Kurs rozszerzony języka Python

Środowisko GTK+

Marcin Młotkowski

11 grudnia 2019

Plan wykładu

1. GUI w Pythonie: GTK+
   - Wprowadzenie do GTK+
   - PyGTK
2. Przykład rysowania
   - Okno główne aplikacji
   - Pakowanie kontrolek
   - Kontrolki
   - Podstawy pracy z Glade
   - Gazpacho
3. Przeglądanie list danych

Plan wykładu

1. GUI w Pythonie: GTK+
   - Wprowadzenie do GTK+
   - PyGTK
2. Przykład rysowania
   - Okno główne aplikacji
   - Pakowanie kontrolek
   - Kontrolki
   - Podstawy pracy z Glade
   - Gazpacho
3. Przeglądanie list danych

Biblioteki okienkowe w Pythonie

- curses: interfejs tekstowy
- Tkinter (Tk interface): biblioteka okienkowa Tk + Tix (Tk extension)
- Pygtk, pygnome: API do środowiska Gtk/Gnome
- PyQT: API do QT
- wxWindows
- OpenGL
- PyWin32

GTK+/GNU

GTK+

The **G**IMP **T**ool**k**it

GNOME

GNU Network Object Model Environment

Moduły towarzyszące

Moduły współistniejące z GTK+

- GObject
- ATK
- Pango
- Cairo
- Glade

Środowisko GTK+

Elementy składowe

