydaje, ale nie zawsze jest w stanie pomóc
- własne, dziedziczone ze standardowych
  - także złożone („multiwidgets”)
  - hej, można je dodawać do palety Glade3!

Glade

- „budowniczy” interfejsów użytkownika
- Glade2 pomoże utworzyć szkielet okna i zdefiniować sygnały
  - nie wszystkie widgety obsługiwane w pełni (m.in. `gtk.ComboBox`, `gtk.TreeView`, `gtk.TextView`)
- Glade3 podobno to (i wiele innych rzeczy) już potrafi
- ale „sprawia wrażenie” niestabilnego

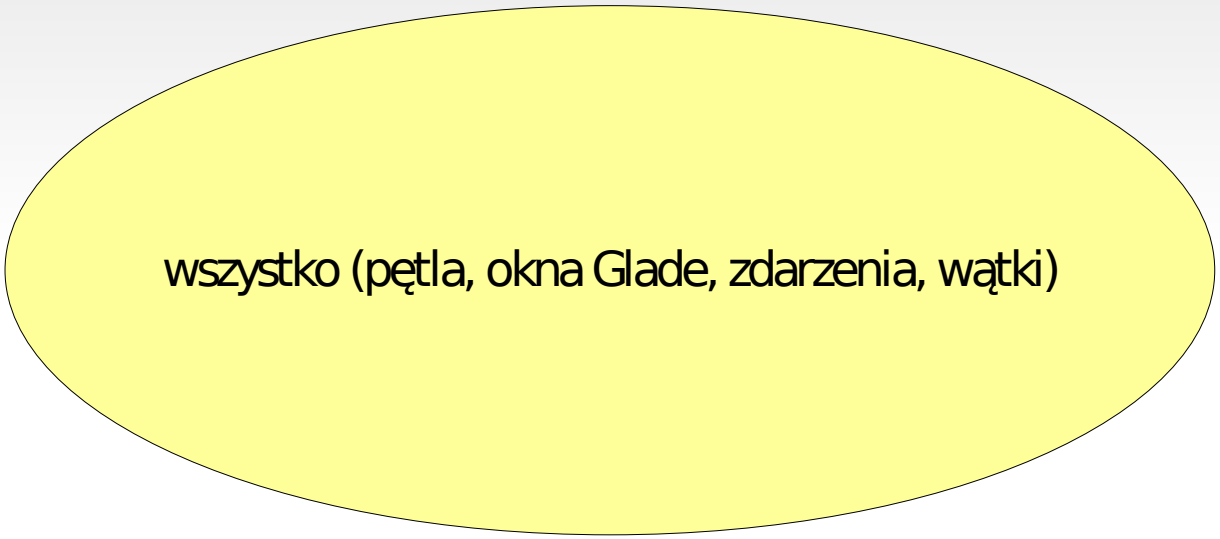
Typowe cechy aplikacji

- sterowana zdarzeniami GUI
- wielowątkowa lub asynchroniczna
  - to nieprawda, że GIL zjada psy i pije krew dzieci
  - wątki są prostsze od pętli asynchronicznych
    - z wyjątkiem przypadków, kiedy nie są:
      - dzięki ułatwieniom GnomeVFS
      - integracja z Twisted (`gtk2reactor`, yeah!)
- o standaryzowanym wyglądzie (HIG)

Jak używać PyGTK?

- nie ma „jedynego słusznego sposobu”
- najczęściej: główna pętla i obsługa zdarzeń w kodzie, definicja okien w Glade
- można chaotycznie (Gajim)
  - sprawdza się w małych i bardzo małych programach
- można strukturalnie (QuodLibet)
  - sposób najczęściej spotykany
- można w ramówce typu MVC (PIDA)

Sposób 1: (nie)kontrolowany chaos



Sposób 2: struktura

- klasy zarządzają aplikacją, oknami, wątkami
- klasa aplikacji: dane globalne aplikacji
  - twór sztuczny, dodany jedynie dla elegancji
- klasy okien: wyświetlanie elementów i zdarzenia
  - może być tak obiektowo, jak tylko się chce
- klasy wątków

Sposób 3: ramówka

- obszerny temat na któreś z przyszłych spotkań
- przeznaczona dla dużych i bardzo dużych aplikacji
- paradygmaty MVC (pygtkmvc), Visual Proxy (Kiwi)
- wymusza (lub sugeruje) strukturę aplikacji

Aplikacja: główna pętla

- inicjalizacja GTK
- włączenie ustawień lokalnych (i18n)
- dodatkowe zadania (m. in.):
  - zezwolenie na jedną instancję programu
  - obsługa sygnałów systemowych
  - utworzenie globalnego egzemplarza klasy aplikacji
  - sprzątanie

Aplikacja: klasa aplikacji

- dodana wyłącznie dla elegancji
  - a my jesteśmy eleganccy, więc ją lubimy
- zarządza globalną konfiguracją programu
- zarządza tworzeniem głównego okna aplikacji

Aplikacja: klasy okien

- jedna klasa na każde okno/dialog
- można wywieść z `gtk.Window` i `gtk.Dialog`, ale…
- …kompozycja jest wygodniejsza
  - klasa przejmuje wtedy rolę kontrolera
  - wiązanie metod do sygnałów z Glade: `signal_autoconnect(self)` – fajne!

Aplikacja: klasa głównego okna

- wydzielona z powodu funkcji i czasu życia (poza tym to zwykła klasa okna)
- zdarzenia menu, pasków narzędzi
  - sposób tradycyjny: Glade
  - sposób polecany ostatnio: `gtk.UIManager` + Glade
    - scentralizowane zarządzanie zdarzeniami

Aplikacja: wątki (1)

- strach przed wątkami jest gorszy niż wątki
- strach przed GIL jest gorszy niż GIL i wątki razem
- GIL == globalna blokada interpretera
  - tylko CPython
  - chroni wewnętrzny stan interpretera
  - zwalniany okresowo i w szczególnych przypadkach
    - rozszerzenia w C (PyGTK właśnie!)
    - oczekiwanie na IO

Aplikacja: wątki (2)

- niestety, nie jest różowo (zwyczajowe ograniczenia)
  - bezpieczne zmiany w UI tylko z głównego wątku aplikacji
  - brak komunikacji między wątkami (tylko przez kolejkę)
- jest szaroróżowo (sporo ułatwień)
  - `gobject.idle_add()`
  - `gobject.io_add()`
  - `gobject.timeout_add()`
  - `gtk.main_iteration()`

Dokumentacja i pomoc

- w miarę pełna, dużo artykułów wprowadzających
- http://pygtk.org/
- http://www.pygtk.org/pygtk2reference/
- http://www.moeraki.com/pygtktutorial/pyg
- http://faq.pygtk.org/
- http://www.daa.com.au/mailman/listinfo/py
  - http://www.mail-archive.com/pygtk@daa.com

I to by było na tyle

- zapraszam na pl.comp.lang.python
- a za 2 tygodnie kolejne spotkanie z Pythonem!