GUIs

David Croft

Coventry University

david.croft@coventry.ac.uk

January 18, 2016

Overview

GUIs

David Croft

GUIs
Hello World!

Layout
Containers

Events
Event arguments
Loops

Recap

1 GUIs
- Hello World!

2 Layout
- Containers

3 Events
- Event arguments
- Loops

4 Recap

Coventry
University

GUIs

David Croft

GUIs
Hello World!

Layout
Containers

Events
Event arguments
Loops

Recap

Event driven   C

You're programs so far have followed a procedural pattern.

- Program is a series of steps.
- Moves through those steps in a predetermined pattern.
- Expects user input in a very specific order.

start $\longrightarrow$ (Step 1) $\rightarrow$ (Step 2) $\rightarrow$ (Step 3) $\longrightarrow$ end

Coventry
University

GUIs

David Croft

GUIs
  Hello World!

Layout
  Containers

Events
  Event arguments
  Loops

Recap

Event driven II

Going to look at event driven programming.

- Program reacts to events.
- Events have actions associated with them.
- Order and frequency of events is unpredictable.
- Does not have a predefined sequence of actions to perform.
- Does not have a predefined end.

GUIs

David Croft

GUIs
  Hello World!

Layout
  Containers

Events
  Event arguments
  Loops

Recap

Benefits

C
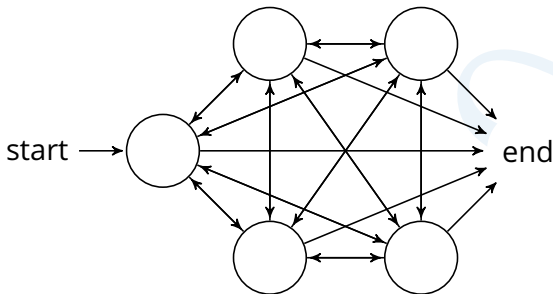
What sort of applications would benefit from an event driven paradigm?

- GUIs
- Control systems
- Embedded systems
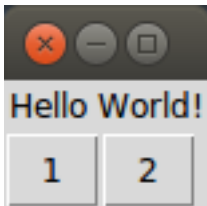
Coventry
University

Events examples

C

GUI events would include...

- ■ Button presses
- ■ Text entry
- ■ Keyboard events
  - ■ Pressing a key
  - ■ Releasing a key
- ■ Mouse events
  - ■ Pressing a button
  - ■ Releasing a button
  - ■ Moving
  - ■ Scrolling

Coventry
University

Hello World!

C

How to create a GUI.

- Wide range of different libraries available.
  - Depends on language and platform.
- Tkinter is the built-in Python default.

■ Window
■ Component/widget/element

Coventry
University

GUIs

David Croft

GUIs
Hello World!

Layout
Containers

Events
Event arguments
Loops

Recap

Hello World!   C

```python
import sys
from tkinter import *

def main():
    root = Tk()

    label = Label(root, text='Hello World!')
    label.pack()

    root.mainloop()

if __name__ == '__main__':
    sys.exit(main())
```
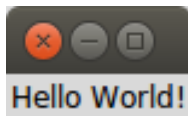
lec_getting_started.py

GUI code should be structured as a class.

- Become clear later.

```python
class Gui:
    def __init__(self, root):
        self.root = root

        self.label = Label(self.root, \
                        text='Hello World!')
        self.label.pack()

def main():
    root = Tk()
    gui = Gui(root)
    root.mainloop()
```

lec_classes.py

GUIs

David Croft

GUIs
  Hello World!

Layout
  Containers

Events
  Event arguments
  Loops

Recap

Coventry
University

GUIs

David Croft

GUIs
Hello World!

Layout
Containers

Events
Event arguments
Loops

Recap

So far we have seen how elements are added to window.

```python
class Gui:
    def __init__(self, root):
        self.root = root

        for i in range(1,10):
            button = Button(self.root, text=i)
            button.pack()
```
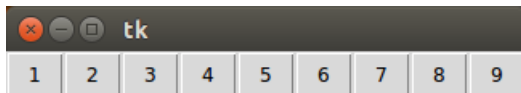lec_layout.py

Coventry
University

GUIs

David Croft

GUIs
Hello World!

Layout
Containers

Events
Event arguments
Loops

Recap

Layout II

Can use the `side` parameter for `.pack()`.

- ■ `TOP` (default).
- ■ Also `LEFT`, `RIGHT` and `BOTTOM`.

```python
class Gui:
    def __init__(self, root):
        self.root = root

        for i in range(1,10):
            button = Button(self.root, text=i)
            button.pack(side=LEFT)
```

lec_layout2.py

Coventry
University

Use `side` to control layout?

```python
class Gui:
  def __init__(self, root):
    self.root = root

    Button(self.root, text=1).pack(side=TOP)
    Button(self.root, text=2).pack(side=LEFT)
    Button(self.root, text=3).pack(side=LEFT)
    Button(self.root, text=4).pack(side=TOP)
    Button(self.root, text=5).pack(side=LEFT)
    Button(self.root, text=6).pack(side=LEFT)
    Button(self.root, text=7).pack(side=TOP)
    Button(self.root, text=8).pack(side=LEFT)
    Button(self.root, text=9).pack(side=LEFT)
```

lec_layout3.py

Need to learn about containers.