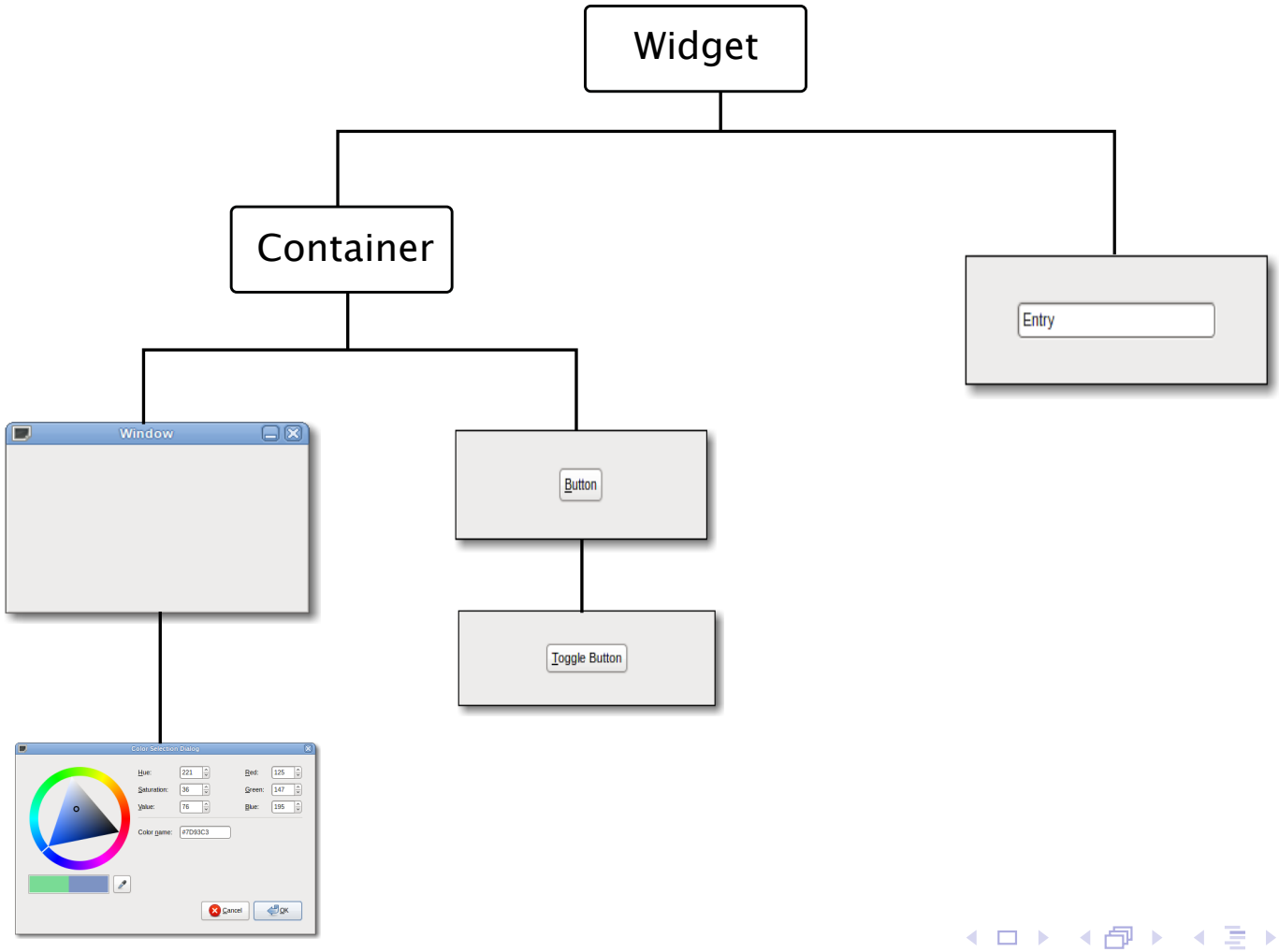
- okna;
- kontrolki;
- zdarzenia

Okna i kontrolki

Rola kontrolki (bardzo nieformalnie):

- kontrolka ma być (najlepiej ładna);
- kontrolka czasem ma reagować, np. na kliknięcie myszą;
- kontrolka czasem może zawierać inne kontrolki.

Hierarchia kontrolek (uproszczone)



GTK+ i Python

Biblioteka `PyGTK`

PyGTK: podstawowe elementy

Biblioteka się nazywa `gtk`.

PyGTK: podstawowe elementy

Biblioteka się nazywa `gtk`.

W bibliotece są odpowiednie klasy `Window`, `Entry`, `Button` etc.

Plan wykładu

1. GUI w Pythonie: GTK+
   - Wprowadzenie do GTK+
   - PyGTK
2. Przykład rysowania
   - Okno główne aplikacji
   - Pakowanie kontrolek
   - Kontrolki
   - Podstawy pracy z Glade
   - Gazpacho
3. Przeglądanie list danych

Przykładowa aplikacja

Specyfikacja

Aplikacja powinna:

- rysować zadane figury;
- podpisywać rysunki;
- kończyć pracę po kliknięciu klawisza na przycisk koniec.

Nagłówki

```
import gi
gi.require_version('Gtk', '3.0')
from gi.repository import Gtk
```

