Лекция 10
Графические интерфейсы

20 апреля 2017 г.

Обзор графических библиотек

Важные особенности:

- Кросс-платформенность
- Документированность (с учетом StackOverflow)
- Open source

Обзор графических библиотек

Важные особенности:

- Кросс-платформенность
- Документированность (с учетом StackOverflow)
- Open source

Подробные обзоры: [ссылка] и [еще ссылка] и еще [ссылка для Web-приложений].

Некоторые библиотеки:

- **Tkinter**
- PyQt/PySide
- wxPython
- PyGTK
- pygame
- **Kivy**
- (Web) Django

Графические интерфейсы: основы

Минимальные необходимые возможности

- Вывод на экран
  (вывести пиксель цвета `C` в точке `(x, y)`)
- Опрашивание устройств ввода
  (позиция курсора, состояние клавиш)

Структура программы

- Инициализация и загрузка всех компонент
- Основной цикл исполнения
  - Обновление ввода
  - Обновление внутреннего состояния программы
  - Вывод изменений интерфейса на экран
- Завершение работы

Графические интерфейсы: основы

Структура программы

- Инициализация и загрузка всех компонент
- Основной цикл исполнения
  - Обновление ввода
  - Обновление внутреннего состояния программы
  - Вывод изменений интерфейса на экран
- Завершение работы

Основные сложности

- Реализация простейших элементов GUI
- Встраивание ресурсоемких процессов в цикл
- Скорость работы

Модель на основе событий

Графическая библиотека

- Реализация основного цикла (возможно, в несколько потоков)

Модель на основе событий

Графическая библиотека

- Реализация основного цикла (возможно, в несколько потоков)
- Свой код уже в цикл писать нельзя

Модель на основе событий

Графическая библиотека

- Реализация основного цикла
  (возможно, в несколько потоков)
- Свой код уже в цикл писать нельзя
- В процессе работы программы происходят *события*,
  можно делать для них *обработчики*

Примеры событий

- Нажатие клавиши
- Клик мыши
- Запуск или окончание программы
- Событие созданное другой частью программы

Общая структура программ с GUI

- Описание элементов интерфейса
  - Описание статических элементов (*виджетов*)
  - Генерация динамических виджетов
  - Код для нестандартных элементов
- Основная логика программы
  - Логика вне интерфейса (бизнес-логика)
  - Обработчики взаимодействия с интерфейсом
  - Обработчики для фоновых процессов

Популярный шаблон проектирования:
Model-View-Controller (MVC)

Особенности GUI в Web-приложениях

- Нет прямой работы с устройствами ввода-вывода.
  Общение с пользователем через браузер — через HTML и браузерные скрипты (например, JavaScript).
- Обязательный параллелизм.
  Одновременная работа с несколькими пользователями.
- Работа с сетью.
  Протоколы, ошибки соединения, а также авторизация и т.п.

Библиотеки для Web-приложений пытаются свести эти проблемы к минимуму.

Библиотека Tkinter

Основные особенности

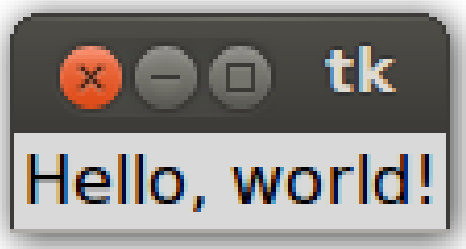
- Tkinter = tk interface
  (библиотека `tcl/tk`)
- Старая библиотека
  (многое осталось для совместимости)
- Есть в стандартной библиотеке Python
  (но все равно нужно ставить `tk`)
- Легко писать простые программы
- Большие программы часто очень запутанные

Hello, world!

```
from Tkinter import *

def main():
    root = Tk()
    label = Label(root, text='Hello, world!')
    label.pack()
    root.mainloop()

main()
```



Класс Frame

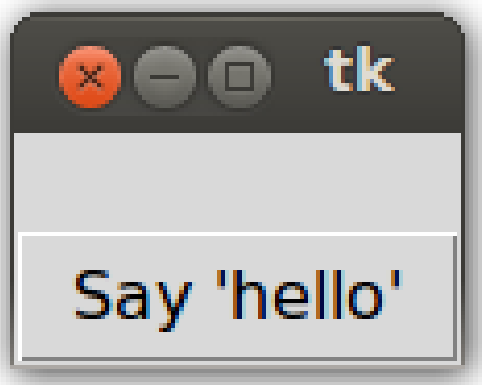
```
class Main(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.pack()
        self.create_widgets()
```

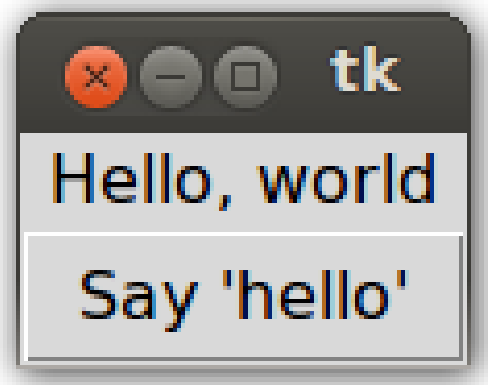
Класс Frame

```
class Main(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.pack()
        self.create_widgets()
    def create_widgets(self):
        self.label = Label(self)
        self.label.pack()
        self.button = Button(self,
            text="Say 'hello'",
            command=self.say_hello)
        self.button.pack()
    def say_hello(self):
        self.label.configure(
            text='Hello, world')
```

Класс Frame

```
root = Tk()
frame = Main(root)
frame.mainloop()
```





Упаковка - pack

```
def create_widgets(self):
    button1 = Button(self, text='1')
    button1.pack(side='left', fill='y',
                 expand=True)
    button2 = Button(self, text='2')
    button2.pack(side='top')
    button3 = Button(self, text='3')
    button3.pack(side='left')
    button4 = Button(self, text='4')
    button4.pack(side='right')
```

Упаковка - grid

```
def create_widgets(self):
    button1 = Button(self, text='1')
    button1.grid(row=0, column=1)
    button2 = Button(self, text='2')
    button2.grid(row=1, column=0,
                 columnspan=2, sticky='nsew')
    button3 = Button(self, text='3')
    button3.grid(row=1, column=2, rowspan=2)
    button4 = Button(self, text='4')
    button4.grid(row=2, column=0)
```

Еще полезные функции

- Упаковщик `place()` для размещения элементов в любой точке
- `bind()` для создания обработчиков произвольных событий
- Стили, темы: `ttk`, `ttk.Style`.
- Другие виджеты: `Text`, `ListBox`, `Checkbutton`, `Radiobutton`, `Frame`, `Scrollbar`...
- Изображения: `BitmapImage`, `PhotoImage`.

Библиотека Kivy

Основные особенности

- Новая и активно развивающаяся
- Поддерживает в том числе мобильные платформы
- Высокая эффективность

Пример программы

Файл `my.kv` — описание интерфейса

```
<MainWindow>:
    BoxLayout:
        size: root.size
        pos: root.pos
        orientation: 'vertical'

        Label:
            text: str(root.counter)
            font_size: 40

        Button:
            text: "Увеличить счетчик"
            on_press: root.increase()
            font_size: 40
```

Пример программы

Файл `main.py` — основная программа

```
from kivy.app import App
from kivy.uix.widget import Widget
from kivy.properties import NumericProperty

class MainWindow(Widget):
    counter = NumericProperty(0)

    def increase(self):
        self.counter += 1

class MyApp(App):  # (!) MyApp -> my.kv
    def build(self):
        return MainWindow()

MyApp().run()
```

Пример программы