- Windows are containers.
    - Elements are 'contained' inside.
- Tkinter also has frames.
    - Special type of element.
    - Contains other elements.
- Group elements together using frames.
    - Can be visible/invisible.

Coventry
University

GUIs

David Croft

GUIs
Hello World!

Layout
Containers

Events
Event arguments
Loops

Recap

Frames

```python
class Gui:
  def __init__(self, root):
    self.root = root

    self.frame1 = Frame(self.root)
    self.frame1.pack()

    self.frame2 = Frame(self.root)
    self.frame2.pack()
```
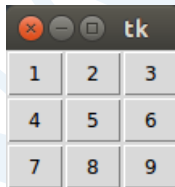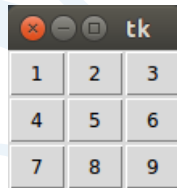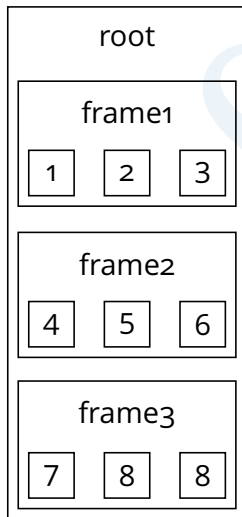
```python
    Button(self.frame1, text=1).pack(side=LEFT)
    Button(self.frame1, text=2).pack(side=LEFT)
    Button(self.frame1, text=3).pack(side=LEFT)
```
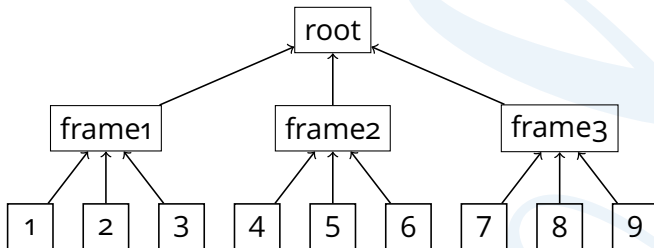
```python
    Button(self.frame3, text=7).pack(side=LEFT)
    Button(self.frame3, text=8).pack(side=LEFT)
    Button(self.frame3, text=9).pack(side=LEFT)
```

lec_frames.py

So what's happening?

- Elements are nested in containers.
- Containers are nested in other containers.

root

frame₁

| 1 | 2 | 3 |

frame₂

| 4 | 5 | 6 |

frame₃

| 7 | 8 | 8 |

tk

1	2	3
4	5	6
7	8	9

Coventry
University

GUIs

David Croft

GUIs
Hello World!

Layout
Containers

Events
Event arguments
Loops

Recap

Hierarchical structure

How do we get our code to actually DO stuff?

- Using Python/Tkinter.
- Other languages/frameworks == different syntax.
  - Same concepts.
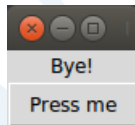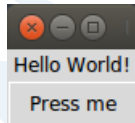- Event handling.
  - Bind events to elements.

Coventry
University

GUIs

David Croft

GUIs
Hello World!

Layout
Containers

Events
Event arguments
Loops

Recap

Events II   C

```python
class Gui:
  def __init__(self, root):
    self.root = root

    self.label = Label(self.root, text='Hello World!')
    self.label.pack()

    self.button = Button(self.root, text='Press me')
    self.button.bind('<Button-1>', self.say_bye)
    self.button.pack()

  def say_bye(self, event):
    self.label.config(text='Bye!')
```

lec_events.py

GUIs

David Croft

GUIs
Hello World!

Layout
Containers

Events
Event arguments
Loops

Recap

Callbacks

Callbacks are how we respond to events.
- ■ Functions that are passed to another function as an argument.

```python
class Gui:
    def __init__(self, root):
        self.root = root

        self.label = Label(self.root, text='Hello World!')
        self.label.pack()

        self.button = Button(self.root, text='Press me')
        self.button.bind('<Button-1>', self.say_bye)
        self.button.pack()

    def say_bye(self, event):
        self.label.config(text='Bye!')
```
lec_events.py

User ⟶ Event ⟶ Listener ⟶ Callback

Coventry
University

GUIs

David Croft

GUIs
  Hello World!

Layout
  Containers

Events
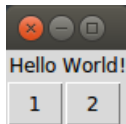 Event arguments
  Loops

Recap

Standard behaviour

User actions can trigger multiple events.