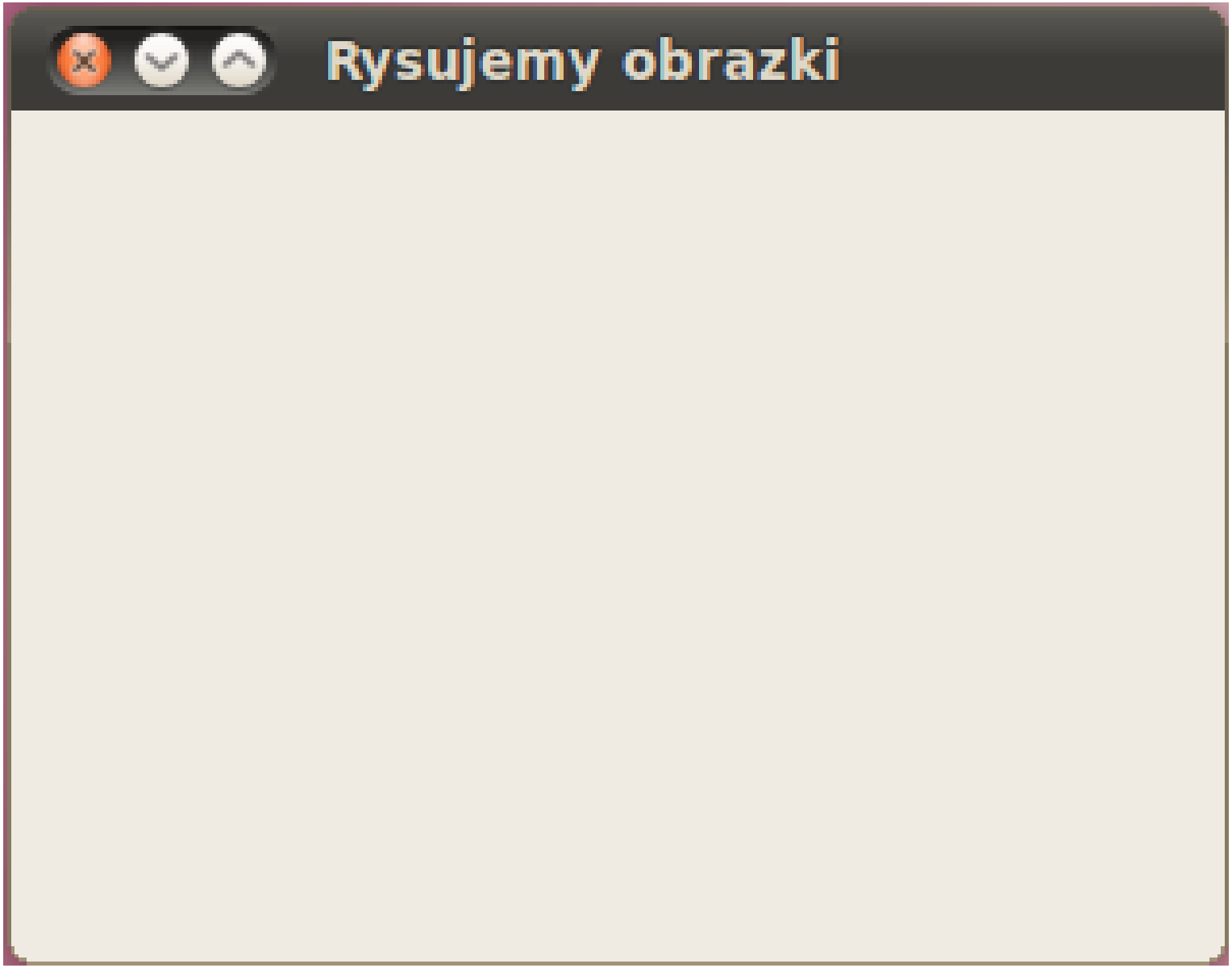
Budowanie okna

```
class Rysownik(Gtk.Window):
    def __init__(self):
        super(Rysownik, self).__init__()
        self.set_title("Rysujemy obrazki")
        self.set_position(Gtk.WIN_POS_CENTER)
        self.connect("destroy", Gtk.main_quit)
        self.kontrolki()
        self.show_all()
```

Uruchomienie aplikacji

```
r = Rysownik()
Gtk.main()
```

Efekt



Dokładanie kontrolek

- `Window` jest kontrolką "widzialną";
- `Window` jest też kontenerem, można wstawić element;
- do `Window` można wstawić tylko jeden element.

Pudełka

Do układania elementów służą pudełka pionowe i poziome.

Pudełka

Do układania elementów służą pudełka pionowe i poziome.

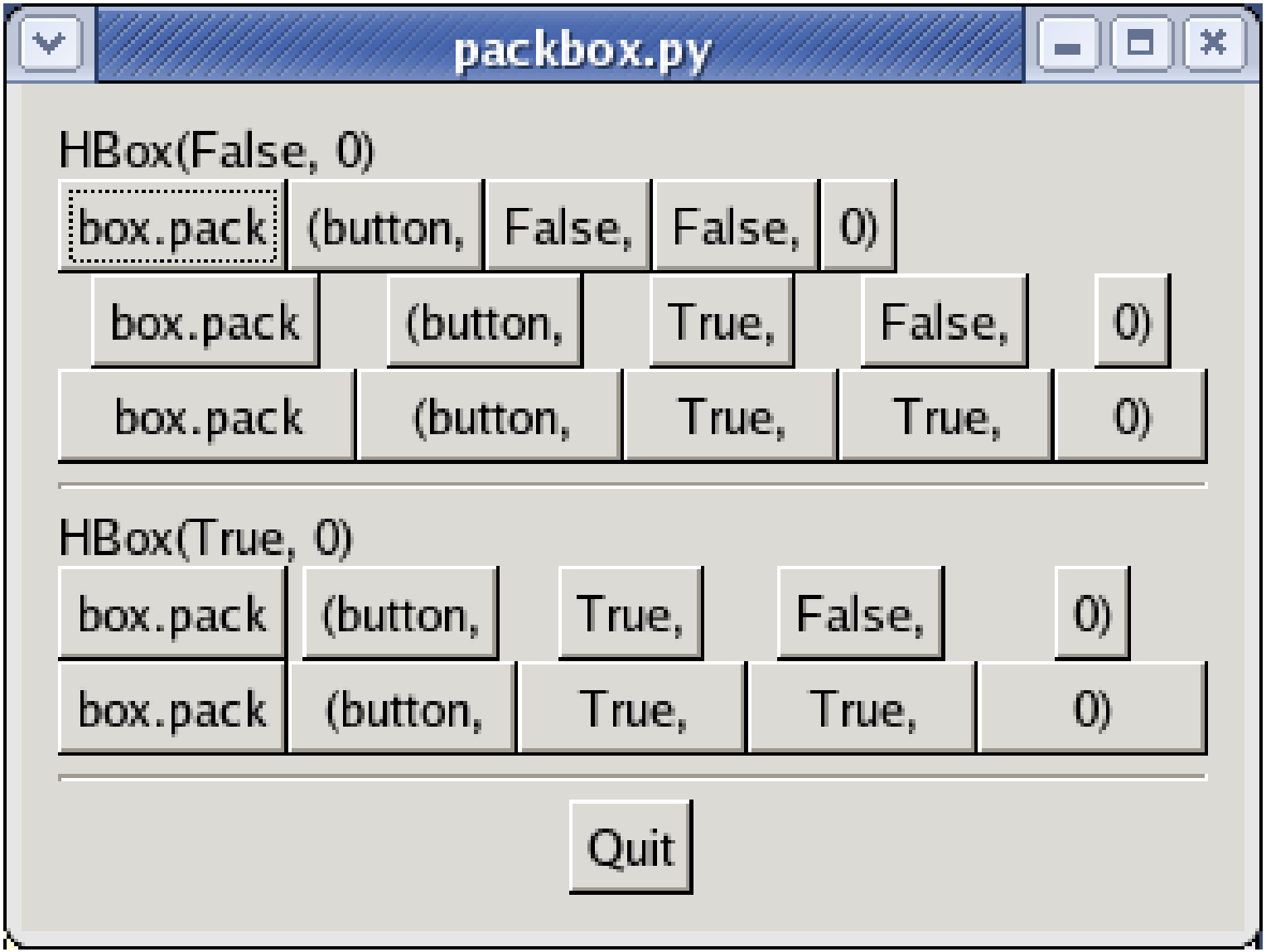
Gtk.Box: Gtk.VBox i Gtk.HBox

.pack_start(*Widget*, *expand*, *fill*, *padding*)
.pack_end(*Widget*, *expand*, *fill*, *padding*)

gdzie

- expand(bool): kontrolki włożone do pudełka mają wypełniają całe pudełko, wypełnieniem jest pusta przestrzeń;
- fill (bool): kontrolki wypełniają całą przestrzeń, ale przy okazji powiększane są kontrolki;
- padding (int): dodatkowy odstęp od sąsiada

Ilustracja pakowania



Parę uwag dodatkowych

Pudełka można pakować w pudełka (pionowe w poziome, poziome w pionowe etc)

Parę uwag dodatkowych

Pudełka można pakować w pudełka (pionowe w poziome, poziome w pionowe etc)

Alternatywa: Gtk.Grid

"Kratka" komórek, do których wkłada się kontrolki.

Wprowadzanie tekstu

Gtk.Entry

- .get_text(): pobranie tekstu z bufora kontrolki;
- .set_text("text"): wyczyszczenie bufora i wstawienie tekstu;
- .insert_text("insert", pos): wstawienie tekstu od *pos*.

Przyciski

Gtk.Button('tekst')

```
endb = Gtk.Button("Koniec")
endb.connect("clicked", lambda x: self.destroy())
```

Sygnały (ang. *signals*)

Sygnał w GTK+

Informacja, że zaszło jakieś zdarzenie, np. kliknięcie przycisku, likwidacja jakiejś kontrolki.

Sygnały (ang. *signals*)

Sygnał w GTK+

Informacja, że zaszło jakieś zdarzenie, np. kliknięcie przycisku, likwidacja jakiejś kontrolki.

- Sygnały są związane z kontrolkami.
- Sygnały mają swoje nazwy.

Funkcje zwrotne (ang. *callbacks*)

Funkcje zwrotne to są funkcje wywoływane jako reakcja na sygnały.

Funkcje zwrotne (ang. *callbacks*)

Funkcje zwrotne to są funkcje wywoływane jako reakcja na sygnały.