- I.e. clicking on button.
    1. Press LMB whilst pointer over button.
    2. Release LMB whilst pointer over button.
- Standard interaction code included in Tkinter.
    - Use `command` parameter.

```python
class Gui:
    def __init__(self, root):
        self.root = root

        self.button = Button(self.root, text='Press me' , \
                        command=self.say_bye)
        self.button.pack()

    def say_bye(self):
        self.label.config(text='Bye!')
```

`lec_events2.py`

GUIs

David Croft

GUIs
Hello World!

Layout
Containers

Events
Event arguments
Loops

Recap

Event arguments

Hello World!


```python
class Gui:
    def __init__(self, root):


        Button(self.root, text='1', \
            command=self.pressed_1).pack(side=LEFT)
        Button(self.root, text='2', \
            command=self.pressed_2).pack(side=LEFT)

    def pressed_1(self):
        self.label.config(text='Pressed 1')


    def pressed_2(self):
        self.label.config(text='Pressed 2')
```
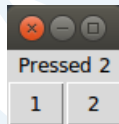
lec_event_args.py

GUIs

David Croft

GUIs
Hello World!

Layout
Containers

Events
Event arguments
Loops

Recap

Event arguments II

Much better to have one function.

- Function takes argument.
- Reuse of each button.
- Doesn't work.
  - Calls function immediately.
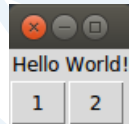- DEMO

```python
class Gui:
    def __init__(self, root):

        Button(self.root, text='1', \
            command=self.pressed_button(1)).pack(side=LEFT)
        Button(self.root, text='2', \
            command=self.pressed_button(2)).pack(side=LEFT)

    def pressed_button(self, number):
        self.label.config(text='Pressed %d' % number)
```

lec_event_args2.py

GUIs

David Croft

GUIs
Hello World!

Layout
Containers

Events
Event arguments
Loops

Recap

Event arguments III

`lambda` functions.

- Only calls function when button is pressed.

```python
class Gui:
    def __init__(self, root):

        Button(self.root, text='1', \
            command=lambda: self.pressed_button(1)).pack(side=LEFT)
        Button(self.root, text='2', \
            command=lambda: self.pressed_button(2)).pack(side=LEFT)

    def pressed_button(self, number):
        self.label.config(text='Pressed %d' % number)
```
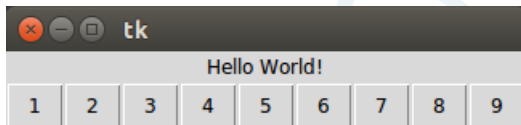
lec_event_args3.py

Already seen we can use create elements in loops.

- Create lots of elements easily.
- How can we combine this with callback arguments?

GUIs

David Croft

GUIs
Hello World!

Layout
Containers

Events
Event arguments
Loops

Recap

Loop arguments

A

`lambda` function in loop.

- What happens when any button is pressed?
  - DEMO.


```python
class Gui:
    def __init__(self, root):

        for i in range(1,10):
            b = Button(self.root, text=i, \
                command=lambda: self.pressed_button(i))
            b.pack(side=LEFT)

    def pressed_button(self, number):
        self.label.config(text='Pressed %d' % number)
```

lec_loop_args.py

GUIs

David Croft

GUIs
  Hello World!

Layout
  Containers

Events
  Event arguments
  Loops

Recap

Loop arguments II

A

```
for i in range(1,10):
    b = Button(self.root, text=i, \
        command=lambda: self.pressed_button(i))
    b.pack(side=LEFT)
```
`lec_loop_args.py`

- Each button will call a `lamda` function when pressed.
- The `lambda` function will call `self.pressed_button(i)`.
- `pressed_button()` will change the label using the value of `i`.
  - What is the value of `i`?
- It's whatever it was at the end of the loop, i.e. 9.
  - No matter what button we press, `i` is always 9.

Coventry
University

GUIs

David Croft

GUIs
Hello World!

Layout
Containers

Events
Event arguments
Loops

Recap

Loop arguments III

A

`lamda` arguments.

- The `lambda` function for each button copies the value of `i` right then.
- Uses that value when it runs in the future.

```python
class Gui:
  def __init__(self, root):

    for i in range(1,10):
      b = Button(self.root, text=i, \
        command=lambda n=i: self.pressed_button(n))
      b.pack(side=LEFT)

  def pressed_button(self, number):
    self.label.config(text='Pressed %d' % number)
```

`lec_loop_args2.py`

Quiz

GUIs

David Croft

GUIs
  Hello World!
Layout
  Containers
Events
  Event arguments
  Loops
Recap

Recap

- GUIs are an example of event driven programming.
- GUI elements are arranged in containers.
- Containers can hold other containers.
- User actions generate events.
- Callbacks are functions that are run in response to events.

Coventry
University

The End