Postać funkcji zwrotnej

def funkcja_zwrotna(*kontrolka*, *dane*)

Funkcje zwrotne (ang. *callbacks*)

Funkcje zwrotne to są funkcje wywoływane jako reakcja na sygnały.

Postać funkcji zwrotnej

def funkcja_zwrotna(*kontrolka*, *dane*)

Łączenie kontrolek, sygnałów i funkcji zwrotnych

kontrolka.connect("nazwa sygnału", *funkcja zwrotna*, *dane*)

Są jeszcze zdarzenia: **Events**.

Są jeszcze zdarzenia: **Events**.

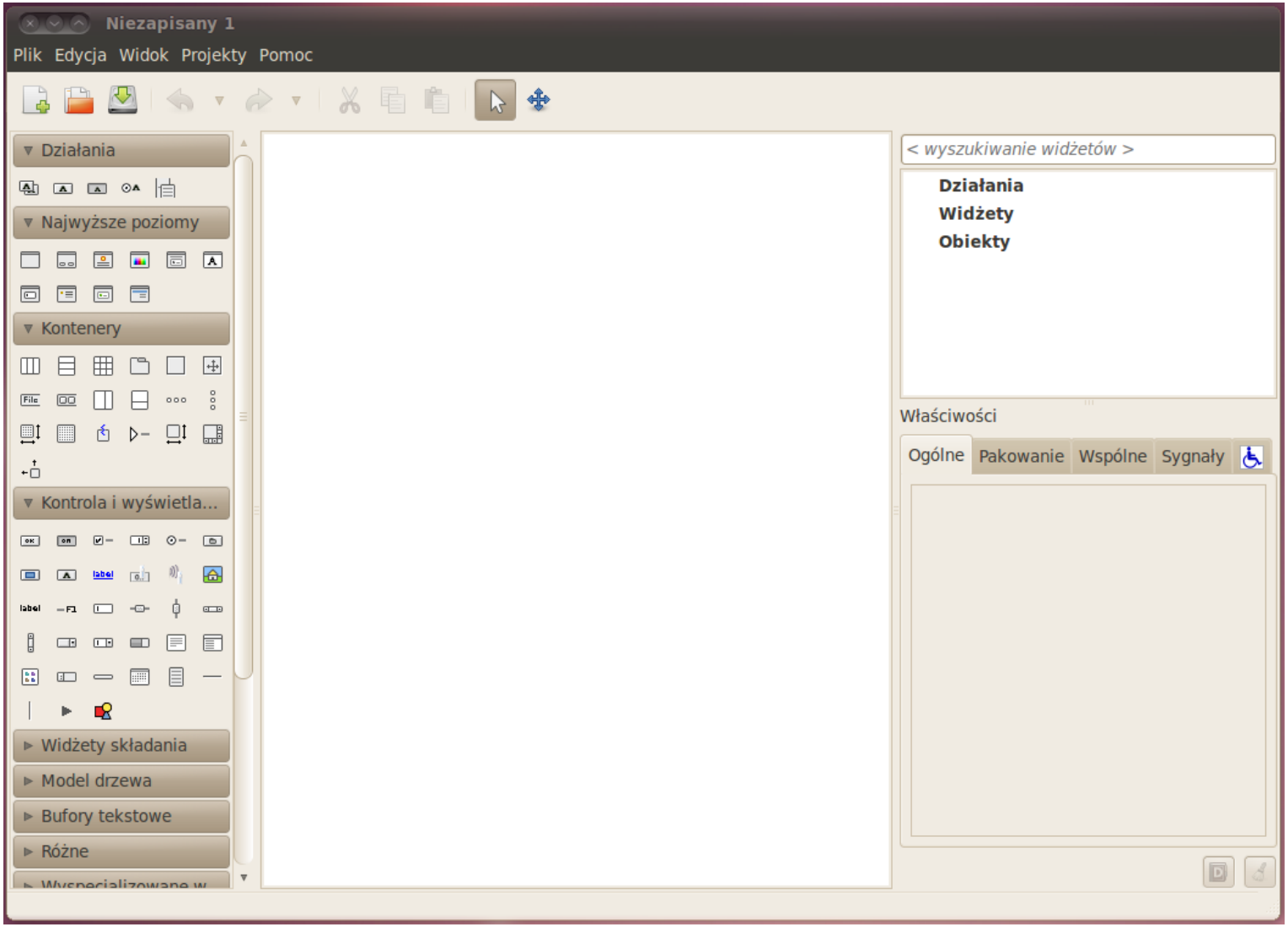
Ale nie będę o nich mówił.

Co to jest

Glade to graficzne narzędzie do projektowania interfejsów dla środowiska GTK+/GNOME.

Schemat działania (Glade-3)

- Glade produkuje plik XML, w którym jest opisany interfejs;
- Aplikacja "wczytuje" ten plik i buduje interfejs;
- Glade-3 jest niezależny od języka.


Niezapisany 1
Plik Edycja Widok Projekty Pomoc
Działania
Najwyższe poziomy
Kontenery
Kontrola i wyświetla...
Widżety składania
Model drzewa
Bufory tekstowe
Różne
< wyszukiwanie widżetów >
Działania
Widżety
Obiekty
Właściwości
Ogólne
Pakowanie
Wspólne
Sygnały

Użycie projektu

GtkBuilder

Biblioteka budująca z XML interfejs graficzny.

Użycie projektu

GtkBuilder

Biblioteka budująca z XML interfejs graficzny.

libglade

Poprzednia biblioteka, używa innego, niekompatybilnego XML'a.

Użycie projektu

GtkBuilder

Biblioteka budująca z XML interfejs graficzny.

libglade

Poprzednia biblioteka, używa innego, niekompatybilnego XML'a.

Konwersja plików

`gtk_builder_convert`

Budowanie okna

Ważne

Trzeba pamiętać, że kontrolki mają swoje nazwy.

Budowanie okna

Ważne

Trzeba pamiętać, że kontrolki mają swoje nazwy.

```
builder = Gtk.Builder()
builder.add_from_file("wyklad.glade")
window = builder.get_object("okno")
window.show()
Gtk.main()
```

Podłączanie sygnałów

Ważne

Podczas budowania interfejsu trzeba wskazać, jakim sygnałom odpowiadają jakie procedury obsługi (handlery).

Podłączanie sygnałów

Ważne

Podczas budowania interfejsu trzeba wskazać, jakim sygnałom odpowiadają jakie procedury obsługi (handlery).

```
builder = Gtk.Builder()
builder.add_from_file("wyklad.glade")
builder.connect_signals({ "on_window_destroy" : Gtk.main_quit })
window = builder.get_object("okno")
window.show()
Gtk.main()
```

Podłączanie menu

Łatwe

Tak samo jak w przypadku innych sygnałów.

Podłączanie menu

Łatwe

Tak samo jak w przypadku innych sygnałów.

Dokładniej:

- trzeba w Glade wskazać procedurę obsługi (wpisać jej nazwę) dla sygnału 'activated';
- powiązać nazwę z prawdziwą procedurą:

```
builder.connect_signals({ "on_window_destroy" : Gtk.main_quit,
                          "on_menu_koniec" : lambda widget : akcja(widget) })
```

Bardziej obiektowo

```
class Rysownik:
    def __init__(self):
        builder = Gtk.Builder()
        builder.add_from_file("wyklad.glade")
        self.window = builder.get_object("okno")
        builder.connect_signals(self)

    def on_window_destroy(self, widget, data=None): pass

    def koniec(self, widget): pass

rysunek = Rysownik()
rysunek.window.show()
Gtk.main()
```

Gazpacho

Inne narzędzie (napisane w PyGTK) do projektowania interfejsów graficznych, produkuje pliki zgodne z GtkBuilder.

Gazpacho

Inne narzędzie (napisane w PyGTK) do projektowania interfejsów graficznych, produkuje pliki zgodne z GtkBuilder.

Kiwi

Z projektem Gazpacho jest związana biblioteka Kiwi (napisana w Pythonie), która w założeniu ma być "lepszym GTK+".

Plan wykładu

1. GUI w Pythonie: GTK+
   - Wprowadzenie do GTK+
   - PyGTK
2. Przykład rysowania
   - Okno główne aplikacji
   - Pakowanie kontrolek
   - Kontrolki
   - Podstawy pracy z Glade
   - Gazpacho
3. Przeglądanie list danych

Kontrolka

Do przeglądania list i drzew służy Gtk.TreeView. Kontrolka wymaga

- kolumn `TreeViewColumn`;
- danych do przeglądania (modelu): `ListStore` (dla zwykłego przeglądania) i `TreeStore` (dla drzewek).

Przykład